```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGG
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     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUUUUUUUUUUUUUUU    GGGGGGGGG
```

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  VV      VV    AAAAAA    LL          UU      UU  EEEEEEEEEE    SSSSSSSS
DDDDDDDD    BBBBBBBB      GGGGGGGG  VV      VV    AAAAAA    LL          UU      UU  EEEEEEEEEE    SSSSSSSS
DD      DD  BB      BB  GG          VV      VV   AA      AA  LL          UU      UU  EE            SS
DD      DD  BB      BB  GG          VV      VV   AA      AA  LL          UU      UU  EE            SS
DD      DD  BB      BB  GG          VV      VV   AA      AA  LL          UU      UU  EE            SS
DD      DD  BB      BB  GG          VV      VV   AA      AA  LL          UU      UU  EE            SS
DD      DD  BBBBBBBB    GG          VV      VV   AA      AA  LL          UU      UU  EEEEEEEE      SSSSSS
DD      DD  BBBBBBBB    GG          VV      VV   AA      AA  LL          UU      UU  EEEEEEEE      SSSSSS
DD      DD  BB      BB  GG  GGGGGG  VV      VV   AAAAAAAAAA  LL          UU      UU  EE                SS
DD      DD  BB      BB  GG  GGGGGG  VV      VV   AAAAAAAAAA  LL          UU      UU  EE                SS
DD      DD  BB      BB  GG      GG    VV  VV     AA      AA  LL          UU      UU  EE                SS
DD      DD  BB      BB  GG      GG    VV  VV     AA      AA  LL          UU      UU  EE                SS
DDDDDDDD    BBBBBBBB      GGGGGG        VV       AA      AA  LLLLLLLLLL  UUUUUUUUUU  EEEEEEEEEE  SSSSSSSS
DDDDDDDD    BBBBBBBB      GGGGGG        VV       AA      AA  LLLLLLLLLL  UUUUUUUUUU  EEEEEEEEEE  SSSSSSSS
```

```
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  DBGVALUES(IDENT = 'V04-000') =
;     2          0002  1 BEGIN
;     3          0003  1 !
;     4          0004  1 !
;     5          0005  1 !*****************************************************************************
;     6          0006  1 !*                                                                           *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                     *
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;    16          0016  1 !*  TRANSFERRED.                                                             *
;    17          0017  1 !*                                                                           *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;    20          0020  1 !*  CORPORATION.                                                             *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*****************************************************************************
;    27          0027  1 !
;    28          0028  1 !
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 !  FACILITY:     VAX-11 DEBUG
;    32          0032  1 !
;    33          0033  1 !  ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       Language-Independent Value Descriptor support routines
;    36          0036  1 !
;    37          0037  1 !  ENVIRONMENT:  VAX/VMS user mode
;    38          0038  1 !
;    39          0039  1 !  AUTHOR:       J. Francis,     CREATION DATE: 19-Apr-1982
;    40          0040  1 !
;    41          0041  1 !  MODIFIED BY:
;    42          0042  1 !
;    43          0043  1 !       001      WC3     21-Jun-83
;    44          0044  1 !                Add support for /PACKED and /DATE_TIME
;    45          0045  1 !
;    46          0046  1 !       002      WC3     15-Jul-83
;    47          0047  1 !                Fix /DATE_TIME to use DBG$CVT_DX_DX
;    48          0048  1 !
;    49          0049  1 !       003      WC3     15-Sep-83
;    50          0050  1 !                Update DBG$GL_CURRENT_PRIMARY for self-referential records
;    51          0051  1 !
;    52          0052  1 !       004      WC3     22-Sep-83
;    53          0053  1 !                Check for variant records that have been optomized away
;    54          0054  1 !                but the DST is still around.
;    55          0055  1 !--
```

```
;    57          0056  1 !
;    58          0057  1 ! Table of Contents
;    59          0058  1 !
;    60          0059  1 FORWARD ROUTINE
;    61          0060  1 ! Global Routines :
;    62          0061  1     dbg$data_length,
;    63          0062  1     dbg$make_skeleton_desc,
;    64          0063  1     dbg$make_integer_desc,
;    65          0064  1     dbg$fill_in_vms_desc,                          ! fill in vms descriptor
;    66          0065  1     dbg$make_val_desc,                             ! Create  value  descriptor
;    67          0066  1     dbg$make_vms_desc,                             ! Create VAX/VMS descriptor
;    68          0067  1     dbg$prim_to_val,                               ! Get value of a primary
;    69          0068  1     dbg$print_aggregate          : NOVALUE,        ! Print aggregate value
;    70          0069  1     dbg$print_value              : NOVALUE,        ! Print value from DEBUG descriptor
;    71          0070  1     dbg$print_value_as_integer   : NOVALUE,        ! Print integer in given radix
;    72          0071  1     dbg$print_vms_value          : NOVALUE;        ! Print value from VMS descriptor
```

```
;    74          0072  1 !
;    75          0073  1 ! INCLUDE FILES
;    76          0074  1 !
;    77          0075  1 REQUIRE 'SRC$:DBGPROLOG';
;    78          0209  1 !
;    79          0210  1 !
;    80          0211  1 ! EXTERNALS
;    81          0212  1 !
;    82          0213  1 EXTERNAL
;    83          0214  1     dbg$gl_current_primary,                 ! Ponter to the primary being processed
;    84          0215  1     dbg$gb_radix        : VECTOR[3, BYTE],  ! Radix settings
;    85          0216  1     dbg$gb_language     : BYTE,             ! Current language
;    86          0217  1     dbg$gv_control      : dbg$control_flags,! DEBUG status information
;    87          0218  1     dbg$gl_call_context,                    ! Context for 'Bound' values
;    88          0219  1     dbg$gl_convert_token,                   ! "Integerize" operator token
;    89          0220  1     dbg$gl_dflttyp,                         ! Default type from "SET TYPE"
;    90          0221  1     dbg$gw_dfltleng     : WORD,             ! Length of default data-type
;    91          0222  1     dbg$gl_sign_flag;                       ! Print '+' before the signed
;    92          0223  1                                             !       variable.
;    93          0224  1
;    94          0225  1 EXTERNAL ROUTINE
;    95          0226  1     dbg$build_primary_subnode   : NOVALUE,  ! Add sub-node to primary
;    96          0227  1     dbg$collect                 : NOVALUE,  ! Sanitize character vectors
;    97          0228  1     dbg$cvt_dx_dx               : NOVALUE,  ! Convert data-types by descriptor
;    98          0229  1     dbg$enum_pos,                           ! Convert value->pos for enum type
;    99          0230  1     dbg$enum_succ,                          ! Find successor of enum type
;   100          0231  1     dbg$enum_val,                           ! Convert pos->value for enum type
;   101          0232  1     dbg$get_tempmem,                        ! Storage space allocator
;   102          0233  1     dbg$is_it_entry,                        ! Check for CALL entry-mask address
;   103          0234  1     dbg$ins_decode,                         ! Print an instruction
;   104          0235  1     dbg$language_format,                    ! Language override output
;   105          0236  1     dbg$newline                 : NOVALUE,  ! Print buffer contents
;   106          0237  1     dbg$nget_radix,                         ! Obtain radix
;   107          0238  1     dbg$print                   : NOVALUE,  ! Print under FAO format
;   108          0239  1     dbg$print_control           : NOVALUE,  ! Control print format
;   109          0240  1     dbg$print_identifier,                   ! Print name of data item
;   110          0241  1     dbg$print_set_value         : NOVALUE,  ! Print value of set
;   111          0242  1     dbg$print_symbol_name       : NOVALUE,  ! Print name from a SYMID
;   112          0243  1     dbg$push_tempmem,                       ! Mark current position
;   113          0244  1     dbg$pop_tempmem             : NOVALUE,  ! Release marked storage
;   114          0245  1     dbg$save_val                : NOVALUE,  ! Save value for %CURVAL
;   115          0246  1     dbg$sta_setcontext          : NOVALUE,  ! Establish RST context
;   116          0247  1     dbg$sta_sympathname         : NOVALUE,  ! Get fully-qualified data name
;   117          0248  1     dbg$sta_symkind             : NOVALUE,  ! Get  kind   of data item
;   118          0249  1     dbg$sta_symname             : NOVALUE,  ! Get  name   of data item
;   119          0250  1     dbg$sta_symsize             : NOVALUE,  ! Get length  of data item
;   120          0251  1     dbg$sta_symtype             : NOVALUE,  ! Get  type   of data item
;   121          0252  1     dbg$sta_symvalue            : NOVALUE,  ! Get address of data item
;   122          0253  1     dbg$sta_typefcode,                      ! Get  FCODE  of data item
;   123          0254  1     dbg$sta_typ_atomic          : NOVALUE,  ! Get symbol table information
;   124          0255  1     dbg$sta_typ_descr           : NOVALUE,  ! Get symbol table information
;   125          0256  1     dbg$sta_typ_enum            : NOVALUE,  ! Get symbol table information
;   126          0257  1     dbg$sta_typ_record          : NOVALUE,  ! Get symbol table information
;   127          0258  1     dbg$sta_typ_subrng          : NOVALUE,  ! Get symbol table information
;   128          0259  1     dbg$sta_typ_typedptr        : NOVALUE,  ! Get symbol table information
;   129          0260  1     dbg$sta_variant_value,                  ! Check value of Variant Tag
;   130          0261  1     dbg$sta_variant_select,                 ! Get variant entry (by tag)
```

```
:  131          0262  1      for$cvt_d_tg,                          ! Conversion routine
:  132          0263  1      for$cvt_g_tg,                          ! Conversion routine
:  133          0264  1      for$cvt_h_tg;                          ! Conversion routine
:  134          0265  1
:  135          0266  1  OWN
:  136          0267  1      signed_dtype        : BITVECTOR[dbg$k_maximum_dtype+1] PRESET(
:  137          0268  1          [dsc$k_dtype_f  ] = 1,
:  138          0269  1          [dsc$k_dtype_d  ] = 1,
:  139          0270  1          [dsc$k_dtype_g  ] = 1,
:  140          0271  1          [dsc$k_dtype_h  ] = 1,
:  141          0272  1          [dsc$k_dtype_b  ] = 1,
:  142          0273  1          [dsc$k_dtype_w  ] = 1,
:  143          0274  1          [dsc$k_dtype_l  ] = 1,
:  144          0275  1          [dsc$k_dtype_q  ] = 1,
:  145          0276  1          [dsc$k_dtype_o  ] = 1,
:  146          0277  1          [dsc$k_dtype_p  ] = 1,
:  147          0278  1          [dsc$k_dtype_nz ] = 1,
:  148          0279  1          [dsc$k_dtype_nl ] = 1,
:  149          0280  1          [dsc$k_dtype_nlo] = 1,
:  150          0281  1          [dsc$k_dtype_nr ] = 1,
:  151          0282  1          [dsc$k_dtype_nro] = 1,
:  152          0283  1          [dsc$k_dtype_sv ] = 1,
:  153          0284  1          [dsc$k_dtype_svu] = 1);
:  154          0285  1
:  155          0286  1
:  156          0287  1  BIND
:  157          0288  1      Format_AC = UPLIT BYTE(%ASCIC '!AC'),
:  158          0289  1      Format_AD = UPLIT BYTE(%ASCIC '!AD');
```

```
;  160     0290  1 GLOBAL ROUTINE DBG$DATA_LENGTH (vms_desc : REF dbg$stg_desc) =
;  161     0291  1 !
;  162     0292  1 ! FUNCTION
;  163     0293  1 !       Given a VMS descriptor, this routine returns the length in
;  164     0294  1 !       bits of the object described by the descriptor.
;  165     0295  1 !
;  166     0296  1 !       Note - for array descriptors, this routine returns the length
;  167     0297  1 !       in bits of an element of the array. Do not change it to
;  168     0298  1 !       return the length of the entire array - things will break.
;  169     0299  1 !
;  170     0300  2     BEGIN
;  171     0301  2     LOCAL length;
;  172     0302  2
;  173     0303  2     ! Obtain the length from the descriptor. We do not yet know whether
;  174     0304  2     ! this length is in bits, nibbles, or bytes.
;  175     0305  2     !
;  176     0306  2     length = .vms_desc[dsc$w_length];
;  177     0307  2
;  178     0308  2     ! Decide whether the length is in bits, nibbles, or bytes, based
;  179     0309  2     ! on the dtype.
;  180     0310  2     !
;  181     0311  2     IF .vms_desc[dsc$b_dtype] EQL dst$k_bool
;  182     0312  2     THEN
;  183     0313  2         length = .length * 1
;  184     0314  2
;  185     0315  2     ELSE CASE .vms_desc[dsc$b_dtype] FROM dbg$k_minimum_dtype TO dbg$k_maximum_dtype OF
;  186     0316  2         SET
;  187     0317  2         [dsc$k_dtype_f  ,dsc$k_dtype_fc ,dsc$k_dtype_d  ,dsc$k_dtype_dc ,
;  188     0318  2          dsc$k_dtype_g  ,dsc$k_dtype_gc ,dsc$k_dtype_h  ,dsc$k_dtype_hc ,
;  189     0319  2          dsc$k_dtype_b  ,dsc$k_dtype_bu ,dsc$k_dtype_w  ,dsc$k_dtype_wu ,
;  190     0320  2          dsc$k_dtype_l  ,dsc$k_dtype_lu ,dsc$k_dtype_q  ,dsc$k_dtype_qu ,
;  191     0321  2          dsc$k_dtype_o  ,dsc$k_dtype_ou ,dsc$k_dtype_t,  dsc$k_dtype_z  ,
;  192     0322  2          dsc$k_dtype_nl ,dsc$k_dtype_nlo,dsc$k_dtype_nr ,dsc$k_dtype_nro,
;  193     0323  2          dsc$k_dtype_nu ,dsc$k_dtype_nz ,dsc$k_dtype_zi ,dsc$k_dtype_zem,
;  194     0324  2          dsc$k_dtype_dsc,dsc$k_dtype_bpv,dsc$k_dtype_blv,dsc$k_dtype_adt]:      ! M002
;  195     0325  2
;  196     0326  2                                  length = .length * %BPUNIT;
;  197     0327  2
;  198     0328  2         [dsc$k_dtype_vt] :       length = (.length + 2) * %BPUNIT;
;  199     0329  2
;  200     0330  2         [dsc$k_dtype_p] :    length = (.length/2 + 1) * %BPUNIT;
;  201     0331  2
;  202     0332  2         [dsc$k_dtype_tf,dsc$k_dtype_vu,dsc$k_dtype_svu] :       0;
;  203     0333  2
;  204     0334  2         [dsc$k_dtype_v ,dsc$k_dtype_sv] :   IF (.length EQL 0) AND
;  205     0335  2                                  (.vms_desc[dsc$b_class] EQL dsc$k_class_z) THEN
;  206     0336  2                                  length = .vms_desc[dsc$l_pos];
;  207     0337  2
;  208     0338  3         [dsc$k_dtype_ac] :       length = (1 +
;  209     0339  2                                  .(.vms_desc[dsc$a_pointer])<0,8,0>) * %BPUNIT;
;  210     0340  2
;  211     0341  3         [dsc$k_dtype_az] :       BEGIN
;  212     0342  3                                  BIND chrvec = vms_desc[dsc$a_pointer] : REF VECTOR [,BYTE];
;  213     0343  3                                  LOCAL index;
;  214     0344  3                                  index = 0;
;  215     0345  3                                  WHILE .index LEQ 2046 DO
;  216     0346  4                                    BEGIN
```

