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

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  LL           AAAAAA    NN      NN    GGGGGGGG    OOOOOO    PPPPPPPP
DDDDDDDD    BBBBBBBB      GGGGGGGG  LL           AAAAAA    NN      NN    GGGGGGGG    OOOOOO    PPPPPPPP
DD      DD  BB      BB  GG          LL          AA    AA  NN      NN  GG          OO      OO  PP      PP
DD      DD  BB      BB  GG          LL          AA    AA  NN      NN  GG          OO      OO  PP      PP
DD      DD  BB      BB  GG          LL          AA    AA  NNNN    NN  GG          OO      OO  PP      PP
DD      DD  BB      BB  GG          LL          AA    AA  NNNN    NN  GG          OO      OO  PP      PP
DD      DD  BBBBBBBB    GG          LL          AA    AA  NN  NN  NN  GG          OO      OO  PPPPPPPP
DD      DD  BBBBBBBB    GG          LL          AA    AA  NN  NN  NN  GG          OO      OO  PPPPPPPP
DD      DD  BB      BB  GG  GGGGGG  LL          AAAAAAAAAA  NN    NNNN  GG  GGGGGG  OO      OO  PP
DD      DD  BB      BB  GG  GGGGGG  LL          AAAAAAAAAA  NN    NNNN  GG  GGGGGG  OO      OO  PP
DD      DD  BB      BB  GG      GG  LL          AA    AA  NN      NN  GG      GG  OO      OO  PP
DD      DD  BB      BB  GG      GG  LL          AA    AA  NN      NN  GG      GG  OO      OO  PP
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  AA    AA  NN      NN    GGGGGG      OOOOOO    PP
DDDDDDDD    BBBBBBBB      GGGGGG    LLLLLLLLLL  AA    AA  NN      NN    GGGGGG      OOOOOO    PP

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 DBGLANGOP (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 ! WRITTEN BY
;    29          0029  1 !       Rich Title      Nov 1982
;    30          0030  1 !
;    31          0031  1 ! MODULE FUNCTION
;    32          0032  1 !       This module contains the routines that are used to evaluate the
;    33          0033  1 !       following language operators in C: * (dereference), & (address of),
;    34          0034  1 !       SIZEOF, and addition and subtraction involving pointers.
;    35          0035  1 !
;    36          0036  1 ! MODIFIED BY
;    37          0037  1 !       B. Becker       Nov 1983        ! Add routines for Ada tick operator support
;    38          0038  1 !       B. Becker       Dec 1983        ! Add arithmetic routines for Scaled Binary.
;    39          0039  1 !
;    40          0040  1
;    41          0041  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    42          0175  1
;    43          0176  1 FORWARD ROUTINE
;    44          0177  1     DBG$ABS_FIXED           : NOVALUE,      ! Perform the absolute value operation on Scaled Binary
;    45          0178  1     DBG$ADD_FIXED_FIXED     : NOVALUE,      ! Perform the add operation on Scaled Binary
;    46          0179  1     DBG$C_ADD_TPTR_L        : NOVALUE,      ! Add an integer to a pointer
;    47          0180  1     DBG$C_ADDRESS_OF        : NOVALUE,      ! Address-of operator
;    48          0181  1     DBG$C_INDIRECTION,                      ! Indirection operator
;    49          0182  1     DBG$C_PRE_DECR_TPTR,                    ! --PTR
;    50          0183  1     DBG$C_PRE_INCR_TPTR,                    ! ++PTR
;    51          0184  1     DBG$C_SIZEOF,                           ! SIZEOF operator
;    52          0185  1     DBG$C_SUB_TPTR_L        : NOVALUE,      ! Subtract an integer from a pointer
;    53          0186  1     DBG$C_SUB_TPTR_TPTR     : NOVALUE,      ! Subtract two pointers
;    54          0187  1     DBG$DIV_FIXED_FIXED     : NOVALUE,      ! Perform the divide operation on Scaled Binary
;    55          0188  1     DBG$ENUM_FIRST,                         ! Find first enumeration element
;    56          0189  1     DBG$ENUM_POS,                           ! Find position of enumeration element
;    57          0190  1     DBG$ENUM_SUCC,                          ! Find successor of enumeration element
```

```
;    58          0191  1      DBG$ENUM_VAL,                          ! Given position, find value of enum element
;    59          0192  1      DBG$EQL_FIXED_FIXED : NOVALUE,         ! Perform the equal evaluation on Scaled Binary
;    60          0193  1      DBG$EVAL_ADA_TICK,                     ! Evaluate an Ada tick operator
;    61          0194  1      DBG$GTR_FIXED_FIXED : NOVALUE,         ! Perform the greater than evaluation on Scaled Binary
;    62          0195  1      DBG$GEQ_FIXED_FIXED : NOVALUE,         ! Perform the greater than or equal evaluation on Scaled Binary
;    63          0196  1      DBG$LEQ_FIXED_FIXED : NOVALUE,         ! Perform the less than or equal evaluation on Scaled Binary
;    64          0197  1      DBG$LSS_FIXED_FIXED : NOVALUE,         ! Perform the less than evaluation on Scaled Binary
;    65          0198  1      DBG$MAKE_VALUE_DESC,                   ! Gets a DST value from a Data Type Comp. entry
;    66          0199  1      DBG$MUL_FIXED_FIXED : NOVALUE,         ! Perform the multiply operation on Scaled Binary
;    67          0200  1      DBG$NEQ_FIXED_FIXED : NOVALUE,         ! Perform the not equal evaluation on Scaled Binary
;    68          0201  1      DBG$NORMALIZE_FIXED : NOVALUE,         ! Normalize a scaled binary
;    69          0202  1      DBG$PRED_ENUM       : NOVALUE,         ! Return the Predecessor of the enumerated type
;    70          0203  1      DBG$SUCC_ENUM       : NOVALUE,         ! Return the Successor of the enumerated type
;    71          0204  1      DBG$SUB_FIXED_FIXED : NOVALUE,         ! Perform the subtract operation on Scaled Binary
;    72          0205  1      DBG$TYPEID_TO_PRIMARY,                 ! Convert typeid to Primary
;    73          0206  1      DBG$UNARY_PLUS_FIXED: NOVALUE,         ! Perform the unary plus operation on Scaled Binary
;    74          0207  1      DBG$UNARY_MINUS_FIXED: NOVALUE,        ! Perform the unary minus operation on Scaled Binary
;    75          0208  1      MATCH_FIXED_BINARYS : NOVALUE;         ! Matches the scales of the fixed binarys
;    76          0209  1
;    77          0210  1  EXTERNAL
;    78          0211  1      DBG$GB_LANGUAGE: BYTE,                 ! Current language setting
;    79          0212  1      DBG$GL_CONVERT_TOKEN;                  ! Language value for call to EVAL_LANG_OPERATOR
;    80          0213  1
;    81          0214  1  LINKAGE
;    82          0215  1      JSB_R1 = JSB (REGISTER = 0, REGISTER = 1): PRESERVE (0, 1);
;    83          0216  1
;    84          0217  1  EXTERNAL ROUTINE
;    85          0218  1      DBG$BUILD_PRIMARY_SUBNODE: NOVALUE,    ! Used in constructing Primary Descriptors
;    86          0219  1      DBG$CVT_CVTLH_R1: JSB_R1 NOVALUE,      ! Convert longword to H_Float
;    87          0220  1      DBG$DATA_LENGTH,                       ! Obtain length from VMS descriptor
;    88          0221  1      DBG$EVAL_LANG_OPERATOR,                ! Convert a primary to a value descriptor
;    89          0222  1      DBG$FILL_IN_VMS_DESC,                  ! Fills in the VMS desc. fields
;    90          0223  1      DBG$GET_BIF_ARGUMENTS,                 ! Obtain Ada tick operator arguments
;    91          0224  1      DBG$GET_TEMPMEM,                       ! Allocate temporary memory
;    92          0225  1      DBG$MAKE_SKELETON_DESC,                ! Make up a descriptor.
;    93          0226  1      DBG$STA_ADDRESS_TO_REGDESCR,           ! Obtain register descriptor
;    94          0227  1      DBG$STA_SYMSIZE: NOVALUE,              ! Obtain length from SYMID
;    95          0228  1      DBG$STA_SYMTYPE: NOVALUE,              ! Obtain type from symid
;    96          0229  1      DBG$STA_SYMVALUE,                      ! Obtain value of the symbol
;    97          0230  1      DBG$STA_TYP_ARRAY: NOVALUE,            ! Obtain info about array
;    98          0231  1      DBG$STA_TYP_ENUM: NOVALUE,             ! Obtain info about enumeration type
;    99          0232  1      DBG$STA_TYP_TYPEDPTR: NOVALUE,         ! Obtain info about typed pointer
;   100          0233  1      DBG$STA_TYPEFCODE,                     ! Obtain FCODE from SYMID
;   101          0234  1      DBG$STA_TYP_SUBRNG: NOVALUE,           ! Obtain Parent typeid for subrange
;   102          0235  1      DBG$TYPEID_FOR_ATOMIC,                 ! Obtain TYPEID for an atomic data type
;   103          0236  1      DBG$TYPEID_FOR_TPTR;                   ! Obtain TYPEID for TPTR data type
;   104          0237  1
;   105          0238  1  BUILTIN
;   106          0239  1      DIVH,
;   107          0240  1      EMUL;
;   108          0241  1
```

```
;  110          0242  1 GLOBAL ROUTINE DBG$ABS_FIXED (ARG_DESC, RESULT_DESC): NOVALUE =
;  111          0243  1 !
;  112          0244  1 ! FUNCTION
;  113          0245  1 !
;  114          0246  1 !       This routine is called to perform the absolute value operation
;  115          0247  1 !       on a scaled binary variable.
;  116          0248  1 !
;  117          0249  1 ! INPUTS
;  118          0250  1 !
;  119          0251  1 !       ARG_DESC        - points to the value descriptor representing the
;  120          0252  1 !                         argument of the operation.
;  121          0253  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  122          0254  1 !                         of the operation.
;  123          0255  1 !
;  124          0256  1 ! OUTPUTS
;  125          0257  1 !
;  126          0258  1 !       The result value descriptor is filled in.
;  127          0259  1 !       No value is returned.
;  128          0260  1 !
;  129          0261  2     BEGIN
;  130          0262  2
;  131          0263  2     MAP
;  132          0264  2         RESULT_DESC     : REF DBG$VALDESC,
;  133          0265  2         ARG_DESC        : REF DBG$VALDESC;
;  134          0266  2
;  135          0267  2
;  136          0268  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = ABS(..ARG_DESC[DBG$L_VALUE_POINTER]);
;  137          0269  2     RESULT_DESC[DBG$B_VALUE_SCALE] = .ARG_DESC[DBG$B_VALUE_SCALE];
;  138          0270  2     RESULT_DESC[DBG$B_VALUE_DTYPE] = .ARG_DESC[DBG$B_VALUE_DTYPE];
;  139          0271  1     END;


                                .TITLE  DBGLANGOP
                                .IDENT  \V04-000\

                                .EXTRN  DBG$GB_LANGUAGE
                                .EXTRN  DBG$GL_CONVERT_TOKEN
                                .EXTRN  DBG$BUILD_PRIMARY_SUBNODE
                                .EXTRN  DBG$CVT_CVTLH_R1
                                .EXTRN  DBG$DATA_LENGTH
                                .EXTRN  DBG$EVAL_LANG_OPERATOR
                                .EXTRN  DBG$FILL_IN_VMS_DESC
                                .EXTRN  DBG$GET_BIF_ARGUMENTS
                                .EXTRN  DBG$GET_TEMPMEM
                                .EXTRN  DBG$MAKE_SKELETON_DESC
                                .EXTRN  DBG$STA_ADDRESS_TO_REGDESCR
                                .EXTRN  DBG$STA_SYMSIZE
                                .EXTRN  DBG$STA_SYMTYPE
                                .EXTRN  DBG$STA_SYMVALUE
                                .EXTRN  DBG$STA_TYP_ARRAY
                                .EXTRN  DBG$STA_TYP_ENUM
                                .EXTRN  DBG$STA_TYP_TYPEDPTR
                                .EXTRN  DBG$STA_TYPEFCODE
                                .EXTRN  DBG$STA_TYP_SUBRNG
                                .EXTRN  DBG$TYPEID_FOR_ATOMIC
                                .EXTRN  DBG$TYPEID_FOR_TPTR
```

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

                                            0004 00000          .ENTRY  DBG$ABS_FIXED, Save R2                  ; 0242
                        52      08      AC  D0 00002            MOVL    RESULT_DESC, R2                         ; 0268
                        50      04      AC  D0 00006            MOVL    ARG_DESC, R0                            ;
                        51      18      B0  D0 0000A            MOVL    @24(R0), R1                             ;
                                        03  18 0000E            BGEQ    1$                                      ;
                        51              51  CE 00010            MNEGL   R1, R1                                  ;
                18      B2              51  D0 00013 1$:        MOVL    R1, @24(R2)                             ;
                1C      A2      1C      A0  90 00017            MOVB    28(R0), 28(R2)                          ; 0269
                16      A2      16      A0  90 0001C            MOVB    22(R0), 22(R2)                          ; 0270
                                            04 00021            RET                                             ; 0271

; Routine Size:  34 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  141          0272  1 GLOBAL ROUTINE DBG$ADD_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;  142          0273  1 !
;  143          0274  1 ! FUNCTION
;  144          0275  1 !
;  145          0276  1 !       This routine is called to perform the add operation
;  146          0277  1 !       on a scaled binary variable.
;  147          0278  1 !
;  148          0279  1 ! INPUTS
;  149          0280  1 !
;  150          0281  1 !       ARG_DESC1       - points to the value descriptor representing the
;  151          0282  1 !                         left argument of the operation.
;  152          0283  1 !       ARG_DESC2       - points to the value descriptor representing the
;  153          0284  1 !                         right argument of the operation.
;  154          0285  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  155          0286  1 !                         of the operation.
;  156          0287  1 !
;  157          0288  1 ! OUTPUTS
;  158          0289  1 !
;  159          0290  1 !       The result value descriptor is filled in.
;  160          0291  1 !       No value is returned.
;  161          0292  1 !
;  162          0293  2     BEGIN
;  163          0294  2
;  164          0295  2     MAP
;  165          0296  2         ARG_DESC1       : REF DBG$VALDESC,
;  166          0297  2         ARG_DESC2       : REF DBG$VALDESC,
;  167          0298  2         RESULT_DESC     : REF DBG$VALDESC;
;  168          0299  2
;  169          0300  2     LOCAL
;  170          0301  2         RESULT_VALUE,
;  171          0302  2         SCALE,
;  172          0303  2         VAL_DESC1       : DBG$STG_DESC,
;  173          0304  2         VAL_DESC2       : DBG$STG_DESC,
;  174          0305  2         VALUE1,
;  175          0306  2         VALUE2;
;  176          0307  2
;  177          0308  2     ! Set up working variables.  This way we don't mess up anything important.
;  178          0309  2     !
;  179          0310  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;  180          0311  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;  181          0312  2
;  182          0313  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;  183          0314  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;  184          0315  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;  185          0316  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;  186          0317  2
;  187          0318  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;  188          0319  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;  189          0320  2
;  190          0321  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;  191          0322  2
;  192          0323  2     ! Do the add.
;  193          0324  2     !
;  194          0325  2     RESULT_VALUE = .VALUE1 + .VALUE2;
;  195          0326  2     SCALE = .VAL_DESC1[DSC$B_SCALE];
;  196          0327  2
;  197          0328  2     ! Has an overflow occured?
```

```
;  198          0329  2     !
;  199          0330  2     IF .RESULT_VALUE<31, 1, 0> NEQ .VALUE1<31, 1, 0> AND
;  200          0331  2        .RESULT_VALUE<31, 1, 0> NEQ .VALUE2<31, 1, 0>
;  201          0332  2     THEN
;  202          0333  3         BEGIN
;  203          0334  3         IF .RESULT_VALUE<0, 1, 0>
;  204          0335  3         THEN
;  205          0336  3             SIGNAL(DBG$_IFIXUND);
;  206          0337  3         RESULT_VALUE = .RESULT_VALUE ^ -1;
;  207          0338  3         SCALE = .SCALE + 1;
;  208          0339  3         RESULT_VALUE<31, 1, 0> = .VALUE1<31, 1, 0>;
;  209          0340  2         END;
;  210          0341  2
;  211          0342  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .RESULT_VALUE;
;  212          0343  2     RESULT_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  213          0344  2     RESULT_DESC[DBG$B_VALUE_SCALE] = .SCALE;
;  214          0345  2
;  215          0346  1     END;


                                    00FC 00000          .ENTRY  DBG$ADD_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 0272
                              5E        20  C2 00002      SUBL2   #32, SP                                     ;
                              57    04  AC  D0 00005      MOVL    ARG_DESC1, R7                               ; 0310
                    14  AE        14  A7  0C  28 00009      MOVC3   #12, 20(R7), VAL_DESC1                      ;
                              56    08  AC  D0 0000F      MOVL    ARG_DESC2, R6                               ; 0311
                    08  AE        14  A6  0C  28 00013      MOVC3   #12, 20(R6), VAL_DESC2                      ;
                              6E    18  B7  D0 00019      MOVL    @24(R7), VALUE1                             ; 0313
                          04  AE    18  B6  D0 0001D      MOVL    @24(R6), VALUE2                             ; 0314
                          18  AE        6E  9E 00022      MOVAB   VALUE1, VAL_DESC1+4                         ; 0315
                          0C  AE    04  AE  9E 00026      MOVAB   VALUE2, VAL_DESC2+4                         ; 0316
                              14  AE    9F 0002B          PUSHAB  VAL_DESC1                                   ; 0318
                      0000V   CF        01  FB 0002E      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                              08  AE    9F 00033          PUSHAB  VAL_DESC2                                   ; 0319
                      0000V   CF        01  FB 00036      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                              08  AE    9F 0003B          PUSHAB  VAL_DESC2                                   ; 0321
                              18  AE    9F 0003E          PUSHAB  VAL_DESC1                                   ;
                      0000V   CF        02  FB 00041      CALLS   #2, MATCH_FIXED_BINARYS                     ;
                    52            6E    04  AE  C1 00046  ADDL3   VALUE2, VALUE1, RESULT_VALUE                ; 0325
                                  53    1C  AE  98 0004B  CVTBL   VAL_DESC1+8, SCALE                          ; 0326
          50        03  AE            01      07  EF 0004F  EXTZV   #7, #1, VALUE1+3, R0                       ; 0330
          50            52            01      1F  ED 00055  CMPZV   #31, #1, RESULT_VALUE, R0                 ;
                                              2F  13 0005A  BEQL    2$                                        ;
          50        07  AE            01      07  EF 0005C  EXTZV   #7, #1, VALUE2+3, R0                       ; 0331
          50            52            01      1F  ED 00062  CMPZV   #31, #1, RESULT_VALUE, R0                 ;
                                              22  13 00067  BEQL    2$                                        ;
                                        0D    52  E9 00069  BLBC    RESULT_VALUE, 1$                          ; 0334
                                    0002878B  8F  DD 0006C  PUSHL   #16577T                                   ; 0336
                          00000000G     00    01  FB 00072  CALLS   #1, LIB$SIGNAL                           ;
                    52            52        FF  8F  78 00079 1$:  ASHL    #-1, RESULT_VALUE, RESULT_VALUE       ; 0337
                                              53  D6 0007E      INCL    SCALE                                 ; 0338
          50        03  AE            01      07  EF 00080  EXTZV   #7, #1, VALUE1+3, R0                       ; 0339
          52            01            1F      50  F0 00086  INSV    R0, #31, #1, RESULT_VALUE                 ;
                                        50    0C  AC  D0 0008B 2$:  MOVL    RESULT_DESC, R0                   ; 0342
                                  18    B0      52  D0 0008F      MOVL    RESULT_VALUE, @24(R0)               ;
```

```
                16   A0        08  90 00093         MOVB    #8, 22(R0)                      ; 0343
                1C   A0        53  90 00097         MOVB    SCALE, 28(R0)                   ; 0344
                                   04 0009B         RET                                     ; 0346

; Routine Size:  156 bytes,    Routine Base:  DBG$CODE + 0022
```

```
;  217     0347  1 GLOBAL ROUTINE DBG$C_ADD_TPTR_L (ARG1_DESC, ARG2_DESC, RESULT_DESC): NOVALUE =
;  218     0348  1 !
;  219     0349  1 ! FUNCTION
;  220     0350  1 !
;  221     0351  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  222     0352  1 !       + operator in C when one of the arguments is a pointer.
;  223     0353  1 !
;  224     0354  1 ! INPUTS
;  225     0355  1 !
;  226     0356  1 !       ARG1_DESC       - points to the value descriptor representing the left
;  227     0357  1 !                         argument of the + operator.
;  228     0358  1 !       ARG2_DESC       - points to the value descriptor representing the right
;  229     0359  1 !                         argument of the + operator.
;  230     0360  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  231     0361  1 !                         of the + operator.
;  232     0362  1 !
;  233     0363  1 ! OUTPUTS
;  234     0364  1 !
;  235     0365  1 !       The result value descriptor is filled in.
;  236     0366  1 !       No value is returned.
;  237     0367  1 !
;  238     0368  2     BEGIN
;  239     0369  2     MAP
;  240     0370  2         ARG1_DESC       : REF DBG$VALDESC,
;  241     0371  2         ARG2_DESC       : REF DBG$VALDESC,
;  242     0372  2         RESULT_DESC     : REF DBG$VALDESC;
;  243     0373  2
;  244     0374  2     LOCAL
;  245     0375  2         ARG1_IS_TPTR,                   ! TRUE if first arg is typed pointer
;  246     0376  2         ARG2_IS_TPTR,                   ! TRUE if second arg is typed pointer
;  247     0377  2         BITSIZE,                        ! Size in bits of pointed to object
;  248     0378  2         FCODE1,                         ! FCODE for first argument
;  249     0379  2         FCODE2,                         ! FCODE for second argument
;  250     0380  2         JUNK: VECTOR[4],                ! Dummy output parameter
;  251     0381  2         SCALE,                          ! Scale factor in the operation
;  252     0382  2         SYMID,                          ! Points to a SYMID
;  253     0383  2         TYPEID1,                        ! Points to a TYPEID
;  254     0384  2         TYPEID2;                        ! Points to a TYPEID
;  255     0385  2
;  256     0386  2     ! Obtain a typeid and fcode for the first argument.
;  257     0387  2     !
;  258     0388  2     TYPEID1 = .ARG1_DESC [DBG$L_DHDR_TYPEID];
;  259     0389  2     IF .TYPEID1 EQL 0
;  260     0390  2     THEN
;  261     0391  2         FCODE1 = 0
;  262     0392  2     ELSE
;  263     0393  2         FCODE1 = DBG$STA_TYPEFCODE (.TYPEID1);
;  264     0394  2
;  265     0395  2     ! Obtain a typeid and fcode for the second argument.
;  266     0396  2     !
;  267     0397  2     TYPEID2 = .ARG2_DESC [DBG$L_DHDR_TYPEID];
;  268     0398  2     IF .TYPEID2 EQL 0
;  269     0399  2     THEN
;  270     0400  2         FCODE2 = 0
;  271     0401  2     ELSE
;  272     0402  2         FCODE2 = DBG$STA_TYPEFCODE (.TYPEID2);
;  273     0403  2
```

```
;  274          0404  2      ! One of the two arguments must be of type array or typed pointer.
;  275          0405  2      ! The other must be an integer. Set the flag ARG1_IS_TPTR if the
;  276          0406  2      ! first argument is the one of type TPTR or array. Set the
;  277          0407  2      ! flag ARG2_IS_TPTR if the second argument is the one which is of
;  278          0408  2      ! type TPTR or array.
;  279          0409  2      !
;  280          0410  2      IF  .FCODE1 EQL RST$K_TYPE_ARRAY OR .FCODE1 EQL RST$K_TYPE_TPTR
;  281          0411  2      THEN
;  282          0412  2          ARG1_IS_TPTR = TRUE
;  283          0413  2      ELSE
;  284          0414  2          ARG1_IS_TPTR = FALSE;
;  285          0415  2      IF .FCODE2 EQL RST$K_TYPE_ARRAY OR .FCODE2 EQL RST$K_TYPE_TPTR
;  286          0416  2      THEN
;  287          0417  2          ARG2_IS_TPTR = TRUE
;  288          0418  2      ELSE
;  289          0419  2          ARG2_IS_TPTR = FALSE;
;  290          0420  2
;  291          0421  2      ! Ensure that exactly one of the flags is TRUE.
;  292          0422  2      !
;  293          0423  2      IF .ARG1_IS_TPTR AND .ARG2_IS_TPTR
;  294          0424  2      THEN
;  295          0425  2          $DBG_ERROR ('DBGLANGOP\both args are pointer');
;  296          0426  3      IF (NOT .ARG1_IS_TPTR) AND (NOT .ARG2_IS_TPTR)
;  297          0427  2      THEN
;  298          0428  2          $DBG_ERROR ('DBGLANGOP\DBG$C_ADD_TPTR_L neither arg is pointer');
;  299          0429  2
;  300          0430  2      ! Obtain a SYMID which describes the type of the object that
;  301          0431  2      ! the typed pointer points to.
;  302          0432  2      !
;  303          0433  2      IF .ARG1_IS_TPTR
;  304          0434  2      THEN
;  305          0435  2          IF .FCODE1 EQL RST$K_TYPE_TPTR
;  306          0436  2          THEN
;  307          0437  2              DBG$STA_TYP_TYPEDPTR (.TYPEID1, SYMID)
;  308          0438  2          ELSE
;  309          0439  2              DBG$STA_TYP_ARRAY (.TYPEID1, JUNK[0], SYMID,    ! The JUNK parameters
;  310          0440  2                                 JUNK[1], JUNK[2], JUNK[3]) ! must be distinct
;  311          0441  2                                                            ! addresses; see note
;  312          0442  2                                                            ! in RSTTYPES
;  313          0443  2      ELSE
;  314          0444  2          IF .ARG2_IS_TPTR
;  315          0445  2          THEN
;  316          0446  2              IF .FCODE2 EQL RST$K_TYPE_TPTR
;  317          0447  2              THEN
;  318          0448  2                  DBG$STA_TYP_TYPEDPTR (.TYPEID2, SYMID)
;  319          0449  2              ELSE
;  320          0450  2                  DBG$STA_TYP_ARRAY (.TYPEID2, JUNK[0], SYMID,
;  321          0451  2                                     JUNK[1], JUNK[2], JUNK[3]);
;  322          0452  2
;  323          0453  2      ! Obtain the bitsize of the object. The scale factor for the addition
;  324          0454  2      ! is in bytes, so convert to bytes.
;  325          0455  2      !
;  326          0456  2      DBG$STA_SYMSIZE (.SYMID, BITSIZE);
;  327          0457  2      SCALE = (.BITSIZE+7) / 8;
;  328          0458  2
;  329          0459  2      ! Now perform the addition, scaling the appropriate operand.
;  330          0460  2      !
```

```
;  331          0461  2     IF .ARG1_IS_TPTR
;  332          0462  2     THEN
;  333          0463  3         BEGIN
;  334          0464  3         RESULT_DESC[DBG$L_VALUE_VALUE0] =
;  335          0465  3                 .ARG1_DESC[DBG$L_VALUE_VALUE0] *
;  336          0466  3             .SCALE * .ARG2_DESC[DBG$L_VALUE_VALUE0];
;  337          0467  3         RESULT_DESC[DBG$L_DHDR_TYPEID] = .ARG1_DESC[DBG$L_DHDR_TYPEID];
;  338          0468  3         END
;  339          0469  2     ELSE
;  340          0470  3         BEGIN
;  341          0471  3         RESULT_DESC[DBG$L_VALUE_VALUE0] =
;  342          0472  3             .SCALE * .ARG1_DESC[DBG$L_VALUE_VALUE0] *
;  343          0473  3                 .ARG2_DESC[DBG$L_VALUE_VALUE0];
;  344          0474  3         RESULT_DESC[DBG$L_DHDR_TYPEID] = .ARG2_DESC[DBG$L_DHDR_TYPEID];
;  345          0475  2         END;
;  346          0476  2
;  347          0477  2     ! Signal an informational message informing the user that scaling has
;  348          0478  2     ! taken place.
;  349          0479  2     !
;  350          0480  2     SIGNAL(DBG$_SCALEADD, 2, .SCALE,
;  351          0481  3         (IF .ARG1_IS_TPTR
;  352          0482  3          THEN UPLIT BYTE (%ASCIC 'right')
;  353          0483  2          ELSE UPLIT BYTE (%ASCIC 'left' )));
;  354          0484  1     END;


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

68 74 6F 62 5C 50 4F 47 4E 41 4C 47 42 44 1F  00000 P.AAA:  .ASCII  <31>\DBGLANGOP\<92>\both args are poin\     ;
   6E 69 6F 70 20 65 72 61 20 73 67 72 61 20  0000F
                              72 65 74        0001D        .ASCII  \ter\                                       ;
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 31  00020 P.AAB:  .ASCII  \1DBGLANGOP\<92>\DBG$C_ADD_TPTR_L neithe\  ;
65 6E 20 4C 5F 52 54 50 54 5F 44 44 41 5F 43  0002F
                              65 68 74 69     0003E
65 74 6E 69 6F 70 20 73 69 20 67 72 61 20 72  00042        .ASCII  \r arg is pointer\                          ;
                              72              00051
                        74 68 67 69 72 05     00052 P.AAC:  .ASCII  <5>\right\                                  ;
                           74 66 65 6C 04     00058 P.AAD:  .ASCII  <4>\left\                                   ;


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

                                          0FFC 00000    .ENTRY  DBG$C_ADD_TPTR_L, Save R2,R3,R4,R5,R6,R7,-  ; 0347
                                                                R8,R9,R10,R11                                 ;
                   5B 00000000G      00   9E 00002    MOVAB   LIB$SIGNAL, R11                               ;
                   5A 00000000'      EF   9E 00009    MOVAB   P.AAA, R10                                    ;
                   5E                18   C2 00010    SUBL2   #24, SP                                       ;
                   55          04    AC   D0 00013    MOVL    ARG1_DESC, R5                                 ; 0388
                   56          08    A5   D0 00017    MOVL    8(R5), TYPEID1                                ;
                                     04   12 0001B    BNEQ    1$                                            ; 0389
                                     58   D4 0001D    CLRL    FCODE1                                        ; 0391
                                     0C   11 0001F    BRB     2$                                            ;
                                     56   DD 00021 1$: PUSHL   TYPEID1                                      ; 0393
                00000000G            00   FB 00023    CALLS   #1, DBG$STA_TYPEFCODE                         ;
                   58                50   D0 0002A    MOVL    R0, FCODE1                                    ;
```

```
                        54      08  AC  D0 0002D 2$:     MOVL    ARG2_DESC, R4                           ; 0397
                        53      08  A4  D0 00031         MOVL    8(R4), TYPEID2
                                    04  12 00035         BNEQ    3$                                      ; 0398
                                    57  D4 00037         CLRL    FCODE2                                  ; 0400
                                    0C  11 00039         BRB     4$
                                    53  DD 0003B 3$:     PUSHL   TYPEID2                                 ; 0402
                00000000G   00          01  FB 0003D         CALLS   #1, DBG$STA_TYPEFCODE
                            57          50  D0 00044         MOVL    R0, FCODE2
                            01          58  D1 00047 4$:     CMPL    FCODE1, #1                          ; 0410
                                        05  13 0004A         BEQL    5$
                            06          58  D1 0004C         CMPL    FCODE1, #6
                                        05  12 0004F         BNEQ    6$
                            59          01  D0 00051 5$:     MOVL    #1, ARG1_IS_TPTR                    ; 0412
                                        02  11 00054         BRB     7$
                                        59  D4 00056 6$:     CLRL    ARG1_IS_TPTR                        ; 0414
                            01          57  D1 00058 7$:     CMPL    FCODE2, #1                          ; 0415
                                        05  13 0005B         BEQL    8$
                            06          57  D1 0005D         CMPL    FCODE2, #6
                                        05  12 00060         BNEQ    9$
                            52          01  D0 00062 8$:     MOVL    #1, ARG2_IS_TPTR                    ; 0417
                                        02  11 00065         BRB     10$
                                        52  D4 00067 9$:     CLRL    ARG2_IS_TPTR                        ; 0419
                            13          59  E9 00069 10$:    BLBC    ARG1_IS_TPTR, 12$                   ; 0423
                            0D          52  E9 0006C         BLBC    ARG2_IS_TPTR, 11$
                                        5A  DD 0006F         PUSHL   R10                                 ; 0425
                                        01  DD 00071         PUSHL   #1
                            00028362    8F  DD 00073         PUSHL   #164706
                            6B          03  FB 00079         CALLS   #3, LIB$SIGNAL
                            14          59  E8 0007C 11$:    BLBS    ARG1_IS_TPTR, 14$                   ; 0426
                            0E          52  E8 0007F 12$:    BLBS    ARG2_IS_TPTR, 13$
                                  20    AA  9F 00082         PUSHAB  P.AAB                               ; 0428
                                        01  DD 00085         PUSHL   #1
                            00028362    8F  DD 00087         PUSHL   #164706
                            6B          03  FB 0008D         CALLS   #3, LIB$SIGNAL
                            1E          59  E9 00090 13$:    BLBC    ARG1_IS_TPTR, 16$                   ; 0433
                            06          58  D1 00093 14$:    CMPL    FCODE1, #6                          ; 0435
                                        06  12 00096         BNEQ    15$
                                  4040  8F  BB 00098         PUSHR   #^M<R6,SP>                          ; 0437
                                        1F  11 0009C         BRB     17$
                                    14  AE  9F 0009E 15$:    PUSHAB  JUNK+12                             ; 0440
                                    14  AE  9F 000A1         PUSHAB  JUNK+8
                                    14  AE  9F 000A4         PUSHAB  JUNK+4
                                    0C  AE  9F 000A7         PUSHAB  SYMID                               ; 0439
                                    18  AE  9F 000AA         PUSHAB  JUNK
                                        56  DD 000AD         PUSHL   TYPEID1
                                        26  11 000AF         BRB     19$
                            2A          52  E9 000B1 16$:    BLBC    ARG2_IS_TPTR, 20$                   ; 0444
                            06          57  D1 000B4         CMPL    FCODE2, #6                          ; 0446
                                        0D  12 000B7         BNEQ    18$
                                  4008  8F  BB 000B9         PUSHR   #^M<R3,SP>                          ; 0448
                00000000G   00          02  FB 000BD 17$:    CALLS   #2, DBG$STA_TYP_TYPEDPTR
                                        18  11 000C4         BRB     20$
                                    14  AE  9F 000C6 18$:    PUSHAB  JUNK+12                             ; 0451
                                    14  AE  9F 000C9         PUSHAB  JUNK+8
                                    14  AE  9F 000CC         PUSHAB  JUNK+4
                                    0C  AE  9F 000CF         PUSHAB  SYMID                               ; 0450
                                    18  AE  9F 000D2         PUSHAB  JUNK
```

```
                                        53   DD 000D5          PUSHL   TYPEID2                                          ;
                00000000G   00          06   FB 000D7 19$:     CALLS   #6, DBG$STA_TYP_ARRAY                           ;
                                  04    AE   9F 000DE 20$:     PUSHAB  BITSIZE                                         ; 0456
                                  04    AE   DD 000E1          PUSHL   SYMID                                           ;
                00000000G   00          02   FB 000E4          CALLS   #2, DBG$STA_SYMSIZE                             ;
           50         04    AE          07   C1 000EB          ADDL3   #7, BITSIZE, R0                                 ; 0457
                                        50   C6 000F0          DIVL2   #8, SCALE                                       ;
                                        51   AC D0 000F3          MOVL    RESULT_DESC, R1                            ; 0464
                                        56   AC D0 000F7          MOVL    RESULT_DESC, R6                            ; 0467
                                        12   E9 000FB          BLBC    ARG1_IS_TPTR, 21$                               ;
           53                           50   A4 C5 000FE          MULL3   32(R4), SCALE, R3                          ; 0466
                                  20    A1   20 B543 9E 00103     MOVAB   @32(R5)[R3], 32(R1)                        ;
                                  08    A6   08   A5 D0 00109     MOVL    8(R5), 8(R6)                               ; 0467
                                        10   11 0010E          BRB     22$                                             ; 0461
           53                           50   A5 C5 00110 21$:     MULL3   32(R5), SCALE, R3                          ; 0472
                                  20    A1   20 B443 9E 00115     MOVAB   @32(R4)[R3], 32(R1)                        ; 0473
                                  08    A6   08   A4 D0 0011B     MOVL    8(R4), 8(R6)                               ; 0474
                                        06   E9 00120 22$:     BLBC    ARG1_IS_TPTR, 23$                               ; 0481
                                        51   52   AA 9F 00123     MOVAB   P.AAC, R1                                  ; 0482
                                        04   11 00127          BRB     24$                                             ;
                                        51   58   AA 9E 00129 23$:     MOVAB   P.AAD, R1                          ; 0483
                                        03   BB 0012D 24$:     PUSHR   #^M<R0,R1>                                      ; 0480
                                        02   DD 0012F          PUSHL   #2                                              ;
                                  0002870B   8F   DD 00131          PUSHL   #165643                                   ;
                                        6B   04   FB 00137          CALLS   #4, LIB$SIGNAL                            ;
                                             04 0013A          RET                                                     ; 0484

; Routine Size:  315 bytes,    Routine Base:  DBG$CODE + 00BE
```

```
;  356     0485  1 GLOBAL ROUTINE DBG$C_ADDRESS_OF (ARG_DESC, RESULT_DESC): NOVALUE =
;  357     0486  1 !
;  358     0487  1 ! FUNCTION
;  359     0488  1 !
;  360     0489  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  361     0490  1 !       & operator in C.
;  362     0491  1 !
;  363     0492  1 ! INPUTS
;  364     0493  1 !
;  365     0494  1 !       ARG_DESC        - points to the value descriptor representing the argument
;  366     0495  1 !                         of the & operator.
;  367     0496  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  368     0497  1 !                         of the & operator.
;  369     0498  1 !
;  370     0499  1 ! OUTPUTS
;  371     0500  1 !
;  372     0501  1 !       The result value descriptor is filled in.
;  373     0502  1 !       No value is returned.
;  374     0503  1 !
;  375     0504  2     BEGIN
;  376     0505  2     MAP
;  377     0506  2         ARG_DESC        : REF DBG$VALDESC,
;  378     0507  2         RESULT_DESC     : REF DBG$VALDESC;
;  379     0508  2
;  380     0509  2     LOCAL
;  381     0510  2         REGDESCR: DBG$REGDESCR;
;  382     0511  2
;  383     0512  2     ! Fill in the value field of the result value descriptor. Since we
;  384     0513  2     ! used the DBG$PRIM_TO_ADDR routine to convert the Primary to
;  385     0514  2     ! a value descriptor, address of the argument is already in
;  386     0515  2     ! the VALUE0 field of the input value descriptor. We thus only
;  387     0516  2     ! need to copy this address.
;  388     0517  2     !
;  389     0518  2     RESULT_DESC[DBG$L_VALUE_VALUE0] = .ARG_DESC[DBG$L_VALUE_VALUE0];
;  390     0519  2
;  391     0520  2     ! Check for typeid of zero. This indicates that the argument was
;  392     0521  2     ! not an identifier (it was instead the result of an expression),
;  393     0522  2     ! and therefore we cannot apply & to it.
;  394     0523  2     !
;  395     0524  2     IF .ARG_DESC[DBG$L_DHDR_TYPEID] EQL 0
;  396     0525  2     THEN
;  397     0526  2         SIGNAL (DBG$_AMPERSAND);
;  398     0527  2
;  399     0528  2     ! Fill in the TYPEID field of the result. This typeid describes
;  400     0529  2     ! the type "pointer to type X", where X is the type of the
;  401     0530  2     ! argument. A routine in RSTTYPES builds this typeid for us.
;  402     0531  2     !
;  403     0532  2     RESULT_DESC[DBG$L_DHDR_TYPEID] =
;  404     0533  2         DBG$TYPEID_FOR_TPTR (.ARG_DESC[DBG$L_DHDR_TYPEID]);
;  405     0534  2
;  406     0535  2     ! Check for result being a register. This is an error since you
;  407     0536  2     ! cannot do & on a variable that is bound to a register.
;  408     0537  2     !
;  409     0538  2     REGDESCR = DBG$STA_ADDRESS_TO_REGDESCR (.RESULT_DESC[DBG$L_VALUE_VALUE0]);
;  410     0539  2     IF .REGDESCR NEQ 0
;  411     0540  2     THEN
;  412     0541  2         SIGNAL (DBG$_ADDRREG, 1, .REGDESCR[DBG$B_REGD_REGNUM]);
```

```
;  413          0542  2
;  414          0543  2     RETURN;
;  415          0544  1     FND;


                                                001C 00000          .ENTRY  DBG$C_ADDRESS_OF, Save R2,R3,R4                 ; 0485
                                    54 00000000G  00   9E 00002     MOVAB   LIB$SIGNAL, R4                                  ;
                                    52        04  AC   7D 00009     MOVQ    ARG_DESC, R2                                    ; 0518
                                20  A3        20  A2   D0 0000D     MOVL    32(R2), 32(R3)                                  ;
                                              08  A2   D5 00012     TSTL    8(R2)                                           ; 0524
                                                  09   12 00015     BNEQ    1$                                              ;
                                    00028F38      8F   DD 00017     PUSHL   #167736                                         ; 0526
                                    64            01   FB 0001D     CALLS   #1, LIB$SIGNAL                                  ;
                                              08  A2   DD 00020 1$:  PUSHL   8(R2)                                          ; 0533
                            00000000G         00   01   FB 00023     CALLS   #1, DBG$TYPEID_FOR_TPTR                         ;
                                  08  A3              50   D0 0002A     MOVL    R0, 8(R3)                                       ;
                                              20  A3   DD 0002E     PUSHL   32(R3)                                          ; 0538
                            00000000G         00   01   FB 00031     CALLS   #1, DBG$STA_ADDRESS_TO_REGDESCR                 ;
                                                   50   D5 00038     TSTL    REGDESCR                                        ; 0539
                                                   10   13 0003A     BEQL    2$                                              ;
        7E          50                     08        08   EF 0003C     EXTZV   #8, #8, REGDESCR, -(SP)                         ; 0541
                                                   01   DD 00041     PUSHL   #1                                              ;
                                    00028F48       8F   DD 00043     PUSHL   #167752                                         ;
                                    64             03   FB 00049     CALLS   #3, LIB$SIGNAL                                  ;
                                                   04   0004C 2$:    RET                                                     ; 0544

; Routine Size:  77 bytes,    Routine Base:  DBG$CODE + 01F9
```

```
;  417          0545  1 GLOBAL ROUTINE DBG$C_INDIRECTION (ARG_DESC) =
;  418          0546  1 !
;  419          0547  1 ! FUNCTION
;  420          0548  1 !
;  421          0549  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  422          0550  1 !       * operation in C.
;  423          0551  1 !
;  424          0552  1 ! INPUTS
;  425          0553  1 !
;  426          0554  1 !       ARG_DESC        - points to the value descriptor representing the argument
;  427          0555  1 !                         of the indirection operation.
;  428          0556  1 !
;  429          0557  1 ! OUTPUTS
;  430          0558  1 !
;  431          0559  1 !       A Primary Descriptor is constructed out of temporary memory representing
;  432          0560  1 !       the result of the indirection. A pointer to this Primary Descriptor
;  433          0561  1 !       is returned.
;  434          0562  1 !
;  435          0563  2     BEGIN
;  436          0564  2     MAP
;  437          0565  2         ARG_DESC        : REF DBG$VALDESC;
;  438          0566  2
;  439          0567  2     LOCAL
;  440          0568  2         ADDRESS,                        ! Address of pointed-to object
;  441          0569  2         ARG_TYPEID,                     ! TYPEID for argument
;  442          0570  2         FCODE,                          ! FCODE for argument
;  443          0571  2         JUNK: VECTOR[4],                ! unused output parameters
;  444          0572  2         RESULT_TYPEID;                  ! TYPEID for result
;  445          0573  2
;  446          0574  2
;  447          0575  2     ! Determine whether the object is a typed pointer or an array.
;  448          0576  2     !
;  449          0577  2     ARG_TYPEID = .ARG_DESC[DBG$L_DHDR_TYPEID];
;  450          0578  2     FCODE = DBG$STA_TYPEFCODE(.ARG_TYPEID);
;  451          0579  2
;  452          0580  2     ! Given the argument TYPEID, obtain a typeid for the pointed-to
;  453          0581  2     ! object. A new typeid will be created for this anonymous object.
;  454          0582  2     !
;  455          0583  2     IF .FCODE EQL RST$K_TYPE_TPTR
;  456          0584  2     THEN
;  457          0585  2         DBG$STA_TYP_TYPEDPTR (.ARG_TYPEID, RESULT_TYPEID)
;  458          0586  2     ELSE IF .FCODE EQL RST$K_TYPE_ARRAY
;  459          0587  2     THEN
;  460          0588  2         DBG$STA_TYP_ARRAY (.ARG_TYPEID, JUNK[0], RESULT_TYPEID,
;  461          0589  2                            JUNK[1], JUNK[2], JUNK[3])
;  462          0590  2     ELSE
;  463          0591  2         $DBG_ERROR ('DBGLANGOP\DBG$C_INDIRECTION arg must be pointer or array');
;  464          0592  2
;  465          0593  2     ! Determine the address of the object. The value of the pointer
;  466          0594  2     ! is the address of the pointed-to object. We have already checked
;  467          0595  2     ! that the argument is of type "pointer", so we assume its longword
;  468          0596  2     ! value is sitting in the normal "VALUE0" field of the descriptor,
;  469          0597  2     ! and we don't bother to do any further type checking, type conversions,
;  470          0598  2     ! bitfield extractions, or whatever.
;  471          0599  2     !
;  472          0600  2     ADDRESS = .ARG_DESC[DBG$L_VALUE_VALUE0];
;  473          0601  2
```

```
;  474          0602  2     ! Convert this TYPEID to a Primary, and return a pointer to this
;  475          0603  2     ! Primary.
;  476          0604  2     !
;  477          0605  2     RETURN DBG$TYPEID_TO_PRIMARY (.RESULT_TYPEID, .ADDRESS);
;  478          0606  1     END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 38  0005D P.AAE:    .ASCII  \8DBGLANGOP\<92>\DBG$C_INDIRECTION arg m\       ;
61 20 4E 4F 49 54 43 45 52 49 44 4E 49 5F 43  0006C                                                              ;
                                  6D 20 67 72 0007B                                                              ;
20 72 65 74 6E 69 6F 70 20 65 62 20 74 73 75  0007F            .ASCII  \ust be pointer or array\                   ;
                        79 61 72 72 61 20 72 6F 0008E                                                           ;


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

                                    000C 00000                  .ENTRY  DBG$C_INDIRECTION, Save R2,R3               ; 0545
                              5E      14   C2 00002              SUBL2   #20, SP
                              52  04   AC   D0 00005              MOVL    ARG_DESC, R2                                ; 0577
                              53  08   A2   D0 00009              MOVL    8(R2), ARG_TYPEID
                                       53   DD 0000D              PUSHL   ARG_TYPEID                                  ; 0578
                00000000G     00       01   FB 0000F              CALLS   #1, DBG$STA_TYPEFCODE
                              06       50   D1 00016              CMPL    FCODE, #6                                   ; 0583
                                       0D   12 00019              BNEQ    1$
                                 4008  8F   BB 0001B              PUSHR   #^M<R3,SP>                                  ; 0585
                00000000G     00       02   FB 0001F              CALLS   #2, DBG$STA_TYP_TYPEDPTR
                                       34   11 00026              BRB     3$
                              01       50   D1 00028 1$:          CMPL    FCODE, #1                                   ; 0586
                                       1A   12 0002B              BNEQ    2$
                                 10    AE   9F 0002D              PUSHAB  JUNK+12                                     ; 0589
                                 10    AE   9F 00030              PUSHAB  JUNK+8
                                 10    AE   9F 00033              PUSHAB  JUNK+4
                                 0C    AE   9F 00036              PUSHAB  RESULT_TYPEID                               ; 0588
                                 14    AE   9F 00039              PUSHAB  JUNK
                                       53   DD 0003C              PUSHL   ARG_TYPEID
                00000000G     00       06   FB 0003E              CALLS   #6, DBG$STA_TYP_ARRAY
                                       15   11 00045              BRB     3$
                          00000000'    EF   9F 00047 2$:          PUSHAB  P.AAE                                       ; 0591
                                       01   DD 0004D              PUSHL   #1
                          00028362     8F   DD 0004F              PUSHL   #164706
                00000000G     00       03   FB 00055              CALLS   #3, LIB$SIGNAL
                              50  20   A2   D0 0005C 3$:          MOVL    32(R2), ADDRESS                             ; 0600
                                       50   DD 00060              PUSHL   ADDRESS                                     ; 0605
                                 04    AE   DD 00062              PUSHL   RESULT_TYPEID
                    0000V     CF       02   FB 00065              CALLS   #2, DBG$TYPEID_TO_PRIMARY
                                       04   0006A                 RET                                                 ; 0606

; Routine Size:  107 bytes,    Routine Base:  DBG$CODE + 0246
```

```
;  480          0607  1 GLOBAL ROUTINE DBG$C_PRE_DECR_TPTR (ARG_DESC) =
;  481          0608  1 !
;  482          0609  1 ! FUNCTION
;  483          0610  1 !
;  484          0611  2     BEGIN
;  485          0612  2     MAP
;  486          0613  2         ARG_DESC: REF DBG$VALDESC;
;  487          0614  2     0
;  488          0615  1     END;


                                                0000 00000          .ENTRY  DBG$C_PRE_DECR_TPTR, Save nothing           ; 0607
                                              50  D4 00002          CLRL    R0                                          ; 0615
                                                  04 00004          RET                                                 ;

; Routine Size:  5 bytes,    Routine Base:  DBG$CODE + 02B1
```

```
;  490          0616  1 GLOBAL ROUTINE DBG$C_PRE_INCR_TPTR (ARG_DESC) =
;  491          0617  1 !
;  492          0618  1 ! FUNCTION
;  493          0619  1 !
;  494          0620  2     BEGIN
;  495          0621  2     MAP
;  496          0622  2         ARG_DESC: REF DBG$VALDESC;
;  497          0623  2     0
;  498          0624  1     END;


                                                0000 00000          .ENTRY  DBG$C_PRE_INCR_TPTR, Save nothing           ; 0616
                                              50  D4 00002          CLRL    R0                                          ; 0624
                                                  04 00004          RET                                                 ;

; Routine Size:  5 bytes,    Routine Base:  DBG$CODE + 02B6
```

```
;  500          0625  1 GLOBAL ROUTINE DBG$C_SIZEOF (ARG_DESC) =
;  501          0626  1 !
;  502          0627  1 ! FUNCTION
;  503          0628  1 !
;  504          0629  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  505          0630  1 !       SIZEOF operation in C.
;  506          0631  1 !
;  507          0632  1 ! INPUTS
;  508          0633  1 !
;  509          0634  1 !       ARG_DESC        - points to the value descriptor representing the argument
;  510          0635  1 !                         of the SIZEOF operator.
;  511          0636  1 !
;  512          0637  1 ! OUTPUTS
;  513          0638  1 !
;  514          0639  1 !       The return value is the result of the SIZEOF operation. This result
;  515          0640  1 !       is the size in bytes of its argument.
;  516          0641  1 !
;  517          0642  2     BEGIN
;  518          0643  2     MAP
;  519          0644  2         ARG_DESC        : REF DBG$VALDESC;
;  520          0645  2
;  521          0646  2     LOCAL
;  522          0647  2         BIT_SIZE;       ! Size in bits of the argument
;  523          0648  2
;  524          0649  2     ! Try obtaining the size from the SYMID.
;  525          0650  2     !
;  526          0651  2     IF .ARG_DESC[DBG$L_DHDR_SYMIDO] NEQ 0
;  527          0652  2     THEN
;  528          0653  2         DBG$STA_SYMSIZE (.ARG_DESC[DBG$L_DHDR_SYMIDO], BIT_SIZE)
;  529          0654  2
;  530          0655  2     ! Try obtaining the size from the TYPEID.
;  531          0656  2     !
;  532          0657  2     ELSE IF .ARG_DESC[DBG$L_DHDR_TYPEID] NEQ 0
;  533          0658  2     THEN
;  534          0659  2         DBG$STA_SYMSIZE (.ARG_DESC[DBG$L_DHDR_TYPEID], BIT_SIZE)
;  535          0660  2
;  536          0661  2     ! Try obtaining the size from the VMS descriptor.
;  537          0662  2     !
;  538          0663  2     ELSE IF .ARG_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC
;  539          0664  2          OR .ARG_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR
;  540          0665  2     THEN
;  541          0666  2         BIT_SIZE = DBG$DATA_LENGTH (ARG_DESC[DBG$A_VALUE_VMSDESC])
;  542          0667  2
;  543          0668  2     ! If the value descriptor did not have a symid or a typeid, and
;  544          0669  2     ! it was not an atomic or vax-standard descriptor type, then
;  545          0670  2     ! we are unable to determine the bit size. This situation should
;  546          0671  2     ! not arise, so we signal an error.
;  547          0672  2     !
;  548          0673  2     ELSE
;  549          0674  2         $DBG_ERROR ('DBGLANGOP\DBG$C_SIZEOF cannot determine bitsize: typeid lost');
;  550          0675  2
;  551          0676  2     ! Return the size in bytes.
;  552          0677  2     !
;  553          0678  2     RETURN .BIT_SIZE / 8;
;  554          0679  1     END;
```