```
;  217     0347  4                     IF .chrvec[.index] EQL 0 THEN EXITLOOP;
;  218     0348  4                     index = .index + 1;
;  219     0349  3                     END;
;  220     0350  3                 length = (.index+1) * %BPUNIT;
;  221     0351  2                 END;
;  222     0352  2
;  223     0353  2         [INRANGE,OUTRANGE] :    length = .length * %BPUNIT;
;  224     0354  2         TES;
;  225     0355  2
;  226     0356  2     RETURN .length;
;  227     0357  1     END;                        ! End of 'dbg$data_length'


                                                .TITLE  DBGVALUES
                                                .IDENT  \V04-000\

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

                                43  41  21  03  00000 P.AAA:  .ASCII  <3>\!AC\                    ;
                                44  41  21  03  00004 P.AAB:  .ASCII  <3>\!AD\                    ;

                                                .PSECT  DBG$OWN,NOEXE,  PIC,2

                                1C  3F  0F  C0  00000 SIGNED_DTYPE:
                                                .BYTE   -64, 15, 63, 28                         ;
                                            00  00004           .BYTE   0                       ;
                                            06  00005           .BYTE   6                       ;

                                        FORMAT_AC=          P.AAA
                                        FORMAT_AD=          P.AAB
                                                .EXTRN  DBG$GL_CURRENT_PRIMARY
                                                .EXTRN  DBG$GB_RADIX, DBG$GB_LANGUAGE
                                                .EXTRN  DBG$GV_CONTROL, DBG$GL_CALL_CONTEXT
                                                .EXTRN  DBG$GL_CONVERT_TOKEN
                                                .EXTRN  DBG$GL_DFLTTYP, DBG$GW_DFLTLENG
                                                .EXTRN  DBG$GL_SIGN_FLAG
                                                .EXTRN  DBG$BUILD_PRIMARY_SUBNODE
                                                .EXTRN  DBG$COLLECT, DBG$CVT_DX_DX
                                                .EXTRN  DBG$ENUM_POS, DBG$ENUM_SUCC
                                                .EXTRN  DBG$ENUM_VAL, DBG$GET_TEMPMEM
                                                .EXTRN  DBG$IS_IT_ENTRY
                                                .EXTRN  DBG$INS_DECODE, DBG$LANGUAGE_FORMAT
                                                .EXTRN  DBG$NEWLINE, DBG$NGET_RADIX
                                                .EXTRN  DBG$PRINT, DBG$PRINT_CONTROL
                                                .EXTRN  DBG$PRINT_IDENTIFIER
                                                .EXTRN  DBG$PRINT_SET_VALUE
                                                .EXTRN  DBG$PRINT_SYMBOL_NAME
                                                .EXTRN  DBG$PUSH_TEMPMEM
                                                .EXTRN  DBG$POP_TEMPMEM
                                                .EXTRN  DBG$SAVE_VAL, DBG$STA_SETCONTEXT
                                                .EXTRN  DBG$STA_SYMPATHNAME
                                                .EXTRN  DBG$STA_SYMKIND
                                                .EXTRN  DBG$STA_SYMNAME
                                                .EXTRN  DBG$STA_SYMSIZE
                                                .EXTRN  DBG$STA_SYMTYPE
                                                .EXTRN  DBG$STA_SYMVALUE
                                                .EXTRN  DBG$STA_TYPEFCODE
```

```
                                                        .EXTRN  DBG$STA_TYP_ATOMIC
                                                        .EXTRN  DBG$STA_TYP_DESCR
                                                        .EXTRN  DBG$STA_TYP_ENUM
                                                        .EXTRN  DBG$STA_TYP_RECORD
                                                        .EXTRN  DBG$STA_TYP_SUBRNG
                                                        .EXTRN  DBG$STA_TYP_TYPEDPTR
                                                        .EXTRN  DBG$STA_VARIANT_VALUE
                                                        .EXTRN  DBG$STA_VARIANT_SELECT
                                                        .EXTRN  FOR$CVT_D_TG, FOR$CVT_G_TG
                                                        .EXTRN  FOR$CVT_H_TG

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

                                    0004 00000          .ENTRY  DBG$DATA_LENGTH, Save R2                        ; 0290
                                52  04   AC  D0 00002   MOVL   VMS_DESC, R2                                    ; 0306
                                51      62  3C 00006   MOVZWL (R2), LENGTH                                    ;
                             9E  8F  02   A2  91 00009   CMPB   2(R2), #158                                     ; 0311
                                     02   7D  13 0000E   BEQL   6$                                             ;
                             2B      00   02   A2  8F 00010   CASEB  2(R2), #0, #43                            ; 0315
                0058        0058        006B        0058    00015 1$:   .WORD  2$-1$,-                          ;
                0058        0058        0058        0058    0001D       5$-1$,-                                 ;
                0058        0058        0058        0058    00025       2$-1$,-                                 ;
                0058        0058        0058        0058    0002D       2$-1$,-                                 ;
                0058        0058        0058        0058    00035       2$-1$,-                                 ;
                0058        0058        0065        0058    0003D       2$-1$,-                                 ;
                0058        0058        0058        0058    00045       2$-1$,-                                 ;
                0058        0058        0058        0058    0004D       2$-1$,-                                 ;
                0058        009C        0058        0058    00055       2$-1$,-                                 ;
                0080        007A        005D        0058    0005D       2$-1$,-                                 ;
                0058        009C        006B        009C    00065       2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        4$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        11$-1$,-                                ;
                                                                        2$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        3$-1$,-                                 ;
```

```
                                                                                7$-1$,-                                 ;
                                                                                8$-1$,-                                 ;
                                                                                11$-1$,-                                ;
                                                                                5$-1$,-                                 ;
                                                                                11$-1$,-                                ;
                                                                                2$-1$                                   ;
                    51              08  C4 0006D 2$:    MULL2   #8, LENGTH                                          ; 0326
                                    3F  11 00070        BRB     11$
                    51              08  C4 00072 3$:    MULL2   #8, LENGTH                                          ; 0328
                    51              10  C0 00075        ADDL2   #16, LENGTH
                                    37  11 00078        BRB     11$
          50        51              02  C7 0007A 4$:    DIVL3   #2, LENGTH, R0                                      ; 0330
                                    2A  11 0007E        BRB     10$
                                    51  D5 00080 5$:    TSTL    LENGTH                                              ; 0334
                                    2D  12 00082        BNEQ    11$
                              03    A2  95 00084        TSTB    3(R2)                                               ; 0335
                                    28  12 00087        BNEQ    11$
                    51        08    A2  D0 00089        MOVL    8(R2), LENGTH                                       ; 0336
                                    22  11 0008D 6$:    BRB     11$                                                 ; 0334
                    50        04    B2  9A 0008F 7$:    MOVZBL  @4(R2), R0                                          ; 0339
                                    15  11 00093        BRB     10$
                                    50  D4 00095 8$:    CLRL    INDEX                                               ; 0344
              000007FE  8F          50  D1 00097 9$:    CMPL    INDEX, #2046                                        ; 0345
                                    0A  14 0009E        BGTR    10$
                              04 B240  95 000A0        TSTB    @4(R2)[INDEX]                                       ; 0347
                                    04  13 000A4        BEQL    10$
                                    50  D6 000A6        INCL    INDEX                                               ; 0348
                                    ED  11 000A8        BRB     9$                                                  ; 0345
          51                        03  78 000AA 10$:    ASHL    #3, INDEX, LENGTH                                   ; 0350
                    51              08  C0 000AE        ADDL2   #8, LENGTH
                    50              51  D0 000B1 11$:    MOVL    LENGTH, R0                                          ; 0356
                                        04 000B4        RET                                                         ; 0357

; Routine Size:  181 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  229          0358  1 GLOBAL ROUTINE DBG$MAKE_SKELETON_DESC(desc_type,data_length) =
;  230          0359  2     BEGIN
;  231          0360  2     BUILTIN ACTUALCOUNT;
;  232          0361  2     LOCAL
;  233          0362  2         desc_length,
;  234          0363  2         result_desc     : REF BLOCK [,LONG] FIELD(dbg$dhdr_fields);
;  235          0364  2
;  236          0365  2     SELECTONE .desc_type OF
;  237          0366  2         SET
;  238          0367  2         [dbg$k_v_value_desc]:   desc_length = %UPVAL*dbg$k_valdesc_base_size + 16;
;  239          0368  2
;  240          0369  3         [dbg$k_value_desc]:     BEGIN
;  241          0370  3                                 desc_length = %UPVAL*dbg$k_valdesc_base_size + 16;
;  242          0371  3                                 IF actualcount() GTR 1 THEN
;  243          0372  3                                   IF .data_length GTR 16 THEN
;  244          0373  3                                     desc_length = .data_length + %UPVAL*dbg$k_valdesc_base_size;
;  245          0374  3                                 END;
;  246          0375  2
;  247          0376  3         [dbg$k_primary_desc]:   BEGIN
;  248          0377  3                                 desc_length = 20;
;  249          0378  3                                 IF actualcount() GTR 1 THEN
;  250          0379  3                                     desc_length = .desc_length + .data_length;
;  251          0380  3                                 END;
;  252          0381  2
;  253          0382  2         [OTHERWISE]:            SIGNAL();
;  254          0383  2         TES;
;  255          0384  2
;  256          0385  2     result_desc = dbg$get_tempmem((.desc_length + (%UPVAL-1)) / %UPVAL);
;  257          0386  2     result_desc[dbg$b_dhdr_lang]   = %X'FF';
;  258          0387  2     result_desc[dbg$b_dhdr_type]   = .desc_type;
;  259          0388  2     result_desc[dbg$w_dhdr_length] = .desc_length;
;  260          0389  2     RETURN .result_desc;
;  261          0390  2
;  262          0391  1     END;                            ! End of 'dbg$make_skeleton_desc'
```

```
                                    000C 00000          .ENTRY  DBG$MAKE_SKELETON_DESC, Save R2,R3          ; 0358
                         53     04 AC  D0 00002        MOVL    DESC_TYPE, R3                               ; 0365
              00000083   8F        53  D1 00006        CMPL    R3, #131                                    ; 0367
                                   05  12 0000D        BNEQ    1$                                          ;
                         52        30  D0 0000F        MOVL    #48, DESC_LENGTH                            ;
                                   3C  11 00012        BRB     4$                                          ;
              0000007A   8F        53  D1 00014 1$:    CMPL    R3, #122                                    ; 0369
                                   15  12 0001B        BNEQ    2$                                          ;
                         52        30  D0 0001D        MOVL    #48, DESC_LENGTH                            ; 0370
                         01        6C  91 00020        CMPB    (AP), #1                                    ; 0371
                                   2B  1B 00023        BLEQU   4$                                          ;
                         10     08 AC  D1 00025        CMPL    DATA_LENGTH, #16                            ; 0372
                                   25  15 00029        BLEQ    4$                                          ;
                52       08     AC  C1 0002B        ADDL3   #32, DATA_LENGTH, DESC_LENGTH               ; 0373
                                   1E  11 00030        BRB     4$                                          ; 0365
              00000079   8F        53  D1 00032 2$:    CMPL    R3, #121                                    ; 0376
                                   0E  12 00039        BNEQ    3$                                          ;
                         52        14  D0 0003B        MOVL    #20, DESC_LENGTH                            ; 0377
```

```
                          01              6C  91 0003E          CMPB    (AP), #1                                ; 0378
                                          0D  1B 00041          BLEQU   4$                                      ;
                          52        08    AC  C0 00043          ADDL2   DATA_LENGTH, DESC_LENGTH                ; 0379
                                          07  11 00047          BRB     4$                                      ; 0365
                00000000G 00              00  FB 00049 3$:      CALLS   #0, LIB$SIGNAL                          ; 0382
                          50        03    A2  9E 00050 4$:      MOVAB   3(R2), R0                               ; 0385
                7E        50              04  C7 00054          DIVL3   #4, R0, -(SP)                           ;
                00000000G 00              01  FB 00058          CALLS   #1, DBG$GET_TEMPMEM                     ;
                       03 A0              01  8E 0005F          MNEGB   #1, 3(RESULT_DESC)                      ; 0386
                       02 A0              53  90 00063          MOVB    R3, 2(RESULT_DESC)                      ; 0387
                          60              52  B0 00067          MOVW    DESC_LENGTH, (RESULT_DESC)              ; 0388
                                              04 0006A          RET                                             ; 0391

; Routine Size:  107 bytes,    Routine Base:  DBG$CODE + 00B5
```

```
;  264          0392  1 GLOBAL ROUTINE DBG$MAKE_INTEGER_DESC(VALUE) =
;  265          0393  1 !
;  266          0394  1 !++
;  267          0395  1 !++
;  268          0396  1 !       Given an integer value, this routine builds a value descriptor
;  269          0397  1 !       for that integer.
;  270          0398  1 !
;  271          0399  1 !--
;  272          0400  1 !
;  273          0401  1 !--
;  274          0402  1 !       A pointer to the constructed value descriptor is returned.
;  275          0403  1 !       The descriptor is built out of temporary memory.
;  276          0404  1 !--
;  277          0405  2     BEGIN
;  278          0406  2     LOCAL
;  279          0407  2         TEMP_DESC: REF DBG$VALDESC;
;  280          0408  2
;  281          0409  2     TEMP_DESC = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC);
;  282          0410  2     TEMP_DESC[DBG$B_DHDR_KIND] = RST$K_DATA;
;  283          0411  2     TEMP_DESC[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
;  284          0412  2     TEMP_DESC[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  285          0413  2     TEMP_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
;  286          0414  2     TEMP_DESC[DBG$W_VALUE_LENGTH] = 4;
;  287          0415  2     TEMP_DESC[DBG$L_VALUE_POINTER] = TEMP_DESC[DBG$A_VALUE_ADDRESS];
;  288          0416  2     TEMP_DESC[DBG$L_VALUE_VALUE0] = .VALUE;
;  289          0417  2     RETURN .TEMP_DESC;
;  290          0418  1     END;


                                    0000 00000  .ENTRY  DBG$MAKE_INTEGER_DESC, Save nothing          ; 0392
                          7E         7A   8F   9A 00002  MOVZBL  #122, -(SP)                         ; 0409
                      8B  AF              01   FB 00006  CALLS   #1, DBG$MAKE_SKELETON_DESC         ;
                      06  A0       0602   8F   B0 0000A  MOVW    #1538, 6(TEMP_DESC)               ; 0411
                      14  A0 01080004    8F   D0 00010  MOVL    #17301508, 20(TEMP_DESC)          ; 0414
                      18  A0        20    A0   9E 00018  MOVAB   32(TEMP_DESC), 24(TEMP_DESC)      ; 0415
                      20  A0        04    AC   D0 0001D  MOVL    VALUE, 32(TEMP_DESC)              ; 0416
                                          04   00022  RET                                          ; 0418

; Routine Size:  35 bytes,    Routine Base:  DBG$CODE + 0120
```

```
;  292     0419  1 GLOBAL ROUTINE DBG$MAKE_VAL_DESC (desc_ptr,target_type) =
;  293     0420  1 !
;  294     0421  1 ! ROUTINE DESCRIPTION
;  295     0422  1 !       Given a VMS descriptor, this routine builds either a Value Descriptor
;  296     0423  1 !       or a Volatile Value Descriptor around the VMS descriptor.
;  297     0424  1 !       In the case where a Value Descriptor is constructed, we need to
;  298     0425  1 !       extract the value represented by the VMS descriptor, and put
;  299     0426  1 !       this value inside the Value Descriptor. So, for descriptors
;  300     0427  1 !       representing bitfields, this routine is where the actual extraction
;  301     0428  1 !       of the bits takes place.
;  302     0429  1 !
;  303     0430  1 ! INPUTS
;  304     0431  1 !       DESC_PTR        - Points to a Vax-standard VMS descriptor
;  305     0432  1 !       TARGET_TYPE     - A constant which can be one of:
;  306     0433  1 !                         DBG$K_VALUE_DESC   or   DBG$K_V_VALUE_DESC
;  307     0434  1 ! OUTPUTS
;  308     0435  1 !       A Value Descriptor or a Volatile Value Descriptor is constructed
;  309     0436  1 !       out of temporary memory. A pointer to this descriptor is returned.
;  310     0437  1 !
;  311     0438  2     BEGIN
;  312     0439  2     LOCAL
;  313     0440  2         vms_desc        : dbg$stg_desc,
;  314     0441  2         bits,bytes,
;  315     0442  2         result_desc     : REF dbg$valdesc;
;  316     0443  2
;  317     0444  2     !+
;  318     0445  2     ! The first thing we do is to 'de-reference' data items of type
;  319     0446  2     ! descriptor which is what we get for arrays of dynamic strings.
;  320     0447  2     !-
;  321     0448  2     ch$move(12,.desc_ptr,vms_desc);
;  322     0449  2     IF .target_type EQL dbg$k_value_desc THEN
;  323     0450  2       WHILE .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_dsc DO
;  324     0451  3         BEGIN
;  325     0452  3         BUILTIN PROBER;
;  326     0453  3         LOCAL addr;
;  327     0454  3         addr = .vms_desc[dsc$a_pointer];
;  328     0455  3         IF NOT PROBER(%REF(0),%REF(8),.addr) THEN SIGNAL(dbg$_noaccessr,1,.addr);
;  329     0456  3         ch$move(8,.addr,vms_desc);
;  330     0457  3         CASE .vms_desc[dsc$b_class] FROM dsc$k_class_z TO dsc$k_class_ubs
;  331     0458  3          OF SET
;  332     0459  3             [dsc$k_class_s,dsc$k_class_d,dsc$k_class_vs] :
;  333     0460  4                 BEGIN
;  334     0461  4                 IF   .vms_desc[dsc$b_class] EQL dsc$k_class_d
;  335     0462  4                 THEN vms_desc[dsc$b_class]  =  dsc$k_class_s;
;  336     0463  4                 IF   .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_vt
;  337     0464  4                 THEN vms_desc[dsc$b_class]  =  dsc$k_class_vs;
;  338     0465  4                 IF   .vms_desc[dsc$b_class] EQL dsc$k_class_vs
;  339     0466  4                 AND .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_t
;  340     0467  4                 THEN vms_desc[dsc$b_dtype]  =  dsc$k_dtype_vt;
;  341     0468  4                 vms_desc[dsc$l_pos] = 0;
;  342     0469  3                 END;
;  343     0470  3
;  344     0471  3             [dsc$k_class_sd,dsc$k_class_ubs] :
;  345     0472  4                 BEGIN
;  346     0473  4                 IF NOT PROBER(%REF(0),%REF(4),.addr+8) THEN SIGNAL(dbg$_noaccessr,1,.addr+8);
;  347     0474  4                 vms_desc[dsc$l_pos] = .(.addr+8)<0,32,0>;
;  348     0475  3                 END;
```

```
;  349     0476  3                [INRANGE,OUTRANGE] :        SIGNAL(dbg$_illtype);
;  350     0477  3                TES;
;  351     0478  3            END;
;  352     0479  2        !+
;  353     0480  2        ! Obtain the length in bits and the length in bytes of the data
;  354     0481  2        ! represented by the VMS descriptor.
;  355     0482  2        !-
;  356     0483  2        bits  = dbg$data_length(vms_desc);
;  357     0484  2        bytes = (.bits + (%BPUNIT-1)) / %BPUNIT;
;  358     0485  2
;  359     0486  2        !+
;  360     0487  2        ! Allocate either enough space for a Volatile Value Descriptor,
;  361     0488  2        ! or enough space for a Value Descriptor which contains the value.
;  362     0489  2        !-
;  363     0490  2        IF (.target_type EQL dbg$k_v_value_desc) OR (.bytes GTR 512)
;  364     0491  2        THEN
;  365     0492  2            result_desc = dbg$make_skeleton_desc(dbg$k_v_value_desc)
;  366     0493  2        ELSE
;  367     0494  2            result_desc = dbg$make_skeleton_desc(dbg$k_value_desc,.bytes);
;  368     0495  2
;  369     0496  2        !+
;  370     0497  2        ! Copy the VMS descriptor into the Value Descriptor. Also fill in
;  371     0498  2        ! the kind and fcode fields.
;  372     0499  2        !-
;  373     0500  2        ch$move(12,vms_desc,result_desc[dbg$a_value_vmsdesc]);
;  374     0501  2        result_desc[dbg$b_dhdr_kind]   = rst$k_data;
;  375     0502  2        result_desc[dbg$b_dhdr_fcode]  = rst$k_type_descr;
;  376     0503  2
;  377     0504  2        !+
;  378     0505  2        ! If the target type is a value descriptor then we want to extract
;  379     0506  2        ! the value represented by the VMS descriptor and
;  380     0507  2        ! right-align the result in the value descriptor.
;  381     0508  2        !-
;  382     0509  2        IF .result_desc[dbg$b_dhdr_type] EQL dbg$k_value_desc THEN
;  383     0510  3            BEGIN
;  384     0511  3            BUILTIN PROBER;
;  385     0512  3            LOCAL addr,pos;
;  386     0513  3            result_desc[dbg$l_value_pointer] = result_desc[dbg$a_value_address];
;  387     0514  3            !+
;  388     0515  3            ! Compute the byte address by adding (bit_offset/8).
;  389     0516  3            ! Then set the bit offset to (bit_offset mod 8).
;  390     0517  3            !-
;  391     0518  3            IF .vms_desc[dsc$b_class] EQL dsc$k_class_ubs
;  392     0519  3            THEN
;  393     0520  3                BEGIN
;  394     0521  4                pos  = .(vms_desc[dsc$l_pos])<0,3,0>;
;  395     0522  4                addr = .vms_desc[dsc$a_pointer] + .(vms_desc[dsc$l_pos])<3,29,1>;
;  396     0523  4                END
;  397     0524  4            ELSE
;  398     0525  3                BEGIN
;  399     0526  4                pos = 0;
;  400     0527  4                addr = .vms_desc[dsc$a_pointer];
;  401     0528  4                END;
;  402     0529  3            !+
;  403     0530  3            ! Check for read access.
;  404     0531  3            !-
;  405     0532  3
```

```
;  406     0533  3          bytes = (.pos + .bits + 7)/8;
;  407     0534  3          IF .bytes NEQ 0
;  408     0535  3          THEN
;  409     0536  3              IF NOT PROBER(%REF(0),bytes,..addr)
;  410     0537  3              THEN
;  411     0538  3                  SIGNAL(dbg$_noaccessr,1,..addr);
;  412     0539  3          !+
;  413     0540  3          ! Don't support bit extractions bigger than 32 bits for now.
;  414     0541  3          ! This restriction may be relaxed at a later time.
;  415     0542  3          !-
;  416     0543  3          IF .bits LEQU 32
;  417     0544  3          THEN
;  418     0545  4              BEGIN
;  419     0546  4              !+
;  420     0547  4              ! Decide whether to do a signed or an unsigned extraction based on
;  421     0548  4              ! the dtype.
;  422     0549  4              !-
;  423     0550  4              SELECTONE .vms_desc[dsc$b_dtype] OF
;  424     0551  4                  SET
;  425     0552  4                  [dsc$k_dtype_sv,dsc$k_dtype_svu,dsc$k_dtype_b,dsc$k_dtype_w] :
;  426     0553  4                      .result_desc[dbg$l_value_pointer] = .(.addr)<.pos,.bits,1>;
;  427     0554  4                  [OTHERWISE] :
;  428     0555  4                      .result_desc[dbg$l_value_pointer] = .(.addr)<.pos,.bits,0>;
;  429     0556  4                  TES;
;  430     0557  4              !+
;  431     0558  4              ! Since we have performed the bit extraction, the bit offset is
;  432     0559  4              ! now zero. Zero the POS field to reflect this.
;  433     0560  4              !-
;  434     0561  4              IF .result_desc[dbg$b_value_class] EQL dsc$k_class_ubs
;  435     0562  4                THEN result_desc[dbg$l_value_pos] = 0;
;  436     0563  4              END
;  437     0564  3          ELSE
;  438     0565  4              BEGIN
;  439     0566  4              !+
;  440     0567  4              ! The value is longer than 32 bits.
;  441     0568  4              !-
;  442     0569  4
;  443     0570  4              IF .pos EQL 0
;  444     0571  4              THEN
;  445     0572  4                !+
;  446     0573  4                ! Copy the bytes.
;  447     0574  4                !-
;  448     0575  4                ch$move(.bytes,..addr,.result_desc[dbg$l_value_pointer])
;  449     0576  4
;  450     0577  4              ELSE
;  451     0578  5                BEGIN
;  452     0579  5                !+
;  453     0580  5                ! We used to disallow this.
;  454     0581  5                ! SIGNAL(dbg$_unimplent);
;  455     0582  5                !-
;  456     0583  5                INCR i FROM 0 TO (.bits-1)/32 DO
;  457     0584  5                  (4*.i + .result_desc[dbg$l_value_pointer]) =
;  458     0585  5                      .(4*.i + .addr)<.pos,32,0>;
;  459     0586  5
;  460     0587  5                !+
;  461     0588  5                ! Since we have performed the bit extraction, the bit offset is
;  462     0589  5                ! now zero. Zero the POS field to reflect this.
```

```
;  463     0590  5              !-
;  464     0591  5              IF .result_desc[dbg$b_value_class] EQL dsc$k_class_ubs
;  465     0592  5                THEN result_desc[dbg$l_value_pos] = 0;
;  466     0593  4              END;
;  467     0594  3            END;
;  468     0595  2        END;
;  469     0596  2
;  470     0597  2        !+
;  471     0598  2        ! We are all done. Return a pointer to the newly constructed Value
;  472     0599  2        ! Descriptor.
;  473     0600  2        !-
;  474     0601  2        RETURN .result_desc;
;  475     0602  1        END;                    ! end of routine dbg$make_val_desc


                                                 OFFC 00000          .ENTRY  DBG$MAKE_VAL_DESC, Save R2,R3,R4,R5,R6,R7,- ; 0419
                                                                             R8,R9,R10,R11
                                                      5E  0C  C2 00002       SUBL2   #12, SP
                                                6E        04  BC  0C  28 00005 MOVC3   #12, @DESC_PTR, VMS_DESC                ; 0448
                                           0000007A       8F  08  AC  D1 0000A CMPL    TARGET_TYPE, #122                       ; 0449
                                                              04  12 00012     BNEQ    2$
                                                      18  02  AE  91 00014 1$:  CMPB    VMS_DESC+2, #24                         ; 0450
                                                              03  13 00018 2$:  BEQL    3$
                                                            0098  31 0001A     BRW     14$
                                                      56  04  AE  D0 0001D 3$:  MOVL    VMS_DESC+4, ADDR                        ; 0454
                                                66    08      00  0C 00021     PROBER  #0, #8, (ADDR)                          ; 0455
                                                              11  12 00025     BNEQ    4$
                                                              56  DD 00027     PUSHL   ADDR
                                                              01  DD 00029     PUSHL   #1
                                                   00028228   8F  DD 0002B     PUSHL   #164392
                                           00000000G  00      03  FB 00031     CALLS   #3, LIB$SIGNAL
                                                6E    66      08  28 00038 4$:  MOVC3   #8, (ADDR), VMS_DESC                    ; 0456
                                                0D    00  03  AE  8F 0003C     CASEB   VMS_DESC+3, #0, #13                     ; 0457
                                           001C      002B      002B  001C  00041 5$:  .WORD   6$-5$,-
                                           001C      001C      001C  001C  00049           8$-5$,-
                                           002B      001C      0054  001C  00051           8$-5$,-
                                                     0054            001C  00059           6$-5$,-
                                                                                           6$-5$,-
                                                                                           6$-5$,-
                                                                                           6$-5$,-
                                                                                           6$-5$,-
                                                                                           6$-5$,-
                                                                                           12$-5$,-
                                                                                           6$-5$,-
                                                                                           8$-5$,-
                                                                                           6$-5$,-
                                                                                           12$-5$
                                                   000287D8   8F  DD 0005D 6$:  PUSHL   #165848                                 ; 0477
                                           00000000G  00      01  FB 00063     CALLS   #1, LIB$SIGNAL
                                                              A8  11 0006A 7$:  BRB     1$
                                                      02  03  AE  91 0006C 8$:  CMPB    VMS_DESC+3, #2                          ; 0461
                                                              04  12 00070     BNEQ    9$
                                                03    AE        01  90 00072     MOVB    #1, VMS_DESC+3                          ; 0462
                                                      25  02  AE  91 00076 9$:  CMPB    VMS_DESC+2, #37                         ; 0463
```