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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 3C 00096 P.AAF:  .ASCII  \<DBGLANGOP\<92>\DBG$C_SIZEOF cannot det\
74 6F 6E 6E 61 63 20 46 4F 45 5A 49 53 5F 43 000A5
                              74 65 64 20 000B4
3A 65 7A 69 73 74 69 62 20 65 6E 69 6D 72 65 000B8          .ASCII  \ermine bitsize: typeid lost\
      74 73 6F 6C 20 64 69 65 70 79 74 20 000C7

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

                                  0004 00000            .ENTRY  DBG$C_SIZEOF, Save R2                   ; 0625
                              5E      04  C2 00002       SUBL2   #4, SP                                  ;
                              52  04  AC  D0 00005       MOVL    ARG_DESC, R2                            ; 0651
                                  0C  A2  D5 00009       TSTL    12(R2)                                  ;
                                      07  13 0000C       BEQL    1$                                      ;
                                      5E  DD 0000E       PUSHL   SP                                      ; 0653
                                  0C  A2  DD 00010       PUSHL   12(R2)                                  ;
                                      0A  11 00013       BRB     2$                                      ;
                                  08  A2  D5 00015 1$:    TSTL    8(R2)                                   ; 0657
                                      0E  13 00018       BEQL    3$                                      ;
                                      5E  DD 0001A       PUSHL   SP                                      ; 0659
                                  08  A2  DD 0001C       PUSHL   8(R2)                                   ;
                00000000G     00        02  FB 0001F 2$:    CALLS   #2, DBG$STA_SYMSIZE                     ;
                                      30  11 00026       BRB     6$                                      ;
                              02  06  A2  91 00028 3$:    CMPB    6(R2), #2                               ; 0663
                                      06  13 0002C       BEQL    4$                                      ;
                              03  06  A2  91 0002E       CMPB    6(R2), #3                               ; 0664
                                      0F  12 00032       BNEQ    5$                                      ;
                                  14  A2  9F 00034 4$:    PUSHAB  20(R2)                                  ; 0666
                00000000G     00        01  FB 00037       CALLS   #1, DBG$DATA_LENGTH                     ;
                              6E        50  D0 0003E       MOVL    R0, BIT_SIZE                            ;
                                      15  11 00041       BRB     6$                                      ;
                    00000000'     EF  9F 00043 5$:    PUSHAB  P.AAF                                   ; 0674
                                  01  DD 00049       PUSHL   #1                                      ;
                    00028362      8F  DD 0004B       PUSHL   #164706                                 ;
                00000000G     00        03  FB 00051       CALLS   #3, LIB$SIGNA'                          ;
                50            6E        08  C7 00058 6$:    DIVL3   #8, BIT_SIZE, R0                        ; 0678
                                          04 0005C       RET                                             ; 0679

; Routine Size:  93 bytes,    Routine Base:  DBG$CODE + 02BB
```

```
;  556     0680  1 GLOBAL ROUTINE DBG$C_SUB_TPTR_L (ARG1_DESC, ARG2_DESC, RESULT_DESC): NOVALUE =
;  557     0681  1 !
;  558     0682  1 ! FUNCTION
;  559     0683  1 !
;  560     0684  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  561     0685  1 !       - operator in C when one of the arguments is a pointer.
;  562     0686  1 !
;  563     0687  1 ! INPUTS
;  564     0688  1 !
;  565     0689  1 !       ARG1_DESC       - points to the value descriptor representing the left
;  566     0690  1 !                         argument of the - operator.
;  567     0691  1 !       ARG2_DESC       - points to the value descriptor representing the right
;  568     0692  1 !                         argument of the - operator.
;  569     0693  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  570     0694  1 !                         of the - operator.
;  571     0695  1 !
;  572     0696  1 ! OUTPUTS
;  573     0697  1 !
;  574     0698  1 !       The result value descriptor is filled in.
;  575     0699  1 !       No value is returned.
;  576     0700  1 !
;  577     0701  2     BEGIN
;  578     0702  2     MAP
;  579     0703  2         ARG1_DESC       : REF DBG$VALDESC,
;  580     0704  2         ARG2_DESC       : REF DBG$VALDESC,
;  581     0705  2         RESULT_DESC     : REF DBG$VALDESC;
;  582     0706  2
;  583     0707  2     LOCAL
;  584     0708  2         BITSIZE,                        ! Size in bits of pointed to object
;  585     0709  2         FCODE1,                         ! FCODE for first argument
;  586     0710  2         JUNK: VECTOR[4],                ! Dummy output parameter
;  587     0711  2         SCALE,                          ! Scale factor in the operation
;  588     0712  2         SYMID,                          ! Points to a SYMID
;  589     0713  2         TYPEID1;                        ! Points to a TYPEID
;  590     0714  2
;  591     0715  2     ! Obtain a typeid and fcode for the first argument.
;  592     0716  2     !
;  593     0717  2     TYPEID1 = .ARG1_DESC [DBG$L_DHDR_TYPEID];
;  594     0718  2     IF .TYPEID1 EQL 0
;  595     0719  2     THEN
;  596     0720  3         BEGIN
;  597     0721  3         $DBG_ERROR ('DBGLANGOP\DBG$C_SUB_TPTR_L no typeid for first arg')
;  598     0722  3         END
;  599     0723  2     ELSE
;  600     0724  2         FCODE1 = DBG$STA_TYPEFCODE (.TYPEID1);
;  601     0725  2
;  602     0726  2     ! Obtain a SYMID which describes the type of the object that
;  603     0727  2     ! the typed pointer points to.
;  604     0728  2     !
;  605     0729  2     IF .FCODE1 EQL RST$K_TYPE_TPTR
;  606     0730  2     THEN
;  607     0731  2         DBG$STA_TYP_TYPEDPTR (.TYPEID1, SYMID)
;  608     0732  2     ELSE
;  609     0733  2         DBG$STA_TYP_ARRAY (.TYPEID1, JUNK[0], SYMID,
;  610     0734  2                            JUNK[1], JUNK[2], JUNK[3]);
;  611     0735  2
;  612     0736  2     ! Obtain the bitsize of the object. The scale factor for the addition
```

```
;  613          0737  2     ! is in bytes, so convert to bytes.
;  614          0738  2     !
;  615          0739  2     DBG$STA_SYMSIZE (.SYMID, BITSIZE);
;  616          0740  2     SCALE = (.BITSIZE+7) / 8;
;  617          0741  2
;  618          0742  2     ! Now perform the subtraction, scaling the second operand.
;  619          0743  2     !
;  620          0744  2     RESULT_DESC[DBG$L_VALUE_VALUE0] =
;  621          0745  2                 .ARG1_DESC[DBG$L_VALUE_VALUE0] -
;  622          0746  2         .SCALE * .ARG2_DESC[DBG$L_VALUE_VALUE0];
;  623          0747  2     RESULT_DESC[DBG$L_DHDR_TYPEID] = .ARG1_DESC[DBG$L_DHDR_TYPEID];
;  624          0748  2
;  625          0749  2     ! Signal an informational telling the user that a scale factor
;  626          0750  2     ! was applied.
;  627          0751  2     !
;  628          0752  2     SIGNAL(DBG$_SCALESUB, 2, .SCALE, UPLIT BYTE(%ASCIC 'right argument'));
;  629          0753  1     END;

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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 32  000D3 P.AAG:  .ASCII  \2DBGLANGOP\<92>\DBG$C_SUB_TPTR_L no typ\   ;
6F 6E 20 4C 5F 52 54 50 54 5F 42 55 53 5F 43  000E2                                                          ;
                                              70 79 74 20  000F1                                              ;
61 20 74 73 72 69 66 20 72 6F 66 20 64 69 65  000F5          .ASCII  \eid for 'first arg\                      ;
                                              67 72        00104                                              ;
74 6E 65 6D 75 67 72 61 20 74 68 67 69 72 0E  00106 P.AAH:  .ASCII  <14>\right argument\                       ;

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

                                  001C 00000            .ENTRY  DBG$C_SUB_TPTR_L, Save R2,R3,R4                 ; 0680
                54 00000000G   00  9E 00002            MOVAB   LIB$SIGNAL, R4                                  ;
                5E             18  C2 00009            SUBL2   #24, SP                                         ;
                53       04    AC  D0 0000C            MOVL    ARG1_DESC, R3                                   ; 0717
                52       08    A3  D0 00010            MOVL    8(R3), TYPEID1                                  ;
                               13  12 00014            BNEQ    1$                                              ; 0718
                   00000000'   EF  9F 00016            PUSHAB  P.AAG                                           ; 0721
                               01  DD 0001C            PUSHL   #1                                              ;
                   00028362    8F  DD 0001E            PUSHL   #164706                                         ;
                64             03  FB 00024            CALLS   #3, LIB$SIGNAL                                  ;
                               09  11 00027            BRB     2$                                              ; 0720
                               52  DD 00029  1$:       PUSHL   TYPEID1                                         ; 0724
00000000G       00             01  FB 0002B            CALLS   #1, DBG$STA_TYPEFCODE                           ;
                06             50  D1 00032  2$:       CMPL    FCODE1, #6                                      ; 0729
                               0D  12 00035            BNEQ    3$                                              ;
                     4004      8F  BB 00037            PUSHR   #^M<R2,SP>                                      ; 0731
00000000G       00             02  FB 0003B            CALLS   #2, DBG$STA_TYP_TYPEDPTR                        ;
                               18  11 00042            BRB     4$                                              ;
                         14    AE  9F 00044  3$:       PUSHAB  JUNK+12                                         ; 0734
                         14    AE  9F 00047            PUSHAB  JUNK+8                                          ;
                         14    AE  9F 0004A            PUSHAB  JUNK+4                                          ;
                         0C    AE  9F 0004D            PUSHAB  SYMID                                           ; 0733
                         18    AE  9F 00050            PUSHAB  JUNK                                            ;
                               52  DD 00053            PUSHL   TYPEID1                                         ;
```

```
                    00000000G  00          06  FB 00055          CALLS   #6, DBG$STA_TYP_ARRAY                    ;
                                        04  AE  9F 0005C 4$:      PUSHAB  BITSIZE                                 ; 0739
                                        04  AE  DD 0005F          PUSHL   SYMID                                   ;
                    00000000G  00           02  FB 00062          CALLS   #2, DBG$STA_SYMSIZE                     ;
                50      04  AE              07  C1 00069          ADDL3   #7, BITSIZE, R0                         ; 0740
                            50              08  C6 0006E          DIVL2   #8, SCALE                               ;
                            51          0C  AC  D0 00071          MOVL    RESULT_DESC, R1                         ; 0744
                            52          08  AC  D0 00075          MOVL    ARG2_DESC, R2                           ; 0746
                52          50          20  A2  C5 00079          MULL3   32(R2), SCALE, R2                       ;
            20  A1      20  A3              52  C3 0007E          SUBL3   R2, 32(R3), 32(R1)                      ;
                        08  A1          08  A3  D0 00084          MOVL    8(R3), 8(R1)                            ; 0747
                                00000000' EF  9F 00089          PUSHAB  P.AAH                                   ; 0752
                                            50  DD 0008F          PUSHL   SCALE                                   ;
                                            02  DD 00091          PUSHL   #2                                      ;
                                00028713    8F  DD 00093          PUSHL   #165651                                 ;
                                        64      04  FB 00099      CALLS   #4, LIB$SIGNAL                          ;
                                                04 0009C          RET                                             ; 0753

; Routine Size:  157 bytes,    Routine Base:  DBG$CODE + 0318
```

```
;  631          0754  1 GLOBAL ROUTINE DBG$C_SUB_TPTR_TPTR (ARG1_DESC, ARG2_DESC, RESULT_DESC): NOVALUE =
;  632          0755  1 !
;  633          0756  1 ! FUNCTION
;  634          0757  1 !
;  635          0758  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  636          0759  1 !       - operator in C when both of the arguments are pointers.
;  637          0760  1 !
;  638          0761  1 ! INPUTS
;  639          0762  1 !
;  640          0763  1 !       ARG1_DESC       - points to the value descriptor representing the left
;  641          0764  1 !                         argument of the - operator.
;  642          0765  1 !       ARG2_DESC       - points to the value descriptor representing the right
;  643          0766  1 !                         argument of the - operator.
;  644          0767  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  645          0768  1 !                         of the - operator.
;  646          0769  1 !
;  647          0770  1 ! OUTPUTS
;  648          0771  1 !
;  649          0772  1 !       The result value descriptor is filled in.
;  650          0773  1 !       No value is returned.
;  651          0774  1 !
;  652          0775  2     BEGIN
;  653          0776  2     MAP
;  654          0777  2         ARG1_DESC       : REF DBG$VALDESC,
;  655          0778  2         ARG2_DESC       : REF DBG$VALDESC,
;  656          0779  2         RESULT_DESC     : REF DBG$VALDESC;
;  657          0780  2
;  658          0781  2     LOCAL
;  659          0782  2         ARG1_IS_TPTR,                   ! TRUE if first arg is typed pointer
;  660          0783  2         ARG2_IS_TPTR,                   ! TRUE if second arg is typed pointer
;  661          0784  2         BITSIZE1,                       ! Size in bits of pointed to object
;  662          0785  2         BITSIZE2,                       ! Size in bits of pointed to object
;  663          0786  2         FCODE1,                         ! FCODE for first argument
;  664          0787  2         FCODE2,                         ! FCODE for second argument
;  665          0788  2         JUNK: VECTOR[4],                ! Dummy output parameter
;  666          0789  2         SCALE,                          ! Scale factor in the operation
;  667          0790  2         SYMID1,                         ! Points to a SYMID
;  668          0791  2         SYMID2,                         ! Points to a SYMID
;  669          0792  2         TYPEID1,                        ! Points to a TYPEID
;  670          0793  2         TYPEID2;                        ! Points to a TYPEID
;  671          0794  2
;  672          0795  2     ! Obtain a typeid and fcode for the first argument
;  673          0796  2     !
;  674          0797  2     TYPEID1 = .ARG1_DESC [DBG$L_DHDR_TYPEID];
;  675          0798  2     IF .TYPEID1 EQL 0
;  676          0799  2     THEN
;  677          0800  3         BEGIN
;  678          0801  3         $DBG_ERROR ('DBGLANGOP\DBG$C_SUB_TPTR_TPTR no typeid for first arg')
;  679          0802  3         END
;  680          0803  2     ELSE
;  681          0804  2         FCODE1 = DBG$STA_TYPEFCODE (.TYPEID1);
;  682          0805  2
;  683          0806  2     ! Obtain a typeid and fcode for the second argument.
;  684          0807  2     !
;  685          0808  2     TYPEID2 = .ARG2_DESC [DBG$L_DHDR_TYPEID];
;  686          0809  2     IF .TYPEID2 EQL 0
;  687          0810  2     THEN
```

```
;  688          0811  3          BEGIN
;  689          0812  3          $DBG_ERROR ('DBGLANGOP\DBG$C_SUB_TPTR_TPTR no typeid for second arg')
;  690          0813  3          END
;  691          0814  2      ELSE
;  692          0815  2          FCODE2 = DBG$STA_TYPEFCODE (.TYPEID2);
;  693          0816  2
;  694          0817  2      ! Both arguments must be of type array or TPTR.
;  695          0818  2      !
;  696          0819  2      IF  .FCODE1 NEQ RST$K_TYPE_ARRAY AND .FCODE1 NEQ RST$K_TYPE_TPTR
;  697          0820  2      THEN
;  698          0821  3          BEGIN
;  699          0822  3          $DBG_ERROR ('DBGLANGOP\DBG$C_SUB_TPTR_TPTR second arg is not ptr')
;  700          0823  2          END;
;  701          0824  2      IF .FCODE2 NEQ RST$K_TYPE_ARRAY AND .FCODE2 NEQ RST$K_TYPE_TPTR
;  702          0825  2      THEN
;  703          0826  3          BEGIN
;  704          0827  3          $DBG_ERROR ('DBGLANGOP\DBG$C_SUB_TPTR_TPTR second arg is not ptr')
;  705          0828  2          END;
;  706          0829  2
;  707          0830  2      ! Obtain a SYMID which describes the type of the object that
;  708          0831  2      ! the typed pointer points to.
;  709          0832  2      !
;  710          0833  2      IF .FCODE1 EQL RST$K_TYPE_TPTR
;  711          0834  2      THEN
;  712          0835  2          DBG$STA_TYP_TYPEDPTR (.TYPEID1, SYMID1)
;  713          0836  2      ELSE
;  714          0837  2          DBG$STA_TYP_ARRAY (.TYPEID1, JUNK[0], SYMID1,
;  715          0838  2                             JUNK[1], JUNK[2], JUNK[3]);
;  716          0839  2      IF .FCODE2 EQL RST$K_TYPE_TPTR
;  717          0840  2      THEN
;  718          0841  2          DBG$STA_TYP_TYPEDPTR (.TYPEID2, SYMID2)
;  719          0842  2      ELSE
;  720          0843  2          DBG$STA_TYP_ARRAY (.TYPEID2, JUNK[0], SYMID2,
;  721          0844  2                             JUNK[1], JUNK[2], JUNK[3]);
;  722          0845  2
;  723          0846  2      ! Check for compatible pointers (?)
;  724          0847  2      !
;  725          0848  2      !<<<---------------------------------
;  726          0849  2
;  727          0850  2      ! Obtain the bitsize of the objects. The scale factor for the addition
;  728          0851  2      ! is in bytes, so convert to bytes.
;  729          0852  2      !
;  730          0853  2      DBG$STA_SYMSIZE (.SYMID1, BITSIZE1);
;  731          0854  2      DBG$STA_SYMSIZE (.SYMID2, BITSIZE2);
;  732          0855  2      IF .BITSIZE1 NEQ .BITSIZE2
;  733          0856  2      THEN
;  734          0857  2          SIGNAL (DBG$_INCOMPPTR);
;  735          0858  2      SCALE = (.BITSIZE1+7) / 8;
;  736          0859  2      IF .SCALE EQL 0
;  737          0860  2      THEN
;  738          0861  2          $DBG_ERROR ('DBGLANGOP\DBG$SUB_TPTR_TPTR scale factor of zero');
;  739          0862  2
;  740          0863  2      ! Now perform the subtraction, scaling the result.
;  741          0864  2      !
;  742          0865  2      RESULT_DESC[DBG$L_VALUE_VALUE0] =
;  743          0866  2          (.ARG1_DESC[DBG$L_VALUE_VALUE0] - .ARG2_DESC[DBG$L_VALUE_VALUE0]) /
;  744          0867  2              .SCALE;
```

```
;  745          0868  2
;  746          0869  2      ! Signal an informational telling the user that scaling was applied
;  747          0870  2      ! to the result.
;  748          0871  2      !
;  749          0872  2      SIGNAL(DBG$_SCALESUB, 2, .SCALE, UPLIT BYTE (%ASCIC 'result'));
;  750          0873  1      END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 35  00115 P.AAI:   .ASCII  \5DBGLANGOP\<92>\DBG$C_SUB_TPTR_TPTR no \ ;
52 54 50 54 5F 52 54 50 54 5F 42 55 53 5F 43  00124                                                        ;
                              20 6F 6E 20     00133                                                        ;
73 72 69 66 20 72 6F 66 20 64 69 65 70 79 74  00137           .ASCII  \typeid for first arg\               ;
                              67 72 61 20 74  00146                                                        ;
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 36  0014B P.AAJ:   .ASCII  \6DBGLANGOP\<92>\DBG$C_SUB_TPTR_TPTR no \ ;
52 54 50 54 5F 52 54 50 54 5F 42 55 53 5F 43  0015A                                                        ;
                              20 6F 6E 20     00169                                                        ;
6F 63 65 73 20 72 6F 66 20 64 69 65 70 79 74  0016D           .ASCII  \typeid for second arg\              ;
                           67 72 61 20 64 6E  0017C                                                        ;
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 33  00182 P.AAK:   .ASCII  \3DBGLANGOP\<92>\DBG$C_SUB_TPTR_TPTR sec\ ;
52 54 50 54 5F 52 54 50 54 5F 42 55 53 5F 43  00191                                                        ;
                              63 65 73 20     001A0                                                        ;
20 74 6F 6E 20 73 69 20 67 72 61 20 64 6E 6F  001A4           .ASCII  \ond arg is not ptr\                 ;
                                    72 74 70  001B3                                                        ;
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 33  001B6 P.AAL:   .ASCII  \3DBGLANGOP\<92>\DBG$C_SUB_TPTR_TPTR sec\ ;
52 54 50 54 5F 52 54 50 54 5F 42 55 53 5F 43  001C5                                                        ;
                              63 65 73 20     001D4                                                        ;
20 74 6F 6E 20 73 69 20 67 72 61 20 64 6E 6F  001D8           .ASCII  \ond arg is not ptr\                 ;
                                    72 74 70  001E7                                                        ;
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 30  001EA P.AAM:   .ASCII  \0DBGLANGOP\<92>\DBG$SUB_TPTR_TPTR scale\ ;
73 20 52 54 50 54 5F 52 54 50 54 5F 42 55 53  001F9                                                        ;
                              65 6C 61 63     00208                                                        ;
6F 72 65 7A 20 66 6F 20 72 6F 74 63 61 66 20  0020C           .ASCII  \ factor of zero\                    ;
                        74 6C 75 73 65 72 06  0021B P.AAN:   .ASCII  <6>\result\                          ;


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

                                    0FFC 00000                  .ENTRY  DBG$C_SUB_TPTR_TPTR, Save R2,R3,R4,R5,R6,-  ; 0754
                                                                        R7,R8,R9,R10,R11                                ;
5B 00000000G   00   9E 00002                MOVAB   DBG$STA_TYP_TYPEDPTR, R11                                       ;
5A 00000000G   00   9E 00009                MOVAB   DBG$STA_TYPEFCODE, R10                                          ;
59 00000000'   EF   9E 00010                MOVAB   P.AAI, R9                                                       ;
58 00000000G   00   9E 00017                MOVAB   LIB$SIGNAL, R8                                                  ;
5E             20   C2 0001E                SUBL2   #32, SP                                                         ;
55       04    AC   D0 00021                MOVL    ARG1_DESC, R5                                                   ; 0797
53       08    A5   D0 00025                MOVL    8(R5), TYPEID1                                                  ;
                0F   12 00029                BNEQ    1$                                                              ; 0798
                59   DD 0002B                PUSHL   R9                                                              ; 0801
                01   DD 0002D                PUSHL   #1                                                              ;
   00028362     8F   DD 0002F                PUSHL   #164706                                                         ;
68              03   FB 00035                CALLS   #3, LIB$SIGNAL                                                  ;
                08   11 00038                BRB     2$                                                              ; 0800
                53   DD 0003A 1$:           PUSHL   TYPEID1                                                         ; 0804
```

```
                6A  01  FB 0003C          CALLS   #1, DBG$STA_TYPEFCODE                   ;
                57  50  D0 0003F          MOVL    R0, FCODE1                              ;
                54  08  AC  D0 00042 2$:   MOVL    ARG2_DESC, R4                           ; 0808
                52  08  A4  D0 00046       MOVL    8(R4), TYPEID2                          ;
                    10  12 0004A          BNEQ    3$                                      ; 0809
                    36  A9  9F 0004C       PUSHAB  P.AAJ                                   ; 0812
                        01  DD 0004F       PUSHL   #1                                      ;
                00028362  8F  DD 00051     PUSHL   #164706                                 ;
                68          03  FB 00057   CALLS   #3, LIB$SIGNAL                          ;
                            08  11 0005A   BRB     4$                                      ; 0811
                            52  DD 0005C 3$:  PUSHL   TYPEID2                             ; 0815
                6A          01  FB 0005E   CALLS   #1, DBG$STA_TYPEFCODE                  ;
                56          50  D0 00061   MOVL    R0, FCODE2                              ;
                01          57  D1 00064 4$:  CMPL    FCODE1, #1                          ; 0819
                            13  13 00067   BEQL    5$                                      ;
                06          57  D1 00069   CMPL    FCODE1, #6                              ;
                            0E  13 0006C   BEQL    5$                                      ;
                    6D  A9  9F 0006E       PUSHAB  P.AAK                                   ; 0822
                        01  DD 00071       PUSHL   #1                                      ;
                00028362  8F  DD 00073     PUSHL   #164706                                 ;
                68          03  FB 00079   CALLS   #3, LIB$SIGNAL                          ;
                01          56  D1 0007C 5$:  CMPL    FCODE2, #1                          ; 0824
                            14  13 0007F   BEQL    6$                                      ;
                06          56  D1 00081   CMPL    FCODE2, #6                              ;
                            0F  13 00084   BEQL    6$                                      ;
                  00A1  C9  9F 00086       PUSHAB  P.AAL                                   ; 0827
                        01  DD 0008A       PUSHL   #1                                      ;
                00028362  8F  DD 0008C     PUSHL   #164706                                 ;
                68          03  FB 00092   CALLS   #3, LIB$SIGNAL                          ;
                06          57  D1 00095 6$:  CMPL    FCODE1, #6                          ; 0833
                            09  12 00098   BNEQ    7$                                      ;
                  4008  8F  BB 0009A       PUSHR   #^M<R3,SP>                              ; 0835
                6B          02  FB 0009E   CALLS   #2, DBG$STA_TYP_TYPEDPTR              ;
                            18  11 000A1   BRB     8$                                      ;
                    1C  AE  9F 000A3 7$:  PUSHAB  JUNK+12                                 ; 0838
                    1C  AE  9F 000A6       PUSHAB  JUNK+8                                  ;
                    1C  AE  9F 000A9       PUSHAB  JUNK+4                                  ;
                    0C  AE  9F 000AC       PUSHAB  SYMID1                                  ; 0837
                    20  AE  9F 000AF       PUSHAB  JUNK                                    ;
                        53  DD 000B2       PUSHL   TYPEID1                                 ;
00000000G       00          06  FB 000B4   CALLS   #6, DBG$STA_TYP_ARRAY                  ;
                06          56  D1 000BB 8$:  CMPL    FCODE2, #6                          ; 0839
                            0A  12 000BE   BNEQ    9$                                      ;
                    04  AE  9F 000C0       PUSHAB  SYMID2                                  ; 0841
                        52  DD 000C3       PUSHL   TYPEID2                                 ;
                6B          02  FB 000C5   CALLS   #2, DBG$STA_TYP_TYPEDPTR              ;
                            18  11 000C8   BRB     10$                                     ;
                    1C  AE  9F 000CA 9$:  PUSHAB  JUNK+12                                 ; 0844
                    1C  AE  9F 000CD       PUSHAB  JUNK+8                                  ;
                    1C  AE  9F 000D0       PUSHAB  JUNK+4                                  ;
                    10  AE  9F 000D3       PUSHAB  SYMID2                                  ; 0843
                    20  AE  9F 000D6       PUSHAB  JUNK                                    ;
                        52  DD 000D9       PUSHL   TYPEID2                                 ;
00000000G       00          06  FB 000DB   CALLS   #6, DBG$STA_TYP_ARRAY                  ;
                    08  AE  9F 000E2 10$:  PUSHAB  BITSIZE1                              ; 0853
                    04  AE  DD 000E5       PUSHL   SYMID1                                  ;
00000000G       00          02  FB 000E8   CALLS   #2, DBG$STA_SYMSIZE                   ;
```

```
                          0C   AE  9F 000EF          PUSHAB  BITSIZE2                                 ; 0854
                          08   AE  DD 000F2          PUSHL   SYMID2                                   ;
          00000000G  00        02  FB 000F5          CALLS   #2, DBG$STA_SYMSIZE                      ;
                0C   AE   08   AE  D1 000FC          CMPL    BITSIZE1, BITSIZE2                       ; 0855
                               09  13 00101          BEQL    11$                                      ;
                    00028F30   8F  DD 00103          PUSHL   #167728                                  ; 0857
                     68        01  FB 00109          CALLS   #1, LIB$SIGNAL                           ;
       50       08   AE        07  C1 0010C  11$:    ADDL3   #7, BITSIZE1, R0                         ; 0858
       52            50        08  C7 00111          DIVL3   #8, R0, SCALE                            ;
                               0F  12 00115          BNEQ    12$                                      ; 0859
                        00D5   C9  9F 00117          PUSHAB  P.AAM                                    ; 0861
                               01  DD 0011B          PUSHL   #1                                       ;
                    00028362   8F  DD 0011D          PUSHL   #164706                                  ;
                     68        03  FB 00123          CALLS   #3, LIB$SIGNAL                           ;
                     50   0C   AC  D0 00126  12$:    MOVL    RESULT_DESC, R0                          ; 0865
       51       20   A5   20   A4  C3 0012A          SUBL3   32(R4), 32(R5), R1                       ; 0866
  20   A0            51        52  C7 00130          DIVL3   SCALE, R1, 32(R0)                        ; 0867
                        0106   C9  9F 00135          PUSHAB  P.AAN                                    ; 0872
                               52  DD 00139          PUSHL   SCALE                                    ;
                               02  DD 0013B          PUSHL   #2                                       ;
                    00028713   8F  DD 0013D          PUSHL   #165651                                  ;
                     68        04  FB 00143          CALLS   #4, LIB$SIGNAL                           ;
                                  04 00146          RET                                              ; 0873

; Routine Size:  327 bytes,    Routine Base:  DBG$CODE + 03B5
```

```
;  752     0874  1 GLOBAL ROUTINE DBG$DIV_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;  753     0875  1 !
;  754     0876  1 ! FUNCTION
;  755     0877  1 !
;  756     0878  1 !       This routine is called to perform the divide operation
;  757     0879  1 !       on a scaled binary variable.
;  758     0880  1 !
;  759     0881  1 ! INPUTS
;  760     0882  1 !
;  761     0883  1 !       ARG_DESC1       - points to the value descriptor representing the
;  762     0884  1 !                         left argument of the operation.
;  763     0885  1 !       ARG_DESC2       - points to the value descriptor representing the
;  764     0886  1 !                         right argument of the operation.
;  765     0887  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  766     0888  1 !                         of the operation.
;  767     0889  1 !
;  768     0890  1 ! OUTPUTS
;  769     0891  1 !
;  770     0892  1 !       The result value descriptor is filled in.
;  771     0893  1 !       No value is returned.
;  772     0894  1 !
;  773     0895  2     BEGIN
;  774     0896  2
;  775     0897  2     MAP
;  776     0898  2         ARG_DESC1       : REF DBG$VALDESC,
;  777     0899  2         ARG_DESC2       : REF DBG$VALDESC,
;  778     0900  2         RESULT_DESC     : REF DBG$VALDESC;
;  779     0901  2
;  780     0902  2     LOCAL
;  781     0903  2         RESULT_VALUE,
;  782     0904  2         SCALE,
;  783     0905  2         SIGN,
;  784     0906  2         TEMP_RESULT     : VECTOR [4,LONG],
;  785     0907  2         TEMP_VAL1       : VECTOR [4,LONG],
;  786     0908  2         TEMP_VAL2       : VECTOR [4,LONG],
;  787     0909  2         TEMP_DESC       : DBG$STG_DESC,
;  788     0910  2         VAL_DESC1       : DBG$STG_DESC,
;  789     0911  2         VAL_DESC2       : DBG$STG_DESC,
;  790     0912  2         VALUE1,
;  791     0913  2         VALUE2;
;  792     0914  2
;  793     0915  2     ! Set up working variables.  This way we don't mess up anything important.
;  794     0916  2     !
;  795     0917  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;  796     0918  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;  797     0919  2
;  798     0920  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;  799     0921  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;  800     0922  2
;  801     0923  2     IF .VALUE2 EQL 0
;  802     0924  2     THEN
;  803     0925  2         SIGNAL(DBG$_DIVBYZERO);
;  804     0926  2
;  805     0927  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;  806     0928  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;  807     0929  2
;  808     0930  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
```

```
;  809          0931  2      DBG$NORMALIZE_FIXED(VAL_DESC2);
;  810          0932  2
;  811          0933  2      MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;  812          0934  2
;  813          0935  2      DBG$CVT_CVTLH_R1(VALUE1, TEMP_VAL1);
;  814          0936  2      DBG$CVT_CVTLH_R1(VALUE2, TEMP_VAL2);
;  815          0937  2
;  816          0938  2      ! Do the divide.
;  817          0939  2      !
;  818          0940  2      DIVH(TEMP_VAL2, TEMP_VAL1, TEMP_RESULT);
;  819          0941  2
;  820          0942  2      ! Now convert the H_float to Fixed Binary.
;  821          0943  2      ! This is not pretty, but it's better that trying to set up to call
;  822          0944  2      ! CVT_DX_DX to do the same thing.  This will be a lot faster.
;  823          0945  2      !
;  824          0946  2      RESULT_VALUE = 0;
;  825          0947  2      SIGN = .TEMP_RESULT<15, 1, 0>;
;  826          0948  2      TEMP_RESULT<15, 1, 0> = 0;
;  827          0949  2      SCALE = .TEMP_RESULT<0, 16, 0> - 16384;
;  828          0950  2      SCALE = .SCALE - 31;
;  829          0951  2      RESULT_VALUE<30, 1, 0> = 1;
;  830          0952  2      RESULT_VALUE<14, 16, 0> = .TEMP_RESULT<16, 16, 0>;
;  831          0953  2      RESULT_VALUE<0, 14, 0> = .(TEMP_RESULT+4)<18, 14, 0>;
;  832          0954  2      IF .SIGN THEN RESULT_VALUE = 0 - .RESULT_VALUE;
;  833          0955  2
;  834          0956  2      ! Normalize the result.  Makes the output look neater.
;  835          0957  2      ! (We don't need all the fields filled in.)
;  836          0958  2      !
;  837          0959  2      TEMP_DESC[DSC$W_LENGTH] = 4;
;  838          0960  2      TEMP_DESC[DSC$B_SCALE] = .SCALE;
;  839          0961  2      TEMP_DESC[DSC$A_POINTER] = RESULT_VALUE;
;  840          0962  2      DBG$NORMALIZE_FIXED(TEMP_DESC);
;  841          0963  2
;  842          0964  2      .RESULT_DESC[DBG$L_VALUE_POINTER] = .RESULT_VALUE;
;  843          0965  2      RESULT_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  844          0966  2      RESULT_DESC[DBG$B_VALUE_SCALE] = .TEMP_DESC[DSC$B_SCALE];
;  845          0967  2
;  846          0968  1      END;


                                        0FFC 00000          .ENTRY  DBG$DIV_FIXED_FIXED, Save R2,R3,R4,R5,R6,-      ; 0874
                                                                    R7,R8,R9,R10,R11                                    ;
                          59     0000V  CF   9E 00002          MOVAB   DBG$NORMALIZE_FIXED, R9                         ;
                          58 00000000G  00   9E 00007          MOVAB   DBG$CVT_CVTLH_R1, R8                            ;
                          5E       A0   AE   9E 0000E          MOVAB   -96(SP), SP                                     ;
                          57       04   AC   D0 00012          MOVL    ARG_DESC1, R7                                   ; 0917
                18   AE        14   A7        0C   28 00016    MOVC3   #12, 20(R7), VAL_DESC1                          ;
                          56       08   AC   D0 0001C          MOVL    ARG_DESC2, R6                                   ; 0918
                0C   AE        14   A6        0C   28 00020    MOVC3   #12, 20(R6), VAL_DESC2                          ;
                          6E       18   B7   D0 00026          MOVL    @24(R7), VALUE1                                 ; 0920
                       04   AE   18   B6   D0 0002A          MOVL    @24(R6), VALUE2                                 ; 0921
                                        0D   12 0002F          BNEQ    1$                                              ; 0923
                          00028240   8F   DD 00031          PUSHL   #164416                                         ; 0925
                00000000G  00        01   FB 00037          CALLS   #1, LIB$SIGNAL                                  ;
```

```
                                1C  AE          6E  9E 0003E 1$:   MOVAB   VALUE1, VAL_DESC1+4                    ; 0927
                                10  AE       04 AE  9E 00042      MOVAB   VALUE2, VAL_DESC2+4                    ; 0928
                                    18 AE    9F 00047            PUSHAB  VAL_DESC1                              ; 0930
                                69          01 FB 0004A          CALLS   #1, DBG$NORMALIZE_FIXED                ;
                                    0C AE    9F 0004D            PUSHAB  VAL_DESC2                              ; 0931
                                69          01 FB 00050          CALLS   #1, DBG$NORMALIZE_FIXED                ;
                                    0C AE    9F 00053            PUSHAB  VAL_DESC2                              ; 0933
                                    1C AE    9F 00056            PUSHAB  VAL_DESC1                              ;
                          0000V CF          02 FB 00059          CALLS   #2, MATCH_FIXED_BINARYS                ;
                                51       40 AE  9E 0005E        MOVAB   TEMP_VAL1, R1                          ; 0935
                                50          6E  9E 00062        MOVAB   VALUE1, R0                             ;
                                            68  16 00065        JSB     DBG$CVT_CVTLH_R1                       ;
                                51       30 AE  9E 00067        MOVAB   TEMP_VAL2, R1                          ; 0936
                                50       04 AE  9E 0006B        MOVAB   VALUE2, R0                             ;
                                            68  16 0006F        JSB     DBG$CVT_CVTLH_R1                       ;
                50  AE      40 AE    30 AE67FD 00071            DIVH3   TEMP_VAL2, TEMP_VAL1, TEMP_RESULT      ; 0940
                                        08 AE  D4 00079         CLRL    RESULT_VALUE                           ; 0946
          51    51  AE                01  07  EF 0007C         EXTZV   #7, #1, TEMP_RESULT+1, SIGN            ; 0947
                                51 AE   80  8F  8A 00082        BICB2   #128, TEMP_RESULT+1                    ; 0948
                                50      50  AE  3C 00087        MOVZWL  TEMP_RESULT, SCALE                     ; 0949
                                50    BFE1  C0  9E 0008B        MOVAB   -164T5(R0), SCALE                      ; 0950
                                0B AE   40  8F  88 00090        BISB2   #64, RESULT_VALUE+3                    ; 0951
    09 AE           10              06  52  AE  F0 00095        INSV    TEMP_RESULT+2, #6, #16, RESULT_VALUE+1 ; 0952
       52   56  AE                  0E      02  EF 0009C        EXTZV   #2, #14, TEMP_RESULT+6, R2             ; 0953
    08 AE           0E              00      52  F0 000A2        INSV    R2, #0, #14, RESULT_VALUE              ;
                                    05      51  E9 000A8        BLBC    SIGN, 2$                               ; 0954
                                08 AE   08  AE  CE 000AB        MNEGL   RESULT_VALUE, RESULT_VALUE             ;
                                24 AE       04  B0 000B0 2$:    MOVW    #4, TEMP_DESC                          ; 0959
                                2C AE       50  90 000B4        MOVB    SCALE, TEMP_DESC+8                     ; 0960
                                28 AE   08  AE  9E 000B8        MOVAB   RESULT_VALUE, TEMP_DESC+4              ; 0961
                                    24 AE   9F 000BD            PUSHAB  TEMP_DESC                              ; 0962
                                69          01 FB 000C0         CALLS   #1, DBG$NORMALIZE_FIXED                ;
                                50      0C  AC  D0 000C3        MOVL    RESULT_DESC, R0                        ; 0964
                                18 B0   08  AE  D0 000C7        MOVL    RESULT_VALUE, @24(R0)                  ;
                                16 A0       08  90 000CC        MOVB    #8, 22(R0)                             ; 0965
                                1C A0   2C  AE  90 000D0        MOVB    TEMP_DESC+8, 28(R0)                    ; 0966
                                            04  05 000D5        RET                                            ; 0968

; Routine Size:  214 bytes,    Routine Base:  DBG$CODE + 04FC
```

```
;  848          0969  1 GLOBAL ROUTINE DBG$ENUM_FIRST (TYPEID: REF RST$ENTRY) =
;  849          0970  1 !
;  850          0971  1 ! FUNCTION
;  851          0972  1 !       This routine finds the first enumeration element in a set of
;  852          0973  1 !       enumeration elements, and returns the integer value of that
;  853          0974  1 !       element. It thus does the same thing as the '"FIRST" operator
;  854          0975  1 !       in ADA. The code is taken from the TOKEN$K_TICK_FIRST case
;  855          0976  1 !       in the DBG$EVAL_ADA_TICK routine in this module.
;  856          0977  1 !
;  857          0978  1 !       This routine is needed when indexing ADA arrays that are
;  858          0979  1 !       subscripted by enumeration types.
;  859          0980  1 ! INPUTS
;  860          0981  1 !       TYPEID  - describes the enumeration type for which we
;  861          0982  1 !                 want the first element.
;  862          0983  1 ! OUTPUTS
;  863          0984  1 !       The value of the first element is returned.
;  864          0985  1 !
;  865          0986  2     BEGIN
;  866          0987  2     LOCAL
;  867          0988  2         ADR_KIND,
;  868          0989  2         COMPONENT_LIST: REF VECTOR[],
;  869          0990  2         DST_VALUE: VECTOR[3],
;  870          0991  2         DUMMY,
;  871          0992  2         HIGHBOUND,
;  872          0993  2         LOWBOUND;
;  873          0994  2
;  874          0995  2     ! If we do not have a typeid then just return 0.
;  875          0996  2     !
;  876          0997  2     IF .TYPEID EQL 0
;  877          0998  2     THEN
;  878          0999  2         RETURN 0;
;  879          1000  2
;  880          1001  2     ! If we have a subrange type, get the parent type.
;  881          1002  2     !
;  882          1003  2     WHILE .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
;  883          1004  2         DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUMMY);
;  884          1005  2
;  885          1006  2     ! If we do not have an enumeration type then just return 0.
;  886          1007  2     !
;  887          1008  2     IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  888          1009  2     THEN
;  889          1010  2         RETURN 0;
;  890          1011  2
;  891          1012  2     ! Obtain the component list.
;  892          1013  2     !
;  893          1014  2     COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  894          1015  2
;  895          1016  2     ! Return the value of the first element.
;  896          1017  2     !
;  897          1018  2     DBG$STA_SYMVALUE(.COMPONENT_LIST[0], DST_VALUE, ADR_KIND);
;  898          1019  2     IF .ADR_KIND NEQ DBG$K_VAL_LITERAL
;  899          1020  2     THEN
;  900          1021  2         $DBG_ERROR('DBGLANGOP\DBG$ENUM_FIRST');
;  901          1022  2     RETURN ..DST_VALUE[0];
;  902          1023  1     END;
```

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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 18  00222 P.AAO:  .ASCII  <24>\DBGLANGOP\<92>\DBG$ENUM_FIRST\          ;
                  54 53 52 49 46 5F 4D 55 4E 45  00231                                                          ;


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

                                                  0000 00000            .ENTRY  DBG$ENUM_FIRST, Save nothing      ; 0969
                                           5E        1C   C2 00002        SUBL2   #28, SP                         ;
                                              04     AC   D5 00005        TSTL    TYPEID                          ; 0997
                                                     5C   13 00008        BEQL    4$                              ;
                                           50        04   AC   D0 0000A 1$:     MOVL    TYPEID, R0              ; 1003
                                           09        18   A0   91 0000E        CMPB    24(R0), #9              ;
                                                          16   12 00012        BNEQ    2$                      ;
                                                          5E   DD 00014        PUSHL   SP                      ; 1004
                                                     08   AE   9F 00016        PUSHAB  HIGHBOUND               ;
                                                     10   AE   9F 00019        PUSHAB  LOWBOUND                ;
                                                     04   AC   9F 0001C        PUSHAB  TYPEID                  ;
                                                          50   DD 0001F        PUSHL   R0                      ;
                                  00000000G        00        05   FB 00021        CALLS   #5, DBG$STA_TYP_SUBRNG  ;
                                                          E0   11 00028        BRB     1$                      ;
                                           50        04   AC   D0 0002A 2$:     MOVL    TYPEID, R0              ; 1008
                                           04        18   A0   91 0002E        CMPB    24(R0), #4              ;
                                                          32   12 00032        BNEQ    4$                      ;
                                           50             2C   C0 00034        ADDL2   #44, COMPONENT_LIST     ; 1014
                                                     0C   AE   9F 00037        PUSHAB  ADR_KIND                ; 1018
                                                     14   AE   9F 0003A        PUSHAB  DST_VALUE               ;
                                                          60   DD 0003D        PUSHL   (COMPONENT_LIST)        ;
                                  00000000G        00        03   FB 0003F        CALLS   #3, DBG$STA_SYMVALUE    ;
                                           01        0C   AE   D1 00046        CMPL    ADR_KIND, #1            ; 1019
                                                          15   13 0004A        BEQL    3$                      ;
                                              00000000'   EF   9F 0004C        PUSHAB  P.AAO                   ; 1021
                                                          01   DD 00052        PUSHL   #1                      ;
                                              00028362    8F   DD 00054        PUSHL   #164706                 ;
                                  00000000G        00        03   FB 0005A        CALLS   #3, LIB$SIGNAL          ;
                                           50        10   BE   D0 00061 3$:     MOVL    @DST_VALUE, R0          ; 1022
                                                          04   00065        RET                             ;
                                                          50   D4 00066 4$:     CLRL    R0                      ; 1023
                                                          04   00068        RET                             ;

; Routine Size:  105 bytes,    Routine Base:  DBG$CODE + 05D2
```

```
;  904          1024  1 GLOBAL ROUTINE DBG$ENUM_POS (TYPEID: REF RST$ENTRY, VALUE) =
;  905          1025  1 !
;  906          1026  1 ! FUNCTION
;  907          1027  1 !       Given an enumeration value, this routine finds which position
;  908          1028  1 !       in the list of enumeration values corresponds to that value.
;  909          1029  1 !
;  910          1030  1 !       This is needed in subscripting ADA arrays, where we want
;  911          1031  1 !       to index by enumeration position and not by value. This
;  912          1032  1 !       routine does the same thing as the "'POS" function in ADA.
;  913          1033  1 ! INPUTS
;  914          1034  1 !       TYPEID  - describes the enumeration type for which we
;  915          1035  1 !                 are doing this operation.
;  916          1036  1 !       VALUE   - The enumeration value, expressed as an integer,
;  917          1037  1 !                 for which we want the position.
;  918          1038  1 ! OUTPUTS
;  919          1039  1 !       The position of the enumeration element is returned.
;  920          1040  1 !
;  921          1041  2     BEGIN
;  922          1042  2     LOCAL
;  923          1043  2         ADR_KIND,
;  924          1044  2         COMPONENT_LIST: REF VECTOR[],
;  925          1045  2         DST_VALUE: VECTOR[3],
;  926          1046  2         DUMMY,
;  927          1047  2         HIGHBOUND,
;  928          1048  2         INDEX,
;  929          1049  2         LOWBOUND;
;  930          1050  2
;  931          1051  2     ! If we do not have a typeid then just return the input.
;  932          1052  2     !
;  933          1053  2     IF .TYPEID EQL 0
;  934          1054  2     THEN
;  935          1055  2         RETURN .VALUE;
;  936          1056  2
;  937          1057  2     ! If we have a subrange type, get the parent type.
;  938          1058  2     !
;  939          1059  2     WHILE .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
;  940          1060  2         DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUMMY);
;  941          1061  2
;  942          1062  2     ! If we do not have an enumeration type then just return the input.
;  943          1063  2     !
;  944          1064  2     IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  945          1065  2     THEN
;  946          1066  2         RETURN .VALUE;
;  947          1067  2
;  948          1068  2     ! Obtain the component list.
;  949          1069  2     !
;  950          1070  2     COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  951          1071  2
;  952          1072  2     ! Loop throug  the component list looking for a value that matches.
;  953          1073  2     !
;  954          1074  2     INDEX = 0;
;  955          1075  2     INCR I FROM 0 TO .TYPEID[RST$L_TYPCOMPCNT]-1 DO
;  956          1076  3         BEGIN
;  957          1077  3         DBG$STA_SYMVALUE(.COMPONENT_LIST[.INDEX], DST_VALUE, ADR_KIND);
;  958          1078  3         IF .ADR_KIND NEQ DBG$K_VAL_LITERAL
;  959          1079  3         THEN
;  960          1080  3             $DBG_ERROR('DBGLANGOP\DBG$ENUM_FIRST');
```

```
;  961          1081  3          IF .VALUE EQL ..DST_VALUE[0]
;  962          1082  3          THEN
;  963          1083  3              RETURN .INDEX
;  964          1084  3          ELSE
;  965          1085  3              INDEX = .INDEX + 1;
;  966          1086  2          END;
;  967          1087  2
;  968          1088  2      ! If we get here we did not find a matching enumeration value.
;  969          1089  2      ! In this case, just return the input value.
;  970          1090  2      !
;  971          1091  2      RETURN .VALUE;
;  972          1092  1      END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 18  0023B P.AAP:  .ASCII  <24>\DBGLANGOP\<92>\DBG$ENUM_FIRST\          ;
                  54 53 52 49 46 5F 4D 55 4E 45  0024A                                                          ;


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

                                    003C 00000          .ENTRY  DBG$ENUM_POS, Save R2,R3,R4,R5                  ; 1024
                              5E     1C  C2 00002          SUBL2   #28, SP                                         ;
                        04    AC  D5 00005          TSTL    TYPEID                                          ; 1053
                              72  13 00008          BEQL    7$                                              ;
                        50    04  AC D0 0000A 1$:   MOVL    TYPEID, R0                                      ; 1059
                        09    18  A0 91 0000E       CMPB    24(R0), #9                                      ;
                              16  12 00012          BNEQ    2$                                              ;
                              5E  DD 00014          PUSHL   SP                                              ; 1060
                              08  AE 9F 00016       PUSHAB  HIGHBOUND                                       ;
                              10  AE 9F 00019       PUSHAB  LOWBOUND                                        ;
                              04  AC 9F 0001C       PUSHAB  TYPEID                                          ;
                              50  DD 0001F          PUSHL   R0                                              ;
00000000G                     00  FB 00021          CALLS   #5, DBG$STA_TYP_SUBRNG                          ;
                              E0  11 00028          BRB     1$                                              ;
                        54    04  AC D0 0002A 2$:   MOVL    TYPEID, R4                                      ; 1064
                        04    18  A4 91 0002E       CMPB    24(R4), #4                                      ;
                              48  12 00032          BNEQ    7$                                              ;
                        52    2C  A4 9E 00034       MOVAB   44(R4), COMPONENT_LIST                          ; 1070
                              53  D4 00038          CLRL    INDEX                                           ; 1074
                        55          01  CE 0003A    MNEGL   #1, I                                           ; 1081
                              38  11 0003D          BRB     6$                                              ;
                              0C  AE 9F 0003F 3$:   PUSHAB  ADR_KIND                                        ; 1077
                              14  AE 9F 00042       PUSHAB  DST_VALUE                                       ;
                                6243  DD 00045          PUSHL   (COMPONENT_LIST)[INDEX]                         ;
00000000G                     00  FB 00048          CALLS   #3, DBG$STA_SYMVALUE                            ;
                        01    0C  AE D1 0004F       CMPL    ADR_KIND, #1                                    ; 1078
                              15  13 00053          BEQL    4$                                              ;
                    00000000'     EF 9F 00055       PUSHAB  P.AAP                                           ; 1080
                              01  DD 0005B          PUSHL   #1                                              ;
                    00028362      8F DD 0005D       PUSHL   #164706                                         ;
00000000G                     00  FB 00063          CALLS   #3, LIB$SIGNAL                                  ;
          10    BE        08  AC D1 0006A 4$:   CMPL    VALUE, @DST_VALUE                               ; 1081
                              04  12 0006F          BNEQ    5$                                              ;
```

```
                        50              53  D0 00071            MOVL    INDEX, R0                           ; 1083
                                            04 00074            RET                                         ;
                                        53  D6 00075 5$:         INCL    INDEX                               ; 1085
                    C3  55      28      A4  F2 00077 6$:         AOBLSS  40(R4), I, 3$                       ; 1075
                        50      08      AC  D0 0007C 7$:         MOVL    VALUE, R0                           ; 1091
                                            04 00080            RET                                         ; 1092

; Routine Size:  129 bytes,    Routine Base:  DBG$CODE + 063B
```

```
;  974          1093  1 GLOBAL ROUTINE DBG$ENUM_SUCC (TYPEID: REF RST$ENTRY, VALUE) =
;  975          1094  1 !
;  976          1095  1 ! FUNCTION
;  977          1096  1 !       Given an enumeration value, this routine finds the enumeration
;  978          1097  1 !       value of the successor element. This corresponds to the '"SUCC"
;  979          1098  1 !       function in ADA.
;  980          1099  1 !
;  981          1100  1 !       This routine is needed when we do an aggregate examine, in ADA,
;  982          1101  1 !       of an array indexed by an enumeration type.
;  983          1102  1 ! INPUTS
;  984          1103  1 !       TYPEID  - describes the enumeration type for which we
;  985          1104  1 !                 are doing this operation.
;  986          1105  1 !       VALUE   - The enumeration value, expressed as an integer,
;  987          1106  1 !                 for which we want the successor.
;  988          1107  1 !
;  989          1108  1 ! OUTPUTS
;  990          1109  1 !       The value of the successor is returned.
;  991          1110  1 !
;  992          1111  2     BEGIN
;  993          1112  2     LOCAL
;  994          1113  2         ADR_KIND,
;  995          1114  2         COMPONENT_LIST: REF VECTOR[],
;  996          1115  2         DST_VALUE: VECTOR[3],
;  997          1116  2         DUMMY,
;  998          1117  2         FOUND_FLAG,
;  999          1118  2         HIGHBOUND,
; 1000          1119  2         INDEX,
; 1001          1120  2         LOWBOUND;
; 1002          1121  2
; 1003          1122  2     ! If we do not have a typeid then just return the value + 1.
; 1004          1123  2     !
; 1005          1124  2     IF .TYPEID EQL 0
; 1006          1125  2     THEN
; 1007          1126  2         RETURN .VALUE+1;
; 1008          1127  2
; 1009          1128  2     ! If we have a subrange type, get the parent type.
; 1010          1129  2     !
; 1011          1130  2     WHILE .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
; 1012          1131  2         DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUMMY);
; 1013          1132  2
; 1014          1133  2     ! If we do not have an enumeration type then just return the value + 1.
; 1015          1134  2     !
; 1016          1135  2     IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
; 1017          1136  2     THEN
; 1018          1137  2         RETURN .VALUE+1;
; 1019          1138  2
; 1020          1139  2     ! Obtain the component list.
; 1021          1140  2     !
; 1022          1141  2     COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
; 1023          1142  2
; 1024          1143  2     ! Loop through the component list looking for a value that matches.
; 1025          1144  2     !
; 1026          1145  2     INDEX = 0;
; 1027          1146  2     FOUND_FLAG = FALSE;
; 1028          1147  2     INCR I FROM 0 TO .TYPEID[RST$L_TYPCOMPCNT]-1 DO
; 1029          1148  3         BEGIN
; 1030          1149  3         DBG$STA_SYMVALUE(.COMPONENT_LIST[.INDEX], DST_VALUE, ADR_KIND);
```

```
;  1031          1150  3          IF .ADR_KIND NEQ DBG$K_VAL_LITERAL
;  1032          1151  3          THEN
;  1033          1152  3              $DBG_ERROR('DBGLANGOP\DBG$ENUM_FIRST');
;  1034          1153  3
;  1035          1154  3          ! If we found a value that matched last time around,
;  1036          1155  3          ! then return the value we computed this time around.
;  1037          1156  3          !
;  1038          1157  3          IF .FOUND_FLAG
;  1039          1158  3          THEN
;  1040          1159  3              RETURN ..DST_VALUE[0];
;  1041          1160  3
;  1042          1161  3          ! If we match this time around, set the flag saying we want to
;  1043          1162  3          ! stop next time around.
;  1044          1163  3          !
;  1045          1164  3          IF .VALUE EQL ..DST_VALUE[0]
;  1046          1165  3          THEN
;  1047          1166  3              FOUND_FLAG = TRUE;
;  1048          1167  3          INDEX = .INDEX + 1;
;  1049          1168  2          END;
;  1050          1169  2
;  1051          1170  2      ! If we get here we did not find a matching enumeration value.
;  1052          1171  2      ! In this case, just return the input value + 1.
;  1053          1172  2      !
;  1054          1173  2      RETURN .VALUE + 1;
;  1055          1174  1      END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 18  00254 P.AAQ:  .ASCII  <24>\DBGLANGOP\<92>\DBG$ENUM_FIRST\          ;
                54 53 52 49 46 5F 4D 55 4E 45  00263                                                          ;


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

                                                 007C 00000     .ENTRY  DBG$ENUM_SUCC, Save R2,R3,R4,R5,R6    ; 1093
                                          5E      1C   C2 00002 SUBL2   #28, SP                               ;
                                      04  AC   D5 00005 TSTL    TYPEID                                      ; 1124
                                          7B   13 00008 BEQL    8$                                          ;
                                      50  04   AC   D0 0000A 1$:  MOVL    TYPEID, R0                          ; 1130
                                      09  18   A0   91 0000E CMPB    24(R0), #9                              ;
                                          16   12 00012 BNEQ    2$                                          ;
                                          5E   DD 00014 PUSHL   SP                                          ; 1131
                                      08  AE   9F 00016 PUSHAB  HIGHBOUND                                    ;
                                      10  AE   9F 00019 PUSHAB  LOWBOUND                                     ;
                                      04  AC   9F 0001C PUSHAB  TYPEID                                      ;
                                          50   DD 0001F PUSHL   R0                                          ;
                          00000000G   00       05   FB 00021 CALLS   #5, DBG$STA_TYP_SUBRNG                 ;
                                          E0   11 00028 BRB     1$                                          ;
                                      53  04   AC   D0 0002A 2$:  MOVL    TYPEID, R3                          ; 1135
                                      04  18   A3   91 0002E CMPB    24(R3), #4                              ;
                                          51   12 00032 BNEQ    8$                                          ;
                                      52  2C   A3   9E 00034 MOVAB   44(R3), COMPONENT_LIST                  ; 1141
                                          54   D4 00038 CLRL    INDEX                                       ; 1145
                                          56   D4 0003A CLRL    FOUND_FLAG                                  ; 1146
```

```
                         55              01  CE 0003C          MNEGL   #1, I                                   ; 1164
                                         3F  11 0003F          BRB     7$                                      ;
                                   0C    AE  9F 00041 3$:      PUSHAB  ADR_KIND                                ; 1149
                                   14    AE  9F 00044          PUSHAB  DST_VALUE                               ;
                                       6244  DD 00047          PUSHL   (COMPONENT_LIST)[INDEX]                 ;
                00000000G        00        03  FB 0004A          CALLS   #3, DBG$STA_SYMVALUE                    ;
                                 01    0C    AE  D1 00051          CMPL    ADR_KIND, #1                        ; 1150
                                         15  13 00055          BEQL    4$                                      ;
                                   00000000' EF  9F 00057          PUSHAB  P.AAQ                               ; 1152
                                         01  DD 0005D          PUSHL   #1                                      ;
                                   00028362  8F  DD 0005F          PUSHL   #164706                             ;
                00000000G        00        03  FB 00065          CALLS   #3, LIB$SIGNAL                          ;
                                 05        56  E9 0006C 4$:      BLBC    FOUND_FLAG, 5$                          ; 1157
                                 50    10  BE  D0 0006F          MOVL    @DST_VALUE, R0                          ; 1159
                                         04  00073              RET                                             ;
                           10    BE    08  AC  D1 00074 5$:      CMPL    VALUE, @DST_VALUE                       ; 1164
                                         03  12 00079          BNEQ    6$                                      ;
                                 56        01  D0 0007B          MOVL    #1, FOUND_FLAG                          ; 1166
                                         54  D6 0007E 6$:       INCL    INDEX                                   ; 1167
                      BC                 55  28  A3  F2 00080 7$:      AOBLSS  40(R3), I, 3$                   ; 1147
                      50           08    AC      01  C1 00085 8$:      ADDL3   #1, VALUE, R0                   ; 1173
                                         04  0008A              RET                                             ; 1174

; Routine Size:  139 bytes,    Routine Base:  DBG$CODE + 06BC
```

```
;  1057          1175  1 GLOBAL ROUTINE DBG$ENUM_VAL (TYPEID: REF RST$ENTRY, POSITION) =
;  1058          1176  1 !
;  1059          1177  1 ! FUNCTION
;  1060          1178  1 !       Given a position in a list of enumeration values, this routine
;  1061          1179  1 !       returns the enumeration value in that position. This corresponds
;  1062          1180  1 !       to the ""VAL" function in ADA.
;  1063          1181  1 !
;  1064          1182  1 !       This routine is used for bounds-checking of ADA arrays that
;  1065          1183  1 !       are indexed by enumeration types.
;  1066          1184  1 ! INPUTS
;  1067          1185  1 !       TYPEID  - describes the enumeration type for which we
;  1068          1186  1 !                 are doing this operation.
;  1069          1187  1 !       POSITION- The position number of the enumeration value.
;  1070          1188  1 !
;  1071          1189  1 ! OUTPUTS
;  1072          1190  1 !       The enumeration value at that position is returned.
;  1073          1191  2     BEGIN
;  1074          1192  2     LOCAL
;  1075          1193  2         ADR_KIND,
;  1076          1194  2         COMPONENT_LIST: REF VECTOR[],
;  1077          1195  2         DST_VALUE: VECTOR[3],
;  1078          1196  2         DUMMY,
;  1079          1197  2         HIGHBOUND,
;  1080          1198  2         LOWBOUND;
;  1081          1199  2
;  1082          1200  2     ! If we do not have a typeid then just return the input.
;  1083          1201  2     !
;  1084          1202  2     IF .TYPEID EQL 0
;  1085          1203  2     THEN
;  1086          1204  2         RETURN .POSITION;
;  1087          1205  2
;  1088          1206  2     ! If we have a subrange type, get the parent type.
;  1089          1207  2     !
;  1090          1208  2     WHILE .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
;  1091          1209  2         DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUMMY);
;  1092          1210  2
;  1093          1211  2     ! If we do not have an enumeration type then just return the input.
;  1094          1212  2     !
;  1095          1213  2     IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1096          1214  2     THEN
;  1097          1215  2         RETURN .POSITION;
;  1098          1216  2
;  1099          1217  2     ! Obtain the component list, look up the Nth element, and return
;  1100          1218  2     ! its value.
;  1101          1219  2     !
;  1102          1220  2     COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  1103          1221  2     DBG$STA_SYMVALUE(.COMPONENT_LIST[.POSITION], DST_VALUE, ADR_KIND);
;  1104          1222  2     IF .ADR_KIND NEQ DBG$K_VAL_LITERAL
;  1105          1223  2     THEN
;  1106          1224  2         $DBG_ERROR('DBGLANGOP\DBG$ENUM_FIRST');
;  1107          1225  2     RETURN ..DST_VALUE[0];
;  1108          1226  1     END;


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