```
                                                04  12  0007A          BNEQ    10$                                             ;
                                        03      0B  90  0007C          MOVB    #11, VMS_DESC+3                                 ; 0464
                                    03  AE      0B  91  00080  10$:     CMPB    VMS_DESC+3, #11                                 ; 0465
                                                0A  12  00084          BNEQ    11$                                             ;
                                    02  AE      0E  91  00086          CMPB    VMS_DESC+2, #14                                 ; 0466
                                                04  12  0008A          BNEQ    11$                                             ;
                                        AE      02  90  0008C          MOVB    #37, VMS_DESC+2                                 ; 0467
                                    08  AE      D4  00090  11$:        CLRL    VMS_DESC+8                                      ; 0468
                                                D5  11  00093          BRB     7$                                              ; 0457
                                04      A6      00  0C  00095  12$:     PROBER  #0, #4, 8(ADDR)                                 ; 0473
                                                12  12  0009A          BNEQ    13$                                             ;
                                    08  A6      9F  0009C              PUSHAB  8(ADDR)                                         ;
                                                01  DD  0009F          PUSHL   #1                                              ;
                            00028228            8F  DD  000A1          PUSHL   #164392                                         ;
                    00000000G   00              03  FB  000A7          CALLS   #3, LIB$SIGNAL                                  ;
                        08  AE      08  A6      D0  000AE  13$:        MOVL    8(ADDR), VMS_DESC+8                             ; 0474
                                                B5  11  000B3          BRB     7$                                              ; 0450
                                                5E  DD  000B5  14$:     PUSHL   SP                                              ; 0484
                        FE01    CF              01  FB  000B7          CALLS   #1, DBG$DATA_LENGTH                             ;
                                5A              50  D0  000BC          MOVL    R0, BITS                                        ;
                                50      07      AA  9E  000BF          MOVAB   7(R10), R0                                      ; 0485
                5B              50      08      C7  000C3              DIVL3   #8, R0, BYTES                                   ;
                    00000083    8F      08      AC  D1  000C7          CMPL    TARGET_TYPE, #131                               ; 0491
                                                09  13  000CF          BEQL    15$                                             ;
                    00000200    8F              5B  D1  000D1          CMPL    BYTES, #512                                     ;
                                                0B  15  000D8          BLEQ    16$                                             ;
                                7E      83      8F  9A  000DA  15$:     MOVZBL  #131, -(SP)                                     ; 0493
                        FE8F    CF              01  FB  000DE          CALLS   #1, DBG$MAKE_SKELETON_DESC                      ;
                                                0B  11  000E3          BRB     17$                                             ;
                                                5B  DD  000E5  16$:     PUSHL   BYTES                                           ; 0495
                                7E      7A      8F  9A  000E7          MOVZBL  #122, -(SP)                                     ;
                        FE82    CF              02  FB  000EB          CALLS   #2, DBG$MAKE_SKELETON_DESC                      ;
                                57              50  D0  000F0  17$:     MOVL    R0, RESULT_DESC                                 ;
            14  A7              6E              0C  28  000F3          MOVC3   #12, VMS_DESC, 20(RESULT_DESC)                  ; 0501
                        06      A7      0603    8F  B0  000F8          MOVW    #1539, 6(RESULT_DESC)                           ; 0503
                        7A      8F      02      A7  91  000FE          CMPB    2(RESULT_DESC), #122                            ; 0510
                                                7E  12  00103          BNEQ    25$                                             ;
                                56      18      A7  9E  00105          MOVAB   24(RESULT_DESC), R6                             ; 0514
                                66      20      A7  9E  00109          MOVAB   32(R7), (R6)                                    ;
                                0D      03      AE  91  0010D          CMPB    VMS_DESC+3, #13                                 ; 0519
                                                12  12  00111          BNEQ    18$                                             ;
    58      08  AE              03              00  EF  00113          EXTZV   #0, #3, VMS_DESC+8, POS                         ; 0522
                59      08      AE      FD      8F  78  00119          ASHL    #-3, VMS_DESC+8, ADDR                           ; 0523
                                59      04      AE  C0  0011F          ADDL2   VMS_DESC+4, ADDR                                ;
                                                06  11  00123          BRB     19$                                             ; 0519
                                                58  D4  00125  18$:     CLRL    POS                                             ; 0527
                                59      04      AE  D0  00127          MOVL    VMS_DESC+4, ADDR                                ; 0528
                                50      07  AA48    9E  0012B  19$:     MOVAB   7(BITS)[POS], R0                                ; 0533
                    5B          50                  08  C7  00130      DIVL3   #8, R0, BYTES                                   ;
                                                    17  13  00134      BEQL    20$                                             ; 0534
                    69          5B                  00  0C  00136      PROBER  #0, BYTES, (ADDR)                               ; 0536
                                                    11  12  0013A      BNEQ    20$                                             ;
                                                    59  DD  0013C      PUSHL   ADDR                                            ; 0538
                                                    01  DD  0013E      PUSHL   #1                                              ;
                                        00028228    8F  DD  00140      PUSHL   #164392                                         ;
                            00000000G   00              03  FB  00146  CALLS   #3, LIB$SIGNAL                                  ;
                                20                      5A  D1  0014D  20$:     CMPL    BITS, #32                               ; 0543
```

```
                                                28  1A 00150          BGTRU   24$                                                    ;
                                            50  02  AE 9A 00152          MOVZBL  VMS_DESC+2, R0                                         ; 0550
                                            06      50 91 00156          CMPB    R0, #6                                                 ; 0552
                                                    05 1F 00159          BLSSU   21$                                                    ;
                                            07      50 91 0015B          CMPB    R0, #7                                                 ;
                                                    0A 1B 0015E          BLEQU   22$                                                    ;
                                            29      50 91 00160 21$:     CMPB    R0, #41                                                ;
                                                    0D 1F 00163          BLSSU   23$                                                    ;
                                            2A      50 91 00165          CMPB    R0, #42                                                ;
                                                    08 1A 00168          BGTRU   23$                                                    ;
    00  B6         69                       5A      58 EE 0016A 22$:     EXTV    POS, BITS, (ADDR), @0(R6)                              ; 0553
                                                    2D 11 00170          BRB     29$                                                    ;
    00  B6         69                       5A      58 EF 00172 23$:     EXTZV   POS, BITS, (ADDR), @0(R6)                              ; 0555
                                                    25 11 00178          BRB     29$                                                    ; 0561
                                                    58 D5 0017A 24$:     TSTL    POS                                                    ; 0570
                                                    07 12 0017C          BNEQ    26$                                                    ;
                00  B6         69               5B      28 0017E          MOVC3   BYTES, (ADDR), @0(R6)                                  ; 0575
                                                    23 11 00183 25$:     BRB     30$                                                    ;
                                            50  FF  AA 9E 00185 26$:     MOVAB   -1(R10), R0                                            ; 0583
                                            50      20 C6 00189          DIVL2   #32, R0                                                ;
                                            51      01 CE 0018C          MNEGL   #1, I                                                  ;
                                                    0A 11 0018F          BRB     28$                                                    ;
                                                6941    DF 00191 27$:     PUSHAL  (ADDR)[I]                                              ; 0585
    00 B641         9E                      20      58 EF 00194          EXTZV   POS, #32, @(SP)+, @0(R6)[I]                            ;
                    F2                      51      50 F3 0019B 28$:     AOBLEQ  R0, I, 27$                                             ; 0584
                                            0D  17  A7 91 0019F 29$:     CMPB    23(RESULT_DESC), #13                                   ; 0591
                                                    03 12 001A3          BNEQ    30$                                                    ;
                                                1C  A7 D4 001A5          CLRL    28(RESULT_DESC)                                        ; 0592
                                            50      57 D0 001A8 30$:     MOVL    RESULT_DESC, R0                                        ; 0601
                                                    04 001AB             RET                                                            ; 0602

; Routine Size:  428 bytes,    Routine Base:  DBG$CODE + 0143
```

```
;  477     0603  1 GLOBAL ROUTINE DBG$FILL_IN_VMS_DESC(fcode,typeid,symid,
;  478     0604  1                                       vms_desc,bit_length,bit_offset) =
;  479     0605  1 !
;  480     0606  1 ! ROUTINE DESCRIPTION
;  481     0607  1 !
;  482     0608  2     BEGIN
;  483     0609  2     MAP
;  484     0610  2         symid           : REF rst$entry,
;  485     0611  2         vms_desc        : REF dbg$stg_desc,
;  486     0612  2         bit_length      : REF VECTOR [1,LONG],
;  487     0613  2         bit_offset      : REF VECTOR [1,LONG];
;  488     0614  2
;  489     0615  2     CASE .fcode FROM rst$k_type_minimum TO rst$k_type_maximum OF
;  490     0616  2         SET
;  491     0617  2         [rst$k_type_atomic] :
;  492     0618  3             BEGIN
;  493     0619  3             LOCAL typecode;
;  494     0620  3             !+
;  495     0621  3             ! Atomic data types - the routine dbg$sta_typ_atomic can be
;  496     0622  3             ! used to obtain the dtype and length in bits. Class is set
;  497     0623  3             ! to S or VS here; it may later be changed to UBS if there
;  498     0624  3             ! is a bit offset present.
;  499     0625  3             !-
;  500     0626  3             dbg$sta_typ_atomic(.typeid,typecode,bit_length[0]);
;  501     0627  3             IF .typecode EQL dst$k_bool THEN
;  502     0628  4                 BEGIN
;  503     0629  4                 vms_desc[dsc$b_class]  = dsc$k_class_s;
;  504     0630  4                 vms_desc[dsc$b_dtype]  = dsc$k_dtype_tf;
;  505     0631  4                 vms_desc[dsc$w_length] = .bit_length[0];
;  506     0632  4                 END
;  507     0633  3             ELSE
;  508     0634  4                 BEGIN
;  509     0635  4                 vms_desc[dsc$b_dtype]  = .typecode;
;  510     0636  4                 IF .typecode EQL dsc$k_dtype_vt
;  511     0637  4                 THEN
;  512     0638  4                     vms_desc[dsc$b_class] = dsc$k_class_vs
;  513     0639  4                 ELSE
;  514     0640  4                     vms_desc[dsc$b_class] = dsc$k_class_s;
;  515     0641  4                 !+
;  516     0642  4                 ! Length is in bits for the five data types below, and
;  517     0643  4                 ! bytes for all others.
;  518     0644  4                 !-
;  519     0645  4                 vms_desc[dsc$w_length] = .bit_length[0]/
;  520     0646  5                     (IF .typecode EQL dsc$k_dtype_v
;  521     0647  5                      OR .typecode EQL dsc$k_dtype_vu
;  522     0648  5                      OR .typecode EQL dsc$k_dtype_sv
;  523     0649  5                      OR .typecode EQL dsc$k_dtype_svu
;  524     0650  5                      OR .typecode EQL dsc$k_dtype_tf
;  525     0651  5                        THEN 1
;  526     0652  4                        ELSE %BPUNIT);
;  527     0653  3                 END;
;  528     0654  2             END;
;  529     0655  2
;  530     0656  2
;  531     0657  2         [rst$k_type_pict] :
;  532     0658  3             BEGIN
;  533     0659  3             !+
```

```
;  534     0660  3              ! "Pictured" data item. This is really an item of data-type
;  535     0661  3              ! dsc$k_dtype_t, but we need to retain extra information in
;  536     0662  3              ! case we want to deposit a numeric value into this item.
;  537     0663  3              !-
;  538     0664  3              dbg$sta_symsize (.typeid,bit_length[0]);
;  539     0665  3              vms_desc[dsc$b_class]  = dsc$k_class_s;
;  540     0666  3              vms_desc[dsc$b_dtype]  = dsc$k_dtype_t;
;  541     0667  3              vms_desc[dsc$w_length] = .bit_length[0]/%BPUNIT;
;  542     0668  2              END;
;  543     0669  2
;  544     0670  2
;  545     0671  2          [rst$k_type_record,rst$k_type_ptr,rst$k_type_tptr,
;  546     0672  2           rst$k_type_enum,rst$k_type_set,rst$k_type_subrng,
;  547     0673  2           rst$k_type_file, rst$k_type_rfa]:
;  548     0674  2              !+
;  549     0675  2              ! Non-atomic (i.e., non-VAX standard) data types. For these,
;  550     0676  2              ! we do not attempt to fill in the VMS descriptor. We just
;  551     0677  2              ! fill in the bit_length.
;  552     0678  2              !-
;  553     0679  2              dbg$sta_symsize(.typeid,bit_length[0]);
;  554     0680  2
;  555     0681  2
;  556     0682  2          ! For variants, there is nothing to fill in.
;  557     0683  2          !
;  558     0684  2          [rst$k_type_variant]:
;  559     0685  2              0;
;  560     0686  2
;  561     0687  2          [rst$k_type_descr] :
;  562     0688  3              BEGIN
;  563     0689  3              !+
;  564     0690  3              ! Types described by descriptors. We use the routine dbg$sta_typ_descr
;  565     0691  3              ! to obtain the class, dtype, and length information.
;  566     0692  3              !-
;  567     0693  3              LOCAL dst_desc : REF dbg$stg_desc;
;  568     0694  3              dbg$sta_typ_descr(.typeid,dst_desc);
;  569     0695  3
;  570     0696  3              ! If we got a symid passed in to this routine,
;  571     0697  3              ! try calling SYMVALUE to get a
;  572     0698  3              ! descriptor. If we get one, then use this
;  573     0699  3              ! descriptor instead of the one we got back from STA_TYP_DESCR.
;  574     0700  3              !
;  575     0701  3              ! Note - normally, these 2 descriptors will be the same.
;  576     0702  3              ! However, for dynamic arrays in PASCAL, the runtime descriptor
;  577     0703  3              ! (which we get back when we call SYMVALUE with the symid) is
;  578     0704  3              ! correct, but the compile-time descriptor (which is part of
;  579     0705  3              ! the typespec) is wrong. This code is a workaround for this
;  580     0706  3              ! problem in the PASCAL DST. The same workaround appears
;  581     0707  3              ! in DBGPARSER for array descriptors.
;  582     0708  3              !
;  583     0709  3              IF .SYMID NEQ 0
;  584     0710  3              THEN
;  585     0711  4                  BEGIN
;  586     0712  4                  LOCAL
;  587     0713  4                      DESC: VECTOR[3],
;  588     0714  4                      RSTPTR: REF RST$ENTRY,
;  589     0715  4                      VALUE_KIND;
;  590     0716  4                  RSTPTR = .SYMID;
```