```
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 18 00026D P.AAR:  .ASCII  <24>\DBGLANGOP\<92>\DBG$ENUM_FIRST\
                  54 53 52 49 46 5F 4D 55 4E 45 00027C

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

                                      0000 00000    .ENTRY  DBG$ENUM_VAL, Save nothing                  ; 1175
                                5E      1C   C2 00002    SUBL2   #28, SP                             ;
                             04   AC   D5 00005    TSTL    TYPEID                                      ; 1202
                                  2A   13 00008    BEQL    3$                                          ;
                             50   04   AC   D0 0000A 1$:  MOVL    TYPEID, R0                          ; 1208
                             09   18   A0   91 0000E    CMPB    24(R0), #9                          ;
                                  16   12 00012    BNEQ    2$                                          ;
                                  5E   DD 00014    PUSHL   SP                                          ; 1209
                             08   AE   9F 00016    PUSHAB  HIGHBOUND                                   ;
                             10   AE   9F 00019    PUSHAB  LOWBOUND                                    ;
                             04   AC   9F 0001C    PUSHAB  TYPEID                                      ;
                                  50   DD 0001F    PUSHL   R0                                          ;
                   00000000G 00       05   FB 00021    CALLS   #5, DBG$STA_TYP_SUBRNG                  ;
                                  E0   11 00028    BRB     1$                                          ;
                             50   04   AC   D0 0002A 2$:  MOVL    TYPEID, R0                          ; 1213
                             04   18   A0   91 0002E    CMPB    24(R0), #4                          ;
                                  05   13 00032    BEQL    4$                                          ;
                             50   08   AC   D0 00034 3$:  MOVL    POSITION, R0                        ; 1215
                                       04 00038    RET                                                 ;
                             50        2C   C0 00039 4$:  ADDL2   #44, COMPONENT_LIST                 ; 1220
                             0C   AE   9F 0003C    PUSHAB  ADR_KIND                                    ; 1221
                             14   AE   9F 0003F    PUSHAB  DST_VALUE                                   ;
                             51   08   AC   D0 00042    MOVL    POSITION, R1                           ;
                                  6041 DD 00046    PUSHL   (COMPONENT_LIST)[R1]                        ;
                   00000000G 00       03   FB 00049    CALLS   #3, DBG$STA_SYMVALUE                    ;
                             01   0C   AE   D1 00050    CMPL    ADR_KIND, #1                        ; 1222
                                  15   13 00054    BEQL    5$                                          ;
                        00000000'      EF   9F 00056    PUSHAB  P.AAR                               ; 1224
                                       01   DD 0005C    PUSHL   #1                                  ;
                        00028362       8F   DD 0005E    PUSHL   #164706                             ;
                   00000000G 00       03   FB 00064    CALLS   #3, LIB$SIGNAL                      ;
                             50   10   BE   D0 0006B 5$:  MOVL    @DST_VALUE, R0                      ; 1225
                                       04 0006F    RET                                                 ; 1226

; Routine Size:  112 bytes,    Routine Base:  DBG$CODE + 0747
```

```
;1110     1227  1 GLOBAL ROUTINE DBG$EQL_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;1111     1228  1 !
;1112     1229  1 ! FUNCTION
;1113     1230  1 !
;1114     1231  1 !       This routine is called to perform the equal to operation
;1115     1232  1 !       on a scaled binary variable.
;1116     1233  1 !
;1117     1234  1 ! INPUTS
;1118     1235  1 !
;1119     1236  1 !       ARG_DESC1       - points to the value descriptor representing the
;1120     1237  1 !                         left argument of the operation.
;1121     1238  1 !       ARG_DESC2       - points to the value descriptor representing the
;1122     1239  1 !                         right argument of the operation.
;1123     1240  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;1124     1241  1 !                         of the operation.
;1125     1242  1 !
;1126     1243  1 ! OUTPUTS
;1127     1244  1 !
;1128     1245  1 !       The result value descriptor is filled in.
;1129     1246  1 !       No value is returned.
;1130     1247  1 !
;1131     1248  2     BEGIN
;1132     1249  2
;1133     1250  2     MAP
;1134     1251  2         ARG_DESC1       : REF DBG$VALDESC,
;1135     1252  2         ARG_DESC2       : REF DBG$VALDESC,
;1136     1253  2         RESULT_DESC     : REF DBG$VALDESC;
;1137     1254  2
;1138     1255  2     LOCAL
;1139     1256  2         VAL_DESC1       : DBG$STG_DESC,
;1140     1257  2         VAL_DESC2       : DBG$STG_DESC,
;1141     1258  2         VALUE1,
;1142     1259  2         VALUE2;
;1143     1260  2
;1144     1261  2     ! Set up working variables.  This way we don't mess up anything important.
;1145     1262  2     !
;1146     1263  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;1147     1264  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;1148     1265  2
;1149     1266  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;1150     1267  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;1151     1268  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;1152     1269  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;1153     1270  2
;1154     1271  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;1155     1272  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;1156     1273  2
;1157     1274  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;1158     1275  2
;1159     1276  2     ! Do the Equal evaluation.
;1160     1277  2     !
;1161     1278  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 EQL .VALUE2;
;1162     1279  2
;1163     1280  1     END;
```

```
                                                00FC 00000          .ENTRY  DBG$EQL_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 1227
                                                     5E 00002 20 C2 SUBL2   #32, SP                                     ;
                                                     57 00005 04 AC MOVL    ARG_DESC1, R7                               ; 1263
                                                     0C 00009 28 MOVC3   #12, 20(R7), VAL_DESC1                         ;
                                                     56 0000F 08 AC MOVL    ARG_DESC2, R6                               ; 1264
                                                     0C 00013 28 MOVC3   #12, 20(R6), VAL_DESC2                         ;
                                                     6E 00019 18 B7 MOVL    @24(R7), VALUE1                             ; 1266
                                                     18 0001D 18 B6 MOVL    @24(R6), VALUE2                             ; 1267
                                                     6E 00022 9E MOVAB   VALUE1, VAL_DESC1+4                            ; 1268
                                                     04 AE 00026 9E MOVAB   VALUE2, VAL_DESC2+4                         ; 1269
                                                     14 AE 0002B 9F PUSHAB  VAL_DESC1                                   ; 1271
                                                     01 FB 0002E CALLS   #1, DBG$NORMALIZE_FIXED                        ;
                                                     08 AE 00033 9F PUSHAB  VAL_DESC2                                   ; 1272
                                                     01 FB 00036 CALLS   #1, DBG$NORMALIZE_FIXED                        ;
                                                     08 AE 0003B 9F PUSHAB  VAL_DESC2                                   ; 1274
                                                     18 AE 0003E 9F PUSHAB  VAL_DESC1                                   ;
                                                     02 FB 00041 CALLS   #2, MATCH_FIXED_BINARYS                        ;
                                                     51 00046 0C AC MOVL    RESULT_DESC, R1                             ; 1278
                                                     50 D4 0004A CLRL    R0                                             ;
                                                     6E D1 0004C CMPL    VALUE1, VALUE2                                 ;
                                                     02 12 00050 BNEQ    1$                                             ;
                                                     50 D6 00052 INCL    R0                                             ;
                                                     50 D0 00054 1$: MOVL    R0, @24(R1)                                ;
                                                        04 00058 RET                                                    ; 1280

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 07B7
```

```
;1165     1281  1 GLOBAL ROUTINE DBG$EVAL_ADA_TICK (TYPEID, OPERATOR) =
;1166     1282  1 !
;1167     1283  1 ! FUNCTION
;1168     1284  1 !       This routine does the evaluation of an Ada tick operator.
;1169     1285  1 !
;1170     1286  1 !       It receives the operand type, and  selects the actual Ada tick routine
;1171     1287  1 !       to do the operation based on the operator sub-code and the typeid.
;1172     1288  1 !       DBG$PERFORM_TICK_xxxxx routine is called to actually perform the
;1173     1289  1 !       specific operation.
;1174     1290  1 !
;1175     1291  1 !       This routine cases on the TOKEN$W_SUBCODE value of this operator to
;1176     1292  1 !       determine which actual routine will do the operation with the operand.
;1177     1293  1 !       If a routine is not found to handle the operation a error is signaled.
;1178     1294  1 !
;1179     1295  1 !       A Primary or a Value Descriptor may be returned as the routine value.
;1180     1296  1 !
;1181     1297  1 ! INPUTS
;1182     1298  1 !       TYPEID   - The typeid of the previous operand.
;1183     1299  1 !
;1184     1300  1 !       OPERATOR - A pointer to the Ada tick operator Token Entry for the
;1185     1301  1 !                       operator to be evaluated.
;1186     1302  1 !
;1187     1303  1 ! OUTPUTS
;1188     1304  1 !       A pointer to the Value Descriptor which results from the evaluation of
;1189     1305  1 !               the operator is returned as this routine's result.  Or,
;1190     1306  1 !       A pointer to the Primary Descritpor.
;1191     1307  1 !
;1192     1308  1
;1193     1309  2     BEGIN
;1194     1310  2
;1195     1311  2     MAP
;1196     1312  2         OPERATOR        : REF TOKEN$ENTRY,      ! Token Entry for operator to perform
;1197     1313  2         TYPEID          : REF RST$ENTRY;        ! RST entry for the operand
;1198     1314  2
;1199     1315  2     LOCAL
;1200     1316  2         ARG_LIST        : REF VECTOR [,LONG],   ! Counted vector of arguments
;1201     1317  2         ARG_VALUE,                              ! Value of the argument
;1202     1318  2         BOUNDVEC        : REF VECTOR [,LONG],   ! Pointer to bounds vector in
;1203     1319  2                                                 ! array descriptor.
;1204     1320  2         COMPONENT_LIST  : REF VECTOR [,LONG],   ! Vector of RST type components
;1205     1321  2         DSCADDR         : REF BLOCK [,BYTE],    ! Array Descriptor
;1206     1322  2         DST_VALUE       : VECTOR [3, LONG],     ! Value contained in DST
;1207     1323  2         DUM1,DUM2,DUM3,                         ! Dummy Variables for calls
;1208     1324  2         HIGHBOUND,                              ! Higher bound of the range
;1209     1325  2         INDEX,                                  ! Index for component list
;1210     1326  2         LOWBOUND,                               ! Lower bound of the range
;1211     1327  2         NDIMS,                                  ! Number of dimensions in array
;1212     1328  2         OPCODE,                                 ! Operator sub-code for current operator
;1213     1329  2         RESULT          : REF DBG$VALDESC,      ! Pointer to result value descriptor
;1214     1330  2         RSTPTR          : REF RST$ENTRY,        ! Temp pointer to an RST entry
;1215     1331  2         STRIDEVEC       : REF VECTOR [,LONG],   ! Pointer to stride vector in
;1216     1332  2                                                 ! array descriptor.
;1217     1333  2         TEMP_VAL_DESC   : REF DBG$VALDESC;      ! Temp value desc.
;1218     1334  2
;1219     1335  2
;1220     1336  2
;1221     1337  2     ! Pick up the operator sub-code and case on it to determine what operator
```

```
;  1222          1338  2     ! routine to call.
;  1223          1339  2     !
;  1224          1340  2     OPCODE = .OPERATOR[TOKEN$W_SUBCODE];
;  1225          1341  2
;  1226          1342  2     SELECTONE .OPCODE OF
;  1227          1343  2         SET
;  1228          1344  2
;  1229          1345  2         [TOKEN$K_TICK_CONSTRAINED]:
;  1230          1346  2             BEGIN
;  1231          1347  2             SIGNAL(DBG$_NOTIMPLAN, 1, UPLIT BYTE (%ASCIC '''CONSTRAINED'));
;  1232          1348  2             END;
;  1233          1349  2
;  1234          1350  2         [TOKEN$K_TICK_FIRST]:
;  1235          1351  2             BEGIN
;  1236          1352  2
;  1237          1353  2             ! Check to see that the incoming typeid is one of the possible
;  1238          1354  2             ! RSTs for this operator.
;  1239          1355  2             !
;  1240          1356  2             IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1241          1357  2             THEN
;  1242          1358  2                 DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1243          1359  2
;  1244          1360  2             IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
;  1245          1361  2                .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG AND
;  1246          1362  2                .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ARRAY
;  1247          1363  2             THEN
;  1248          1364  2                 SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1249          1365  2
;  1250          1366  2             ! If the input typeid is enumeration, then execute the code for
;  1251          1367  2             ! this, otherwise it will be an array type which requires different
;  1252          1368  2             ! processing.
;  1253          1369  2             !
;  1254          1370  3             IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_ENUM OR
;  1255          1371  3                .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1256          1372  3             THEN
;  1257          1373  4                 BEGIN
;  1258          1374  4
;  1259          1375  4                 ! There should not be an argument list.  If there is one,
;  1260          1376  4                 ! signal the error.
;  1261          1377  4                 !
;  1262          1378  4                 IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1263          1379  4                 THEN
;  1264          1380  4                     SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1265          1381  4
;  1266          1382  4                 ! Make a value descriptor for the result.
;  1267          1383  4                 !
;  1268          1384  4                 RESULT = DBG$MAKE_VALUE_DESC(.TYPEID, 0, RST$K_TYPE_ENUM);
;  1269          1385  4
;  1270          1386  4                 ! Check to see if the type is subrange of enumeration.  If so
;  1271          1387  4                 ! handle the 'FIRST for the subrange.
;  1272          1388  4                 !
;  1273          1389  4                 IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1274          1390  4                 THEN
;  1275          1391  5                     BEGIN
;  1276          1392  5                     DBG$STA_TYP_SUBRNG(.TYPEID, RSTPTR, LOWBOUND, HIGHBOUND, DUM1);
;  1277          1393  5                     IF .RSTPTR[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1278          1394  5                     THEN
```

```
;1279     1395  5                    SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;1280     1396  5
;1281     1397  5                ! Fix up typeid to point to the parent typeid not the subrange
;1282     1398  5                !
;1283     1399  5                RESULT[DBG$L_DHDR_TYPEID]  = .RSTPTR;
;1284     1400  5
;1285     1401  5                RESULT[DBG$L_VALUE_VALUE0] = ..LOWBOUND;
;1286     1402  5                END
;1287     1403  4            ELSE
;1288     1404  5                BEGIN
;1289     1405  5
;1290     1406  5                ! Go get the first element of this enumeration type and
;1291     1407  5                ! return it.
;1292     1408  5                !
;1293     1409  5                COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;1294     1410  5                RSTPTR = .COMPONENT_LIST[0];
;1295     1411  5                DBG$STA_SYMVALUE(.RSTPTR, DST_VALUE[0], DUM1);
;1296     1412  5                RESULT[DBG$L_VALUE_VALUE0] = ..DST_VALUE[0];
;1297     1413  4                END;
;1298     1414  4            END
;1299     1415  4
;1300     1416  3        ELSE
;1301     1417  4            BEGIN
;1302     1418  4
;1303     1419  4            ! If there's an argument list, go get the argument.  Otherwise,
;1304     1420  4            ! set the default value of the 'FIRST function to one.
;1305     1421  4            !
;1306     1422  4            IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;1307     1423  4            THEN
;1308     1424  5                BEGIN
;1309     1425  5                ! Get the argument
;1310     1426  5                !
;1311     1427  5                ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;1312     1428  5                TEMP_VAL_DESC = .ARG_LIST[1];
;1313     1429  5                ARG_VALUE = .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0];
;1314     1430  5                END
;1315     1431  4            ELSE
;1316     1432  4                ARG_VALUE = 1;
;1317     1433  4
;1318     1434  4            ! Get array information.
;1319     1435  4            !
;1320     1436  4            DBG$STA_TYP_ARRAY(.TYPEID, DSCADDR, DUM2, NDIMS, COMPONENT_LIST, DUM3);
;1321     1437  4
;1322     1438  4            ! Check to see that the input value in the proper range.
;1323     1439  4            ! If not signal novalue.
;1324     1440  4            !
;1325     1441  4            IF .ARG_VALUE LEQ 0 OR .ARG_VALUE GTR .NDIMS
;1326     1442  4            THEN
;1327     1443  4                SIGNAL(DBG$_INVARRDIM);
;1328     1444  4
;1329     1445  4            ARG_VALUE = .ARG_VALUE - 1;
;1330     1446  4
;1331     1447  4            ! Make a value descriptor for the first array subscript value
;1332     1448  4            !
;1333     1449  4            IF .COMPONENT_LIST[.ARG_VALUE] EQL 0
;1334     1450  4            THEN
;1335     1451  4                RSTPTR = DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_L, 32, FALSE)
```

```
;  1336     1452  4                ELSE
;  1337     1453  5                    BEGIN
;  1338     1454  5                    RSTPTR = .COMPONENT_LIST[.ARG_VALUE];
;  1339     1455  5                    IF .RSTPTR[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1340     1456  5                    THEN
;  1341     1457  5                        DBG$STA_TYP_SUBRNG(.RSTPTR, RSTPTR, DUM1, DUM2, DUM3);
;  1342     1458  4                    END;
;  1343     1459  4
;  1344     1460  4                RESULT = DBG$MAKE_VALUE_DESC(.RSTPTR, 0, .RSTPTR[RST$B_FCODE]);
;  1345     1461  4                STRIDEVEC = .DSCADDR + 20;
;  1346     1462  4                BOUNDVEC = .STRIDEVEC + 4 * .NDIMS;
;  1347     1463  4                RESULT[DBG$L_VALUE_VALUE0] = .BOUNDVEC[2 * .ARG_VALUE];
;  1348     1464  4
;  1349     1465  3                END;
;  1350     1466  2            END;
;  1351     1467  2
;  1352     1468  2        [TOKEN$K_TICK_LAST]:
;  1353     1469  2            BEGIN
;  1354     1470
;  1355     1471                ! Check to see that the incoming typeid is one of the possible
;  1356     1472                ! RSTs for this operator.
;  1357     1473                !
;  1358     1474                IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1359     1475                THEN
;  1360     1476                    DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1361     1477
;  1362     1478                IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
;  1363     1479                   .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG AND
;  1364     1480                   .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ARRAY
;  1365     1481                THEN
;  1366     1482                    SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1367     1483
;  1368     1484                ! If the input typeid is enumeration, then execute the code for
;  1369     1485                ! this, otherwise it will be an array type which requires different
;  1370     1486                ! processing.
;  1371     1487                !
;  1372     1488                IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_ENUM OR
;  1373     1489                   .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1374     1490                THEN
;  1375     1491  4                 BEGIN
;  1376     1492  4
;  1377     1493  4                 ! There should not be an argument list.  If there is one,
;  1378     1494  4                 ! signal the error.
;  1379     1495  4                 !
;  1380     1496  4                 IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1381     1497  4                 THEN
;  1382     1498  4                     SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1383     1499  4
;  1384     1500  4                 ! Make a value descriptor for the result.
;  1385     1501  4                 !
;  1386     1502  4                 RESULT = DBG$MAKE_VALUE_DESC(.TYPEID, 0, RST$K_TYPE_ENUM);
;  1387     1503  4
;  1388     1504  4                 ! Check to see if the type is subrange of enumeration.  If so
;  1389     1505  4                 ! handle the 'LAST for the subrange.
;  1390     1506  4                 !
;  1391     1507  4                 IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1392     1508  4                 THEN
```

```
;  1393          1509  5                BEGIN
;  1394          1510  5                DBG$STA_TYP_SUBRNG(.TYPEID, RSTPTR, LOWBOUND, HIGHBOUND, DUM1);
;  1395          1511  5                IF .RSTPTR[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1396          1512  5                THEN
;  1397          1513  5                    SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1398          1514  5
;  1399          1515  5                ! Fix up typeid to point to the parent typeid not the subrange
;  1400          1516  5                !
;  1401          1517  5                RESULT[DBG$L_DHDR_TYPEID]  = .RSTPTR;
;  1402          1518  5
;  1403          1519  5                RESULT[DBG$L_VALUE_VALUE0] = ..HIGHBOUND;
;  1404          1520  5                END
;  1405          1521  4            ELSE
;  1406          1522  5                BEGIN
;  1407          1523  5
;  1408          1524  5                ! Go get the first element of this enumeration type and
;  1409          1525  5                ! return it.
;  1410          1526  5                !
;  1411          1527  5                COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  1412          1528  5                RSTPTR = .COMPONENT_LIST[.TYPEID[RST$L_TYPCOMPCNT] - 1];
;  1413          1529  5                DBG$STA_SYMVALUE(.RSTPTR, DST_VALUE[0], DUM1);
;  1414          1530  5                RESULT[DBG$L_VALUE_VALUE0] = ..DST_VALUE[0];
;  1415          1531  4                END;
;  1416          1532  4            END
;  1417          1533  4
;  1418          1534  3        ELSE
;  1419          1535  4            BEGIN
;  1420          1536  4
;  1421          1537  4            ! If there's an argument list, go get the argument.  Otherwise,
;  1422          1538  4            ! set the default value of the 'LAST function to one.
;  1423          1539  4            !
;  1424          1540  4            IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1425          1541  4            THEN
;  1426          1542  5                BEGIN
;  1427          1543  5                ! Get the argument
;  1428          1544  5                !
;  1429          1545  5                ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1430          1546  5                TEMP_VAL_DESC = .ARG_LIST[1];
;  1431          1547  5                ARG_VALUE = .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0];
;  1432          1548  5                END
;  1433          1549  4            ELSE
;  1434          1550  4                ARG_VALUE = 1;
;  1435          1551  4
;  1436          1552  4            ! Get array information.
;  1437          1553  4            !
;  1438          1554  4            DBG$STA_TYP_ARRAY(.TYPEID, DSCADDR, DUM2, NDIMS, COMPONENT_LIST, DUM3);
;  1439          1555  4
;  1440          1556  4            ! Check to see that the input value in the proper range.
;  1441          1557  4            ! If not signal novalue.
;  1442          1558  4            !
;  1443          1559  4            IF .ARG_VALUE LEQ 0 OR .ARG_VALUE GTR .NDIMS
;  1444          1560  4            THEN
;  1445          1561  4                SIGNAL(DBG$_INVARRDIM);
;  1446          1562  4
;  1447          1563  4            ARG_VALUE = .ARG_VALUE - 1;
;  1448          1564  4
;  1449          1565  4            ! Make a value descriptor for the first array subscript value
```

```
;  1450          1566  4                 !
;  1451          1567  4                 IF .COMPONENT_LIST[.ARG_VALUE] EQL 0
;  1452          1568  4                 THEN
;  1453          1569  4                     RSTPTR = DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_L, 32, FALSE)
;  1454          1570  4                 ELSE
;  1455          1571  5                     BEGIN
;  1456          1572  5                     RSTPTR = .COMPONENT_LIST[.ARG_VALUE];
;  1457          1573  5                     IF .RSTPTR[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1458          1574  5                     THEN
;  1459          1575  5                         DBG$STA_TYP_SUBRNG(.RSTPTR, RSTPTR, DUM1, DUM2, DUM3);
;  1460          1576  4                     END;
;  1461          1577  4
;  1462          1578  4                 RESULT = DBG$MAKE_VALUE_DESC(.RSTPTR, 0, .RSTPTR[RST$B_FCODE]);
;  1463          1579  4                 STRIDEVEC = .DSCADDR + 20;
;  1464          1580  4                 BOUNDVEC = .STRIDEVEC + 4 * .NDIMS;
;  1465          1581  4                 RESULT[DBG$L_VALUE_VALUE0] = .BOUNDVEC[2 * .ARG_VALUE + 1];
;  1466          1582  4
;  1467          1583  3                 END;
;  1468          1584  2             END;
;  1469          1585  2
;  1470          1586  2         [TOKEN$K_TICK_LENGTH]:
;  1471          1587  3             BEGIN
;  1472          1588  3
;  1473          1589  3             ! Check to see that the incoming typeid is one of the possible
;  1474          1590  3             ! RSTs for this operator.
;  1475          1591  3             !
;  1476          1592  3             IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1477          1593  3             THEN
;  1478          1594  3                 DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1479          1595  3
;  1480          1596  3             IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ARRAY
;  1481          1597  3             THEN
;  1482          1598  3                 SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1483          1599  3
;  1484          1600  3             ! If there's an argument list.  Get the argument.
;  1485          1601  3             ! Otherwise let the default dimension value be 1.
;  1486          1602  3             !
;  1487          1603  3             IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1488          1604  3             THEN
;  1489          1605  4                 BEGIN
;  1490          1606  4
;  1491          1607  4                 ! Get the argument
;  1492          1608  4                 !
;  1493          1609  4                 ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1494          1610  4                 TEMP_VAL_DESC = .ARG_LIST[1];
;  1495          1611  4                 ARG_VALUE = .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0];
;  1496          1612  4                 END
;  1497          1613  3             ELSE
;  1498          1614  3                 ARG_VALUE = 1;
;  1499          1615  3
;  1500          1616  3             ! Get array information.
;  1501          1617  3             !
;  1502          1618  3             DBG$STA_TYP_ARRAY(.TYPEID, DSCADDR, DUM1, NDIMS, DUM2, DUM3);
;  1503          1619  3
;  1504          1620  3             ! Check to see that the input value in the proper range.
;  1505          1621  3             ! If not signal novalue.
;  1506          1622  3             !
```

```
;  1507          1623  3            IF .ARG_VALUE LEQ 0 OR .ARG_VALUE GTR .NDIMS
;  1508          1624  3            THEN
;  1509          1625  3                SIGNAL(DBG$_INVARRDIM);
;  1510          1626  3
;  1511          1627  3            ! Make a typeid for the result desc.
;  1512          1628  3            !
;  1513          1629  3            RSTPTR = DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_L, 32, FALSE);
;  1514          1630  3
;  1515          1631  3            ! Make a value descriptor for the result.
;  1516          1632  3            !
;  1517          1633  3            RESULT = DBG$MAKE_VALUE_DESC(.RSTPTR, 0, RST$K_TYPE_ATOMIC);
;  1518          1634  3
;  1519          1635  3            ! Set up pointers.
;  1520          1636  3            !
;  1521          1637  3            STRIDEVEC = .DSCADDR + 20;
;  1522          1638  3            BOUNDVEC = .STRIDEVEC + 4 * .NDIMS;
;  1523          1639  3
;  1524          1640  3            ! Calculate the length.
;  1525          1641  3            !
;  1526          1642  3            ARG_VALUE = .ARG_VALUE - 1;
;  1527          1643  3            RESULT[DBG$L_VALUE_VALUE0] =
;  1528          1644  3                .BOUNDVEC[2 * .ARG_VALUE + 1] - .BOUNDVEC[2 * .ARG_VALUE] + 1;
;  1529          1645  3
;  1530          1646  2            END;
;  1531          1647  2
;  1532          1648  2        [TOKEN$K_TICK_POS]:
;  1533          1649  2            BEGIN
;  1534          1650  2
;  1535          1651  3            ! Check to see that the incoming typeid is one of the possible
;  1536          1652  3            ! RSTs for this operator.
;  1537          1653  3            !
;  1538          1654  3            IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1539          1655  3            THEN
;  1540          1656  3                DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1541          1657  3
;  1542          1658  3            IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
;  1543          1659  3               .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
;  1544          1660  3            THEN
;  1545          1661  3                SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1546          1662  3
;  1547          1663  3            ! There should be an argument list of one.  If there is not,
;  1548          1664  3            ! signal the error now.
;  1549          1665  3            !
;  1550          1666  3            IF NOT .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1551          1667  3            THEN
;  1552          1668  3                SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1553          1669  3
;  1554          1670  3            ! Make a typeid for the result desc.
;  1555          1671  3            !
;  1556          1672  3            RSTPTR = DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_L, 32, FALSE);
;  1557          1673  3
;  1558          1674  3            ! Make a value descriptor for the result.
;  1559          1675  3            !
;  1560          1676  3            RESULT = DBG$MAKE_VALUE_DESC(.RSTPTR, 0, RST$K_TYPE_ATOMIC);
;  1561          1677  3
;  1562          1678  3            ! Get the argument
;  1563          1679  3            !
```

```
;  1564     1680  3          ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1565     1681  3
;  1566     1682  3          IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1567     1683  3          THEN
;  1568     1684  4              BEGIN
;  1569     1685  4
;  1570     1686  4              ! Make sure the parent type of the subrange is enumeration.
;  1571     1687  4              ! And make the parent type the new type.
;  1572     1688  4              !
;  1573     1689  4              DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUM3);
;  1574     1690  4              IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1575     1691  4              THEN
;  1576     1692  4                  SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1577     1693  3              END;
;  1578     1694  3
;  1579     1695  3          ! Find out which component in the typeid list is the one specified
;  1580     1696  3          ! by the symid in the argument list.  Then return the index as the
;  1581     1697  3          ! position of this element.  If the component was not found signal
;  1582     1698  3          ! an error.
;  1583     1699  3          !
;  1584     1700  3          TEMP_VAL_DESC = .ARG_LIST[1];
;  1585     1701  3          IF .TEMP_VAL_DESC[DBG$L_DHDR_TYPEID] NEQ .TYPEID
;  1586     1702  3          THEN
;  1587     1703  3              SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1588     1704  3
;  1589     1705  3          INDEX = 0;
;  1590     1706  3          COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  1591     1707  3          WHILE .INDEX LEQ .TYPEID[RST$L_TYPCOMPCNT]-1 DO
;  1592     1708  4              BEGIN
;  1593     1709  4              DBG$STA_SYMVALUE(.COMPONENT_LIST[.INDEX], DST_VALUE[0], DUM1);
;  1594     1710  4              IF .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0] EQL ..DST_VALUE[0]
;  1595     1711  4              THEN
;  1596     1712  5                  BEGIN
;  1597     1713  5                  RESULT[DBG$L_VALUE_VALUE0] = .INDEX;
;  1598     1714  5                  EXITLOOP;
;  1599     1715  5                  END
;  1600     1716  4              ELSE
;  1601     1717  4                  INDEX = .INDEX + 1;
;  1602     1718  3              END;
;  1603     1719  3
;  1604     1720  3              IF .INDEX GEQ .TYPEID[RST$L_TYPCOMPCNT]
;  1605     1721  3              THEN
;  1606     1722  3                  SIGNAL(DBG$_CMPNOTFND);
;  1607     1723  3
;  1608     1724  2              END;
;  1609     1725  2
;  1610     1726  2          [TOKEN$K_TICK_PRED]:
;  1611     1727  2              BEGIN
;  1612     1728  2
;  1613     1729  2              ! Check to see that the incoming typeid is one of the possible
;  1614     1730  2              ! RSTs for this operator.
;  1615     1731  2              !
;  1616     1732  2              IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1617     1733  2              THEN
;  1618     1734  2                  DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1619     1735  2
;  1620     1736  3              IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
```

```
;  1621          1737  3                .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
;  1622          1738  3            THEN
;  1623          1739  3                SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1624          1740  3
;  1625          1741  3            ! Make a value descriptor for the result.
;  1626          1742  3            !
;  1627          1743  3            RESULT = DBG$MAKE_VALUE_DESC(.TYPEID, 0, RST$K_TYPE_ENUM);
;  1628          1744  3
;  1629          1745  3            ! We expect an argument list.  If there isn't one, signal the error.
;  1630          1746  3            !
;  1631          1747  3            IF NOT .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1632          1748  3            THEN
;  1633          1749  3                SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1634          1750  3
;  1635          1751  3            ! Get the argument
;  1636          1752  3            !
;  1637          1753  3            ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1638          1754  3
;  1639          1755  3            IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1640          1756  3            THEN
;  1641          1757  4                BEGIN
;  1642          1758  4
;  1643          1759  4                ! Make sure the parent type of the subrange is enumeration
;  1644          1760  4                !
;  1645          1761  4                DBG$STA_TYP_SUBRNG(.TYPEID, RSTPTR, LOWBOUND, HIGHBOUND, DUM3);
;  1646          1762  4                IF .RSTPTR[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1647          1763  4                THEN
;  1648          1764  4                    SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1649          1765  3                END;
;  1650          1766  3
;  1651          1767  3            ! Perform the operation
;  1652          1768  3            !
;  1653          1769  3            DBG$PRED_ENUM(.ARG_LIST[1], .RESULT);
;  1654          1770  3
;  1655          1771  2            END;
;  1656          1772  2
;  1657          1773  2        [TOKEN$K_TICK_SIZE]:
;  1658          1774  3            BEGIN
;  1659          1775  3
;  1660          1776  3            ! There should be no argument list.  If there is one, signal the
;  1661          1777  3            ! error now.
;  1662          1778  3            !
;  1663          1779  3            IF .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1664          1780  3            THEN
;  1665          1781  3                SIGNAL(DBG$_SYNERREXPR, 1, UPLIT BYTE (%ASCIC '(') );
;  1666          1782  3
;  1667          1783  3            ! Make a typeid for the result desc.
;  1668          1784  3            !
;  1669          1785  3            RSTPTR = DBG$TYPEID_FOR_ATOMIC(DSC$K_DTYPE_L, 32, FALSE);
;  1670          1786  3
;  1671          1787  3            ! Make a value descriptor for the result.
;  1672          1788  3            !
;  1673          1789  3            RESULT = DBG$MAKE_VALUE_DESC(.RSTPTR, 0, RST$K_TYPE_ATOMIC);
;  1674          1790  3
;  1675          1791  3            ! Get the bit size of the input typeid. This may be a symbol or a
;  1676          1792  3            ! type.
;  1677          1793  3            !
```

```
;  1678     1794  3              DBG$STA_SYMSIZE(.TYPEID, RESULT[DBG$A_VALUE_ADDRESS]);
;  1679     1795  3
;  1680     1796  2              END;
;  1681     1797  2
;  1682     1798  2          [TOKEN$K_TICK_SUCC]:
;  1683     1799  3              BEGIN
;  1684     1800  3
;  1685     1801  3              ! Check to see that the incoming typeid is one of the possible
;  1686     1802  3              ! RSTs for this operator.
;  1687     1803  3              !
;  1688     1804  3              IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1689     1805  3              THEN
;  1690     1806  3                  DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1691     1807  3
;  1692     1808  3              IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
;  1693     1809  3                 .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
;  1694     1810  3              THEN
;  1695     1811  3                  SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1696     1812  3
;  1697     1813  3              ! Make a value descriptor for the result.
;  1698     1814  3              !
;  1699     1815  3              RESULT = DBG$MAKE_VALUE_DESC(.TYPEID, 0, RST$K_TYPE_ENUM);
;  1700     1816  3
;  1701     1817  3              ! We expect an argument list.  If there isn't one, signal the error.
;  1702     1818  3              !
;  1703     1819  3              IF NOT .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1704     1820  3              THEN
;  1705     1821  3                  SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1706     1822  3
;  1707     1823  3              ! Get the argument
;  1708     1824  3              !
;  1709     1825  3              ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1710     1826  3
;  1711     1827  3              IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1712     1828  3              THEN
;  1713     1829  4                  BEGIN
;  1714     1830  4
;  1715     1831  4                  ! Make sure the parent type of the subrange is enumeration
;  1716     1832  4                  !
;  1717     1833  4                  DBG$STA_TYP_SUBRNG(.TYPEID, RSTPTR, LOWBOUND, HIGHBOUND, DUM3);
;  1718     1834  4                  IF .RSTPTR[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1719     1835  4                  THEN
;  1720     1836  4                      SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1721     1837  3                  END;
;  1722     1838  3
;  1723     1839  3              ! Perform the operation
;  1724     1840  3              !
;  1725     1841  3              DBG$SUCC_ENUM(.ARG_LIST[1], .RESULT);
;  1726     1842  3
;  1727     1843  2              END;
;  1728     1844  2
;  1729     1845  2          [TOKEN$K_TICK_VAL]:
;  1730     1846  3              BEGIN
;  1731     1847  3
;  1732     1848  3              ! Check to see that the incoming typeid is one of the possible
;  1733     1849  3              ! RSTs for this operator.
;  1734     1850  3              !
```

```
;  1735          1851  3          IF .TYPEID[RST$B_KIND] EQL RST$K_DATA
;  1736          1852               THEN
;  1737          1853                   DBG$STA_SYMTYPE(.TYPEID, DUM1, TYPEID);
;  1738          1854
;  1739          1855               IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM AND
;  1740          1856                  .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
;  1741          1857               THEN
;  1742          1858                   SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1743          1859
;  1744          1860               ! We expect an argument list.  If there isn't one, signal the error.
;  1745          1861               !
;  1746          1862               IF NOT .OPERATOR[TOKEN$V_ARGUMENT_LIST]
;  1747          1863               THEN
;  1748          1864                   SIGNAL(DBG$_INVARGLIS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1749          1865
;  1750          1866               ! Get the argument
;  1751          1867               !
;  1752          1868               ARG_LIST = DBG$GET_BIF_ARGUMENTS(1, OPERATOR[TOKEN$B_OPLEN]);
;  1753          1869
;  1754          1870               IF .TYPEID[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  1755          1871               THEN
;  1756          1872   4               BEGIN
;  1757          1873   4
;  1758          1874   4               ! Make sure the parent type of the subrange is enumeration.
;  1759          1875   4               ! And make the parent type the new type.
;  1760          1876   4               !
;  1761          1877   4               DBG$STA_TYP_SUBRNG(.TYPEID, TYPEID, LOWBOUND, HIGHBOUND, DUM3);
;  1762          1878   4               IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;  1763          1879   4               THEN
;  1764          1880   4                   SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  1765          1881   3               END;
;  1766          1882   3
;  1767          1883   3           ! We must now convert the value of the argument to an integer.
;  1768          1884   3           ! To do this we must set up a new value descriptor.  We allocate
;  1769          1885   3           ! a skeleton descriptor and fill in some of the fields.
;  1770          1886   3           !
;  1771          1887   3           TEMP_VAL_DESC = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
;  1772          1888   3           TEMP_VAL_DESC[DBG$B_DHDR_KIND] = RST$K_DATA;
;  1773          1889   3           TEMP_VAL_DESC[DBG$L_DHDR_TYPEID] = 0;
;  1774          1890   3           TEMP_VAL_DESC[DBG$L_VALUE_POINTER] = TEMP_VAL_DESC[DBG$A_VALUE_ADDRESS];
;  1775          1891   3           TEMP_VAL_DESC[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
;  1776          1892   3           TEMP_VAL_DESC[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  1777          1893   3           TEMP_VAL_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  1778          1894   3           TEMP_VAL_DESC[DBG$W_VALUE_LENGTH] = 4;
;  1779          1895   3
;  1780          1896   3           ! Now call the conversion routine.  Put the result back into
;  1781          1897   3           ! the temporary descriptor.
;  1782          1898   3           !
;  1783          1899   3           TEMP_VAL_DESC = DBG$EVAL_LANG_OPERATOR(DBG$GL_CONVERT_TOKEN,
;  1784          1900   3                                                  .ARG_LIST[1], .TEMP_VAL_DESC);
;  1785          1901   3
;  1786          1902   3           ! Check to see that the input value is in the proper range.
;  1787          1903   3           ! If not signal novalue.
;  1788          1904   3           !
;  1789          1905   3           IF .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0] LSS 0 OR
;  1790          1906   3              .TEMP_VAL_DESC[DBG$L_VALUE_VALUE0] GEQ .TYPEID[RST$L_TYPCOMPCNT]
;  1791          1907   3           THEN
```

```
;  1792          1908  3                SIGNAL(DBG$_NOVALUE);
;  1793          1909  3
;  1794          1910  3            ! Get the symbol RST from the type RST
;  1795          1911  3            !
;  1796          1912  3            COMPONENT_LIST = TYPEID[RST$A_TYPCOMPLST];
;  1797          1913  3            RSTPTR = .COMPONENT_LIST[.TEMP_VAL_DESC[DBG$L_VALUE_VALUE0]];
;  1798          1914  3
;  1799          1915  3            ! Make a value descriptor for the result.
;  1800          1916  3            !
;  1801          1917  3            RESULT = DBG$MAKE_VALUE_DESC(.TYPEID, .RSTPTR, .TYPEID[RST$B_FCODE]);
;  1802          1918  3
;  1803          1919  3            ! Get the value of this RST element.
;  1804          1920  3            !
;  1805          1921  3            DBG$STA_SYMVALUE(.RSTPTR, DST_VALUE[0], DUM1);
;  1806          1922  3            RESULT[DBG$L_VALUE_VALUE0] = .DST_VALUE[0];
;  1807          1923  3
;  1808          1924  3            END;
;  1809          1925  2
;  1810          1926  2        [OTHERWISE]:
;  1811          1927  3            BEGIN
;  1812          1928  3            $DBG_ERROR('DBGLANGOP\DBG$EVAL_ADA_TICK, invalid token sub-code');
;  1813          1929  3            END;
;  1814          1930  2
;  1815          1931  2        TES;
;  1816          1932  2
;  1817          1933  2    ! Return a pointer to the result value descriptor.
;  1818          1934  2    !
;  1819          1935  2    RETURN .RESULT;
;  1820          1936  1    END;


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

                44  45  4E  49  41  52  54  53  4E  4F  43  27  0C  00286 P.AAS:  .ASCII  <12>\'CONSTRAINED\                    ;
                                                    28  01  00293 P.AAT:  .ASCII  <1>\(\                                ;
24  47  42  44  5C  50  4F  47  4E  41  4C  47  42  44  33  00295 P.AAU:  .ASCII  \3DBGLANGOP\<92>\DBG$EVAL_ADA_TICK, inva\  ;
20  2C  4B  43  49  54  5F  41  44  41  5F  4C  41  56  45  002A4                                                          ;
                                            61  76  6E  69  002B3                                                          ;
63  2D  62  75  73  20  6E  65  6B  6F  74  20  64  69  6C  002B7          .ASCII  \lid token sub-code\                       ;
                                            65  64  6F  002C6                                                          ;


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

                                                OFFC 00000          .ENTRY  DBG$EVAL_ADA_TICK, Save R2,R3,R4,R5,R6,R7,-  ; 1281
                                                                            R8,R9,R10,R11                                  ;
                                5B 00000000G    00   9E 00002          MOVAB   DBG$STA_SYMTYPE, R11                       ;
                                5A 00000000G    00   9E 00009          MOVAB   DBG$STA_TYP_SUBRNG, R10                    ;
                                59 00000000G    00   9E 00010          MOVAB   LIB$SIGNAL, R9                             ;
                                5E              30   C2 00017          SUBL2   #48, SP                                    ;
                                54        08    AC   D0 0001A          MOVL    OPERATOR, R4                               ; 1340
                                52        06    A4   3C 0001E          MOVZWL  6(R4), OPCODE                              ;
                                01              52   D1 00022          CMPL    OPCODE, #1                                 ; 1345
                                                11   12 00025          BNEQ    1$                                         ;
                                    00000000'   EF   9F 00027          PUSHAB  P.AAS                                      ; 1347
```

```
                         01   DD 0002D          PUSHL   #1
                 00028250    8F   DD 0002F      PUSHL   #164432
                       0754  31   00035         BRW     75$
02                       52  D1   00038 1$:     CMPL    OPCODE, #2                      ;1350
                         03  13   0003B         BEQL    2$
                       0159  31   0003D         BRW     16$
50                 04    AC  D0   00040 2$:     MOVL    TYPEID, R0                      ;1356
06                 14    A0  91   00044         CMPB    20(R0), #6
                         0B  12   00048         BNEQ    3$
                    04    AC  9F   0004A         PUSHAB  TYPEID                          ;1358
                    24    AE  9F   0004D         PUSHAB  DUM1
                         50  DD   00050         PUSHL   R0
6B                       03  FB   00052         CALLS   #3, DBG$STA_SYMTYPE
53                 04    AC  D0   00055 3$:     MOVL    TYPEID, R3                      ;1360
52                 18    A3  9A   00059         MOVZBL  24(R3), R2
04                       52  91   0005D         CMPB    R2, #4
                         18  13   00060         BEQL    4$
09                       52  91   00062         CMPB    R2, #9                          ;1361
                         13  13   00065         BEQL    4$
01                       52  91   00067         CMPB    R2, #1                          ;1362
                         0E  13   0006A         BEQL    4$
                    0C    A4  9F   0006C         PUSHAB  12(R4)                          ;1364
                         01  DD   0006F         PUSHL   #1
                 000289CA    8F   DD 00071      PUSHL   #166346
69                       03  FB   00077         CALLS   #3, LIB$SIGNAL
04                       52  91   0007A 4$:     CMPB    R2, #4                          ;1370
                         05  13   0007D         BEQL    5$
09                       52  91   0007F         CMPB    R2, #9                          ;1371
                         67  12   00082         BNEQ    9$
0E                 64    0B  E1   00084 5$:     BBC     #11, (R4), 6$                   ;1378
                    0C    A4  9F   00088         PUSHAB  12(R4)                          ;1380
                         01  DD   0008B         PUSHL   #1
                 00028838    8F   DD 0008D      PUSHL   #165944
                69               03   FB 00093  CALLS   #3, LIB$SIGNAL
                                 04   DD 00096 6$:     PUSHL   #4                      ;1384
                                 7E   D4 00098         CLRL    -(SP)
                                 53   DD 0009A         PUSHL   R3
0000V           CF               03   FB 0009C         CALLS   #3, DBG$MAKE_VALUE_DESC
                56               50   D0 000A1         MOVL    R0, RESULT
                09        18     A3   91 000A4         CMPB    24(R3), #9              ;1389
                                 35   12 000A8         BNEQ    8$
                          20     AE   9F 000AA         PUSHAB  DUM1                    ;1392
                          1C     AE   9F 000AD         PUSHAB  HIGHBOUND
                          24     AE   9F 000B0         PUSHAB  LOWBOUND
                          1C     AE   9F 000B3         PUSHAB  RSTPTR
                                 53   DD 000B6         PUSHL   R3
                6A               05   FB 000B8         CALLS   #5, DBG$STA_TYP_SUBRNG
                52        10     AE   D0 000BB         MOVL    RSTPTR, R2              ;1393
                04        18     A2   91 000BF         CMPB    24(R2), #4
                                 0E   13 000C3         BEQL    7$
                          0C     A4   9F 000C5         PUSHAB  12(R4)                  ;1395
                                 01   DD 000C8         PUSHL   #1
                       000289CA  8F   DD 000CA         PUSHL   #166346
                69               03   FB 000D0         CALLS   #3, LIB$SIGNAL
08              A6               52   D0 000D3 7$:     MOVL    R2, 8(RESULT)           ;1399
20              A6        1C     BE   D0 000D7         MOVL    @LOWBOUND, 32(RESULT)   ;1401
                               06B0   31 000DC         BRW     76$                     ;1389
```

```
                    6E        2C    A3  9E 000DF 8$:      MOVAB   44(R3), COMPONENT_LIST                ; 1409
                 10 AE        00    BE  D0 000E3          MOVL    @COMPONENT_LIST, RSTPTR               ; 1410
                                  067C  31 000E8          BRW     73$                                   ; 1411
              19                  64    E1 000EB 9$:      BBC     #11, (R4), 10$                        ; 1422
                                  0C    A4 9F 000EF       PUSHAB  12(R4)                                ; 1427
                                        01 DD 000F2       PUSHL   #1                                    ;
                 00000000G        00    02 FB 000F4       CALLS   #2, DBG$GET_BIF_ARGUMENTS             ;
                                  57    50 D0 000FB       MOVL    R0, ARG_LIST                          ;
                                  55    04 A7 D0 000FE    MOVL    4(ARG_LIST), TEMP_VAL_DESC            ; 1428
                                  52    20 A5 D0 00102    MOVL    32(TEMP_VAL_DESC), ARG_VALUE          ; 1429
                                        03 11 00106       BRB     11$                                   ; 1422
                                  52          01 D0 00108 10$:    MOVL    #1, ARG_VALUE                 ; 1432
                                        14 AE 9F 0010B 11$:   PUSHAB  DUM3                            ; 1436
                                        04 AE 9F 0010E        PUSHAB  COMPONENT_LIST                  ;
                                        10 AE 9F 00111        PUSHAB  NDIMS                           ;
                                        10 AE 9F 00114        PUSHAB  DUM2                            ;
                                        1C AE 9F 00117        PUSHAB  DSCADDR                         ;
                                           53 DD 0011A        PUSHL   R3                              ;
                 00000000G        00       06 FB 0011C       CALLS   #6, DBG$STA_TYP_ARRAY           ;
                                           52 D5 00123       TSTL    ARG_VALUE                       ; 1441
                                           06 15 00125       BLEQ    13$                             ;
                          08    AE         52 D1 00127       CMPL    ARG_VALUE, NDIMS                ;
                                           09 15 0012B       BLEQ    13$                             ;
                                00028850    8F DD 0012D 12$:  PUSHL   #165968                         ; 1443
                                  69        01 FB 00133       CALLS   #1, LIB$SIGNAL                  ;
                                           52 D7 00136 13$:   DECL    ARG_VALUE                       ; 1445
                                  50    00 BE42 D0 00138      MOVL    @COMPONENT_LIST[ARG_VALUE], R0  ; 1449
                                           12 12 0013D       BNEQ    14$                             ;
                                  7E          20 7D 0013F    MOVQ    #32, -(SP)                      ; 1451
                                              08 DD 00142    PUSHL   #8                               ;
                 00000000G        00          03 FB 00144    CALLS   #3, DBG$TYPEID_FOR_ATOMIC       ;
                          10    AE            50 D0 0014B    MOVL    R0, RSTPTR                      ;
                                              1B 11 0014F    BRB     15$                             ;
                          10    AE            50 D0 00151 14$:    MOVL    R0, RSTPTR                ; 1454
                                  09    18    A0 91 00155       CMPB    24(R0), #9                    ; 1455
                                              11 12 00159       BNEQ    15$                           ;
                                        14    AE 9F 0015B       PUSHAB  DUM3                          ; 1457
                                        08    AE 9F 0015E       PUSHAB  DUM2                          ;
                                        28    AE 9F 00161       PUSHAB  DUM1                          ;
                                        1C    AE 9F 00164       PUSHAB  RSTPTR                        ;
                                              50 DD 00167       PUSHL   R0                            ;
                                  6A          05 FB 00169       CALLS   #5, DBG$STA_TYP_SUBRNG        ;
                                  50    10    AE D0 0016C 15$:  MOVL    RSTPTR, R0                    ; 1460
                                  7E    18    A0 9A 00170       MOVZBL  24(R0), -(SP)                 ;
                                              7E D4 00174       CLRL    -(SP)                         ;
                                              50 DD 00176       PUSHL   R0                            ;
                          0000V   CF          03 FB 00178       CALLS   #3, DBG$MAKE_VALUE_DESC       ;
                                  56          50 D0 0017D       MOVL    R0, RESULT                    ;
                       58    0C   AE          14 C1 00180       ADDL3   #20, DSCADDR, STRIDEVEC       ; 1461
                                  50    08    AE D0 00185       MOVL    NDIMS, R0                     ; 1462
                                  53       6840 DE 00189       MOVAL   (STRIDEVEC)[R0], BOUNDVEC     ;
                       50         52          01 78 0018D       ASHL    #1, ARG_VALUE, R0             ; 1463
                                  20    A6       6340 D0 00191  MOVL    (BOUNDVEC)[R0], 32(RESULT)    ;
                                              05F6 31 00196     BRW     76$                           ; 1342
                                  03             52 D1 00199 16$:    CMPL    OPCODE, #3               ; 1468
                                                 03 13 0019C       BEQL    17$                           ;
                                              0162 31 0019E       BRW     31$                           ;
```

```
                50        04    AC  D0 001A1 17$:    MOVL    TYPEID, R0                          ; 1474
                06        14    A0  91 001A5        CMPB    20(R0), #6                          ;
                                08  12 001A9        BNEQ    18$                                 ;
                                04  AC 9F 001AB     PUSHAB  TYPEID                              ; 1476
                                24  AE 9F 001AE     PUSHAB  DUM1                                ;
                                50  DD 001B1        PUSHL   R0                                  ;
                6B                  03  FB 001B3     CALLS   #3, DBG$STA_SYMTYPE                 ;
                53        04    AC  D0 001B6 18$:    MOVL    TYPEID, R3                          ; 1478
                55        18    A3  9A 001BA        MOVZBL  24(R3), R5                          ;
                04                  55  91 001BE     CMPB    R5, #4                              ;
                                    18  13 001C1     BEQL    19$                                 ;
                09                  55  91 001C3     CMPB    R5, #9                              ; 1479
                                    13  13 001C6     BEQL    19$                                 ;
                01                  55  91 001C8     CMPB    R5, #1                              ; 1480
                                    0E  13 001CB     BEQL    19$                                 ;
                              0C    A4  9F 001CD     PUSHAB  12(R4)                              ; 1482
                                    01  DD 001D0     PUSHL   #1                                  ;
                        000289CA    8F  DD 001D2     PUSHL   #166346                             ;
                69                  03  FB 001D8     CALLS   #3, LIB$SIGNAL                      ;
                04                  55  91 001DB 19$:    CMPB    R5, #4                          ; 1488
                                    05  13 001DE     BEQL    20$                                 ;
                09                  55  91 001E0     CMPB    R5, #9                              ; 1489
                                    70  12 001E3     BNEQ    24$                                 ;
          0E    64                  0B  E1 001E5 20$:    BBC     #11, (R4), 21$                  ; 1496
                              0C    A4  9F 001E9     PUSHAB  12(R4)                              ; 1498
                                    01  DD 001EC     PUSHL   #1                                  ;
                        00028838    8F  DD 001EE     PUSHL   #165944                             ;
                69                  03  FB 001F4     CALLS   #3, LIB$SIGNAL                      ;
                                    04  DD 001F7 21$:    PUSHL   #4                              ; 1502
                                    7E  D4 001F9     CLRL    -(SP)                               ;
                                    53  DD 001FB     PUSHL   R3                                  ;
          0000V     CF              03  FB 001FD     CALLS   #3, DBG$MAKE_VALUE_DESC             ;
                56                  50  D0 00202     MOVL    R0, RESULT                          ;
                09        18    A3  91 00205        CMPB    24(R3), #9                          ; 1507
                                    35  12 00209     BNEQ    23$                                 ;
                                20  AE 9F 0020B     PUSHAB  DUM1                                ; 1510
                                1C  AE 9F 0020E     PUSHAB  HIGHBOUND                           ;
                                24  AE 9F 00211     PUSHAB  LOWBOUND                            ;
                                1C  AE 9F 00214     PUSHAB  RSTPTR                              ;
                                    53  DD 00217     PUSHL   R3                                  ;
                6A                  05  FB 00219     CALLS   #5, DBG$STA_TYP_SUBRNG              ;
                53        10    AE  D0 0021C        MOVL    RSTPTR, R3                          ; 1511
                04        18    A3  91 00220        CMPB    24(R3), #4                          ;
                                    0E  13 00224     BEQL    22$                                 ;
                              0C    A4  9F 00226     PUSHAB  12(R4)                              ; 1513
                                    01  DD 00229     PUSHL   #1                                  ;
                        000289CA    8F  DD 0022B     PUSHL   #166346                             ;
                69                  03  FB 00231     CALLS   #3, LIB$SIGNAL                      ;
          08    A6                  53  D0 00234 22$:    MOVL    R3, 8(RESULT)                   ; 1517
          20    A6        18        BE  D0 00238        MOVL    @HIGHBOUND, 32(RESULT)          ; 1519
                              054F  31  00 0023D        BRW     76$                             ; 1507
                6E        2C    A3  9E 00240 23$:    MOVAB   44(R3), COMPONENT_LIST              ; 1527
                50        28    A3  D0 00244        MOVL    40(R3), R0                          ; 1528
                50        00 BE40   DE 00248        MOVAL   @COMPONENT_LIST[R0], R0             ;
          10    AE        FC    A0  D0 0024D        MOVL    -4(R0), RSTPTR                      ;
                              0512  31 00252        BRW     73$                                 ; 1529
    19          64                  0B  E1 00255 24$:    BBC     #11, (R4), 25$                  ; 1540
```

```
                    0C   A4  9F 00259          PUSHAB  12(R4)                                  ; 1545
                         01  DD 0025C          PUSHL   #1
00000000G   00           02  FB 0025E          CALLS   #2, DBG$GET_BIF_ARGUMENTS
            57           50  D0 00265          MOVL    R0, ARG_LIST
            55       04  A7  D0 00268          MOVL    4(ARG_LIST), TEMP_VAL_DESC              ; 1546
            52       20  A5  D0 0026C          MOVL    32(TEMP_VAL_DESC), ARG_VALUE            ; 1547
                         03  11 00270          BRB     26$                                     ; 1540
            52           01  D0 00272 25$:     MOVL    #1, ARG_VALUE                           ; 1550
                    14   AE  9F 00275 26$:     PUSHAB  DUM3                                    ; 1554
                    04   AE  9F 00278          PUSHAB  COMPONENT_LIST
                    10   AE  9F 0027B          PUSHAB  NDIMS
                    10   AE  9F 0027E          PUSHAB  DUM2
                    1C   AE  9F 00281          PUSHAB  DSCADDR
                         53  DD 00284          PUSHL   R3
00000000G   00           06  FB 00286          CALLS   #6, DBG$STA_TYP_ARRAY
                         52  D5 0028D          TSTL    ARG_VALUE                               ; 1559
                         06  15 0028F          BLEQ    27$
        08  AE           52  D1 00291          CMPL    ARG_VALUE, NDIMS
                         09  15 00295          BLEQ    28$
                00028850 8F  DD 00297 27$:     PUSHL   #165968                                 ; 1561
            69           01  FB 0029D          CALLS   #1, LIB$SIGNAL
                         52  D7 002A0 28$:     DECL    ARG_VALUE                               ; 1563
            50       00 BE42 D0 002A2          MOVL    @COMPONENT_LIST[ARG_VALUE], R0          ; 1567
                         12  12 002A7          BNEQ    29$
            7E           20  7D 002A9          MOVQ    #32, -(SP)                              ; 1569
                         08  DD 002AC          PUSHL   #8
00000000G   00           03  FB 002AE          CALLS   #3, DBG$TYPEID_FOR_ATOMIC
        10  AE           50  D0 002B5          MOVL    R0, RSTPTR
                         1B  11 002B9          BRB     30$
        10  AE           50  D0 002BB 29$:     MOVL    R0, RSTPTR                              ; 1572
            09       18  A0  91 002BF          CMPB    24(R0), #9                              ; 1573
                         11  12 002C3          BNEQ    30$
                    14   AE  9F 002C5          PUSHAB  DUM3                                    ; 1575
                    08   AE  9F 002C8          PUSHAB  DUM2
                    28   AE  9F 002CB          PUSHAB  DUM1
                    1C   AE  9F 002CE          PUSHAB  RSTPTR
                         50  DD 002D1          PUSHL   R0
            6A           05  FB 002D3          CALLS   #5, DBG$STA_TYP_SUBRNG
            50       10  AE  D0 002D6 30$:     MOVL    RSTPTR, R0                              ; 1578
            7E       18  A0  9A 002DA          MOVZBL  24(R0), -(SP)
                         7E  D4 002DE          CLRL    -(SP)
                         50  DD 002E0          PUSHL   R0
        0000V    CF          03  FB 002E2          CALLS   #3, DBG$MAKE_VALUE_DESC
            56           50  D0 002E7          MOVL    R0, RESULT
58      0C   AE          14  C1 002EA          ADDL3   #20, DSCADDR, STRIDEVEC                 ; 1579
            50       08  AE  D0 002EF          MOVL    NDIMS, R0                               ; 1580
            53       6840 DE 002F3          MOVAL   (STRIDEVEC)[R0], BOUNDVEC
            52           02  C4 002F7          MULL2   #2, R2                                  ; 1581
        20  A6       04  A342 D0 002FA         MOVL    4(BOUNDVEC)[R2], 32(RESULT)
                         048C 31 00300          BRW     76$                                     ; 1342
            04           52  D1 00303 31$:     CMPL    OPCODE, #4                              ; 1586
                         03  13 00306          BEQL    32$
                         00BC 31 00308          BRW     39$
            50       04  AC  D0 0030B 32$:     MOVL    TYPEID, R0                              ; 1592
            06       14  A0  91 0030F          CMPB    20(R0), #6
                         0B  12 00313          BNEQ    33$
                    04   AC  9F 00315          PUSHAB  TYPEID                                  ; 1594
```

```
                                 24   AE  9F 00318          PUSHAB  DUM1                                    ;
                                      50  DD 0031B          PUSHL   R0                                      ;
                             6B       03  FB 0031D          CALLS   #3, DBG$STA_SYMTYPE                     ;
                             53   04  AC  D0 00320 33$:     MOVL    TYPEID, R3                              ; 1596
                             01   18  A3  91 00324          CMPB    24(R3), #1                              ;
                                  0E  13 00328              BEQL    34$                                     ;
                                 0C   A4  9F 0032A          PUSHAB  12(R4)                                  ; 1598
                                      01  DD 0032D          PUSHL   #1                                      ;
                          000289CA    8F  DD 0032F          PUSHL   #166346                                 ;
                             69       03  FB 00335          CALLS   #3, LIB$SIGNAL                          ;
                19           64       0B  E1 00338 34$:     BBC     #11, (R4), 35$                          ; 1603
                                 0C   A4  9F 0033C          PUSHAB  12(R4)                                  ; 1609
                                      01  DD 0033F          PUSHL   #1                                      ;
                 00000000G    00       02  FB 00341          CALLS   #2, DBG$GET_BIF_ARGUMENTS               ;
                             57       50  D0 00348          MOVL    R0, ARG_LIST                            ;
                             55   04  A7  D0 0034B          MOVL    4(ARG_LIST), TEMP_VAL_DESC              ; 1610
                             52   20  A5  D0 0034F          MOVL    32(TEMP_VAL_DESC), ARG_VALUE            ; 1611
                                  03  11 00353              BRB     36$                                     ; 1603
                             52       01  D0 00355 35$:     MOVL    #1, ARG_VALUE                           ; 1614
                                 14   AE  9F 00358 36$:     PUSHAB  DUM3                                    ; 1618
                                 08   AE  9F 0035B          PUSHAB  DUM2                                    ;
                                 10   AE  9F 0035E          PUSHAB  NDIMS                                   ;
                                 2C   AE  9F 00361          PUSHAB  DUM1                                    ;
                                 1C   AE  9F 00364          PUSHAB  DSCADDR                                 ;
                                      53  DD 00367          PUSHL   R3                                      ;
                 00000000G    00       06  FB 00369          CALLS   #6, DBG$STA_TYP_ARRAY                   ;
                                      52  D5 00370          TSTL    ARG_VALUE                               ; 1623
                                      06  15 00372          BLEQ    37$                                     ;
                         08   AE       52  D1 00374          CMPL    ARG_VALUE, NDIMS                        ;
                                      09  15 00378          BLEQ    38$                                     ;
                          00028850    8F  DD 0037A 37$:     PUSHL   #165968                                 ; 1625
                             69       01  FB 00380          CALLS   #1, LIB$SIGNAL                          ;
                             7E       20  7D 00383 38$:     MOVQ    #32, -(SP)                              ; 1629
                                      08  DD 00386          PUSHL   #8                                      ;
                 00000000G    00       03  FB 00388          CALLS   #3, DBG$TYPEID_FOR_ATOMIC               ;
                         10   AE       50  D0 0038F          MOVL    R0, RSTPTR                              ;
                                      02  DD 00393          PUSHL   #2                                      ; 1633
                                      7E  D4 00395          CLRL    -(SP)                                   ;
                                 18   AE  DD 00397          PUSHL   RSTPTR                                  ;
                          0000V       CF  03  FB 0039A          CALLS   #3, DBG$MAKE_VALUE_DESC                 ;
                                      56  D0 0039F          MOVL    R0, RESULT                              ;
                58        0C   AE       14  C1 003A2          ADDL3   #20, DSCADDR, STRIDEVEC                 ; 1637
                                 50   08  AE  D0 003A7          MOVL    NDIMS, R0                               ; 1638
                                 53       6840  DE 003AB          MOVAL   (STRIDEVEC)[R0], BOUNDVEC               ;
                                      52  D7 003AF          DECL    ARG_VALUE                               ; 1642
                50                    52  01  78 003B1          ASHL    #1, ARG_VALUE, R0                       ; 1644
                                      52  02  C4 003B5          MULL2   #2, R2                                  ;
                53        04  A340       6342  C3 003B8          SUBL3   (BOUNDVEC)[R2], 4(BOUNDVEC)[R0], R3     ;
                             20   A6  01   A3  9E 003BF          MOVAB   1(R3), 32(RESULT)                       ;
                                      03C8  31 003C4          BRW     76$                                     ; 1342
                                 05       52  D1 003C7 39$:     CMPL    OPCODE, #5                              ; 1648
                                      03  13 003CA          BEQL    40$                                     ;
                                      0104  31 003CC          BRW     50$                                     ;
                                 50   04  AC  D0 003CF 40$:     MOVL    TYPEID, R0                              ; 1654
                                 06   14  A0  91 003D3          CMPB    20(R0), #6                              ;
                                      0B  12 003D7          BNEQ    41$                                     ;
                                 04   AC  9F 003D9          PUSHAB  TYPEID                                  ; 1656
```

```
                        24   AE  9F 003DC          PUSHAB  DUM1                                    ;
                             50  DD 003DF          PUSHL   R0                                      ;
                        6B       03  FB 003E1          CALLS   #3, DBG$STA_SYMTYPE                 ;
                        52   04  AC  D0 003E4 41$:    MOVL    TYPEID, R2                          ; 1658
                        04   18  A2  91 003E8          CMPB    24(R2), #4                          ;
                             14  13 003EC          BEQL    42$                                     ;
                        09   18  A2  91 003EE          CMPB    24(R2), #9                          ; 1659
                             0E  13 003F2          BEQL    42$                                     ;
                        0C   A4  9F 003F4          PUSHAB  12(R4)                                  ; 1661
                             01  DD 003F7          PUSHL   #1                                      ;
                 000289CA    8F  DD 003F9          PUSHL   #166346                                 ;
                        69       03  FB 003FF          CALLS   #3, LIB$SIGNAL                      ;
                  0E    64       0B  E0 00402 42$:    BBS     #11, (R4), 43$                      ; 1666
                        0C   A4  9F 00406          PUSHAB  12(R4)                                  ; 1668
                             01  DD 00409          PUSHL   #1                                      ;
                 00028838    8F  DD 0040B          PUSHL   #165944                                 ;
                        69       03  FB 00411          CALLS   #3, LIB$SIGNAL                      ;
                        7E       20  7D 00414 43$:    MOVQ    #32, -(SP)                          ; 1672
                             08  DD 00417          PUSHL   #8                                      ;
          00000000G     00       03  FB 00419          CALLS   #3, DBG$TYPEID_FOR_ATOMIC           ;
                 10   AE       50  D0 00420          MOVL    R0, RSTPTR                          ;
                             02  DD 00424          PUSHL   #2                                      ; 1676
                             7E  D4 00426          CLRL    -(SP)                                   ;
                        18   AE  DD 00428          PUSHL   RSTPTR                                  ;
              0000V     CF       03  FB 0042B          CALLS   #3, DBG$MAKE_VALUE_DESC             ;
                        56       50  D0 00430          MOVL    R0, RESULT                          ;
                        0C   A4  9F 00433          PUSHAB  12(R4)                                  ; 1680
                             01  DD 00436          PUSHL   #1                                      ;
          00000000G     00       02  FB 00438          CALLS   #2, DBG$GET_BIF_ARGUMENTS           ;
                        57       50  D0 0043F          MOVL    R0, ARG_LIST                        ;
                        09   18  A2  91 00442          CMPB    24(R2), #9                          ; 1682
                             29  12 00446          BNEQ    44$                                     ;
                        14   AE  9F 00448          PUSHAB  DUM3                                    ; 1689
                        1C   AE  9F 0044B          PUSHAB  HIGHBOUND                               ;
                        24   AE  9F 0044E          PUSHAB  LOWBOUND                                ;
                        04   AC  9F 00451          PUSHAB  TYPEID                                  ;
                             52  DD 00454          PUSHL   R2                                      ;
                        6A       05  FB 00456          CALLS   #5, DBG$STA_TYP_SUBRNG              ;
                        50   04  AC  D0 00459          MOVL    TYPEID, R0                          ; 1690
                        04   18  A0  91 0045D          CMPB    24(R0), #4                          ;
                             0E  13 00461          BEQL    44$                                     ;
                        0C   A4  9F 00463          PUSHAB  12(R4)                                  ; 1692
                             01  DD 00466          PUSHL   #1                                      ;
                 000289CA    8F  DD 00468          PUSHL   #166346                                 ;
                        69       03  FB 0046E          CALLS   #3, LIB$SIGNAL                      ;
                        55   04  A7  D0 00471 44$:    MOVL    4(ARG_LIST), TEMP_VAL_DESC          ; 1700
                   04   AC   08  A5  D1 00475          CMPL    8(TEMP_VAL_DESC), TYPEID            ; 1701
                             0E  13 0047A          BEQL    45$                                     ;
                        0C   A4  9F 0047C          PUSHAB  12(R4)                                  ; 1703
                             01  DD 0047F          PUSHL   #1                                      ;
                 00028838    8F  DD 00481          PUSHL   #165944                                 ;
                        69       03  FB 00487          CALLS   #3, LIB$SIGNAL                      ;
                             53  D4 0048A 45$:    CLRL    INDEX                                   ; 1705
                  6E   04  AC       2C  C1 0048C          ADDL3   #44, TYPEID, COMPONENT_LIST         ; 1706
                        52   04  AC  D0 00491          MOVL    TYPEID, R2                          ; 1707
                  50    28  A2       01  C3 00495 46$:    SUBL3   #1, 40(R2), R0                      ;
                             50  53  D1 0049A          CMPL    INDEX, R0                           ;
```