```
;  591     0717  4                WHILE .RSTPTR[RST$B_KIND] NEQ RST$K_MODULE DO
;  592     0718  4                    RSTPTR = .RSTPTR[RST$L_UPSCOPEPTR];
;  593     0719  4                IF .RSTPTR[RST$B_LANGUAGE] EQL DBG$K_PASCAL
;  594     0720  4                THEN
;  595     0721  5                    BEGIN
;  596     0722  5                    DBG$STA_SETCONTEXT(.SYMID);
;  597     0723  5                    DBG$STA_SYMVALUE(.SYMID, DESC, VALUE_KIND);
;  598     0724  5                    IF .VALUE_KIND EQL DBG$K_VAL_DESCR
;  599     0725  5                    THEN
;  600     0726  5                        DST_DESC = .DESC[0];
;  601     0727  4                    END;
;  602     0728  3                END;
;  603     0729  3
;  604     0730  3            vms_desc[dsc$b_class]  = .dst_desc[dsc$b_class];
;  605     0731  3            vms_desc[dsc$b_dtype]  = .dst_desc[dsc$b_dtype];
;  606     0732  3            vms_desc[dsc$w_length] = .dst_desc[dsc$w_length];
;  607     0733  3
;  608     0734  3            !+
;  609     0735  3            ! Fix things up so that dtype VT always corresponds to class VS.
;  610     0736  3            ! (This seems to be necessary for PL/I varying strings).
;  611     0737  3            !-
;  612     0738  3            IF .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_vt
;  613     0739  3            THEN
;  614     0740  3              vms_desc[dsc$b_class] = dsc$k_class_vs;
;  615     0741  3
;  616     0742  3            SELECTONE .vms_desc[dsc$b_class] OF
;  617     0743  3                SET
;  618     0744  3                [dsc$k_class_s,dsc$k_class_d,dsc$k_class_vs] : 0;
;  619     0745  3
;  620     0746  3                [dsc$k_class_sd] :
;  621     0747  4                    BEGIN
;  622     0748  4                    vms_desc[dsc$b_digits] = .dst_desc[dsc$b_digits];
;  623     0749  4                    vms_desc[dsc$b_scale]  = .dst_desc[dsc$b_scale];
;  624     0750  4                    vms_desc[dsc$v_fl_binscale] = .dst_desc[dsc$v_fl_binscale];
;  625     0751  4
;  626     0752  4                    !+
;  627     0753  4                    !*** Workaround for a problem in the PL/I DST.
;  628     0754  4                    !*** The scale they are giving us is the negative
;  629     0755  4                    !*** of what we expect.
;  630     0756  4                    !-
;  631     0757  4                    IF .symid NEQ 0
;  632     0758  4                    THEN
;  633     0759  5                        BEGIN
;  634     0760  5                        WHILE .symid[rst$b_kind] NEQ rst$k_module DO
;  635     0761  5                            symid = .symid[rst$l_upscopeptr];
;  636     0762  5                        IF (.symid[rst$b_language] EQL dbg$k_pli) AND
;  637     0763  5                            .symid[rst$v_oldpliflag]
;  638     0764  5                        THEN
;  639     0765  5                            vms_desc[dsc$b_scale] = - .dst_desc[dsc$b_scale];
;  640     0766  4                        END;
;  641     0767  3                    END;
;  642     0768  3
;  643     0769  3                [dsc$k_class_ubs] :
;  644     0770  3                    SELECTONE .dst_desc[dsc$b_dtype] OF
;  645     0771  3                        SET
;  646     0772  3                        [dsc$k_dtype_syu,dsc$k_dtype_vu,dsc$k_dtype_tf]:
;  647     0773  3                            bit_offset[0] = .bit_offset[0] + .dst_desc[dsc$l_pos];
```

```
;  648     0774  3                    [dsc$k_dtype_ubs]:
;  649     0775  3                        BEGIN
;  650     0776  4                        bit_offset[0] = .bit_offset[0] + .(.dst_desc+8)<0,16,1>;
;  651     0777  4                        IF .(.dst_desc+10)<0,1,0>
;  652     0778  4                            THEN vms_desc[dsc$b_dtype] = dsc$k_dtype_svu
;  653     0779  4                            ELSE vms_desc[dsc$b_dtype] = dsc$k_dtype_vu;
;  654     0780  4                        END;
;  655     0781  3
;  656     0782  3                    [OTHERWISE]:
;  657     0783  3                        0;
;  658     0784  3                    TES;
;  659     0785  3
;  660     0786  3                [OTHERWISE] :
;  661     0787  3                    SIGNAL(dbg$_unimplent);
;  662     0788  3                TES;
;  663     0789  3
;  664     0790  3            IF .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_bpv
;  665     0791  3            THEN
;  666     0792  3                BEGIN
;  667     0793  4                vms_desc[dsc$b_class]  = dsc$k_class_z;
;  668     0794  4                vms_desc[dsc$b_dtype]  = dsc$k_dtype_zem;
;  669     0795  4                vms_desc[dsc$w_length] = 2;
;  670     0796  4                dbg$gl_call_context = .dst_desc[dsc$a_frame];
;  671     0797  4                END
;  672     0798  4            ELSE IF .vms_desc[dsc$b_dtype] EQL dsc$k_dtype_blv
;  673     0799  3            THEN
;  674     0800  3                BEGIN
;  675     0801  4                vms_desc[dsc$b_class]  = dsc$k_class_z;
;  676     0802  4                vms_desc[dsc$b_dtype]  = dsc$k_dtype_zi;
;  677     0803  4                vms_desc[dsc$w_length] =
;  678     0804  4                    (dbg$ins_decode(.vms_desc[dsc$a_pointer],false,false) -
;  679     0805  5                                    .vms_desc[dsc$a_pointer]);
;  680     0806  4                dbg$gl_call_context = .dst_desc[dsc$a_frame];
;  681     0807  4                END;
;  682     0808  3
;  683     0809  3                bit_length[0] = dbg$data_length(.vms_desc);
;  684     0810  3                END;
;  685     0811  2
;  686     0812  2            ! Self relative labels in PL/I (i.e., arrays of labels).
;  687     0813  2            ! The value of one of these is equal to the contents of the
;  688     0814  2            ! memory location plus its own address. In other words, the
;  689     0815  2            ! values actually stored in the label array are offsets to
;  690     0816  2            ! the actual place to branch to.
;  691     0817  2            !
;  692     0818  2            [rst$k_type_self_rel_lab]:
;  693     0819  2                BEGIN
;  694     0820  3                vms_desc[dsc$b_class] = dsc$k_class_z;
;  695     0821  3                vms_desc[dsc$b_dtype] = dsc$k_dtype_zi;
;  696     0822  3                vms_desc[dsc$a_pointer] = .vms_desc[dsc$a_pointer] +
;  697     0823  3                    .(.vms_desc[dsc$a_pointer]);
;  698     0824  3                vms_desc[dsc$w_length] =
;  699     0825  3                    (dbg$ins_decode(.vms_desc[dsc$a_pointer],false,false) -
;  700     0826  4                                    .vms_desc[dsc$a_pointer]);
;  701     0827  3                bit_length[0] = .vms_desc[dsc$w_length] * 8;
;  702     0828  3                END;
;  703     0829  2
;  704     0830  2
```

```
;  705          0831  2          ! We do not handle any other fcodes.
;  706          0832  2          !
;  707          0833  2          [INRANGE,OUTRANGE] :
;  708          0834  2              SIGNAL(dbg$_unimplent);
;  709          0835  2          TES;
;  710          0836  2
;  711          0837  2      RETURN sts$k_success;
;  712          0838  1      END;                            ! End of routine 'dbg$fill_in_vms_desc'



                                                03FC 00000          .ENTRY  DBG$FILL_IN_VMS_DESC, Save R2,R3,R4,R5,R6,- ; 0603
                                                                            R7,R8,R9
                                    59 00000000G   00   9E 00002    MOVAB   LIB$SIGNAL, R9                              ;
                                    58 00000000G   00   9E 00009    MOVAB   DBG$INS_DECODE, R8                          ;
                                    57 00000000G   00   9E 00010    MOVAB   DBG$STA_SYMSIZE, R7                         ;
                                    5E             18   C2 00017    SUBL2   #24, SP                                     ;
                              15          01   04   AC   CF 0001A    CASEL   FCODE, #1, #21                              ; 0615
                00C6          00D2      0037        002C     0001F 1$:  .WORD   2$-1$,-                                  ;
                00C6          00C6      00C6        00A9     00027              3$-1$,-                                  ;
                002C          002C      002C        00C6     0002F              14$-1$,-                                 ;
                00C6          00C6      002C        002C     00037              12$-1$,-                                 ;
                00C6          0234      002C        002C     0003F              11$-1$,-                                 ;
                                        002C        0212     00047              12$-1$,-                                 ;
                                                                                12$-1$,-                                 ;
                                                                                12$-1$,-                                 ;
                                                                                12$-1$,-                                 ;
                                                                                2$-1$,-                                  ;
                                                                                12$-1$,-                                 ;
                                                                                2$-1$,-                                  ;
                                                                                2$-1$,-                                  ;
                                                                                2$-1$,-                                  ;
                                                                                2$-1$,-                                  ;
                                                                                12$-1$,-                                 ;
                                                                                12$-1$,-                                 ;
                                                                                2$-1$,-                                  ;
                                                                                2$-1$,-                                  ;
                                                                                33$-1$,-                                 ;
                                                                                12$-1$,-                                 ;
                                                                                32$-1$,-                                 ;
                                                                                2$-1$                                    ;
                          00028800      8F   DD 0004B 2$:      PUSHL   #165888                                     ; 0834
                      69                01   FB 00051          CALLS   #1, LIB$SIGNAL                              ;
                                        70   11 00054          BRB     10$                                         ;
                                  14    AC   DD 00056 3$:      PUSHL   BIT_LENGTH                                  ; 0626
                                  04    AE   9F 00059          PUSHAB  TYPECODE                                    ;
                                  08    AC   DD 0005C          PUSHL   TYPEID                                      ;
                00000000G         00        03   FB 0005F          CALLS   #3, DBG$STA_TYP_ATOMIC                      ;
                                  50    10   AC   D0 00066          MOVL    VMS_DESC, R0                                ; 0629
                                  51    10   AC   D0 0006A          MOVL    VMS_DESC, R1                                ; 0630
                                  52        6E   D0 0006E          MOVL    TYPECODE, R2                                ; 0627
                0000009E          8F        52   D1 00071          CMPL    R2, #158                                    ;
                                            0F   12 00078          BNEQ    4$                                          ;
                          03      A0        01   90 0007A          MOVB    #1, 3(R0)                                   ; 0629
                          02      A1        28   90 0007E          MOVB    #40, 2(R1)                                  ; 0630
                          10      BC   14   BC   B0 00082          MOVW    @BIT_LENGTH, @VMS_DESC                      ; 0631
```

```
                                65  11 00087          BRB     13$                                     ; 0627
                          02    A1                    52  90 00089 4$:      MOVB    R2, 2(R1)       ; 0635
                          25                          52  D1 0008D          CMPL    R2, #37         ; 0636
                                                      06  12 00090          BNEQ    5$
                          03    A0                    0B  90 00092          MOVB    #11, 3(R0)      ; 0638
                                                      04  11 00096          BRB     6$
                          03    A0                    01  90 00098 5$:      MOVB    #1, 3(R0)       ; 0640
                          01                          52  D1 0009C 6$:      CMPL    R2, #1          ; 0646
                                                      14  13 0009F          BEQL    7$
                          22                          52  D1 000A1          CMPL    R2, #34         ; 0647
                                                      0F  13 000A4          BEQL    7$
                          29                          52  D1 000A6          CMPL    R2, #41         ; 0648
                                                      0A  13 000A9          BEQL    7$
                          2A                          52  D1 000AB          CMPL    R2, #42         ; 0649
                                                      05  13 000AE          BEQL    7$
                          28                          52  D1 000B0          CMPL    R2, #40         ; 0650
                                                      05  12 000B3          BNEQ    8$
                          50                          01  D0 000B5 7$:      MOVL    #1, R0          ; 0646
                                                      03  11 000B8          BRB     9$
                          50                          08  D0 000BA 8$:      MOVL    #8, R0
                51        14    BC                    50  C7 000BD 9$:      DIVL3   R0, @BIT_LENGTH, R1
                          10    BC                    51  B0 000C2          MOVW    R1, @VMS_DESC
                                                      26  11 000C6 10$:      BRB     13$             ; 0615
                                              14      AC  DD 000C8 11$:      PUSHL   BIT_LENGTH      ; 0664
                                              08      AC  DD 000CB          PUSHL   TYPEID
                          67                          02  FB 000CE          CALLS   #2, DBG$STA_SYMSIZE
                          50                  10      AC  D0 000D1          MOVL    VMS_DESC, R0    ; 0665
                          02    A0            010E    8F  B0 000D5          MOVW    #270, 2(R0)     ; 0666
                51        14    BC                    08  C7 000DB          DIVL3   #8, @BIT_LENGTH, R1 ; 0667
                                60                    51  B0 000E0          MOVW    R1, (R0)
                                                      09  11 000E3          BRB     13$             ; 0615
                                              14      AC  DD 000E5 12$:      PUSHL   BIT_LENGTH      ; 0679
                                              08      AC  DD 000E8          PUSHL   TYPEID
                                      67              02  FB 000EB          CALLS   #2, DBG$STA_SYMSIZE
                                            0162      31  000EE 13$:         BRW     33$
                                              04      AE  9F 000F1 14$:      PUSHAB  DST_DESC        ; 0694
                                              08      AC  DD 000F4          PUSHL   TYPEID
                00000000G       00                    02  FB 000F7          CALLS   #2, DBG$STA_TYP_DESCR
                                52            0C      AC  D0 000FE          MOVL    SYMID, R2       ; 0709
                                                      56  D4 00102          CLRL    R6
                                                      52  D5 00104          TSTL    R2
                                                      3A  13 00106          BEQL    17$
                                                      56  D6 00108          INCL    R6
                                50                    52  D0 0010A          MOVL    R2, RSTPTR      ; 0716
                                01            14      A0  91 0010D 15$:      CMPB    20(RSTPTR), #1  ; 0717
                                                      06  13 00111          BEQL    16$
                                50            10      A0  D0 00113          MOVL    16(RSTPTR), RSTPTR ; 0718
                                                      F4  11 00117          BRB     15$
                                06            29      A0  91 00119 16$:      CMPB    41(RSTPTR), #6  ; 0719
                                                      23  12 0011D          BNEQ    17$
                                                      52  DD 0011F          PUSHL   R2              ; 0722
                00000000G       00                    01  FB 00121          CALLS   #1, DBG$STA_SETCONTEXT
                                              08      AE  9F 00128          PUSHAB  VALUE_KIND      ; 0723
                                              10      AE  9F 0012B          PUSHAB  DESC
                                                      52  DD 0012E          PUSHL   R2
                00000000G       00                    03  FB 00130          CALLS   #3, DBG$STA_SYMVALUE
                                03            08      AE  D1 00137          CMPL    VALUE_KIND, #3  ; 0724
```