```
                          22    14 0049D        BGTR    48$                                   ;
                    20    AE    9F 0049F        PUSHAB  DUM1                                  ; 1709
                    28    AE    9F 004A2        PUSHAB  DST_VALUE                             ;
                    08  BE43    DD 004A5        PUSHL   @COMPONENT_LIST[INDEX]                ;
     00000000G  00          03  FB 004A9        CALLS   #3, DBG$STA_SYMVALUE                  ;
            24  BE     20   A5  D1 004B0        CMPL    32(TEMP_VAL_DESC), @DST_VALUE         ; 1710
                            06  12 004B5        BNEQ    47$                                   ;
            20  A6          53  D0 004B7        MOVL    INDEX, 32(RESULT)                     ; 1713
                            04  11 004BB        BRB     48$                                   ; 1712
                            53  D6 004BD 47$:   INCL    INDEX                                 ; 1717
                            D4  11 004BF        BRB     46$                                   ; 1707
            28  A2          53  D1 004C1 48$:   CMPL    INDEX, 40(R2)                         ; 1720
                            09  19 004C5        BLSS    49$                                   ;
                  00028848  8F  DD 004C7        PUSHL   #165960                               ; 1722
                69          01  FB 004CD        CALLS   #1, LIB$SIGNAL                        ;
                          02BC  31 004D0 49$:   BRW     76$                                   ; 1342
                06          52  D1 004D3 50$:   CMPL    OPCODE, #6                            ; 1726
                            03  13 004D6        BEQL    51$                                   ;
                          009D  31 004D8        BRW     56$                                   ;
                50     04   AC  D0 004DB 51$:   MOVL    TYPEID, R0                            ; 1732
                06     14   A0  91 004DF        CMPB    20(R0), #6                            ;
                            0B  12 004E3        BNEQ    52$                                   ;
                       04   AC  9F 004E5        PUSHAB  TYPEID                                ; 1734
                       24   AE  9F 004E8        PUSHAB  DUM1                                  ;
                            50  DD 004EB        PUSHL   R0                                    ;
                6B          03  FB 004ED        CALLS   #3, DBG$STA_SYMTYPE                   ;
                52     04   AC  D0 004F0 52$:   MOVL    TYPEID, R2                            ; 1736
                04     18   A2  91 004F4        CMPB    24(R2), #4                            ;
                            14  13 004F8        BEQL    53$                                   ;
                09     18   A2  91 004FA        CMPB    24(R2), #9                            ; 1737
                            0E  13 004FE        BEQL    53$                                   ;
                       0C   A4  9F 00500        PUSHAB  12(R4)                                ; 1739
                            01  DD 00503        PUSHL   #1                                    ;
                  000289CA  8F  DD 00505        PUSHL   #166346                               ;
                69          03  FB 0050B        CALLS   #3, LIB$SIGNAL                        ;
                            04  DD 0050E 53$:   PUSHL   #4                                    ; 1743
                            7E  D4 00510        CLRL    -(SP)                                 ;
                            52  DD 00512        PUSHL   R2                                    ;
         0000V  CF          03  FB 00514        CALLS   #3, DBG$MAKE_VALUE_DESC               ;
                56          50  D0 00519        MOVL    R0, RESULT                            ;
   0E           64          0B  E0 0051C        BBS     #11, (R4), 54$                        ; 1747
                       0C   A4  9F 00520        PUSHAB  12(R4)                                ; 1749
                            01  DD 00523        PUSHL   #1                                    ;
                  00028838  8F  DD 00525        PUSHL   #165944                               ;
                69          03  FB 0052B        CALLS   #3, LIB$SIGNAL                        ;
                       0C   A4  9F 0052E 54$:   PUSHAB  12(R4)                                ; 1753
                            01  DD 00531        PUSHL   #1                                    ;
     00000000G  00          02  FB 00533        CALLS   #2, DBG$GET_BIF_ARGUMENTS             ;
                57          50  D0 0053A        MOVL    R0, ARG_LIST                          ;
                09     18   A2  91 0053D        CMPB    24(R2), #9                            ; 1755
                            29  12 00541        BNEQ    55$                                   ;
                       14   AE  9F 00543        PUSHAB  DUM3                                  ; 1761
                       1C   AE  9F 00546        PUSHAB  HIGHBOUND                             ;
                       24   AE  9F 00549        PUSHAB  LOWBOUND                              ;
                       1C   AE  9F 0054C        PUSHAB  RSTPTR                                ;
                            52  DD 0054F        PUSHL   R2                                    ;
                6A          05  FB 00551        CALLS   #5, DBG$STA_TYP_SUBRNG                ;
```

```
                    50        10    AE  D0 00554          MOVL    RSTPTR, R0                          ; 1762
                    04        18    A0  91 00558          CMPB    24(R0), #4
                                    0E  13 0055C          BEQL    55$
                              0C    A4  9F 0055E          PUSHAB  12(R4)                              ; 1764
                                    01  DD 00561          PUSHL   #1
                          000289CA  8F  DD 00563          PUSHL   #166346
                    69              03  FB 00569          CALLS   #3, LIB$SIGNAL
                                    56  DD 0056C 55$:     PUSHL   RESULT                              ; 1769
                              04    A7  DD 0056E          PUSHL   4(ARG_LIST)
              0000V CF              02  FB 00571          CALLS   #2, DBG$PRED_ENUM
                                    46  11 00576          BRB     58$                                 ; 1342
                    07              52  D1 00578 56$:     CMPL    OPCODE, #7                          ; 1773
                                    44  12 0057B          BNEQ    59$
                11  64              0B  E1 0057D          BBC     #11, (R4), 57$                      ; 1779
                          00000000'  EF  9F 00581          PUSHAB  P.AAT                               ; 1781
                                    01  DD 00587          PUSHL   #1
                          000289E2  8F  DD 00589          PUSHL   #166370
                    69              03  FB 0058F          CALLS   #3, LIB$SIGNAL
                    7E              20  7D 00592 57$:     MOVQ    #32, -(SP)                          ; 1785
                                    08  DD 00595          PUSHL   #8
          00000000G 00              03  FB 00597          CALLS   #3, DBG$TYPEID_FOR_ATOMIC
                 10 AE              50  D0 0059E          MOVL    R0, RSTPTR
                                    02  DD 005A2          PUSHL   #2                                  ; 1789
                                    7E  D4 005A4          CLRL    -(SP)
                              18    AE  DD 005A6          PUSHL   RSTPTR
              0000V CF              03  FB 005A9          CALLS   #3, DBG$MAKE_VALUE_DESC
                    56              50  D0 005AE          MOVL    R0, RESULT
                              20    A6  9F 005B1          PUSHAB  32(RESULT)                          ; 1794
                              04    AC  DD 005B4          PUSHL   TYPEID
          00000000G 00              02  FB 005B7          CALLS   #2, DBG$STA_SYMSIZE
                                01CE  31  DD 005BE 58$:     BRW     76$                                 ; 1342
                    08              52  D1 005C1 59$:     CMPL    OPCODE, #8                          ; 1798
                                    03  13 005C4          BEQL    60$
                                009E  31  DD 005C6          BRW     65$
                    50        04    AC  D0 005C9 60$:     MOVL    TYPEID, R0                          ; 1804
                    06        14    A0  91 005CD          CMPB    20(R0), #6
                                    0B  12 005D1          BNEQ    61$
                              04    AC  9F 005D3          PUSHAB  TYPEID                              ; 1806
                              24    AE  9F 005D6          PUSHAB  DUM1
                                    50  DD 005D9          PUSHL   R0
                    6B              03  FB 005DB          CALLS   #3, DBG$STA_SYMTYPE
                    52        04    AC  D0 005DE 61$:     MOVL    TYPEID, R2                          ; 1808
                    04        18    A2  91 005E2          CMPB    24(R2), #4
                                    14  13 005E6          BEQL    62$
                    09        18    A2  91 005E8          CMPB    24(R2), #9                          ; 1809
                                    0E  13 005EC          BEQL    62$
                              0C    A4  9F 005EE          PUSHAB  12(R4)                              ; 1811
                                    01  DD 005F1          PUSHL   #1
                          000289CA  8F  DD 005F3          PUSHL   #166346
                    69              03  FB 005F9          CALLS   #3, LIB$SIGNAL
                                    04  DD 005FC 62$:     PUSHL   #4                                  ; 1815
                                    7E  D4 005FE          CLRL    -(SP)
                                    52  DD 00600          PUSHL   R2
              0000V CF              03  FB 00602          CALLS   #3, DBG$MAKE_VALUE_DESC
                    56              50  D0 00607          MOVL    R0, RESULT
                0E  64              0B  E0 0060A          BBS     #11, (R4), 63$                      ; 1819
                              0C    A4  9F 0060E          PUSHAB  12(R4)                              ; 1821
```

```
                        01  DD 00611          PUSHL   #1                                          ;
                00028838    8F  DD 00613          PUSHL   #165944                                 ;
                    69      03  FB 00619          CALLS   #3, LIB$SIGNAL                          ;
                        0C  A4  9F 0061C 63$:      PUSHAB  12(R4)                                  ; 1825
                            01  DD 0061F          PUSHL   #1                                      ;
00000000G               00      02  FB 00621          CALLS   #2, DBG$GET_BIF_ARGUMENTS           ;
                        57      50  D0 00628          MOVL    R0, ARG_LIST                        ;
                        09  16  A2  91 0062B          CMPB    24(R2), #9                          ; 1827
                                29  12 0062F          BNEQ    64$                                 ;
                            14  AE  9F 00631          PUSHAB  DUM3                                ; 1833
                            1C  AE  9F 00634          PUSHAB  HIGHBOUND                           ;
                            24  AE  9F 00637          PUSHAB  LOWBOUND                            ;
                            1C  AE  9F 0063A          PUSHAB  RSTPTR                              ;
                            52  DD 0063D          PUSHL   R2                                      ;
                        6A      05  FB 0063F          CALLS   #5, DBG$STA_TYP_SUBRNG              ;
                        50  10  AE  D0 00642          MOVL    RSTPTR, R0                          ; 1834
                        04  18  A0  91 00646          CMPB    24(R0), #4                          ;
                                0E  13 0064A          BEQL    64$                                 ;
                            0C  A4  9F 0064C          PUSHAB  12(R4)                              ; 1836
                            01  DD 0064F          PUSHL   #1                                      ;
                00028CCA    8F  DD 00651          PUSHL   #166346                                 ;
                    69      03  FB 00657          CALLS   #3, LIB$SIGNAL                          ;
                        56  DD 0065A 64$:      PUSHL   RESULT                                  ; 1841
                        04  A7  DD 0065C          PUSHL   4(ARG_LIST)                             ;
                0000V   CF      02  FB 0065F          CALLS   #2, DBG$SUCC_ENUM                   ;
                            0128  31 00664          BRW     76$                                 ; 1342
                        09      52  D1 00667 65$:      CMPL    OPCODE, #9                          ; 1845
                                03  13 0066A          BEQL    66$                                 ;
                            010F  31 0066C          BRW     74$                                 ;
                        50  04  AC  D0 0066F 66$:      MOVL    TYPEID, R0                          ; 1851
                        06  14  A0  91 00673          CMPB    20(R0), #6                          ;
                                0B  12 00677          BNEQ    67$                                 ;
                            04  AC  9F 00679          PUSHAB  TYPEID                              ; 1853
                            24  AE  9F 0067C          PUSHAB  DUM1                                ;
                            50  DD 0067F          PUSHL   R0                                      ;
                        6B      03  FB 00681          CALLS   #3, DBG$STA_SYMTYPE                 ;
                        52  04  AC  D0 00684 67$:      MOVL    TYPEID, R2                          ; 1855
                        04  18  A2  91 00688          CMPB    24(R2), #4                          ;
                                14  13 0068C          BEQL    68$                                 ;
                        09  18  A2  91 0068E          CMPB    24(R2), #9                          ; 1856
                                0E  13 00692          BEQL    68$                                 ;
                            0C  A4  9F 00694          PUSHAB  12(R4)                              ; 1858
                            01  DD 00697          PUSHL   #1                                      ;
                00028CCA    8F  DD 00699          PUSHL   #166346                                 ;
                    69      03  FB 0069F          CALLS   #3, LIB$SIGNAL                          ;
            0E          64      0B  E0 006A2 68$:      BBS     #11, (R4), 69$                      ; 1862
                        0C  A4  9F 006A6          PUSHAB  12(R4)                              ; 1864
                            01  DD 006A9          PUSHL   #1                                      ;
                00028838    8F  DD 006AB          PUSHL   #165944                                 ;
                    69      03  FB 006B1          CALLS   #3, LIB$SIGNAL                          ;
                        0C  A4  9F 006B4 69$:      PUSHAB  12(R4)                              ; 1868
                            01  DD 006B7          PUSHL   #1                                      ;
            00000000G       00      02  FB 006B9          CALLS   #2, DBG$GET_BIF_ARGUMENTS           ;
                        57      50  D0 006C0          MOVL    R0, ARG_LIST                        ;
                        09  18  A2  91 006C3          CMPB    24(R2), #9                          ; 1870
                                29  12 006C7          BNEQ    70$                                 ;
                            14  AE  9F 006C9          PUSHAB  DUM3                                ; 1877
```

```
                        1C   AE  9F 006CC          PUSHAB  HIGHBOUND                                       ;
                        24   AE  9F 006CF          PUSHAB  LOWBOUND                                        ;
                        04   AC  9F 006D2          PUSHAB  TYPEID                                          ;
                             52  DD 006D5          PUSHL   R2                                              ;
                        6A       05  FB 006D7          CALLS   #5, DBG$STA_TYP_SUBRNG                      ;
                        50   04  AC  D0 006DA          MOVL    TYPEID, R0                                  ; 1878
                        04   18  A0  91 006DE          CMPB    24(R0), #4                                  ;
                             0E  13 006E2          BEQL    70$                                             ;
                        0C   A4  9F 006E4          PUSHAB  12(R4)                                          ; 1880
                             01  DD 006E7          PUSHL   #1                                              ;
                   000289CA  8F  DD 006E9          PUSHL   #166346                                         ;
                        69       03  FB 006EF          CALLS   #3, LIB$SIGNAL                              ;
                             04  DD 006F2  70$:     PUSHL   #4                                              ; 1887
                        7E   7A  8F  9A 006F4          MOVZBL  #122, -(SP)                                 ;
              00000000G 00       02  FB 006F8          CALLS   #2, DBG$MAKE_SKELETON_DESC                  ;
                        55       50  D0 006FF          MOVL    R0, TEMP_VAL_DESC                           ;
                  07    A5       06  90 00702          MOVB    #6, 7(TEMP_VAL_DESC)                        ; 1888
                        08   A5  D4 00706          CLRL    8(TEMP_VAL_DESC)                            ; 1889
                  18    A5   20  A5  9E 00709          MOVAB   32(R5), 24(TEMP_VAL_DESC)                   ; 1890
                  06    A5       02  90 0070E          MOVB    #2, 6(TEMP_VAL_DESC)                        ; 1891
                  14    A5 01080004  8F  D0 00712          MOVL    #17301508, 20(TEMP_VAL_DESC)            ; 1894
                             55  DD 0071A          PUSHL   TEMP_VAL_DESC                                   ; 1900
                        04   A7  DD 0071C          PUSHL   4(ARG_LIST)                                     ;
                   00000000G 00  9F 0071F          PUSHAB  DBG$GL_CONVERT_TOKEN                            ; 1899
              00000000G 00       03  FB 00725          CALLS   #3, DBG$EVAL_LANG_OPERATOR                  ;
                        55       50  D0 0072C          MOVL    R0, TEMP_VAL_DESC                           ;
                        52   20  A5  D0 0072F          MOVL    32(TEMP_VAL_DESC), R2                       ; 1905
                             0A  19 00733          BLSS    71$                                             ;
                        50   04  AC  D0 00735          MOVL    TYPEID, R0                                  ; 1906
                  28    A0       52  D1 00739          CMPL    R2, 40(R0)                                  ;
                             09  19 0073D          BLSS    72$                                             ;
                   000287F8  8F  DD 0073F  71$:     PUSHL   #165880                                         ; 1908
                        69       01  FB 00745          CALLS   #1, LIB$SIGNAL                              ;
                        50   04  AC  D0 00748  72$:     MOVL    TYPEID, R0                                  ; 1912
                        6E   2C  A0  9E 0074C          MOVAB   44(R0), COMPONENT_LIST                      ;
                  10    AE   00 BE42  D0 00750          MOVL    @COMPONENT_LIST[R2], RSTPTR                 ; 1913
                        7E   18  A0  9A 00756          MOVZBL  24(R0), -(SP)                               ; 1917
                             14  AE  DD 0075A          PUSHL   RSTPTR                                          ;
                             50  DD 0075D          PUSHL   R0                                              ;
                  0000V     CF       03  FB 0075F          CALLS   #3, DBG$MAKE_VALUE_DESC                     ;
                        56       50  D0 00764          MOVL    R0, RESULT                                  ;
                             20  AE  9F 00767  73$:     PUSHAB  DUM1                                            ; 1921
                             28  AE  9F 0076A          PUSHAB  DST_VALUE                                       ;
                             18  AE  DD 0076D          PUSHL   RSTPTR                                          ;
              00000000G 00       03  FB 00770          CALLS   #3, DBG$STA_SYMVALUE                        ;
                  20    A6   24  BE  D0 00777          MOVL    @DST_VALUE, 32(RESULT)                      ; 1922
                             11  11 0077C          BRB     76$                                             ; 1342
                   00000000'  EF  9F 0077E  74$:     PUSHAB  P.AAU                                           ; 1928
                             01  DD 00784          PUSHL   #1                                              ;
                   00028362  8F  DD 00786          PUSHL   #164706                                         ;
                        69       03  FB 0078C  75$:     CALLS   #3, LIB$SIGNAL                              ;
                        50       56  D0 0078F  76$:     MOVL    RESULT, R0                                  ; 1935
                                 04 00792          RET                                                     ; 1936

; Routine Size:  1939 bytes,    Routine Base:  DBG$CODE + 0810
```

```
;  1822          1937  1 GLOBAL ROUTINE DBG$GEQ_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;  1823          1938  1 !
;  1824          1939  1 ! FUNCTION
;  1825          1940  1 !
;  1826          1941  1 !       This routine is called to perform the greater than or equal to operation
;  1827          1942  1 !       on a scaled binary variable.
;  1828          1943  1 !
;  1829          1944  1 ! INPUTS
;  1830          1945  1 !
;  1831          1946  1 !       ARG_DESC1       - points to the value descriptor representing the
;  1832          1947  1 !                         left argument of the operation.
;  1833          1948  1 !       ARG_DESC2       - points to the value descriptor representing the
;  1834          1949  1 !                         right argument of the operation.
;  1835          1950  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  1836          1951  1 !                         of the operation.
;  1837          1952  1 !
;  1838          1953  1 ! OUTPUTS
;  1839          1954  1 !
;  1840          1955  1 !       The result value descriptor is filled in.
;  1841          1956  1 !       No value is returned.
;  1842          1957  1 !
;  1843          1958  2     BEGIN
;  1844          1959  2
;  1845          1960  2     MAP
;  1846          1961  2         ARG_DESC1       : REF DBG$VALDESC,
;  1847          1962  2         ARG_DESC2       : REF DBG$VALDESC,
;  1848          1963  2         RESULT_DESC     : REF DBG$VALDESC;
;  1849          1964  2
;  1850          1965  2     LOCAL
;  1851          1966  2         VAL_DESC1       : DBG$STG_DESC,
;  1852          1967  2         VAL_DESC2       : DBG$STG_DESC,
;  1853          1968  2         VALUE1,
;  1854          1969  2         VALUE2;
;  1855          1970  2
;  1856          1971  2     ! Set up working variables.  This way we don't mess up anything important.
;  1857          1972  2     !
;  1858          1973  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;  1859          1974  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;  1860          1975  2
;  1861          1976  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;  1862          1977  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;  1863          1978  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;  1864          1979  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;  1865          1980  2
;  1866          1981  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;  1867          1982  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;  1868          1983  2
;  1869          1984  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;  1870          1985  2
;  1871          1986  2     ! Do the Greater than or Equal to evaluation.
;  1872          1987  2     !
;  1873          1988  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 GEQ .VALUE2;
;  1874          1989  2
;  1875          1990  1     END;
```

```
                                    00FC 00000          .ENTRY  DBG$GEQ_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 1937
                                5E      20  C2 00002    SUBL2   #32, SP                                       ;
                                57  04  AC  D0 00005    MOVL    ARG_DESC1, R7                                 ; 1973
                            14  AE  14  A7  0C  28 00009 MOVC3   #12, 20(R7), VAL_DESC1                       ;
                                56  08  AC  D0 0000F    MOVL    ARG_DESC2, R6                                 ; 1974
                            08  AE  14  A6  0C  28 00013 MOVC3   #12, 20(R6), VAL_DESC2                       ;
                                6E  18  B7  D0 00019    MOVL    @24(R7), VALUE1                               ; 1976
                            04  AE  18  B6  D0 0001D    MOVL    @24(R6), VALUE2                               ; 1977
                            18  AE      6E  9E 00022    MOVAB   VALUE1, VAL_DESC1+4                           ; 1978
                            0C  AE  04  AE  9E 00026    MOVAB   VALUE2, VAL_DESC2+4                           ; 1979
                                    14  AE  9F 0002B    PUSHAB  VAL_DESC1                                     ; 1981
                        0000V   CF      01  FB 0002E    CALLS   #1, DBG$NORMALIZE_FIXED                       ;
                                    08  AE  9F 00033    PUSHAB  VAL_DESC2                                     ; 1982
                        0000V   CF      01  FB 00036    CALLS   #1, DBG$NORMALIZE_FIXED                       ;
                                    08  AE  9F 0003B    PUSHAB  VAL_DESC2                                     ; 1984
                                    18  AE  9F 0003E    PUSHAB  VAL_DESC1                                     ;
                        0000V   CF      02  FB 00041    CALLS   #2, MATCH_FIXED_BINARYS                       ;
                                51  0C  AC  D0 00046    MOVL    RESULT_DESC, R1                               ; 1988
                                        50  D4 0004A    CLRL    R0                                            ;
                            04  AE      6E  D1 0004C    CMPL    VALUE1, VALUE2                                ;
                                        02  19 00050    BLSS    1$                                            ;
                                        50  D6 00052    INCL    R0                                            ;
                            18  B1      50  D0 00054 1$: MOVL    R0, @24(R1)                                   ;
                                            04 00058    RET                                                   ; 1990

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 0FA3
```

```
; 1877          1991  1 GLOBAL ROUTINE DBG$GTR_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
; 1878          1992  1 !
; 1879          1993  1 ! FUNCTION
; 1880          1994  1 !
; 1881          1995  1 !       This routine is called to perform the greater than operation
; 1882          1996  1 !       on a scaled binary variable.
; 1883          1997  1 !
; 1884          1998  1 ! INPUTS
; 1885          1999  1 !
; 1886          2000  1 !       ARG_DESC1       - points to the value descriptor representing the
; 1887          2001  1 !                         left argument of the operation.
; 1888          2002  1 !       ARG_DESC2       - points to the value descriptor representing the
; 1889          2003  1 !                         right argument of the operation.
; 1890          2004  1 !       RESULT_DESC     - points to the value descriptor representing the result.
; 1891          2005  1 !                         of the operation.
; 1892          2006  1 !
; 1893          2007  1 ! OUTPUTS
; 1894          2008  1 !
; 1895          2009  1 !       The result value descriptor is filled in.
; 1896          2010  1 !       No value is returned.
; 1897          2011  1 !
; 1898          2012  2     BEGIN
; 1899          2013  2
; 1900          2014  2     MAP
; 1901          2015  2         ARG_DESC1       : REF DBG$VALDESC,
; 1902          2016  2         ARG_DESC2       : REF DBG$VALDESC,
; 1903          2017  2         RESULT_DESC     : REF DBG$VALDESC;
; 1904          2018  2
; 1905          2019  2     LOCAL
; 1906          2020  2         VAL_DESC1       : DBG$STG_DESC,
; 1907          2021  2         VAL_DESC2       : DBG$STG_DESC,
; 1908          2022  2         VALUE1,
; 1909          2023  2         VALUE2;
; 1910          2024  2
; 1911          2025  2     ! Set up working variables.  This way we don't mess up anything important.
; 1912          2026  2     !
; 1913          2027  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
; 1914          2028  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
; 1915          2029  2
; 1916          2030  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
; 1917          2031  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
; 1918          2032  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
; 1919          2033  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
; 1920          2034  2
; 1921          2035  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
; 1922          2036  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
; 1923          2037  2
; 1924          2038  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
; 1925          2039  2
; 1926          2040  2     ! Do the Greater Than evaluation.
; 1927          2041  2     !
; 1928          2042  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 GTR .VALUE2;
; 1929          2043  2
; 1930          2044  1     END;
```

```
                                                00FC 00000          .ENTRY  DBG$GTR_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 1991
                                             5E 20   C2 00002       SUBL2   #32, SP                                     ;
                                             57 04   AC  D0 00005   MOVL    ARG_DESC1, R7                               ; 2027
                                          14 AE      0C  28 00009   MOVC3   #12, 20(R7), VAL_DESC1                      ;
                                             56 08   AC  D0 0000F   MOVL    ARG_DESC2, R6                               ; 2028
                                          08 AE 14   A6  0C  28 00013   MOVC3   #12, 20(R6), VAL_DESC2                  ;
                                             6E 18   B7  D0 00019   MOVL    @24(R7), VALUE1                             ; 2030
                                          04 AE 18   B6  D0 0001D   MOVL    @24(R6), VALUE2                             ; 2031
                                          18 AE      6E  9E 00022   MOVAB   VALUE1, VAL_DESC1+4                         ; 2032
                                          0C AE 04   AE  9E 00026   MOVAB   VALUE2, VAL_DESC2+4                         ; 2033
                                             14 AE   9F  0002B      PUSHAB  VAL_DESC1                                   ; 2035
                                       0000V CF      01  FB 0002E   CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                             08 AE   9F  00033      PUSHAB  VAL_DESC2                                   ; 2036
                                       0000V CF      01  FB 00036   CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                             08 AE   9F  0003B      PUSHAB  VAL_DESC2                                   ; 2038
                                             18 AE   9F  0003E      PUSHAB  VAL_DESC1                                   ;
                                       0000V CF      02  FB 00041   CALLS   #2, MATCH_FIXED_BINARYS                     ;
                                             51 0C   AC  D0 00046   MOVL    RESULT_DESC, R1                             ; 2042
                                                50   D4  0004A      CLRL    R0                                          ;
                                          04 AE      6E  D1 0004C   CMPL    VALUE1, VALUE2                              ;
                                                02   15  00050      BLEQ    1$                                          ;
                                                50   D6  00052      INCL    R0                                          ;
                                          18 B1      50  D0 00054 1$: MOVL    R0, @24(R1)                                 ;
                                                04 00058           RET                                                 ; 2044

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 0FFC
```

```
;1932        2045  1 GLOBAL ROUTINE DBG$LEQ_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;1933        2046  1 !
;1934        2047  1 ! FUNCTION
;1935        2048  1 !
;1936        2049  1 !       This routine is called to perform the less than or equal to operation
;1937        2050  1 !       on a scaled binary variable.
;1938        2051  1 !
;1939        2052  1 ! INPUTS
;1940        2053  1 !
;1941        2054  1 !       ARG_DESC1       - points to the value descriptor representing the
;1942        2055  1 !                         left argument of the operation.
;1943        2056  1 !       ARG_DESC2       - points to the value descriptor representing the
;1944        2057  1 !                         right argument of the operation.
;1945        2058  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;1946        2059  1 !                         of the operation.
;1947        2060  1 !
;1948        2061  1 ! OUTPUTS
;1949        2062  1 !
;1950        2063  1 !       The result value descriptor is filled in.
;1951        2064  1 !       No value is returned.
;1952        2065  1 !
;1953        2066  2     BEGIN
;1954        2067  2
;1955        2068  2     MAP
;1956        2069  2         ARG_DESC1       : REF DBG$VALDESC,
;1957        2070  2         ARG_DESC2       : REF DBG$VALDESC,
;1958        2071  2         RESULT_DESC     : REF DBG$VALDESC;
;1959        2072  2
;1960        2073  2     LOCAL
;1961        2074  2         VAL_DESC1       : DBG$STG_DESC,
;1962        2075  2         VAL_DESC2       : DBG$STG_DESC,
;1963        2076  2         VALUE1,
;1964        2077  2         VALUE2;
;1965        2078  2
;1966        2079  2     ! Set up working variables.  This way we don't mess up anything important.
;1967        2080  2     !
;1968        2081  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;1969        2082  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;1970        2083  2
;1971        2084  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;1972        2085  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;1973        2086  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;1974        2087  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;1975        2088  2
;1976        2089  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;1977        2090  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;1978        2091  2
;1979        2092  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;1980        2093  2
;1981        2094  2     ! Do the Less Than or Equal to evaluation.
;1982        2095  2     !
;1983        2096  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 LEQ .VALUE2;
;1984        2097  2
;1985        2098  1     END;
```

```
                                    00FC 00000          .ENTRY  DBG$LEQ_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 2045
                                5E      20   C2 00002          SUBL2   #32, SP
                                57      04   AC  D0 00005      MOVL    ARG_DESC1, R7                          ; 2081
                14    AE          14    A7   0C  28 00009      MOVC3   #12, 20(R7), VAL_DESC1
                                56      08   AC  D0 0000F      MOVL    ARG_DESC2, R6                          ; 2082
                08    AE          14    A6   0C  28 00013      MOVC3   #12, 20(R6), VAL_DESC2
                                6E      18   B7  D0 00019      MOVL    @24(R7), VALUE1                        ; 2084
                            04  AE      18   B6  D0 0001D      MOVL    @24(R6), VALUE2                        ; 2085
                            18  AE          6E  9E 00022      MOVAB   VALUE1, VAL_DESC1+4                    ; 2086
                            0C  AE      04  AE  9E 00026      MOVAB   VALUE2, VAL_DESC2+4                    ; 2087
                                        14  AE  9F 0002B      PUSHAB  VAL_DESC1                              ; 2089
                        0000V   CF          01  FB 0002E      CALLS   #1, DBG$NORMALIZE_FIXED
                                        08  AE  9F 00033      PUSHAB  VAL_DESC2                              ; 2090
                        0000V   CF          01  FB 00036      CALLS   #1, DBG$NORMALIZE_FIXED
                                        08  AE  9F 00 3B      PUSHAB  VAL_DESC2                              ; 2092
                                        18  AE  9F 0003E      PUSHAB  VAL_DESC1
                        0000V   CF          02  FB 00041      CALLS   #2, MATCH_FIXED_BINARYS
                                51      0C  AC  D0 00046      MOVL    RESULT_DESC, R1                        ; 2096
                                            50  D4 0004A      CLRL    R0
                            04  AE          6E  D1 0004C      CMPL    VALUE1, VALUE2
                                            02  14 00050      BGTR    1$
                                            50  D6 00052      INCL    R0
                            18  B1          50  D0 00054 1$:  MOVL    R0, @24(R1)
                                                04 00058      RET                                            ; 2098

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 1055
```

```
;1987        2099  1 GLOBAL ROUTINE DBG$LSS_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;1988        2100  1 !
;1989        2101  1 ! FUNCTION
;1990        2102  1 !
;1991        2103  1 !       This routine is called to perform the less than operation
;1992        2104  1 !       on a scaled binary variable.
;1993        2105  1 !
;1994        2106  1 ! INPUTS
;1995        2107  1 !
;1996        2108  1 !       ARG_DESC1       - points to the value descriptor representing the
;1997        2109  1 !                         left argument of the operation.
;1998        2110  1 !       ARG_DESC2       - points to the value descriptor representing the
;1999        2111  1 !                         right argument of the operation.
;2000        2112  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;2001        2113  1 !                         of the operation.
;2002        2114  1 !
;2003        2115  1 ! OUTPUTS
;2004        2116  1 !
;2005        2117  1 !       The result value descriptor is filled in.
;2006        2118  1 !       No value is returned.
;2007        2119  1 !
;2008        2120  2     BEGIN
;2009        2121  2
;2010        2122  2     MAP
;2011        2123  2         ARG_DESC1       : REF DBG$VALDESC,
;2012        2124  2         ARG_DESC2       : REF DBG$VALDESC,
;2013        2125  2         RESULT_DESC     : REF DBG$VALDESC;
;2014        2126  2
;2015        2127  2     LOCAL
;2016        2128  2         VAL_DESC1       : DBG$STG_DESC,
;2017        2129  2         VAL_DESC2       : DBG$STG_DESC,
;2018        2130  2         VALUE1,
;2019        2131  2         VALUE2;
;2020        2132  2
;2021        2133  2     ! Set up working variables.  This way we don't mess up anything important.
;2022        2134  2     !
;2023        2135  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;2024        2136  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;2025        2137  2
;2026        2138  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;2027        2139  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;2028        2140  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;2029        2141  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;2030        2142  2
;2031        2143  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;2032        2144  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;2033        2145  2
;2034        2146  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;2035        2147  2
;2036        2148  2     ! Do the Less Than evaluation.
;2037        2149  2     !
;2038        2150  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 LSS .VALUE2;
;2039        2151  2
;2040        2152  1     END;
```

```
                                00FC 00000          .ENTRY  DBG$LSS_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 2099
                             5E    20   C2 00002    SUBL2   #32, SP                                     ;
                             57 04 AC   D0 00005    MOVL    ARG_DESC1, R7                               ; 2135
                          14 AE    14 A7   0C   28 00009    MOVC3   #12, 20(R7), VAL_DESC1              ;
                             56 08 AC   D0 0000F    MOVL    ARG_DESC2, R6                               ; 2136
                          08 AE    14 A6   0C   28 00013    MOVC3   #12, 20(R6), VAL_DESC2              ;
                             6E 18 B7   D0 00019    MOVL    @24(R7), VALUE1                             ; 2138
                          04 AE    18 B6   D0 0001D    MOVL    @24(R6), VALUE2                             ; 2139
                          18 AE       6E   9E 00022    MOVAB   VALUE1, VAL_DESC1+4                         ; 2140
                          0C AE    04 AE   9E 00026    MOVAB   VALUE2, VAL_DESC2+4                         ; 2141
                             14 AE    9F 0002B    PUSHAB  VAL_DESC1                                   ; 2143
                       0000V CF       01   FB 0002E    CALLS   #1, DBG$NORMALIZE_FIXED             ;
                             08 AE    9F 00033    PUSHAB  VAL_DESC2                                   ; 2144
                       0000V CF       01   FB 00036    CALLS   #1, DBG$NORMALIZE_FIXED             ;
                             08 AE    9F 0003B    PUSHAB  VAL_DESC2                                   ; 2146
                             18 AE    9F 0003E    PUSHAB  VAL_DESC1                                   ;
                       0000V CF       02   FB 00041    CALLS   #2, MATCH_FIXED_BINARYS             ;
                             51 0C AC   D0 00046    MOVL    RESULT_DESC, R1                             ; 2150
                                50   D4 0004A    CLRL    R0                                          ;
                          04 AE       6E   D1 0004C    CMPL    VALUE1, VALUE2                              ;
                                02   18 00050    BGEQ    1$                                          ;
                                50   D6 00052    INCL    R0                                          ;
                          18 B1       50   D0 00054 1$:  MOVL    R0, @24(R1)                                 ;
                                      04 00058    RET                                                 ; 2152

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 10AE
```

```
;  2042          2153  1 GLOBAL ROUTINE DBG$MAKE_VALUE_DESC (TYPEID, SYMID, FCODE) =
;  2043          2154  1 !
;  2044          2155  1 ! FUNCTION
;  2045          2156  1 !       Allocates space for a value descriptor of the given type, and
;  2046          2157  1 !       fills in the fields.
;  2047          2158  1 !
;  2048          2159  1 ! INPUTS
;  2049          2160  1 !       TYPEID  - RST Type Entry
;  2050          2161  1 !       SYMID   - RST Symbol Entry (May be zero)
;  2051          2162  1 !       FCODE   - Format code for value descriptor
;  2052          2163  1 !
;  2053          2164  1 ! OUTPUTS
;  2054          2165  1 !       Returns the address of a value descriptor allocated out of temporary
;  2055          2166  1 !       memory.
;  2056          2167  1 !;
;  2057          2168  1
;  2058          2169  2     BEGIN
;  2059          2170  2
;  2060          2171  2     MAP
;  2061          2172  2         TYPEID  : REF RST$ENTRY;        ! RST Entry type for the value descriptor
;  2062          2173  2
;  2063          2174  2     LITERAL
;  2064          2175  2         DESC_LENGTH = 10;               ! Length of the descriptor
;  2065          2176  2
;  2066          2177  2     LOCAL
;  2067          2178  2         DUM1,DUM2,                      ! Dummy variables for routine call
;  2068          2179  2         RESULT  : REF DBG$VALDESC;      ! Address of the result descriptor
;  2069          2180  2
;  2070          2181  2
;  2071          2182  2     ! Get temporary memory for the new descriptor
;  2072          2183  2     !
;  2073          2184  2     RESULT = DBG$GET_TEMPMEM (DESC_LENGTH);
;  2074          2185  2
;  2075          2186  2     ! Fill in the fields of the new value descriptor.
;  2076          2187  2     !
;  2077          2188  2     RESULT [DBG$B_DHDR_LANG]    = .DBG$GB_LANGUAGE;
;  2078          2189  2     RESULT [DBG$B_DHDR_TYPE]    = DBG$K_VALUE_DESC;
;  2079          2190  2     RESULT [DBG$W_DHDR_LENGTH]  = 4 * DESC_LENGTH;
;  2080          2191  2     RESULT [DBG$B_DHDR_KIND]    = RST$K_DATA;
;  2081          2192  2     RESULT [DBG$B_DHDR_FCODE]   = .FCODE;
;  2082          2193  2     RESULT [DBG$L_DHDR_TYPEID]  = .TYPEID;
;  2083          2194  2     RESULT [DBG$L_DHDR_SYMID0]  = .SYMID;
;  2084          2195  2
;  2085          2196  2     IF NOT DBG$FILL_IN_VMS_DESC(.FCODE, .TYPEID, 0,
;  2086          2197  2                                 RESULT[DBG$A_VALUE_VMSDESC],DUM1, DUM2)
;  2087          2198  2     THEN
;  2088          2199  2         $DBG_ERROR ('DBGLANGOP\DBG$MAKE_VALUE_DESC could not fill in the VMS fields');
;  2089          2200  2
;  2090          2201  2     RESULT [DBG$L_VALUE_POINTER] = RESULT[DBG$A_VALUE_ADDRESS];
;  2091          2202  2
;  2092          2203  2     RETURN .RESULT;
;  2093          2204  1     END;


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

```
24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 3E  002C9 P.AAV:  .ASCII  \>DBGLANGOP\<92>\DBG$MAKE_VALUE_DESC cou\
43 53 45 44 5F 45 55 4C 41 56 5F 45 4B 41 4D  002D8
                                  75 6F 63 20  002E7
20 6E 69 20 6C 6C 69 66 20 74 6F 6E 20 64 6C  002EB          .ASCII  \ld not fill in the VMS fields\
   73 64 6C 65 69 66 20 53 4D 56 20 65 68 74  002FA


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

                                     0004 00000            .ENTRY  DBG$MAKE_VALUE_DESC, Save R2                ; 2153
                          5E          08  C2 00002        SUBL2   #8, SP                                      ;
                                      0A  DD 00005        PUSHL   #10                                         ; 2184
                00000000G 00          01  FB 00007        CALLS   #1, DBG$GET_TEMPMEM                         ;
                          52          50  D0 0000E        MOVL    R0, RESULT                                  ;
                      03  A2 00000000G 00  90 00011        MOVB    DBG$GB_LANGUAGE, 3(RESULT)                  ; 2188
                      02  A2        7A   8F  90 00019        MOVB    #122, 2(RESULT)                             ; 2189
                          62              28  B0 0001E        MOVW    #40, (RESULT)                               ; 2190
                      07  A2              06  90 00021        MOVB    #6, 7(RESULT)                               ; 2191
                      06  A2        0C   AC  90 00025        MOVB    FCODE, 6(RESULT)                            ; 2192
                      08  A2        04   AC  7D 0002A        MOVQ    TYPEID, 8(RESULT)                           ; 2193
                                      5E  DD 0002F        PUSHL   SP                                          ; 2197
                                  08  AE  9F 00031        PUSHAB  DUM1                                        ;
                                  14  A2  9F 00034        PUSHAB  20(RESULT)                                  ;
                                      7E  D4 00037        CLRL    -(SP)                                       ;
                                  04  AC  DD 00039        PUSHL   TYPEID                                      ;
                                  0C  AC  DD 0003C        PUSHL   FCODE                                       ;
                00000000G 00              06  FB 00C3F        CALLS   #6, DBG$FILL_IN_VMS_DESC                    ;
                          15              50  E8 00046        BLBS    R0, 1$                                      ;
                             00000000'    EF  9F 00049        PUSHAB  P.AAV                                       ; 2199
                                          01  DD 0004F        PUSHL   #1                                          ;
                             00028362     8F  DD 00051        PUSHL   #164706                                     ;
                00000000G 00              03  FB 00057        CALLS   #3, LIB$SIGNAL                              ;
                      18  A2        20   A2  9E 0005E 1$:     MOVAB   32(R2), 24(RESULT)                          ; 2201
                          50              52  D0 00063        MOVL    RESULT, R0                                  ; 2203
                                          04  00066        RET                                                 ; 2204

; Routine Size:  103 bytes,    Routine Base:  DBG$CODE + 1107
```

```
;2095     2205  1 GLOBAL ROUTINE DBG$MUL_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;2096     2206  1 !
;2097     2207  1 !++ FUNCTION
;2098     2208  1 !
;2099     2209  1 !       This routine is called to perform the multiply operation
;2100     2210  1 !       on a scaled binary variable.
;2101     2211  1 !
;2102     2212  1 !-- INPUTS
;2103     2213  1 !
;2104     2214  1 !       ARG_DESC1       - points to the value descriptor representing the
;2105     2215  1 !                         left argument of the operation.
;2106     2216  1 !       ARG_DESC2       - points to the value descriptor representing the
;2107     2217  1 !                         right argument of the operation.
;2108     2218  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;2109     2219  1 !                         of the operation.
;2110     2220  1 !
;2111     2221  1 !-- OUTPUTS
;2112     2222  1 !
;2113     2223  1 !       The result value descriptor is filled in.
;2114     2224  1 !       No value is returned.
;2115     2225  1 !
;2116     2226  2     BEGIN
;2117     2227  2
;2118     2228  2     MAP
;2119     2229  2         ARG_DESC1       : REF DBG$VALDESC,
;2120     2230  2         ARG_DESC2       : REF DBG$VALDESC,
;2121     2231  2         RESULT_DESC     : REF DBG$VALDESC;
;2122     2232  2
;2123     2233  2     LOCAL
;2124     2234  2         INDEX,
;2125     2235  2         RESULT_VALUE    : VECTOR[2,LONG],
;2126     2236  2         SCALE,
;2127     2237  2         TEMP_VAL        : BITVECTOR[32],
;2128     2238  2         VAL_DESC1       : DBG$STG_DESC,
;2129     2239  2         VAL_DESC2       : DBG$STG_DESC,
;2130     2240  2         VALUE1,
;2131     2241  2         VALUE2;
;2132     2242  2
;2133     2243  2     ! Set up working variables.  This way we don't mess up anything important.
;2134     2244  2     !
;2135     2245  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;2136     2246  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;2137     2247  2
;2138     2248  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;2139     2249  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;2140     2250  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;2141     2251  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;2142     2252  2
;2143     2253  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;2144     2254  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;2145     2255  2
;2146     2256  2     ! Do the add.
;2147     2257  2     !
;2148     2258  2     EMUL(VALUE1, VALUE2, %REF(0), RESULT_VALUE);
;2149     2259  2     SCALE = .VAL_DESC1[DSC$B_SCALE] + .VAL_DESC2[DSC$B_SCALE];
;2150     2260  2
;2151     2261  2     ! Now it gets tricky...
```

```
; 2152          2262  2      ! We've got to put this quadword result into a longword and check
; 2153          2263  2      ! for a loss of precision.
; 2154          2264  2      !
; 2155          2265  2      ! Let's start by finding the most significant bit in the second longword.
; 2156          2266  2      !     First: if the value of the second longword is 0 or -1, we needn't
; 2157          2267  2      !     waste our time.
; 2158          2268  2      !
; 2159          2269  2      IF .RESULT_VALUE[1] EQL 0 OR
; 2160          2270  2         .RESULT_VALUE[1] EQL -1
; 2161          2271  2      THEN
; 2162          2272  3          BEGIN
; 2163          2273  3          TEMP_VAL = .RESULT_VALUE[0];
; 2164          2274  3          IF .TEMP_VAL[31] NEQ .(RESULT_VALUE+4)<0, 1, 0>
; 2165          2275  3          THEN
; 2166          2276  4              BEGIN
; 2167          2277  4              IF .TEMP_VAL[0]
; 2168          2278  4              THEN
; 2169          2279  4                  SIGNAL(DBG$_IFIXUND);
; 2170          2280  4              TEMP_VAL = .TEMP_VAL ^ -1;
; 2171          2281  4              SCALE = .SCALE + 1;
; 2172          2282  3              END;
; 2173          2283  3          END
; 2174          2284  2      ELSE
; 2175          2285  3          BEGIN
; 2176          2286  3          TEMP_VAL = .RESULT_VALUE[1];
; 2177          2287  3          INDEX = 30;
; 2178          2288  3          WHILE (.TEMP_VAL[.INDEX] NEQ .TEMP_VAL[31]) AND (.INDEX GEQ 0) DO
; 2179          2289  3              INDEX = .INDEX - 1;
; 2180          2290  3          IF .INDEX EQL -1
; 2181          2291  3          THEN
; 2182          2292  3              $DBG_ERROR('DBGLANGOP\DBG$MUL_FIXED_FIXED should never be here');
; 2183          2293  3
; 2184          2294  3              ! Move the 32 bits into our temporary location, and modify the scale.
; 2185          2295  3              !
; 2186          2296  3              TEMP_VAL = .RESULT_VALUE<(.INDEX + 2), 32, 0>;
; 2187          2297  3              SCALE = .SCALE + .INDEX + 2;
; 2188          2298  3              IF .RESULT_VALUE<0, (.INDEX + 2), 0> NEQ 0
; 2189          2299  3              THEN
; 2190          2300  3                  SIGNAL(DBG$_IFIXUND);
; 2191          2301  2              END;
; 2192          2302  2
; 2193          2303  2          .RESULT_DESC[DBG$L_VALUE_POINTER] = .TEMP_VAL;
; 2194          2304  2          RESULT_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
; 2195          2305  2          RESULT_DESC[DBG$B_VALUE_SCALE] = .SCALE;
; 2196          2306  2
; 2197          2307  1          END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 32  00308 P.AAW:  .ASCII  \2DBGLANGOP\<92>\DBG$MUL_FIXED_FIXED sho\  ;
44 45 58 49 46 5F 44 45 58 49 46 5F 4C 55 4D  00317                                                          ;
                                  6F 68 73 20  00326                                                          ;
65 68 20 65 62 20 72 65 76 65 6E 20 64 6C 75  0032A          .ASCII  \uld never be here\                      ;
                                     65 72     00339                                                          ;
```

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

                                                        01FC 00000      .ENTRY  DBG$MUL_FIXED_FIXED, Save R2,R3,R4,R5,R6,-      ; 2205
                                                                                R7,R8
                                                58 00000000G    00  9E 00002    MOVAB   LIB$SIGNAL, R8                          ;
                                                5E              28  C2 00009    SUBL2   #40, SP                                 ;
                                                57          04  AC  D0 0000C    MOVL    ARG_DESC1, R7                           ; 2245
                                        14  AE      14      A7      0C 28 00010    MOVC3   #12, 20(R7), VAL_DESC1                  ;
                                                56          08  AC  D0 00016    MOVL    ARG_DESC2, R6                           ; 2246
                                        08  AE      14      A6      0C 28 0001A    MOVC3   #12, 20(R6), VAL_DESC2                  ;
                                                6E          18  B7  D0 00020    MOVL    @24(R7), VALUE1                         ; 2248
                                                    04      AE  B6  D0 00024    MOVL    @24(R6), VALUE2                         ; 2249
                                                    18      AE  6E  9E 00029    MOVAB   VALUE1, VAL_DESC1+4                     ; 2250
                                                    0C      AE      04  AE  9E 0002D    MOVAB   VALUE2, VAL_DESC2+4             ; 2251
                                                            14  AE  9F 00032    PUSHAB  VAL_DESC1                               ; 2253
                                                0000V   CF          01  FB 00035    CALLS   #1, DBG$NORMALIZE_FIXED                 ;
                                                            08  AE  9F 0003A    PUSHAB  VAL_DESC2                               ; 2254
                                                0000V   CF          01  FB 0003D    CALLS   #1, DBG$NORMALIZE_FIXED                 ;
                                    20  AE          00      04  AE  6E  7A 00042    EMUL    VALUE1, VALUE2, #0, RESULT_VALUE        ; 2258
                                                53          1C  AE  98 00049    CVTBL   VAL_DESC1+8, SCALE                      ; 2259
                                                50          10  AE  98 0004D    CVTBL   VAL_DESC2+8, R0                         ;
                                                53              50  C0 00051    ADDL2   R0, SCALE                               ;
                                                50          24  AE  D0 00054    MOVL    RESULT_VALUE+4, R0                      ; 2269
                                                            09  13 00058    BEQL    1$                                      ;
                                            FFFFFFFF    8F          50  D1 0005A    CMPL    R0, #-1                             ; 2270
                                                            26  12 00061    BNEQ    3$                                      ;
                                                54          20  AE  D0 00063 1$:   MOVL    RESULT_VALUE, TEMP_VAL              ; 2273
                                        50      24  AE          01      00  EF 00067    EXTZV   #0, #1, RESULT_VALUE+4, R0      ; 2274
                                        50          54          01      1F  ED 0006D    CMPZV   #31, #1, TEMP_VAL, R0           ;
                                                            69  13 00072    BEQL    7$                                      ;
                                                09              54  E9 00074    BLBC    TEMP_VAL, 2$                            ; 2277
                                                00028788    8F      DD 00077    PUSHL   #165771                                 ; 2279
                                                68              01  FB 0007D    CALLS   #1, LIB$SIGNAL                          ;
                                        54              54      FF  8F  78 00080 2$:    ASHL    #-1, TEMP_VAL, TEMP_VAL         ; 2280
                                                            53  D6 00085    INCL    SCALE                                   ; 2281
                                                            54  11 00087    BRB     7$                                      ; 2269
                                                54              50  D0 00089 3$:    MOVL    R0, TEMP_VAL                        ; 2286
                                                52              1E  D0 0008C    MOVL    #30, INDEX                              ; 2287
                                    50          54          01      52  EF 0008F 4$:    EXTZV   INDEX, #1, TEMP_VAL, R0         ; 2288
                                    50          54          01      1F  ED 00094    CMPZV   #31, #1, TEMP_VAL, R0           ;
                                                            08  13 00099    BEQL    5$                                      ;
                                                            52  D5 0009B    TSTL    INDEX                                   ;
                                                            04  19 0009D    BLSS    5$                                      ;
                                                            52  D7 0009F    DECL    INDEX                                   ; 2289
                                                            EC  11 000A1    BRB     4$                                      ;
                                            FFFFFFFF    8F          52  D1 000A3 5$:    CMPL    INDEX, #-1                      ; 2290
                                                            11  12 000AA    BNEQ    6$                                      ;
                                                00000000'   EF      9F 000AC    PUSHAB  P.AAW                                   ; 2292
                                                            01  DD 000B2    PUSHL   #1                                      ;
                                                00028362    8F      DD 000B4    PUSHL   #164706                                 ;
                                                68              03  FB 000BA    CALLS   #3, LIB$SIGNAL                          ;
                                                51          02  A2  9E 000BD 6$:    MOVAB   2(R2), R1                           ; 2296
                                    54      20  AE          20      51  EF 000C1    EXTZV   R1, #32, RESULT_VALUE, TEMP_VAL         ;
                                                53          02 A243 9E 000C7    MOVAB   2(INDEX)[SCALE], SCALE                  ; 2297
                                    50      20  AE          51          00  EF 000CC    EXTZV   #0, R1, RESULT_VALUE, R0        ; 2298
```

```
                                    09  13 000D2          BEQL    7$                                      ;
                          0002878B  8F  DD 000D4          PUSHL   #165771                                 ; 2300
                       68           01  FB 000DA          CALLS   #1, LIB$SIGNAL                          ;
                       50        0C  AC  D0 000DD 7$:      MOVL    RESULT_DESC, R0                         ; 2303
                    18 B0           54  D0 000E1          MOVL    TEMP_VAL, @24(R0)                       ;
                    16 A0           08  90 000E5          MOVB    #8, 22(R0)                              ; 2304
                    1C A0           53  90 000E9          MOVB    SCALE, 28(R0)                           ; 2305
                                        04 000ED          RET                                             ; 2307

; Routine Size:  238 bytes,    Routine Base:  DBG$CODE + 116E
```

```
;  2199          2308  1 GLOBAL ROUTINE DBG$NEQ_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;  2200          2309  1 !
;  2201          2310  1 ! FUNCTION
;  2202          2311  1 !
;  2203          2312  1 !       This routine is called to perform the not equal to operation
;  2204          2313  1 !       on a scaled binary variable.
;  2205          2314  1 !
;  2206          2315  1 ! INPUTS
;  2207          2316  1 !
;  2208          2317  1 !       ARG_DESC1       - points to the value descriptor representing the
;  2209          2318  1 !                         left argument of the operation.
;  2210          2319  1 !       ARG_DESC2       - points to the value descriptor representing the
;  2211          2320  1 !                         right argument of the operation.
;  2212          2321  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  2213          2322  1 !                         of the operation.
;  2214          2323  1 !
;  2215          2324  1 ! OUTPUTS
;  2216          2325  1 !
;  2217          2326  1 !       The result value descriptor is filled in.
;  2218          2327  1 !       No value is returned.
;  2219          2328  1 !
;  2220          2329  2     BEGIN
;  2221          2330  2
;  2222          2331  2     MAP
;  2223          2332  2         ARG_DESC1       : REF DBG$VALDESC,
;  2224          2333  2         ARG_DESC2       : REF DBG$VALDESC,
;  2225          2334  2         RESULT_DESC     : REF DBG$VALDESC;
;  2226          2335  2
;  2227          2336  2     LOCAL
;  2228          2337  2         VAL_DESC1       : DBG$STG_DESC,
;  2229          2338  2         VAL_DESC2       : DBG$STG_DESC,
;  2230          2339  2         VALUE1,
;  2231          2340  2         VALUE2;
;  2232          2341  2
;  2233          2342  2     ! Set up working variables.  This way we don't mess up anything important.
;  2234          2343  2     !
;  2235          2344  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;  2236          2345  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;  2237          2346  2
;  2238          2347  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;  2239          2348  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;  2240          2349  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;  2241          2350  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;  2242          2351  2
;  2243          2352  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;  2244          2353  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;  2245          2354  2
;  2246          2355  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;  2247          2356  2
;  2248          2357  2     ! Do the Not Equal evaluation.
;  2249          2358  2     !
;  2250          2359  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = .VALUE1 NEQ .VALUE2;
;  2251          2360  2
;  2252          2361  1     END;
```

```
                                                00FC 00000          .ENTRY  DBG$NEQ_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 2308
                                             5E    20  C2 00002      SUBL2   #32, SP                                     ;
                                             57 04  AC  D0 00005      MOVL    ARG_DESC1, R7                               ; 2344
                                          14 AE 14  A7  0C  28 00009 MOVC3   #12, 20(R7), VAL_DESC1                      ;
                                             56 08  AC  D0 0000F      MOVL    ARG_DESC2, R6                               ; 2345
                                          08 AE 14  A6  0C  28 00013 MOVC3   #12, 20(R6), VAL_DESC2                      ;
                                             6E 18  B7  D0 00019      MOVL    @24(R7), VALUE1                             ; 2347
                                          04 AE 18  B6  D0 0001D      MOVL    @24(R6), VALUE2                             ; 2348
                                          18 AE     6E  9E 00022      MOVAB   VALUE1, VAL_DESC1+4                         ; 2349
                                          0C AE 04  AE  9E 00026      MOVAB   VALUE2, VAL_DESC2+4                         ; 2350
                                             14 AE  9F  0002B         PUSHAB  VAL_DESC1                                   ; 2352
                                       0000V CF     01  FB 0002E      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                             08 AE  9F  00033         PUSHAB  VAL_DESC2                                   ; 2353
                                       0000V CF     01  FB 00036      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                             08 AE  9F  0003B         PUSHAB  VAL_DESC2                                   ; 2355
                                             18 AE  9F  0003E         PUSHAB  VAL_DESC1                                   ;
                                       0000V CF     02  FB 00041      CALLS   #2, MATCH_FIXED_BINARYS                     ;
                                          51 0C     AC  D0 00046      MOVL    RESULT_DESC, R1                             ; 2359
                                             50     D4  0004A         CLRL    R0                                          ;
                                          04 AE     6E  D1 0004C      CMPL    VALUE1, VALUE2                              ;
                                                    02  13 00050      BEQL    1$                                          ;
                                                    50  D6 00052      INCL    R0                                          ;
                                          18 B1     50  D0 00054 1$:  MOVL    R0, @24(R1)                                 ;
                                                        04 00058      RET                                                 ; 2361

; Routine Size:  89 bytes,    Routine Base:  DBG$CODE + 125C
```

```
;  2254          2362  1 GLOBAL ROUTINE DBG$NORMALIZE_FIXED (FIXED_DESC): NOVALUE =
;  2255          2363  1 !
;  2256          2364  1 ! FUNCTION
;  2257          2365  1 !
;  2258          2366  1 !       This routine is called to normalize a scaled binary value.
;  2259          2367  1 !       It will shift the bits as far towards bit zero as possible,
;  2260          2368  1 !       by moving the rightmost one bit to bit zero and likewise
;  2261          2369  1 !       adjusting all the rest.  The value of the DSC$B_SCALE field
;  2262          2370  1 !       in the descriptor will be adjusted accordingly.
;  2263          2371  1 !
;  2264          2372  1 ! INPUTS
;  2265          2373  1 !
;  2266          2374  1 !       FIXED_DESC      - points to the scaled binary descriptor to be altered.
;  2267          2375  1 !
;  2268          2376  1 ! OUTPUTS
;  2269          2377  1 !
;  2270          2378  1 !       The descriptor is altered as described above.
;  2271          2379  1 !       No value is returned.
;  2272          2380  1 !
;  2273          2381  2     BEGIN
;  2274          2382  2
;  2275          2383  2     MAP
;  2276          2384  2         FIXED_DESC      : REF DBG$STG_DESC;
;  2277          2385  2
;  2278          2386  2     LOCAL
;  2279          2387  2         FIXED_LENGTH,
;  2280          2388  2         FIXED_VALUE     : REF BITVECTOR[32];    ! Bit vector of the value.
;  2281          2389  2
;  2282          2390  2
;  2283          2391  2     ! Make sure FIXED_LENGTH is a valid length.
;  2284          2392  2     !
;  2285          2393  2     FIXED_LENGTH = .FIXED_DESC[DSC$W_LENGTH];
;  2286          2394  3     IF (.FIXED_LENGTH NEQ 1) AND (.FIXED_LENGTH NEQ 2) AND (.FIXED_LENGTH NEQ 4)
;  2287          2395  2     THEN
;  2288          2396  2         $DBG_ERROR('DBGLANGOP\DBG$NORMALIZE_FIXED, invalid scaled binary size');
;  2289          2397  2
;  2290          2398  2     ! Do the normalization.
;  2291          2399  2     !
;  2292          2400  2     FIXED_LENGTH = .FIXED_LENGTH * 8;
;  2293          2401  2     FIXED_VALUE = .FIXED_DESC[DSC$A_POINTER];
;  2294          2402  2     IF ..FIXED_VALUE EQL 0
;  2295          2403  2     THEN
;  2296          2404  2         FIXED_DESC[DSC$B_SCALE] = 0
;  2297          2405  2     ELSE
;  2298          2406  2         WHILE NOT .FIXED_VALUE[0] DO
;  2299          2407  3             BEGIN
;  2300          2408  3             .FIXED_VALUE = ..FIXED_VALUE ^ -1;
;  2301          2409  3             FIXED_DESC[DSC$B_SCALE] = .FIXED_DESC[DSC$B_SCALE] + 1;
;  2302          2410  2             END;
;  2303          2411  1     END;


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

24  47  42  44  5C  50  4F  47  4E  41  4C  47  42  44  39  0033B P.AAX:  .ASCII  \9DBGLANGOP\<92>\DBG$NORMALIZE_FIXED, in\   ;
44  45  58  49  46  5F  45  5A  49  4C  41  4D  52  4F  4E  0034A                                                            ;
```

```
                                                  6E 69 20 2C  00359
69 62 20 64 65 6C 61 63 73 20 64  69 6C 61 76  00035D          .ASCII  \valid scaled binary size\                      ;
                  65 7A 69 73 20  79 72 61 6E  00036C                                                                 ;


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

                                                 000C 00000     .ENTRY  DBG$NORMALIZE_FIXED, Save R2,R3                 ; 2362
                                52        04   AC   D0 00002     MOVL    FIXED_DESC, R2                                  ; 2393
                                53             62   3C 00006     MOVZWL  (R2), FIXED_LENGTH                              ;
                                01             53   D1 00009     CMPL    FIXED_LENGTH, #1                                ; 2394
                                               1F   13 0000C     BEQL    1$                                              ;
                                02             53   D1 0000E     CMPL    FIXED_LENGTH, #2                                ;
                                               1A   13 00011     BEQL    1$                                              ;
                                04             53   D1 00013     CMPL    FIXED_LENGTH, #4                                ;
                                               15   13 00016     BEQL    1$                                              ;
                                    00000000'  EF   9F 00018     PUSHAB  P.AAX                                           ; 2396
                                               01   DD 0001E     PUSHL   #1                                              ;
                                    00028362   8F   DD 00020     PUSHL   #164706                                         ;
                         00000000G  00             03   FB 00026     CALLS   #3, LIB$SIGNAL                          ;
                                53             08   C4 0002D  1$:  MULL2   #8, FIXED_LENGTH                           ; 2400
                                50        04   A2   D0 00030     MOVL    4(R2), FIXED_VALUE                              ; 2401
                                               60   D5 00034     TSTL    (FIXED_VALUE)                                   ; 2402
                                               04   12 00036     BNEQ    2$                                              ;
                                          08   A2   94 00038     CLRB    8(R2)                                           ; 2404
                                               04   0003B     RET                                                     ;
                                0A             60   E8 0003C  2$:  BLBS    (FIXED_VALUE), 3$                           ; 2406
                      60        60        FF   8F   78 0003F     ASHL    #-1, (FIXED_VALUE), (FIXED_VALUE)               ; 2408
                                          08   A2   96 00044     INCB    8(R2)                                           ; 2409
                                               F3   11 00047     BRB     2$                                              ; 2406
                                               04   00049  3$:  RET                                                     ; 2411

; Routine Size:  74 bytes,    Routine Base:  DBG$CODE + 1285
```

```
;  2305          2412  1 GLOBAL ROUTINE DBG$PRED_ENUM (ARG_DESC, RESULT_DESC): NOVALUE =
;  2306          2413  1 !
;  2307          2414  1 ! FUNCTION
;  2308          2415  1 !
;  2309          2416  1 !       This routine is called to perform the PRED built-in function
;  2310          2417  1 !       or 'PRED function for Ada on an enumerated type variable.
;  2311          2418  1 !
;  2312          2419  1 ! INPUTS
;  2313          2420  1 !
;  2314          2421  1 !       ARG_DESC        - points to the value descriptor representing the
;  2315          2422  1 !                         argument of the PRED built-in function.
;  2316          2423  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  2317          2424  1 !                         of the PRED built-in function operator.
;  2318          2425  1 !
;  2319          2426  1 ! OUTPUTS
;  2320          2427  1 !
;  2321          2428  1 !       The result value descriptor is filled in.
;  2322          2429  1 !       No value is returned.
;  2323          2430  1 !
;  2324          2431  2     BEGIN
;  2325          2432  2     MAP
;  2326          2433  2         ARG_DESC        : REF DBG$VALDESC,
;  2327          2434  2         RESULT_DESC     : REF DBG$VALDESC;
;  2328          2435  2
;  2329          2436  2     LOCAL
;  2330          2437  2         COMPLIST        : REF VECTOR [,LONG],   ! Type component list in the RST
;  2331          2438  2         COMPCOUNT,                              ! Number of components in list
;  2332          2439  2         DST_VALUE       : VECTOR [3,LONG],      ! Value contained in DST
;  2333          2440  2         DUMMY,                                  ! Dummy for DBG$STA_SYMVALUE
;  2334          2441  2         INDEX,                                  ! Index for component search loop
;  2335          2442  2         FCODE,                                  ! FCODE for argument
;  2336          2443  2         TYPEID          : REF RST$ENTRY;        ! Points to a TYPEID
;  2337          2444  2
;  2338          2445  2     ! Obtain a typeid and fcode for the argument.
;  2339          2446  2     !
;  2340          2447  2     DBG$STA_SYMTYPE(.ARG_DESC[DBG$L_DHDR_TYPEID], FCODE, TYPEID);
;  2341          2448  2
;  2342          2449  2     ! Obtain enumeration type info.
;  2343          2450  2     !
;  2344          2451  2     DBG$STA_TYP_ENUM(.TYPEID, COMPCOUNT, COMPLIST, DUMMY);
;  2345          2452  2
;  2346          2453  2     ! Now perform the PRED operation by finding the current enumeration
;  2347          2454  2     ! element and from there finding the value of the previous element and
;  2348          2455  2     ! moving the new value into RESULT_DESC.
;  2349          2456  2     !
;  2350          2457  2     RESULT_DESC[DBG$L_DHDR_TYPEID] = .ARG_DESC[DBG$L_DHDR_TYPEID];
;  2351          2458  2     INDEX = 0;
;  2352          2459  2     WHILE .INDEX LEQ .COMPCOUNT-1 DO
;  2353          2460  3         BEGIN
;  2354          2461  3
;  2355          2462  3         ! Compare the values to see if this is the current enumerated element
;  2356          2463  3         ! Must go down into the DST to get the value.
;  2357          2464  3         !
;  2358          2465  3         DBG$STA_SYMVALUE(.COMPLIST[.INDEX], DST_VALUE[0], DUMMY);
;  2359          2466  3         IF .ARG_DESC[DBG$L_VALUE_VALUE0] EQL ..DST_VALUE[0]
;  2360          2467  3         THEN
;  2361          2468  3             EXITLOOP;
```

```
;  2362          2469  3          INDEX = .INDEX + 1;
;  2363          2470  2          END;
;  2364          2471  2
;  2365          2472  2      IF .INDEX GEQ .COMPCOUNT
;  2366          2473  2      THEN
;  2367          2474  2          $DBG_ERROR ('DBGLANGOP\DBG$PRED_ENUM component not found in the RST');
;  2368          2475  2
;  2369          2476  2      IF .INDEX GTR 0
;  2370          2477  2      THEN
;  2371          2478  3          BEGIN
;  2372          2479  3          DBG$STA_SYMVALUE(.COMPLIST[.INDEX-1], DST_VALUE[0], DUMMY);
;  2373          2480  3          RESULT_DESC[DBG$L_VALUE_VALUE0] = ..DST_VALUE[0];
;  2374          2481  3          RESULT_DESC[DBG$L_DHDR_SYMID0] = .COMPLIST[.INDEX-1];
;  2375          2482  3          END
;  2376          2483  2      ELSE
;  2377          2484  2
;  2378          2485  2          ! If enumeration is out of range, signal the error here.
;  2379          2486  2          !
;  2380          2487  2          SIGNAL(DBG$_ILLENUMVAL);
;  2381          2488  2
;  2382          2489  1      END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 36  00375 P.AAY:  .ASCII  \6DBGLANGOP\<92>\DBG$PRED_ENUM component\   ;
6F 70 6D 6F 63 20 4D 55 4E 45 5F 44 45 52 50  00384                                                          ;
                                  74 6E 65 6E  00393                                                          ;
74 20 6E 69 20 64 6E 75 6F 66 20 74 6F 6E 20  00397          .ASCII  \ not found in the RST\                 ;
                              54 53 52 20 65 68  003A6                                                       ;


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

                                  00FC 00000          .ENTRY  DBG$PRED_ENUM, Save R2,R3,R4,R5,R6,R7       ; 2412
                 57 00000000G  00   9E 00002          MOVAB   LIB$SIGNAL, R7                              ;
                 56 00000000G  00   9E 00009          MOVAB   DBG$STA_SYMVALUE, R6                        ;
                 5E                20   C2 00010      SUBL2   #32, SP                                     ;
                                   5E   DD 00013      PUSHL   SP                                          ; 2447
                           08      AE   9F 00015      PUSHAB  FCODE                                       ;
                 54        04      AC   D0 00018      MOVL    ARG_DESC, R4                                ;
                           08      A4   DD 0001C      PUSHL   8(R4)                                       ;
     00000000G   00                03   FB 0001F      CALLS   #3, DBG$STA_SYMTYPE                         ;
                           10      AE   9F 00026      PUSHAB  DUMMY                                       ; 2451
                           0C      AE   9F 00029      PUSHAB  COMPLIST                                    ;
                           14      AE   9F 0002C      PUSHAB  COMPCOUNT                                   ;
                           0C      AE   DD 0002F      PUSHL   TYPEID                                      ;
     00000000G   00                04   FB 00032      CALLS   #4, DBG$STA_TYP_ENUM                        ;
                 53        08      AC   D0 00039      MOVL    RESULT_DESC, R3                             ; 2457
           08    A3        08      A4   D0 0003D      MOVL    8(R4), 8(R3)                                ;
                                   52   D4 00042      CLRL    INDEX                                       ; 2458
55         0C   AE                01   C3 00044      SUBL3   #1, COMPCOUNT, R5                           ; 2459
                 55                52   D1 00047 1$:  CMPL    INDEX, R5                                   ;
                                   18   14 0004C      BGTR    2$                                          ;
                           10      AE   9F 0004E      PUSHAB  DUMMY                                       ; 2465
```

```
                        18   AE  9F 00051          PUSHAB  DST_VALUE                                ;
                        10 BE42  DD 00054          PUSHL   @COMPLIST[INDEX]                        ;
                66              03  FB 00058          CALLS   #3, DBG$STA_SYMVALUE                    ;
          14    BE        20    A4  D1 0005B          CMPL    32(R4), @DST_VALUE                      ; 2466
                                04  13 00060          BEQL    2$                                      ;
                                52  D6 00062          INCL    INDEX                                   ; 2469
                                E3  11 00064          BRB     1$                                      ; 2459
          0C    AE              52  D1 00066 2$:      CMPL    INDEX, COMPCOUNT                        ; 2472
                                11  19 0006A          BLSS    3$                                      ;
                00000000'       EF  9F 0006C          PUSHAB  P.AAY                                   ; 2474
                                01  DD 00072          PUSHL   #1                                      ;
                00028362        8F  DD 00074          PUSHL   #164706                                 ;
                67              03  FB 0007A          CALLS   #3, LIB$SIGNAL                          ;
                                52  D5 0007D 3$:      TSTL    INDEX                                   ; 2476
                                1C  15 0007F          BLEQ    4$                                      ;
                        10   AE  9F 00081          PUSHAB  DUMMY                                   ; 2479
                        18   AE  9F 00084          PUSHAB  DST_VALUE                               ;
                52      10 BE42  DE 00087          MOVAL   @COMPLIST[INDEX], R2                    ;
                        FC   A2  DD 0008C          PUSHL   -4(R2)                                  ;
                66              03  FB 0008F          CALLS   #3, DBG$STA_SYMVALUE                    ;
          20    A3        14    BE  D0 00092          MOVL    @DST_VALUE, 32(R3)                      ; 2480
          0C    A3        FC    A2  D0 00097          MOVL    -4(R2), 12(R3)                          ; 2481
                                04  00 0009C          RET                                             ; 2476
                00028840        8F  DD 0009D 4$:      PUSHL   #165952                                 ; 2487
                67              01  FB 000A3          CALLS   #1, LIB$SIGNAL                          ;
                                04  00 000A6          RET                                             ; 2489