```
                                                05  12  0013B          BNEQ    17$
                                04  AE      0C  AE  D0  0013D          MOVL    DESC, DST_DESC                          ;0726
                                52  10      AC  D0  00142  17$:       MOVL    VMS_DESC, R2                            ;0730
                                54  03      A2  9E  00146              MOVAB   3(R2), R4
                                53  04      AE  D0  0014A              MOVL    DST_DESC, R3
                                64  03      A3  90  0014E              MOVB    3(R3), (R4)
                                55  02      A2  9E  00152              MOVAB   2(R2), R5                               ;0731
                                50  02      A3  9A  00156              MOVZBL  2(R3), R0
                                65  50      90  0015A                  MOVB    R0, (R5)
                                62  63      B0  0015D                  MOVW    (R3), (R2)                              ;0732
                                25  65      91  00160                  CMPB    (R5), #37                               ;0738
                                    03      12  00163                  BNEQ    18$
                                64  0B      90  00165                  MOVB    #11, (R4)                               ;0740
                                64  95      00168  18$:                TSTB    (R4)                                    ;0744
                                    05      13  0016A                  BEQL    19$
                                02  64      91  0016C                  CMPB    (R4), #2
                                    7A      1B  0016F                  BLEQU   26$
                                0B  64      91  00171  19$:           CMPB    (R4), #11
                                    75      13  00174                  BEQL    26$
                                09  64      91  00176                  CMPB    (R4), #9                                ;0746
                                    3B      12  00179                  BNEQ    22$
                            08  A2      08  A3  B0  0017B              MOVW    8(R3), 8(R2)                            ;0749
    50      0A  A3                  01  03  EF  00180                  EXTZV   #3, #1, 10(R3), R0                      ;0750
0A  A2              01              03  50  F0  00186                  INSV    R0, #3, #1, 10(R2)
                                67  56  E9  0018C                  BLBC    R6, 28$                                 ;0757
                                50  0C  AC  D0  0018F  20$:           MOVL    SYMID, R0                               ;0760
                                01  14  A0  91  00193                  CMPB    20(R0), #1
                                    07  13  00197                      BEQL    21$
                            0C  AC      10  A0  D0  00199              MOVL    16(R0), SYMID                           ;0761
                                    EF  11  0019E                      BRB     20$
                                50  0C  AC  D0  001A0  21$:           MOVL    SYMID, R0                               ;0762
                                05  29  A0  91  001A4                  CMPB    41(R0), #5
                                    4C  12  001A8                      BNEQ    28$
47          28  A0                  05  E1  001AA                  BBC     #5, 40(R0), 28$                         ;0763
            08  A2      08  A3  8E  001AF                  MNEGB   8(R3), 8(R2)                            ;0765
                                    40  11  001B4                  BRB     28$                                     ;0742
                                0D  64  91  001B6  22$:           CMPB    (R4), #13                               ;0769
                                    32  12  001B9                  BNEQ    27$
                                22  50  91  001BB                  CMPB    R0, #34                                 ;0772
                                    0A  13  001BE                  BEQL    23$
                                28  50  91  001C0                  CMPB    R0, #40
                                    05  13  001C3                  BEQL    23$
                                2A  50  91  001C5                  CMPB    R0, #42
                                    07  12  001C8                  BNEQ    24$
                            18  BC      08  A3  C0  001CA  23$:       ADDL2   8(R3), @BIT_OFFSET                      ;0773
                                    25  11  001CF                  BRB     28$
                            A1  8F          50  91  001D1  24$:       CMPB    R0, #161                                ;0775
                                    1F  12  001D5                  BNEQ    28$
                                50  08  A3  32  001D7              CVTWL   8(R3), R0                               ;0777
                            18  BC      50  C0  001DB              ADDL2   R0, @BIT_OFFSET
                                05  0A  A3  E9  001DF              BLBC    10(R3), 25$                             ;0778
                                65  2A  90  001E3                  MOVB    #42, (R5)                               ;0779
                                    0E  11  001E6                  BRB     28$
                                65  22  90  001E8  25$:           MOVB    #34, (R5)                               ;0780
                                    09  11  001EB  26$:           BRB     28$                                     ;0770
                        00028800    8F  DD  001ED  27$:           PUSHL   #165888                                 ;0788
                                69  01  FB  001F3                  CALLS   #1, LIB$SIGNAL
```

```
                                20              65  91 001F6 28$:   CMPB    (R5), #32                       ; 0791
                                                0A  12 001F9        BNEQ    29$
                                                64  94 001FB        CLRB    (R4)                            ; 0794
                                65              17  90 001FD        MOVB    #23, (R5)                       ; 0795
                                62              02  B0 00200        MOVW    #2, (R2)                        ; 0796
                                                17  11 00203        BRB     30$                             ; 0797
                                21              65  91 00205 29$:   CMPB    (R5), #33                       ; 0799
                                                1A  12 00208        BNEQ    31$
                                                64  94 0020A        CLRB    (R4)                            ; 0802
                                65              16  90 0020C        MOVB    #22, (R5)                       ; 0803
                                                7E  7C 0020F        CLRQ    -(SP)                           ; 0805
                            04  A2              DD  00211        PUSHL   4(R2)
                                68              03  FB 00214        CALLS   #3, DBG$INS_DECODE
                        62      50          04  A2  A3 00217        SUBW3   4(R2), R0, (R2)                 ; 0806
                00000000G       00          08  A3  D0 0021C 30$:   MOVL    8(R3), DBG$GL_CALL_CONTEXT      ; 0807
                                                52  DD 00224 31$:   PUSHL   R2                              ; 0810
                      FAE6      CF                  01  FB 00226        CALLS   #1, DBG$DATA_LENGTH
                        14      BC                  50  D0 0022B        MOVL    R0, @BIT_LENGTH
                                                    22  11 0022F        BRB     33$                             ; 0615
                                52          10  AC  D0 00231 32$:   MOVL    VMS_DESC, R2                    ; 0821
                            02  A2                  16  B0 00235        MOVW    #22, 2(R2)                      ; 0822
                            04  A2              04  B2  C0 00239        ADDL2   @4(R2), 4(R2)                   ; 0824
                                                7E  7C 0023E        CLRQ    -(SP)                           ; 0826
                            04  A2              DD  00240        PUSHL   4(R2)
                                68              03  FB 00243        CALLS   #3, DBG$INS_DECODE
                        62      50          04  A2  A3 00246        SUBW3   4(R2), R0, (R2)                 ; 0827
                                50              62  3C 0024B        MOVZWL  (R2), R0                        ; 0828
                    14  BC      50              03  78 0024E        ASHL    #3, R0, @BIT_LENGTH
                                50              01  D0 00253 33$:   MOVL    #1, R0                          ; 0837
                                                    04 00256        RET                                     ; 0838

; Routine Size:  599 bytes,    Routine Base:  DBG$CODE + 02EF
```

```
;  714     0839  1 GLOBAL ROUTINE DBG$MAKE_VMS_DESC (prm_desc,vms_desc) =
;  715     0840  1 !
;  716     0841  1 ! FUNCTIONAL DESCRIPTION:
;  717     0842  1 !
;  718     0843  1 !       This routine constructs a VAX/VMS descriptor that points to the
;  719     0844  1 !       value of a symbol (described by a primary symbol descriptor).
;  720     0845  1 !       It first materializes the address by resolving all array and/or
;  721     0846  1 !       record component references, and then determines the length and
;  722     0847  1 !       data type.  Finally it checks for sub references, and if one is
;  723     0848  1 !       present updates the length and address fields accordingly.
;  724     0849  1 !
;  725     0850  1 ! FORMAL PARAMETERS:
;  726     0851  1 !
;  727     0852  1 !       prm_desc        - A longword containing the address of a primary descriptor
;  728     0853  1 !
;  729     0854  1 !       vms_desc        - A longword containing the address of a block of at least
;  730     0855  1 !                         12 bytes where a VAX/VMS descriptor will be constructed.
;  731     0856  1 !
```

```
;  733     0857  1 !dbg$make_vms_desc (prm_desc,vms_desc)
;  734     0858  2     BEGIN
;  735     0859  2     MAP
;  736     0860  2         prm_desc        : REF dbg$primary,
;  737     0861  2         vms_desc        : REF dbg$stg_desc;
;  738     0862  2
;  739     0863  2     LOCAL
;  740     0864  2         adr_kind,
;  741     0865  2         adr_ptrs        : VECTOR [3,LONG],
;  742     0866  2         addr_offset,
;  743     0867  2         bit_offset,
;  744     0868  2         bit_length,
;  745     0869  2         result_desc     : BLOCK [12,BYTE],
;  746     0870  2         data_subnode    : REF dbg$prim_node,
;  747     0871  2         prim_subnode    : REF dbg$prim_node,
;  748     0872  2         typeid: REF rst$entry,
;  749     0873  2         s_value;
;  750     0874  2
;  751     0875  2     BUILTIN
;  752     0876  2         PROBER;
;  753     0877  2
;  754     0878  2     dbg$gl_current_primary = .prm_desc;                                 ! A003
;  755     0879  2
;  756     0880  2     ! It is illegal to call DBG$MAKE_VMS_DESC with a type.
;  757     0881  2     !
;  758     0882  2     IF .prm_desc[dbg$b_dhdr_kind] EQL rst$k_type
;  759     0883  2     THEN
;  760     0884  3         BEGIN
;  761     0885  3         LOCAL
;  762     0886  3             name;
;  763     0887  3         IF .prm_desc[dbg$l_dhdr_symid0] NEQ 0
;  764     0888  3         THEN
;  765     0889  4             BEGIN
;  766     0890  4             dbg$sta_symname(.prm_desc[dbg$l_dhdr_symid0], name);
;  767     0891  4             SIGNAL (dbg$_novaltyp, 1, .name);
;  768     0892  4             END
;  769     0893  3         ELSE
;  770     0894  3             SIGNAL (dbg$_novalue);
;  771     0895  2         END;
;  772     0896  2
;  773     0897  2     !+
;  774     0898  2     ! The first thing we do is to set the symbol table access context to
;  775     0899  2     ! the correct stack frame (in case of values whose address is given by
;  776     0900  2     ! an offset from an address in a register such as FP or AP), and clear
;  777     0901  2     ! the initial byte and bit addresses and descriptor fields.
;  778     0902  2     !-
;  779     0903  2     dbg$sta_setcontext(.prm_desc[dbg$l_dhdr_symid0]);
;  780     0904  2     bit_offset = bit_length = 0;
;  781     0905  2     ch$fill(0,12,result_desc);
```

```
;  783          0906  2      ! Loop for all except the last sub-node in the primary descriptor,
;  784          0907  2      ! building up the address of the element within the structure.
;  785          0908  2      ! This address is represented by a byte address, stored in the
;  786          0909  2      ! dsc$a_pointer field of result_desc, and a bit offset, stored
;  787          0910  2      ! (for now) in the local variable bit_offset.
;  788          0911  2      !
;  789          0912  2      ! N.B.  This MUST be done 'top-down' to deal with POINTER data-types.
;  790          0913  2      !-
;  791          0914  2      prim_subnode = .prm_desc[dbg$l_prim_flink];
;  792          0915  2      data_subnode = .prm_desc[dbg$l_prim_blink];
;  793          0916  2      WHILE .prim_subnode NEQA .data_subnode DO
;  794          0917  2          BEGIN
;  795          0918  3          result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + .prim_subnode[dbg$l_pnode_reloc];
;  796          0919  3          !+
;  797          0920  3          ! Get the address of this element by calling dbg$sta_symvalue.
;  798          0921  3          ! This should return either the address of a data item (offset
;  799          0922  3          ! from the start of the record, if this is a record component)
;  800          0923  3          ! or the address of a descriptor. Anything else is an error.
;  801          0924  3          !-
;  802          0925  3          IF (.prim_subnode[dbg$l_pnode_symid] NEQ 0) AND
;  803          0926  3          NOT .prim_subnode[dbg$v_pnode_ignore]
;  804          0927  3          THEN
;  805          0928  4              BEGIN
;  806          0929  4              dbg$sta_symvalue(.prim_subnode[dbg$l_pnode_symid],adr_ptrs,adr_kind);
;  807          0930  4              SELECTONE .adr_kind OF
;  808          0931  4                  SET
;  809          0932  4                  [dbg$k_val_addr]:
;  810          0933  5                      BEGIN
;  811          0934  5                      result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + .adr_ptrs[0];
;  812          0935  5                      bit_offset = .bit_offset + .adr_ptrs[1];
;  813          0936  4                      END;
;  814          0937  4                  [dbg$k_val_descr]:
;  815          0938  5                      BEGIN
;  816          0939  5                      BIND adr_desc = adr_ptrs[0] : REF dbg$stg_desc;
;  817          0940  5                      result_desc[dsc$a_pointer]  = .result_desc[dsc$a_pointer] + .adr_desc[dsc$a_pointer];
;  818          0941  4                      END;
;  819          0942  4                  [dbg$k_val_unalloc]:
;  820          0943  5                      BEGIN
;  821          0944  5                      LOCAL name;
;  822          0945  5                      dbg$sta_symname(.prim_subnode[dbg$l_pnode_symid],name);
;  823          0946  5                      SIGNAL(dbg$_unallocated, 1, .name);
;  824          0947  4                      END;
;  825          0948  4                  [OTHERWISE]:
;  826          0949  4                      SIGNAL(dbg$_novalue);
;  827          0950  4                  TES;
;  828          0951  3              END;
```

```
;  830     0952  3          SELECTONE .prim_subnode[dbg$b_pnode_fcode] OF
;  831     0953  3              SET
;  832     0954  3              [rst$k_type_array]:
;  833     0955  4                  BEGIN
;  834     0956  4                  !+
;  835     0957  4                  ! If this is an array (of records or pointers, presumably),
;  836     0958  4                  ! get the address of an individual element of the array.
;  837     0959  4                  !-
;  838     0960  4                  BIND pnsub = prim_subnode[dbg$a_pnarr_svector] : dbg$prim_node_subs;
;  839     0961  4                  addr_offset = .prim_subnode[dbg$l_pnarr_offset];
;  840     0962  4
;  841     0963  4                  ! Loop through the dimensions of the array.
;  842     0964  4                  !
;  843     0965  4                  DECR index FROM .prim_subnode[dbg$b_pnarr_dimcnt]-1 TO 0 DO
;  844     0966  5                    BEGIN
;  845     0967  5                    s_value = .pnsub[.index,dbg$l_pnsub_svalue];
;  846     0968  5                    typeid = .pnsub[.index,dbg$l_pnsub_typeid];
;  847     0969  5
;  848     0970  5                    ! If the array is indexed by an enumeration type,
;  849     0971  5                    ! then index by the position and not by
;  850     0972  5                    ! the value. This cases arises only in ADA for
;  851     0973  5                    ! enumerated types with representation specs.
;  852     0974  5                    !
;  853     0975  5                    IF (.dbg$gb_language EQL dbg$k_ada) AND
;  854     0976  6                       (.typeid NEQ 0)
;  855     0977  5                    THEN
;  856     0978  6                      IF (.typeid[rst$b_fcode] EQL rst$k_type_enum)
;  857     0979  5                      THEN
;  858     0980  5                        s_value = dbg$enum_pos(.typeid,.s_value);
;  859     0981  5
;  860     0982  5                    addr_offset = .addr_offset + (.s_value*.pnsub[.index,dbg$l_pnsub_stride]);
;  861     0983  4                    END;
;  862     0984  4
;  863     0985  4                  IF .prim_subnode[dbg$v_pnarr_bitref]
;  864     0986  4                    THEN bit_offset = .bit_offset + .addr_offset
;  865     0987  4                    ELSE result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + .addr_offset;
;  866     0988  3                  END;
;  867     0989  3
;  868     0990  3              [rst$k_type_ptr,rst$k_type_tptr]:
;  869     0991  4                  BEGIN
;  870     0992  4                  BUILTIN PROBER;
;  871     0993  4                  LOCAL addr;
;  872     0994  4                  !+
;  873     0995  4                  ! Check for read access before trying to fetch a value.
;  874     0996  4                  !-
;  875     0997  4                  addr = .result_desc[dsc$a_pointer] + .bit_offset<3,29,1>;
;  876     0998  4                  IF NOT PROBER(%REF(0),%REF(5),.addr)
;  877     0999  4                    THEN SIGNAL(dbg$_noaccessr,1,.addr);
;  878     1000  4                  !+
;  879     1001  4                  ! If this is a POINTER, fetch a longword value
;  880     1002  4                  !-
;  881     1003  4                  result_desc[dsc$a_pointer] = .(.addr)<.bit_offset,32,0>;
;  882     1004  4                  bit_offset = 0;
;  883     1005  3                  END;
;  884     1006  3
;  885     1007  3              [rst$k_type_record]:
;  886     1008  3                  0;
```