; Routine Size:  167 bytes,    Routine Base:  DBG$CODE + 12FF
```

```
;2384     2490  1 GLOBAL ROUTINE DBG$SUB_FIXED_FIXED (ARG_DESC1, ARG_DESC2, RESULT_DESC): NOVALUE =
;2385     2491  1 !
;2386     2492  1 ! FUNCTION
;2387     2493  1 !
;2388     2494  1 !       This routine is called to perform the subtract operation
;2389     2495  1 !       on a scaled binary variable.
;2390     2496  1 !
;2391     2497  1 ! INPUTS
;2392     2498  1 !
;2393     2499  1 !       ARG_DESC1       - points to the value descriptor representing the
;2394     2500  1 !                         left argument of the operation.
;2395     2501  1 !       ARG_DESC2       - points to the value descriptor representing the
;2396     2502  1 !                         right argument of the operation.
;2397     2503  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;2398     2504  1 !                         of the operation.
;2399     2505  1 !
;2400     2506  1 ! OUTPUTS
;2401     2507  1 !
;2402     2508  1 !       The result value descriptor is filled in.
;2403     2509  1 !       No value is returned.
;2404     2510  1 !
;2405     2511  2     BEGIN
;2406     2512  2
;2407     2513  2     MAP
;2408     2514  2         ARG_DESC1       : REF DBG$VALDESC,
;2409     2515  2         ARG_DESC2       : REF DBG$VALDESC,
;2410     2516  2         RESULT_DESC     : REF DBG$VALDESC;
;2411     2517  2
;2412     2518  2     LOCAL
;2413     2519  2         RESULT_VALUE,
;2414     2520  2         SCALE,
;2415     2521  2         VAL_DESC1       : DBG$STG_DESC,
;2416     2522  2         VAL_DESC2       : DBG$STG_DESC,
;2417     2523  2         VALUE1,
;2418     2524  2         VALUE2;
;2419     2525  2
;2420     2526  2     ! Set up working variables.  This way we don't mess up anything important.
;2421     2527  2     !
;2422     2528  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC1[DBG$A_VALUE_VMSDESC], VAL_DESC1);
;2423     2529  2     CH$MOVE(DBG$K_STG_DESC_SIZE, ARG_DESC2[DBG$A_VALUE_VMSDESC], VAL_DESC2);
;2424     2530  2
;2425     2531  2     VALUE1 = ..ARG_DESC1[DBG$L_VALUE_POINTER];
;2426     2532  2     VALUE2 = ..ARG_DESC2[DBG$L_VALUE_POINTER];
;2427     2533  2     VAL_DESC1[DSC$A_POINTER] = VALUE1;
;2428     2534  2     VAL_DESC2[DSC$A_POINTER] = VALUE2;
;2429     2535  2
;2430     2536  2     DBG$NORMALIZE_FIXED(VAL_DESC1);
;2431     2537  2     DBG$NORMALIZE_FIXED(VAL_DESC2);
;2432     2538  2
;2433     2539  2     MATCH_FIXED_BINARYS(VAL_DESC1, VAL_DESC2);
;2434     2540  2
;2435     2541  2     ! Do the Subtraction.
;2436     2542  2     !
;2437     2543  2     RESULT_VALUE = .VALUE1 - .VALUE2;
;2438     2544  2     SCALE = .VAL_DESC1[DSC$B_SCALE];
;2439     2545  2
;2440     2546  2     ! Has an overflow occured?
```

```
;  2441          2547  2      !
;  2442          2548  2      IF .VALUE1<31, 1, 0> NEQ .VALUE2<31, 1, 0> AND
;  2443          2549  2         .RESULT_VALUE<31, 1, 0> EQL .VALUE2<31, 1, 0>
;  2444          2550  2      THEN
;  2445          2551  3          BEGIN
;  2446          2552  3          IF .RESULT_VALUE<0, 1, 0>
;  2447          2553  3          THEN
;  2448          2554  3              SIGNAL(DBG$_IFIXUND);
;  2449          2555  3          RESULT_VALUE = .RESULT_VALUE ^ -1;
;  2450          2556  3          SCALE = .SCALE + 1;
;  2451          2557  3          RESULT_VALUE<31, 1, 0> = .VALUE1<31, 1, 0>;
;  2452          2558  2          END;
;  2453          2559  2
;  2454          2560  2      .RESULT_DESC[DBG$L_VALUE_POINTER] = .RESULT_VALUE;
;  2455          2561  2      RESULT_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  2456          2562  2      RESULT_DESC[DBG$B_VALUE_SCALE] = .SCALE;
;  2457          2563  2
;  2458          2564  1      END;


                                        00FC 00000          .ENTRY  DBG$SUB_FIXED_FIXED, Save R2,R3,R4,R5,R6,R7 ; 2490
                                  5E        20  C2 00002      SUBL2   #32, SP                                     ;
                                  57    04  AC  D0 00005      MOVL    ARG_DESC1, R7                               ; 2528
                                                0C  28 00009      MOVC3   #12, 20(R7), VAL_DESC1                  ;
                                  56    08  AC  D0 0000F      MOVL    ARG_DESC2, R6                               ; 2529
                                                0C  28 00013      MOVC3   #12, 20(R6), VAL_DESC2                  ;
                                        6E  18  B7  D0 00019      MOVL    @24(R7), VALUE1                             ; 2531
                                  04    AE  18  B6  D0 0001D      MOVL    @24(R6), VALUE2                             ; 2532
                                  18    AE      6E  9E 00022      MOVAB   VALUE1, VAL_DESC1+4                         ; 2533
                                  0C    AE  04  AE  9E 00026      MOVAB   VALUE2, VAL_DESC2+4                         ; 2534
                                            14  AE  9F 0002B      PUSHAB  VAL_DESC1                                   ; 2536
                                FEDC    CF      01  FB 0002E      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                            08  AE  9F 00033      PUSHAB  VAL_DESC2                                   ; 2537
                                FED4    CF      01  FB 00036      CALLS   #1, DBG$NORMALIZE_FIXED                     ;
                                            08  AE  9F 0003B      PUSHAB  VAL_DESC2                                   ; 2539
                                            18  AE  9F 0003E      PUSHAB  VAL_DESC1                                   ;
                               0000V    CF      02  FB 00041      CALLS   #2, MATCH_FIXED_BINARYS                     ;
                          52                6E  04  AE  C3 00046      SUBL3   VALUE2, VALUE1, RESULT_VALUE            ; 2543
                                            53  1C  AE  98 0004B      CVTBL   VAL_DESC1+8, SCALE                      ; 2544
                          50        07      AE  03  AE  8D 0004F      XORB3   VALUE1+3, VALUE2+3, R0                  ; 2548
                                                    2F  18 00055      BGEQ    2$                                  ;
50          07    AE                  01              07  EF 00057      EXTZV   #7, #1, VALUE2+3, R0                    ; 2549
50                52                  01              1F  ED 0005D      CMPZV   #31, #1, RESULT_VALUE, R0           ;
                                                    22  12 00062      BNEQ    2$                                  ;
                                            0D          52  E9 00064      BLBC    RESULT_VALUE, 1$                    ; 2552
                                                0002878B    8F  DD 00067      PUSHL   #165771                             ; 2554
                                00000000G   00              01  FB 0006D      CALLS   #1, LIB$SIGNAL                      ;
                          52                52        FF    8F  78 00074 1$:    ASHL    #-1, RESULT_VALUE, RESULT_VALUE     ; 2555
                                                        53  D6 00079      INCL    SCALE                               ; 2556
50          03    AE                  01              07  EF 0007B      EXTZV   #7, #1, VALUE1+3, R0                    ; 2557
52                01                  1F              50  F0 00081      INSV    R0, #31, #1, RESULT_VALUE           ;
                                            50  0C  AC  D0 00086 2$:    MOVL    RESULT_DESC, R0                     ; 2560
                                  18    B0              52  D0 0008A      MOVL    RESULT_VALUE, @24(R0)               ;
                                  16    A0              08  90 0008E      MOVB    #8, 22(R0)                          ; 2561
```

```
                1C   A0          53  90 00092         MOVB    SCALE, 28(R0)                   ; 2562
                                     04 00096         RET                                     ; 2564

; Routine Size:  151 bytes,    Routine Base:  DBG$CODE + 13A6
```

```
;2460   2565  1 GLOBAL ROUTINE DBG$SUCC_ENUM (ARG_DESC, RESULT_DESC): NOVALUE =
;2461   2566  1 !
;2462   2567  1 ! FUNCTION
;2463   2568  1 !
;2464   2569  1 !       This routine is called to perform the SUCC built-in function
;2465   2570  1 !       or 'SUCC function for Ada on an enumerated type variable.
;2466   2571  1 !
;2467   2572  1 ! INPUTS
;2468   2573  1 !
;2469   2574  1 !       ARG_DESC        - points to the value descriptor representing the
;2470   2575  1 !                         argument of the SUCC built-in function.
;2471   2576  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;2472   2577  1 !                         of the SUCC built-in function operator.
;2473   2578  1 !
;2474   2579  1 ! OUTPUTS
;2475   2580  1 !
;2476   2581  1 !       The result value descriptor is filled in.
;2477   2582  1 !       No value is returned.
;2478   2583  1 !
;2479   2584  2     BEGIN
;2480   2585  2     MAP
;2481   2586  2         ARG_DESC        : REF DBG$VALDESC,
;2482   2587  2         RESULT_DESC     : REF DBG$VALDESC;
;2483   2588  2
;2484   2589  2     LOCAL
;2485   2590  2         COMPLIST        : REF VECTOR [,LONG],   ! Type component list in the RST
;2486   2591  2         COMPCOUNT,                              ! Number of components in list
;2487   2592  2         DST_VALUE       : VECTOR [3,LONG],      ! Value contained in DST
;2488   2593  2         DUMMY,                                  ! Dummy for DBG$STA_XXXXX
;2489   2594  2         INDEX,                                  ! Index for component search loop
;2490   2595  2         FCODE,                                  ! FCODE for argument
;2491   2596  2         TYPEID          : REF RST$ENTRY;        ! Points to a TYPEID
;2492   2597  2
;2493   2598  2     ! Obtain a typeid and fcode for the argument.
;2494   2599  2     !
;2495   2600  2     DBG$STA_SYMTYPE(.ARG_DESC[DBG$L_DHDR_TYPEID], FCODE, TYPEID);
;2496   2601  2
;2497   2602  2     ! Obtain enumeration type info.
;2498   2603  2     !
;2499   2604  2     DBG$STA_TYP_ENUM(.TYPEID, COMPCOUNT, COMPLIST, DUMMY);
;2500   2605  2
;2501   2606  2     ! Now perform the SUCC operation by finding the current enumeration
;2502   2607  2     ! element and from there finding the value of the next element and moving
;2503   2608  2     ! the new value into RESULT_DESC.
;2504   2609  2     !
;2505   2610  2     RESULT_DESC[DBG$L_DHDR_TYPEID] = .ARG_DESC[DBG$L_DHDR_TYPEID];
;2506   2611  2     INDEX = 0;
;2507   2612  2     WHILE .INDEX LEQ .COMPCOUNT-1 DO
;2508   2613  3         BEGIN
;2509   2614  3
;2510   2615  3         ! Compare the values to see if this is the current enumerated element
;2511   2616  3         ! Must go down into the DST to get the value.
;2512   2617  3         !
;2513   2618  3         DBG$STA_SYMVALUE(.COMPLIST[.INDEX], DST_VALUE[0], DUMMY);
;2514   2619  3         IF .ARG_DESC[DBG$L_VALUE_VALUE0] EQL ..DST_VALUE[0]
;2515   2620  3         THEN
;2516   2621  3             EXITLOOP;
```

```
;  2517          2622  3          INDEX = .INDEX + 1;
;  2518          2623  2          END;
;  2519          2624  2
;  2520          2625  2      IF .INDEX GEQ .COMPCOUNT
;  2521          2626  2      THEN
;  2522          2627  2          $DBG_ERROR ('DBGLANGOP\DBG$SUCC_ENUM component not found in the RST');
;  2523          2628  2
;  2524          2629  2      IF .INDEX LSS .COMPCOUNT-1
;  2525          2630  2      THEN
;  2526          2631  3          BEGIN
;  2527          2632  3          DBG$STA_SYMVALUE(.COMPLIST[.INDEX+1], DST_VALUE[0], DUMMY);
;  2528          2633  3          RESULT_DESC[DBG$L_VALUE_VALUE0] = ..DST_VALUE[0];
;  2529          2634  3          RESULT_DESC[DBG$L_DHDR_SYMID0] = .COMPLIST[.INDEX+1];
;  2530          2635  3          END
;  2531          2636  2      ELSE
;  2532          2637  2
;  2533          2638  2          ! If enumeration is out of range, signal the error here.
;  2534          2639  2          !
;  2535          2640  2          SIGNAL(DBG$_ILLENUMVAL);
;  2536          2641  2
;  2537          2642  1      END;


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

24 47 42 44 5C 50 4F 47 4E 41 4C 47 42 44 36  003AC P.AAZ:    .ASCII  \6DBGLANGOP\<92>\DBG$SUCC_ENUM component\   ;
6F 70 6D 6F 63 20 4D 55 4E 45 5F 43 43 55 53  003BB                                                          ;
                                  74 6E 65 6E  003CA                                                          ;
74 20 6E 69 20 64 6E 75 6F 66 20 74 6F 6E 20  003CE            .ASCII  \ not found in the RST\                  ;
                        54 53 52 20 65 68     003DD                                                          ;


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

                                00FC 00000          .ENTRY  DBG$SUCC_ENUM, Save R2,R3,R4,R5,R6,R7           ; 2565
                    57 00000000G   00   9E 00002    MOVAB   LIB$SIGNAL, R7                                  ;
                    56 00000000G   00   9E 00009    MOVAB   DBG$STA_SYMVALUE, R6                            ;
                    5E                  20   C2 00010    SUBL2   #32, SP                                     ;
                                        5E   DD 00013    PUSHL   SP                                          ; 2600
                               08       AE   9F 00015    PUSHAB  FCODE                                       ;
                    54         04       AC   D0 00018    MOVL    ARG_DESC, R4                                ;
                               08       A4   DD 0001C    PUSHL   8(R4)                                       ;
    00000000G       00                  03   FB 0001F    CALLS   #3, DBG$STA_SYMTYPE                         ;
                               10       AE   9F 00026    PUSHAB  DUMMY                                       ; 2604
                               0C       AE   9F 00029    PUSHAB  COMPLIST                                    ;
                               14       AE   9F 0002C    PUSHAB  COMPCOUNT                                   ;
                               0C       AE   DD 0002F    PUSHL   TYPEID                                      ;
    00000000G       00                  04   FB 00032    CALLS   #4, DBG$STA_TYP_ENUM                        ;
                    53         08       AC   D0 00039    MOVL    RESULT_DESC, R3                             ; 2610
          08        A3         08       A4   D0 0003D    MOVL    8(R4), 8(R3)                                ;
                                        52   D4 00042    CLRL    INDEX                                       ; 2611
55        0C        AE                  01   C3 00044    SUBL3   #1, COMPCOUNT, R5                           ; 2612
                    55                  52   D1 00049 1$: CMPL    INDEX, R5                                   ;
                                        18   14 0004C    BGTR    2$                                          ;
                               10       AE   9F 0004E    PUSHAB  DUMMY                                       ; 2618
```

```
                        18   AE  9F 00051          PUSHAB  DST_VALUE                                ;
                        10 BE42  DD 00054          PUSHL   @COMPLIST[INDEX]                        ;
                66              03  FB 00058          CALLS   #3, DBG$STA_SYMVALUE                ;
          14    BE        20    A4  D1 0005B          CMPL    32(R4), @DST_VALUE                  ; 2619
                                04  13 00060          BEQL    2$                                  ;
                                52  D6 00062          INCL    INDEX                               ; 2622
                                E3  11 00064          BRB     1$                                  ; 2612
          0C    AE              52  D1 00066 2$:      CMPL    INDEX, COMPCOUNT                    ; 2625
                                11  19 0006A          BLSS    3$                                  ;
                00000000'       EF  9F 0006C          PUSHAB  P.AAZ                               ; 2627
                                01  DD 00072          PUSHL   #1                                  ;
                00028362        8F  DD 00074          PUSHL   #164706                             ;
                67              03  FB 0007A          CALLS   #3, LIB$SIGNAL                      ;
                55              52  D1 0007D 3$:      CMPL    INDEX, R5                           ; 2629
                                1C  18 00080          BGEQ    4$                                  ;
                        10    AE  9F 00082          PUSHAB  DUMMY                               ; 2632
                        18    AE  9F 00085          PUSHAB  DST_VALUE                           ;
                52      10 BE42  DE 00088          MOVAL   @COMPLIST[INDEX], R2                ;
                        04    A2  DD 0008D          PUSHL   4(R2)                               ;
                66              03  FB 00090          CALLS   #3, DBG$STA_SYMVALUE                ;
          20    A3        14    BE  D0 00093          MOVL    @DST_VALUE, 32(R3)                  ; 2633
          0C    A3        04    A2  D0 00098          MOVL    4(R2), 12(R3)                       ; 2634
                                04  04 0009D          RET                                         ; 2629
                00028840        8F  DD 0009E 4$:      PUSHL   #165952                             ; 2640
                67              01  FB 000A4          CALLS   #1, LIB$SIGNAL                      ;
                                04  04 000A7          RET                                         ; 2642

; Routine Size:  168 bytes,    Routine Base:  DBG$CODE + 143D
```

```
;  2539          2643  1 ROUTINE DBG$TYPEID_TO_PRIMARY (TYPEID, ADDRESS) =
;  2540          2644  1 !
;  2541          2645  1 ! FUNCTION
;  2542          2646  1 !       This routine takes a TYPEID which describes an anonymous object,
;  2543          2647  1 !       and builds a Primary Descriptor for the object.
;  2544          2648  1 !       For example, if we dereference a typed pointer in C with (*PTR)
;  2545          2649  1 !       then we can get a typeid describing the pointed-to object,
;  2546          2650  1 !       and we also know the address of the anonymous object.
;  2547          2651  1 !       This routine turns the typeid into a Primary Descriptor.
;  2548          2652  1 !       It first constructs a Primary root node and then calls
;  2549          2653  1 !       DBG$BUILD_PRIMARY_SUBNODE to create the Primary sub-node.
;  2550          2654  1 !       Finally, it stuffs the address into the RELOC field.
;  2551          2655  1 !
;  2552          2656  1 ! INPUTS
;  2553          2657  1 !       TYPEID  - TYPEID for the object
;  2554          2658  1 !       ADDRESS - A byte address for the object
;  2555          2659  1 !
;  2556          2660  1 ! OUTPUTS
;  2557          2661  1 !       A Primary Descriptor is built out of temporary memory and a
;  2558          2662  1 !       pointer to this descriptor is returned.
;  2559          2663  1 !
;  2560          2664  2     BEGIN
;  2561          2665  2
;  2562          2666  2     LOCAL
;  2563          2667  2         FCODE,                          ! Fcode for this Primary
;  2564          2668  2         NODEPTR: REF DBG$PRIM_NODE,     ! Pointer to a Primary Subnode
;  2565          2669  2         PRIMPTR: REF DBG$PRIMARY,       ! Pointer to a Primary Descriptor
;  2566          2670  2         RSTPTR;                         ! An rstptr (either a symid or a typeid)
;  2567          2671  2
;  2568          2672  2
;  2569          2673  2     ! Obtain the fcode from the typeid.
;  2570          2674  2     !
;  2571          2675  2     FCODE = DBG$STA_TYPEFCODE (.TYPEID);
;  2572          2676  2
;  2573          2677  2     ! Allocate space for the Primary and fill in some of the header fields.
;  2574          2678  2     !
;  2575          2679  2     PRIMPTR = DBG$GET_TEMPMEM (DBG$K_PRIMARY_SIZE);
;  2576          2680  2     PRIMPTR[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
;  2577          2681  2     PRIMPTR[DBG$B_DHDR_TYPE] = DBG$K_PRIMARY_DESC;
;  2578          2682  2     PRIMPTR[DBG$W_DHDR_LENGTH] = DBG$K_PRIMARY_SIZE*%UPVAL;
;  2579          2683  2     PRIMPTR[DBG$B_DHDR_KIND] = RST$K_DATA;
;  2580          2684  2     PRIMPTR[DBG$B_DHDR_FCODE] = .FCODE;
;  2581          2685  2     PRIMPTR[DBG$L_DHDR_TYPEID] = .TYPEID;
;  2582          2686  2     PRIMPTR[DBG$L_DHDR_SYMID0] = 0;
;  2583          2687  2     PRIMPTR[DBG$L_PRIM_FLINK] = PRIMPTR[DBG$A_PRIM_FLINK];
;  2584          2688  2     PRIMPTR[DBG$L_PRIM_BLINK] = PRIMPTR[DBG$A_PRIM_FLINK];
;  2585          2689  2
;  2586          2690  2     ! Call BUILD_PRIMARY_SUBNODE to build a subnode and fill in all of
;  2587          2691  2     ! the subnode information. Note that this routine also fills in
;  2588          2692  2     ! SYMID, KIND, FCODE, and TYPEID for the root node, so we do not have
;  2589          2693  2     ! to do that here.
;  2590          2694  2     !
;  2591          2695  2     DBG$BUILD_PRIMARY_SUBNODE (.PRIMPTR, RST$K_DATA, 0, .FCODE, .TYPEID, 0);
;  2592          2696  2
;  2593          2697  2     ! We already know the address of the object described by the Primary.
;  2594          2698  2     ! This address is put in the "RELOC" field so that PRIM_TO_VAL
;  2595          2699  2     ! can determine the address of this object.
```

```
;  2596          2700  2     !
;  2597          2701  2     NODEPTR = .PRIMPTR[DBG$L_PRIM_BLINK];
;  2598          2702  2     NODEPTR[DBG$L_PNODE_RELOC] = .ADDRESS;
;  2599          2703  2
;  2600          2704  2     ! Return a pointer to a Primary.
;  2601          2705  2     !
;  2602          2706  2     RETURN .PRIMPTR;
;  2603          2707  1     END;


                                        000C 00000 DBG$TYPEID_TO_PRIMARY:
                                                   .WORD   Save R2,R3                          ;  2643
                                04   AC  DD 00002  PUSHL   TYPEID                            ;  2675
00000000G   00                  01   FB  00005  CALLS   #1, DBG$STA_TYPEFCODE                ;
            53                  50   D0  0000C  MOVL    R0, FCODE                           ;
                                09   DD  0000F  PUSHL   #9                                   ;  2679
00000000G   00                  01   FB  00011  CALLS   #1, DBG$GET_TEMPMEM                  ;
            52                  50   D0  00018  MOVL    R0, PRIMPTR                         ;
      03    A2  00000000G   00  90   0001B  MOVB    DBG$GB_LANGUAGE, 3(PRIMPTR)             ;  2680
      02    A2        79    8F  90   00023  MOVB    #121, 2(PRIMPTR)                        ;  2681
            62              24  B0   00028  MOVW    #36, (PRIMPTR)                          ;  2682
      07    A2              06  90   0002B  MOVB    #6, 7(PRIMPTR)                          ;  2683
      06    A2              53  90   0002F  MOVB    FCODE, 6(PRIMPTR)                       ;  2684
      08    A2        04    AC  D0   00033  MOVL    TYPEID, 8(PRIMPTR)                      ;  2685
                      0C    A2  D4   00038  CLRL    12(PRIMPTR)                             ;  2686
      14    A2        14    A2  9E   0003B  MOVAB   20(PRIMPTR), 20(PRIMPTR)                ;  2687
      18    A2        14    A2  9E   00040  MOVAB   20(PRIMPTR), 24(PRIMPTR)                ;  2688
                              7E  D4   00045  CLRL    -(SP)                                   ;  2695
                      04      AC  DD   00047  PUSHL   TYPEID                                  ;
                              53  DD   0004A  PUSHL   FCODE                                   ;
                  7E                  06   7D  0004C  MOVQ    #6, -(SP)                       ;
                              52  DD   0004F  PUSHL   PRIMPTR                                 ;
          00000000G   00          06   FB  00051  CALLS   #6, DBG$BUILD_PRIMARY_SUBNODE       ;
                      50      18  A2   D0  00058  MOVL    24(PRIMPTR), NODEPTR                ;  2701
            14    A0        08  AC   D0  0005C  MOVL    ADDRESS, 20(NODEPTR)                ;  2702
                      50          52   D0  00061  MOVL    PRIMPTR, R0                         ;  2706
                                  04   00064  RET                                            ;  2707

; Routine Size:  101 bytes,    Routine Base:  DBG$CODE + 14E5
```

```
;  2605          2708  1 GLOBAL ROUTINE DBG$UNARY_MINUS_FIXED (ARG_DESC, RESULT_DESC): NOVALUE =
;  2606          2709  1 !
;  2607          2710  1 ! FUNCTION
;  2608          2711  1 !
;  2609          2712  1 !       This routine is called to perform the unary minus operation
;  2610          2713  1 !       on a scaled binary variable.
;  2611          2714  1 !
;  2612          2715  1 ! INPUTS
;  2613          2716  1 !
;  2614          2717  1 !       ARG_DESC        - points to the value descriptor representing the
;  2615          2718  1 !                         argument of the operation.
;  2616          2719  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  2617          2720  1 !                         of the operation.
;  2618          2721  1 !
;  2619          2722  1 ! OUTPUTS
;  2620          2723  1 !
;  2621          2724  1 !       The result value descriptor is filled in.
;  2622          2725  1 !       No value is returned.
;  2623          2726  1 !
;  2624          2727  2     BEGIN
;  2625          2728  2
;  2626          2729  2     MAP
;  2627          2730  2         RESULT_DESC     : REF DBG$VALDESC,
;  2628          2731  2         ARG_DESC        : REF DBG$VALDESC;
;  2629          2732  2
;  2630          2733  2
;  2631          2734  2     .RESULT_DESC[DBG$L_VALUE_POINTER] = 0 - ..ARG_DESC[DBG$L_VALUE_POINTER];
;  2632          2735  2     RESULT_DESC[DBG$B_VALUE_SCALE] = .ARG_DESC[DBG$B_VALUE_SCALE];
;  2633          2736  2     RESULT_DESC[DBG$B_VALUE_DTYPE] = .ARG_DESC[DBG$B_VALUE_DTYPE];
;  2634          2737  1     END;

                                        0000 00000          .ENTRY  DBG$UNARY_MINUS_FIXED, Save nothing          ; 2708
                    50      08  AC      D0 00002          MOVL    RESULT_DESC, R0                              ; 2734
                    51      04  AC      D0 00006          MOVL    ARG_DESC, R1                                 ;
                18  B0      18  B1      CE 0000A          MNEGL   @24(R1), @24(R0)                             ;
                1C  A0      1C  A1      90 0000F          MOVB    28(R1), 28(R0)                               ; 2735
                16  A0      16  A1      90 00014          MOVB    22(R1), 22(R0)                               ; 2736
                                        04 00019          RET                                                  ; 2737

; Routine Size:  26 bytes,    Routine Base:  DBG$CODE + 154A
```

```
;  2636          2738  1 GLOBAL ROUTINE DBG$UNARY_PLUS_FIXED (ARG_DESC, RESULT_DESC): NOVALUE =
;  2637          2739  1 !
;  2638          2740  1 ! FUNCTION
;  2639          2741  1 !
;  2640          2742  1 !       This routine is called to perform the unary plus operation
;  2641          2743  1 !       on a scaled binary variable.
;  2642          2744  1 !
;  2643          2745  1 ! INPUTS
;  2644          2746  1 !
;  2645          2747  1 !       ARG_DESC        - points to the value descriptor representing the
;  2646          2748  1 !                         argument of the operation.
;  2647          2749  1 !       RESULT_DESC     - points to the value descriptor representing the result.
;  2648          2750  1 !                         of the operation.
;  2649          2751  1 !
;  2650          2752  1 ! OUTPUTS
;  2651          2753  1 !
;  2652          2754  1 !       The result value descriptor is filled in.
;  2653          2755  1 !       No value is returned.
;  2654          2756  1 !
;  2655          2757  2     BEGIN
;  2656          2758  2
;  2657          2759  2     MAP
;  2658          2760  2         RESULT_DESC     : REF DBG$VALDESC,
;  2659          2761  2         ARG_DESC        : REF DBG$VALDESC;
;  2660          2762  2
;  2661          2763  2
;  2662          2764  2     RESULT_DESC[DBG$L_VALUE_POINTER] = .ARG_DESC[DBG$L_VALUE_POINTER];
;  2663          2765  2     RESULT_DESC[DBG$B_VALUE_SCALE] = .ARG_DESC[DBG$B_VALUE_SCALE];
;  2664          2766  2     RESULT_DESC[DBG$B_VALUE_DTYPE] = .ARG_DESC[DBG$B_VALUE_DTYPE];
;  2665          2767  1     END;


                                        0000 00000          .ENTRY  DBG$UNARY_PLUS_FIXED, Save nothing          ; 2738
                          50      04    AC   7D 00002       MOVQ    ARG_DESC, R0                                ; 2764
                    18    A1      18    A0   D0 00006       MOVL    24(R0), 24(R1)                              ;
                    1C    A1      1C    A0   90 0000B       MOVB    28(R0), 28(R1)                              ; 2765
                    16    A1      16    A0   90 00010       MOVB    22(R0), 22(R1)                              ; 2766
                                              04 00015       RET                                                 ; 2767

; Routine Size:  22 bytes,    Routine Base:  DBG$CODE + 1564
```

```
;  2667          2768  1 ROUTINE MATCH_FIXED_BINARYS(ARG_DESC1, ARG_DESC2): NOVALUE =
;  2668          2769  1 !
;  2669          2770  1 ! FUNCTION
;  2670          2771  1 !
;  2671          2772  1 !       This routine is called to match the scaling factors of the input
;  2672          2773  1 !       scaled binarys.  We do this by moving the value with the largest
;  2673          2774  1 !       scale down (or the decimal point to the left) until the scales are
;  2674          2775  1 !       equal.  Sometimes this would require shifting out the most
;  2675          2776  1 !       significant bit of that value, in this case we then shift the other
;  2676          2777  1 !       value up (to the right) to match.  This means we would be shifting
;  2677          2778  1 !       out some bits so we signal a message to that effect.
;  2678          2779  1 !
;  2679          2780  1 ! INPUTS
;  2680          2781  1 !
;  2681          2782  1 !       ARG_DESC1       - points to the VMS descriptor representing the
;  2682          2783  1 !                         first argument of the operation.
;  2683          2784  1 !       ARG_DESC2       - points to the VMS descriptor representing the
;  2684          2785  1 !                         second argument of the operation.
;  2685          2786  1 !
;  2686          2787  1 ! OUTPUTS
;  2687          2788  1 !
;  2688          2789  1 !       The VMS descriptors are altered.
;  2689          2790  1 !       No value is returned.
;  2690          2791  1 !
;  2691          2792  2     BEGIN
;  2692          2793  2
;  2693          2794  2     MAP
;  2694          2795  2         ARG_DESC1       : REF DBG$STG_DESC,
;  2695          2796  2         ARG_DESC2       : REF DBG$STG_DESC;
;  2696          2797  2
;  2697          2798  2     LOCAL
;  2698          2799  2         VAL1            : REF BITVECTOR[32],
;  2699          2800  2         VAL2            : REF BITVECTOR[32];
;  2700          2801  2
;  2701          2802  2
;  2702          2803  2     VAL1 = .ARG_DESC1[DSC$A_POINTER];
;  2703          2804  2     VAL2 = .ARG_DESC2[DSC$A_POINTER];
;  2704          2805  2
;  2705          2806  2     WHILE .ARG_DESC1[DSC$B_SCALE] GTR .ARG_DESC2[DSC$B_SCALE] DO
;  2706          2807  3         BEGIN
;  2707          2808  3         IF .VAL1[30] NEQ .VAL1[31]
;  2708          2809  3         THEN
;  2709          2810  4             BEGIN
;  2710          2811  4             SIGNAL(DBG$_IFIXUND);
;  2711          2812  4             WHILE .ARG_DESC2[DSC$B_SCALE] LSS .ARG_DESC1[DSC$B_SCALE] DO
;  2712          2813  5                 BEGIN
;  2713          2814  5                 .VAL2 = ..VAL2 ^ -1;
;  2714          2815  5                 ARG_DESC2[DSC$B_SCALE] = .ARG_DESC2[DSC$B_SCALE] + 1;
;  2715          2816  4                 END;
;  2716          2817  4             EXITLOOP;
;  2717          2818  3             END;
;  2718          2819  3         .VAL1 = ..VAL1 ^ 1;
;  2719          2820  3         ARG_DESC1[DSC$B_SCALE] = .ARG_DESC1[DSC$B_SCALE] - 1;
;  2720          2821  2         END;
;  2721          2822  2
;  2722          2823  2     WHILE .ARG_DESC2[DSC$B_SCALE] GTR .ARG_DESC1[DSC$B_SCALE] DO
;  2723          2824  3         BEGIN
```

```
;  2724          2825  3          IF .VAL2[30] NEQ .VAL2[31]
;  2725          2826  3          THEN
;  2726          2827  4              BEGIN
;  2727          2828  4              SIGNAL(DBG$_IFIXUND);
;  2728          2829  4              WHILE .ARG_DESC1[DSC$B_SCALE] LSS .ARG_DESC2[DSC$B_SCALE] DO
;  2729          2830  5                  BEGIN
;  2730          2831  5                  .VAL1 = ..VAL1 ^ -1;
;  2731          2832  5                  ARG_DESC1[DSC$B_SCALE] = .ARG_DESC1[DSC$B_SCALE] + 1;
;  2732          2833  4                  END;
;  2733          2834  4              EXITLOOP;
;  2734          2835  3              END;
;  2735          2836  3          .VAL2 = ..VAL2 ^ 1;
;  2736          2837  3          ARG_DESC2[DSC$B_SCALE] = .ARG_DESC2[DSC$B_SCALE] - 1;
;  2737          2838  2          END;
;  2738          2839  2
;  2739          2840  1      END;


                                        007C 00000 MATCH_FIXED_BINARYS:
                                                    .WORD   Save R2,R3,R4,R5,R6                 ; 2768
                              56 00000000G   00  9E 00002   MOVAB   LIB$SIGNAL, R6
                              51       04    AC  D0 00009   MOVL    ARG_DESC1, R1               ; 2803
                              55       04    A1  D0 0000D   MOVL    4(R1), VAL1
                              50       08    AC  D0 00011   MOVL    ARG_DESC2, R0               ; 2804
                              54       04    A0  D0 00015   MOVL    4(R0), VAL2
                              53       08    A1  9E 00019   MOVAB   8(R1), R3                  ; 2806
                              52       08    A0  9E 0001D   MOVAB   8(R0), R2
                              62             63  91 00021 1$:  CMPB    (R3), (R2)
                                             2A  15 00024   BLEQ    4$
                    50        65       01      1F EF 00026   EXTZV   #31, #1, (VAL1), R0       ; 2808
                    50        65       01      1E ED 0002B   CMPZV   #30, #1, (VAL1), R0
                                             17  13 00030   BEQL    3$
                                 0002878B    8F  DD 00032   PUSHL   #165771                    ; 2811
                              66             01  FB 00038   CALLS   #1, LIB$SIGNAL
                              63             62  91 0003B 2$:  CMPB    (R2), (R3)            ; 2812
                                             10  18 0003E   BGEQ    4$
                              54       64 FF 8F  78 00040   ASHL    #-1, (VAL2), (VAL2)         ; 2814
                                             62  96 00045   INCB    (R2)                        ; 2815
                                             F2  11 00047   BRB     2$                          ; 2812
                              65             02  C4 00049 3$:  MULL2   #2, (VAL1)              ; 2819
                                             63  97 0004C   DECB    (R3)                        ; 2820
                                             D1  11 0004E   BRB     1$                          ; 2806
                              63             62  91 00050 4$:  CMPB    (R2), (R3)            ; 2823
                                             2A  15 00053   BLEQ    7$
                    50        64       01      1F EF 00055   EXTZV   #31, #1, (VAL2), R0       ; 2825
                    50        64       01      1E ED 0005A   CMPZV   #30, #1, (VAL2), R0
                                             17  13 0005F   BEQL    6$
                                 0002878B    8F  DD 00061   PUSHL   #165771                    ; 2828
                              66             01  FB 00067   CALLS   #1, LIB$SIGNAL
                              62             63  91 0006A 5$:  CMPB    (R3), (R2)            ; 2829
                                             10  18 0006D   BGEQ    7$
                              65       65 FF 8F  78 0006F   ASHL    #-1, (VAL1), (VAL1)         ; 2831
                                             63  96 00074   INCB    (R3)                        ; 2832
                                             F2  11 00076   BRB     5$                          ; 2829
```

```
                        64          02  C4 00078 6$:    MULL2   #2, (VAL2)                ; 2836
                                    62  97 0007B        DECB    (R2)                      ; 2837
                                    D1  11 0007D        BRB     4$                        ; 2823
                                        04 0007F 7$:    RET                               ; 2840

; Routine Size:  128 bytes,    Routine Base:  DBG$CODE + 157A

; 2740          2841  1
; 2741          2842  0 END ELUDOM



                                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$CODE                      5626  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$PLIT                       995  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         5         0       1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                       32         0         0          7       00:00.2
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                     1545       180        11         97       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                      418         0         0         31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                      386        16         4         22       00:00.3



;                               COMMAND QUALIFIERS

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

; Size:          5626 code + 995 data bytes
; Run Time:         01:30.6
; Elapsed Time:     04:13.5
; Lines/CPU Min:     1882
; Lexemes/CPU-Min: 15141
; Memory Used:  528 pages
; Compilation Complete
```

DBGIFTHEN
LIS

DBGLANVEC
LIS

DBGLANGOP
LIS

DBGGEN
LIS

DBGLEVEL1
LIS