```
;  887          1009  3
;  888          1010  3                [rst$k_type_variant]:
;  889          1011  3                    IF NOT .prim_subnode[dbg$v_pnvar_valid] THEN
;  890          1012  4                        BEGIN
;  891          1013  4                        LOCAL tag_value,tag_size,tag_name : REF VECTOR[,BYTE];
;  892          1014  4                        prim_subnode[dbg$v_pnvar_valid] = true;
;  893          1015  4                        IF .prim_subnode[dbg$l_pnvar_tagid] NEQ 0 THEN
;  894          1016  5                            BEGIN
;  895          1017  5                            BUILTIN PROBER;                                                     ! A0
;  896          1018  5                            dbg$sta_symname(.prim_subnode[dbg$l_pnvar_tagid],tag_name);
;  897          1019  5                            dbg$sta_symsize(.prim_subnode[dbg$l_pnvar_tagid],tag_size);
;  898          1020  5                            IF (.tag_size NEQ 0) AND (.tag_name[0] NEQ 0) THEN
;  899          1021  6                                BEGIN
;  900          1022  6                                dbg$sta_symvalue(.prim_subnode[dbg$l_pnvar_tagid],adr_ptrs,adr_kind);
;  901          1023  6                                adr_ptrs[0] = .adr_ptrs[0] + .result_desc[dsc$a_pointer];
;  902          1024  6                                adr_ptrs[1] = .adr_ptrs[1] + .bit_offset;
;  903          1025  6
;  904          1026  6                                !++
;  905          1027  6                                ! Check that the address is accessable
;  906          1028  6                                !--
;  907          1029  6                                IF NOT PROBER( %REF(0), %REF(4), .adr_ptrs[0] )             ! A0
;  908          1030  6                                THEN                                                        ! A0
;  909          1031  6                                    SIGNAL(dbg$_noaccessr,1,.adr_ptrs[0]);                  ! A0
;  910          1032  6
;  911          1033  6                                tag_value = .(.adr_ptrs[0])<.adr_ptrs[1],.tag_size,0>;
;  912          1034  6                                IF NOT dbg$sta_variant_value(.tag_value,.prim_subnode[dbg$l_pnvar_dstptr])
;  913          1035  6                                  THEN SIGNAL(dbg$_badtagval,2,.tag_value,tag_name[0]);
;  914          1036  5                                END;
;  915          1037  4                            END;
;  916          1038  3                        END;
;  917          1039  3
;  918          1040  3                [rst$k_type_file]:
;  919          1041  4                    BEGIN
;  920          1042  4                    BUILTIN PROBER;
;  921          1043  4                    LOCAL addr: REF BITVECTOR[];
;  922          1044  4                    !+
;  923          1045  4                    ! For file types what we have in the vms descriptor is a
;  924          1046  4                    ! pointer to a PASCAL file descriptor. Bit 16 in the
;  925          1047  4                    ! second longword of this descriptor is a "valid" bit which
;  926          1048  4                    ! basically says whether the file is open. If bit 16 is set
;  927          1049  4                    ! then the first longword of the descriptor is a pointer
;  928          1050  4                    ! to a buffer from which we can read the next item
;  929          1051  4                    ! in the file.
;  930          1052  4                    !-
;  931          1053  4                    !+
;  932          1054  4                    ! Note in the calculations below, bit_offset will normally
;  933          1055  4                    ! be zero. It might conceivably be non-zero in obscure cases,
;  934          1056  4                    ! such as a file variable which is an element of a packed
;  935          1057  4                    ! record.
;  936          1058  4                    !-
;  937          1059  4                    !+
;  938          1060  4                    ! Check for read access.
;  939          1061  4                    !-
;  940          1062  4                    addr = .result_desc[dsc$a_pointer] + .bit_offset<3,29,1>;
;  941          1063  4                    bit_offset = .bit_offset<0,3,0>;
;  942          1064  4                    IF NOT PROBER(%REF(0),%REF(8),.addr) THEN SIGNAL(dbg$_illfilptr);
;  943          1065  4                    !+
```

```
;  944          1066  4                ! Check "valid" bit.
;  945          1067  4                !-
;  946          1068  4                IF NOT .addr[48+.bit_offset] THEN SIGNAL(dbg$_illfilptr);
;  947          1069  4                !+
;  948          1070  4                ! Put the buffer address back into result_desc.
;  949          1071  4                !-
;  950          1072  4                result_desc[dsc$a_pointer] = .(.addr)<.bit_offset,32,0>;
;  951          1073  4                bit_offset = 0;
;  952          1074  3                END;
;  953          1075  2
;  954          1076  3            [OTHERWISE]:
;  955          1077  3                SIGNAL(dbg$_illtype);
;  956          1078  2
;  957          1079  3            TES;
;  958          1080  3        prim_subnode = .prim_subnode[dbg$l_pnode_flink];
;  959          1081  2        END;
```

```
;  961          1082  2      !+
;  962          1083  2      ! We now have resolved all earlier record component and array references,
;  963          1084  2      ! and have to determine the actual data item address. The first thing to
;  964          1085  2      ! do is to calculate the base address (much as we did above).
;  965          1086  2      !-
;  966          1087  2      IF (.data_subnode[dbg$l_pnode_symid] NEQ 0) AND
;  967          1088  2      NOT .data_subnode[dbg$v_pnode_ignore]
;  968          1089  2      THEN
;  969          1090  3          BEGIN
;  970          1091  3          dbg$sta_symvalue(.data_subnode[dbg$l_pnode_symid],adr_ptrs,adr_kind);
;  971          1092  3          SELECTONE .adr_kind OF
;  972          1093  3              SET
;  973          1094  3              [dbg$k_val_addr]:
;  974          1095  4                  BEGIN
;  975          1096  4                  result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + .adr_ptrs[0];
;  976          1097  4                  bit_offset = .bit_offset + .adr_ptrs[1];
;  977          1098  3                  END;
;  978          1099  3              [dbg$k_val_descr]:
;  979          1100  4                  BEGIN
;  980          1101  4                  BIND adr_desc = adr_ptrs[0]     : REF dbg$stg_desc;
;  981          1102  4                  result_desc[dsc$a_pointer]  = .result_desc[dsc$a_pointer] + .adr_desc[dsc$a_pointer];
;  982          1103  3                  END;
;  983          1104  3              [dbg$k_val_literal]:
;  984          1105  4                  BEGIN
;  985          1106  4                  result_desc[dsc$a_pointer] = .adr_ptrs[0];
;  986          1107  4                  bit_offset = .adr_ptrs[1];
;  987          1108  3                  END;
;  988          1109  3              [dbg$k_val_unalloc]:
;  989          1110  4                  BEGIN
;  990          1111  4                  LOCAL name;
;  991          1112  4                  dbg$sta_symname(.data_subnode[dbg$l_pnode_symid],name);
;  992          1113  4                  SIGNAL(dbg$_unallocated, 1, .name);
;  993          1114  3                  END;
;  994          1115  3              [OTHERWISE]:
;  995          1116  3                  SIGNAL(dbg$_novalue);
;  996          1117  3              TES;
;  997          1118  2          END;
```

```
;  999          1119  2      !+
; 1000          1120  2      ! Having determined the address of the data object, we now attempt to
; 1001          1121  2      ! fill in the rest of the fields in the VMS descriptor (class, dtype,
; 1002          1122  2      ! and length). We use the information that we have in the bottom subnode:
; 1003          1123  2      ! a kind, an fcode, and a typeid.
; 1004          1124  2      !-
; 1005          1125  2      CASE .data_subnode[dbg$b_pnode_kind] FROM rst$k_kind_minimum TO rst$k_kind_maximum OF
; 1006          1126  2          SET
; 1007          1127  2
; 1008          1128  2          [rst$k_routine,rst$k_block,rst$k_entry,rst$k_line,rst$k_label]:
; 1009          1129  3              BEGIN
; 1010          1130  3              !+
; 1011          1131  3              ! Special case for MACRO - since MACRO declares everything to
; 1012          1132  3              ! be a label, then we want to instead use the default type
; 1013          1133  3              ! that has been specified with a SET TYPE command.
; 1014          1134  3              !-
; 1015          1135  4              IF (.data_subnode[dbg$b_pnode_kind] EQL rst$k_label)
; 1016          1136  4              AND (.prm_desc[dbg$b_dhdr_lang] EQL dbg$k_macro)
; 1017          1137  4              AND (NOT .prm_desc[dbg$v_dhdr_override])
; 1018          1138  4              AND (.dbg$gl_dflttyp NEQ dsc$k_dtype_zi)     ! If instruction, the we
; 1019          1139  4                                                           ! already have correct type
; 1020          1140  4                                                           ! and length
; 1021          1141  3              THEN
; 1022          1142  4                  BEGIN
; 1023          1143  4                  result_desc[dsc$b_class] = dsc$k_class_z;
; 1024          1144  4                  result_desc[dsc$b_dtype]  = .dbg$gl_dflttyp;
; 1025          1145  4                  result_desc[dsc$w_length] = .dbg$gw_dfltleng; ! All default types
; 1026          1146  4                  bit_length = 8*.dbg$gw_dfltleng;             ! have length in bytes.
; 1027          1147  4                  END
; 1028          1148  4
; 1029          1149  3              ELSE
; 1030          1150  4                  BEGIN
; 1031          1151  4                  !+
; 1032          1152  4                  ! This must be a primary representing an instruction or an entry
; 1033          1153  4                  ! mask in the user program.
; 1034          1154  4                  !-
; 1035          1155  4                  IF .bit_offset NEQ 0 THEN SIGNAL(dbg$_unimplent);
; 1036          1156  4                  result_desc[dsc$b_class] = dsc$k_class_z;
; 1037          1157  4                  IF dbg$is_it_entry(.result_desc[dsc$a_pointer])
; 1038          1158  4                  THEN
; 1039          1159  5                      BEGIN
; 1040          1160  5                      result_desc[dsc$b_dtype] = dsc$k_dtype_zem;
; 1041          1161  5                      bit_length = 16;
; 1042          1162  5                      END
; 1043          1163  4                  ELSE
; 1044          1164  5                      BEGIN
; 1045          1165  5                      result_desc[dsc$b_dtype] = dsc$k_dtype_zi;
; 1046          1166  6                      bit_length = %BPUNIT*(dbg$ins_decode(.result_desc[dsc$a_pointer],false,false) -
; 1047          1167  5                                                          .result_desc[dsc$a_pointer]);
; 1048          1168  4                      END;
; 1049          1169  3                  END;
; 1050          1170  2              END;
; 1051          1171  2
; 1052          1172  2          [rst$k_data,rst$k_typcomp]:
; 1053          1173  3              BEGIN
; 1054          1174  3              !+
; 1055          1175  3              ! The Primary represents data in the user program. Note that
```

```
;  1056     1176  3          ! record components come back with kind=typcomp (this is a quirk
;  1057     1177  3          ! in the parsing that may eventually be fixed).
;  1058     1178  3          !-
;  1059     1179  3          IF .data_subnode[dbg$b_pnode_fcode] EQL rst$k_type_array
;  1060     1180  3          THEN
;  1061     1181  4              BEGIN
;  1062     1182  4              BIND pnsub = data_subnode[dbg$a_pnarr_svector] : dbg$prim_node_subs;
;  1063     1183  4              addr_offset = .data_subnode[dbg$l_pnarr_offset];
;  1064     1184  4
;  1065     1185  4              ! Loop through the dimensions of the array.
;  1066     1186  4              !
;  1067     1187  4              DECR index FROM .data_subnode[dbg$b_pnarr_dimcnt]-1 TO 0 DO
;  1068     1188  5                BEGIN
;  1069     1189  5                s_value = .pnsub[.index,dbg$l_pnsub_svalue];
;  1070     1190  5                typeid = .pnsub[.index,dbg$l_pnsub_typeid];
;  1071     1191  5
;  1072     1192  5                ! If the array is indexed by an enumeration type,
;  1073     1193  5                ! then index by the position and not by
;  1074     1194  5                ! the value. This cases arises only in ADA for
;  1075     1195  5                ! enumerated types with representation specs.
;  1076     1196  5                !
;  1077     1197  5                IF (.dbg$gb_language EQL dbg$k_ada) AND
;  1078     1198  6                   (.typeid NEQ 0)
;  1079     1199  5                THEN
;  1080     1200  6                  IF (.typeid[rst$b_fcode] EQL rst$k_type_enum)
;  1081     1201  5                  THEN
;  1082     1202  5                    s_value = dbg$enum_pos(.typeid,.s_value);
;  1083     1203  5
;  1084     1204  5                addr_offset = .addr_offset + (.s_value*.pnsub[.index,dbg$l_pnsub_stride]);
;  1085     1205  4                END;
;  1086     1206  4
;  1087     1207  4          IF .data_subnode[dbg$v_pnarr_bitref]
;  1088     1208  4            THEN bit_offset = .bit_offset + .addr_offset
;  1089     1209  4            ELSE result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + .addr_offset;
;  1090     1210  4
;  1091     1211  4          !+
;  1092     1212  4          ! Figure out the class field. This class may get fixed up later -
;  1093     1213  4          ! we want class=ubs to reflect the fact that there is a bit
;  1094     1214  4          ! offset present. But for now, we just fill in class VS for
;  1095     1215  4          ! dtype vt, class SD if digits or scale are present, and
;  1096     1216  4          ! class s for all other dtypes.
;  1097     1217  4          !-
;  1098     1218  4          IF .data_subnode[dbg$b_pnarr_dtype] EQL dsc$k_dtype_vt
;  1099     1219  4          THEN
;  1100     1220  4              result_desc[dsc$b_class]  = dsc$k_class_vs
;  1101     1221  4          ELSE
;  1102     1222  5              BEGIN
;  1103     1223  5              IF (.data_subnode[dbg$b_pnarr_digits] NEQ 0) OR
;  1104     1224  6                 (.data_subnode[dbg$b_pnarr_scale] NEQ 0)
;  1105     1225  5              THEN
;  1106     1226  6                  BEGIN
;  1107     1227  6                  result_desc[dsc$b_class]  = dsc$k_class_sd;
;  1108     1228  6                  result_desc[dsc$b_digits] = .data_subnode[dbg$b_pnarr_digits];
;  1109     1229  6                  result_desc[dsc$b_scale]  = .data_subnode[dbg$b_pnarr_scale];
;  1110     1230  6                  END
;  1111     1231  5              ELSE
;  1112     1232  5                  result_desc[dsc$b_class]  = dsc$k_class_s;
```

```
;  1113          1233  4                    END;
;  1114          1234  4
;  1115          1235  4
;  1116          1236  4                    ! Fill in dtype and length. For arrays, we have this information
;  1117          1237  4                    ! in the array subnode.
;  1118          1238  4                    !
;  1119          1239  4                    result_desc[dsc$b_dtype]  = .data_subnode[dbg$b_pnarr_dtype];
;  1120          1240  4                    result_desc[dsc$w_length] = .data_subnode[dbg$w_pnarr_length];
;  1121          1241  4                    !+
;  1122          1242  4                    ! Fix up boolean to use the new type code dsc$k_dtype_tf.
;  1123          1243  4                    !-
;  1124          1244  4                    IF .result_desc[dsc$b_dtype] EQL dst$k_bool
;  1125          1245  4                    THEN
;  1126          1246  5                        BEGIN
;  1127          1247  5                        result_desc[dsc$b_class]  = dsc$k_class_s; ! This will get
;  1128          1248  5                                                                  ! changed to UBS
;  1129          1249  5                                                                  ! if there is a
;  1130          1250  5                                                                  ! bit offset
;  1131          1251  5                        result_desc[dsc$b_dtype]  = dsc$k_dtype_tf;
;  1132          1252  5                        result_desc[dsc$w_length] = bit_length = 1;
;  1133          1253  5                        END
;  1134          1254  4                    ELSE
;  1135          1255  4                        bit_length = dbg$data_length(result_desc);
;  1136          1256  4                    END
;  1137          1257  3                ELSE
;  1138          1258  3                    !+
;  1139          1259  3                    ! For everything except arrays, we call a routine to fill
;  1140          1260  3                    ! in the fields of the VMS descriptor, passing to this
;  1141          1261  3                    ! routine all the information we have collected so far.
;  1142          1262  3                    !-
;  1143          1263  3                    dbg$fill_in_vms_desc(.data_subnode[dbg$b_pnode_fcode],
;  1144          1264  3                                         .data_subnode[dbg$l_pnode_typeid],
;  1145          1265  3                                         .prm_desc[dbg$l_dhdr_symid0],
;  1146          1266  3                                         result_desc,bit_length,bit_offset);
```

```
;  1148          1267  3              !+
;  1149          1268  3              ! Fix things up so class VS always has dtype VT. Class VS and
;  1150          1269  3              ! type T is the older way of expressing varying string data
;  1151          1270  3              ! type, so we fix it up to use the newer dtype VT.
;  1152          1271  3              !-
;  1153          1272  3              IF  .result_desc[dsc$b_class] EQL dsc$k_class_vs
;  1154          1273  3              AND .result_desc[dsc$b_dtype] EQL dsc$k_dtype_t
;  1155          1274  3                THEN result_desc[dsc$b_dtype] = dsc$k_dtype_vt;
;  1156          1275  2              END;
;  1157          1276  2
;  1158          1277  2          !+
;  1159          1278  2          ! We should not see other kinds.
;  1160          1279  2          !-
;  1161          1280  2          [INRANGE,OUTRANGE]:
;  1162          1281  2              SIGNAL(dbg$_unimplent);
;  1163          1282  2          TES;
;  1164          1283  2
;  1165          1284  2
;  1166          1285  2      ! Dereference descriptors of type DSC.
;  1167          1286  2      !
;  1168          1287  2      IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_dsc
;  1169          1288  2      THEN
;  1170          1289  3          BEGIN
;  1171          1290  3          IF NOT PROBER(%REF(0),%REF(8),.result_desc[dsc$a_pointer])
;  1172          1291  3          THEN
;  1173          1292  3              SIGNAL(dbg$_noaccessr,1,.result_desc[dsc$a_pointer]);
;  1174          1293  3          ch$move(.result_desc[dsc$w_length],
;  1175          1294  3                  .result_desc[dsc$a_pointer], result_desc);
;  1176          1295  2          END;
;  1177          1296  2
;  1178          1297  2
;  1179          1298  2      ! Dereference descriptors of type BPV or BLV.
;  1180          1299  2      !
;  1181          1300  2      IF (.result_desc[dsc$b_dtype] EQL dsc$k_dtype_blv) OR
;  1182          1301  3         (.result_desc[dsc$b_dtype] EQL dsc$k_dtype_bpv)
;  1183          1302  2      THEN
;  1184          1303  3          BEGIN
;  1185          1304  3          IF NOT PROBER(%REF(0),%REF(8),.result_desc[dsc$a_pointer])
;  1186          1305  3          THEN
;  1187          1306  3              SIGNAL(dbg$_noaccessr,1,.result_desc[dsc$a_pointer]);
;  1188          1307  3          result_desc[dsc$a_pointer] = ..result_desc[dsc$a_pointer];
;  1189          1308  3          IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_blv
;  1190          1309  3          THEN
;  1191          1310  3              result_desc[dsc$b_dtype] = dsc$k_dtype_zi;
;  1192          1311  3          IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_bpv
;  1193          1312  3          THEN
;  1194          1313  3              result_desc[dsc$b_dtype] = dsc$k_dtype_zem;
;  1195          1314  2          END;
;  1196          1315  2
;  1197          1316  2      result_desc[dsc$a_pointer] =
;  1198          1317  2              .result_desc[dsc$a_pointer] + .data_subnode[dbg$l_pnode_reloc];
;  1199          1318  2
```

```
;  1201          1319  2     IF .prm_desc[dbg$v_dhdr_subref] THEN
;  1202          1320  3     BEGIN
;  1203          1321  3     !+
;  1204          1322  3     ! If there was an offset in the Primary (either a bit offset or
;  1205          1323  3     ! a byte offset) then take care of that here.
;  1206          1324  3     !-
;  1207          1325  3     IF .prm_desc[dbg$v_dhdr_bitref]
;  1208          1326  3     THEN
;  1209          1327  4         BEGIN
;  1210          1328  4         bit_offset = .prm_desc[dbg$w_prim_offset] + .bit_offset;
;  1211          1329  4         bit_length = .prm_desc[dbg$w_prim_length];
;  1212          1330  4         result_desc[dsc$b_class]  = dsc$k_class_z;  ! These get fixed up
;  1213          1331  4         result_desc[dsc$b_dtype]  = dsc$k_dtype_z;  ! below.
;  1214          1332  4         result_desc[dsc$w_length] = 0;
;  1215          1333  4         IF (.bit_offset AND (%BPUNIT-1)) EQL 0 THEN
;  1216          1334  4           SELECTONE .bit_length OF
;  1217          1335  4             SET
;  1218          1336  4
;  1219          1337  4             [ 8]:
;  1220          1338  5                 result_desc[dsc$b_dtype] = (IF .prm_desc[dbg$v_dhdr_sgnext]
;  1221          1339  4                                             THEN dsc$k_dtype_b ELSE dsc$k_dtype_bu);
;  1222          1340  4             [16]:
;  1223          1341  5                 result_desc[dsc$b_dtype] = (IF .prm_desc[dbg$v_dhdr_sgnext]
;  1224          1342  4                                             THEN dsc$k_dtype_w ELSE dsc$k_dtype_wu);
;  1225          1343  4             [32]:
;  1226          1344  5                 result_desc[dsc$b_dtype] = (IF .prm_desc[dbg$v_dhdr_sgnext]
;  1227          1345  4                                             THEN dsc$k_dtype_l ELSE dsc$k_dtype_lu);
;  1228          1346  4
;  1229          1347  4             [OTHERWISE]:
;  1230          1348  4                 0;
;  1231          1349  4
;  1232          1350  4             TES;
;  1233          1351  4         END
;  1234          1352  4
;  1235          1353  3     ELSE
;  1236          1354  4         BEGIN
;  1237          1355  4         IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_vt
;  1238          1356  4         THEN
;  1239          1357  5             BEGIN
;  1240          1358  5             result_desc[dsc$b_class]   = dsc$k_class_s;
;  1241          1359  5             result_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1242          1360  5             result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + 2;
;  1243          1361  4             END;
;  1244          1362  4         result_desc[dsc$a_pointer] = .prm_desc[dbg$w_prim_offset] + .result_desc[dsc$a_pointer];
;  1245          1363  4         bit_length = .prm_desc[dbg$w_prim_length] * %BPUNIT;
;  1246          1364  4
;  1247          1365  4         !+
;  1248          1366  4         ! For the string data types (which include ascii and also the
;  1249          1367  4         ! numeric string types NU, NL, NLO, NR, NRO, and NZ), fill
;  1250          1368  4         ! in the length of the result descriptor, but leave the class
;  1251          1369  4         ! and dtype unchanged. Note that the code below relies on the
;  1252          1370  4         ! fact that these dtype codes span the range from 14 (dsc$k_dtype_t)
;  1253          1371  4         ! to 20 (dsc$k_dtype_nz).
;  1254          1372  4         !-
;  1255          1373  4         IF (.result_desc[dsc$b_dtype] GEQ dsc$k_dtype_t) AND
;  1256          1374  5            (.result_desc[dsc$b_dtype] LEQ dsc$k_dtype_nz)
;  1257          1375  4         THEN
```

```
;  1258          1376  4                 result_desc[dsc$w_length] = .prm_desc[dbg$w_prim_length]
;  1259          1377  4             ELSE
;  1260          1378  5                 BEGIN
;  1261          1379  5                 result_desc[dsc$b_class]  = dsc$k_class_z;       ! (0)
;  1262          1380  5                 result_desc[dsc$b_dtype]  = dsc$k_dtype_z;       ! (0)
;  1263          1381  5                 result_desc[dsc$w_length] = 0;
;  1264          1382  4                 END;
;  1265          1383  3             END;
;  1266          1384  2         END;
```

```
;  1268     1385  2      !+
;  1269     1386  2      ! At this point, if the bit_offset variable is not a multiple of 8 then
;  1270     1387  2      ! there really is a bit offset. Fix up the class field to be UBS in
;  1271     1388  2      ! this case.
;  1272     1389  2      !-
;  1273     1390  2      IF (.bit_offset AND (%BPUNIT-1)) NEQ 0 THEN
;  1274     1391  2          BEGIN
;  1275     1392  2          !+
;  1276     1393  2          ! We used to not support unaligned bit fields longer than 32 bits.
;  1277     1394  2          ! IF .bit_length GTRU 32 THEN SIGNAL(dbg$_unimplent);
;  1278     1395  2          !-
;  1279     1396  2          result_desc[dsc$b_class] = dsc$k_class_ubs;
;  1280     1397  2          IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_z
;  1281     1398  2            THEN result_desc[dsc$b_dtype] = dsc$k_dtype_vu;
;  1282     1399  2          !+
;  1283     1400  2          ! Bit length is in bits for these five data types, and in bytes
;  1284     1401  2          ! for all others.
;  1285     1402  2          !-
;  1286     1403  2          result_desc[dsc$w_length] = .bit_length /
;  1287     1404  4            (IF .result_desc[dsc$b_dtype] EQL dsc$k_dtype_vu
;  1288     1405  4             OR .result_desc[dsc$b_dtype] EQL dsc$k_dtype_v
;  1289     1406  4             OR .result_desc[dsc$b_dtype] EQL dsc$k_dtype_svu
;  1290     1407  4             OR .result_desc[dsc$b_dtype] EQL dsc$k_dtype_sv
;  1291     1408  4             OR .result_desc[dsc$b_dtype] EQL dsc$k_dtype_tf
;  1292     1409  4             THEN 1 ELSE 8);
;  1293     1410  2          result_desc[dsc$l_pos]   = .bit_offset;
;  1294     1411  2          END
;  1295     1412  2      ELSE
;  1296     1413  2          BEGIN
;  1297     1414  2          !+
;  1298     1415  2          ! If we get here then we have byte-aligned data.
;  1299     1416  2          ! Fix up the pointer field to point to the byte where the data
;  1300     1417  2          ! data actually begins.
;  1301     1418  2          !-
;  1302     1419  2          result_desc[dsc$a_pointer] = .result_desc[dsc$a_pointer] + (.bit_offset/%BPUNIT);
;  1303     1420  4          IF (.result_desc[dsc$b_class] EQL dsc$k_class_z)
;  1304     1421  2          THEN
;  1305     1422  4              BEGIN
;  1306     1423  5              IF ((.bit_length AND (%BPUNIT-1)) EQL 0)
;  1307     1424  4              THEN
;  1308     1425  4                  !+
;  1309     1426  4                  ! If the length of the data is exactly a multiple of 8 then
;  1310     1427  4                  ! leave the dtype Z and express the length in bytes.
;  1311     1428  4                  !-
;  1312     1429  4                  result_desc[dsc$w_length] = .bit_length/%BPUNIT
;  1313     1430  4              ELSE
;  1314     1431  5                  BEGIN
;  1315     1432  5                  !+
;  1316     1433  5                  ! If the length is not expressible in bytes then change the dtype
;  1317     1434  5                  ! to V and fill in the length field with a bit length.
;  1318     1435  5                  !-
;  1319     1436  5                  result_desc[dsc$b_dtype] = dsc$k_dtype_v;
;  1320     1437  5                  IF .bit_length LSSU %X'10000'
;  1321     1438  5                  THEN
;  1322     1439  6                      BEGIN
;  1323     1440  6                      result_desc[dsc$w_length] = .bit_length;
;  1324     1441  6                      result_desc[dsc$l_pos]    = 0;
```

```
;  1325          1442  6                    END
;  1326          1443  5                ELSE
;  1327          1444  6                    BEGIN
;  1328          1445  6                    !+
;  1329          1446  6                    ! Special handling for the case where the length does
;  1330          1447  6                    ! not fit in the word field.
;  1331          1448  6                    !-
;  1332          1449  6                    result_desc[dsc$w_length] = 0;
;  1333          1450  6                    result_desc[dsc$l_pos] = .bit_length;
;  1334          1451  5                END;
;  1335          1452  4            END;
;  1336          1453  3        END;
;  1337          1454  2    END;
;  1338          1455  2
;  1339          1456  2    ch$move(12,result_desc,.vms_desc);
;  1340          1457  2
;  1341          1458  2    RETURN sts$k_success;
;  1342          1459  2
;  1343          1460  1    END;                ! End of routine dbg$make_vms_desc


                                    0FFC 00000        .ENTRY  DBG$MAKE_VMS_DESC, Save R2,R3,R4,R5,R6,R7,- ; 0839
                                                              R8,R9,R10,R11
                5B 00000000G    00  9E 00002        MOVAB   LIB$SIGNAL, R11                               ;
                5E              00  C2 00009        SUBL2   #56, SP                                       ;
                57        04    AC  D0 0000C        MOVL    PRM_DESC, R7                                  ; 0878
00000000G       00              57  D0 00010        MOVL    R7, DBG$GL_CURRENT_PRIMARY                    ;
                5A        04    A7  9E 00017        MOVAB   4(R7), R10                                    ; 0882
                07        03    AA  91 0001B        CMPB    3(R10), #7                                    ;
                                29  12 0001F        BNEQ    2$                                            ;
                                0C  A7  D5 00021        TSTL    12(R7)                                    ; 0887
                                1B  13 00024        BEQL    1$                                            ;
                                5E  DD 00026        PUSHL   SP                                            ; 0890
                                0C  A7  DD 00028        PUSHL   12(R7)                                    ;
                00000000G       00  02  FB 0002B        CALLS   #2, DBG$STA_SYMNAME                       ;
                                    6E  DD 00032        PUSHL   NAME                                      ; 0891
                                    01  DD 00034        PUSHL   #1                                        ;
                            00028168    8F  DD 00036        PUSHL   #164200                               ;
                        6B              03  FB 0003C        CALLS   #3, LIB$SIGNAL                        ;
                                        09  11 0003F        BRB     2$                                    ; 0887
                            000287F8    8F  DD 00041  1$:   PUSHL   #165880                               ; 0894
                        6B              01  FB 00047        CALLS   #1, LIB$SIGNAL                        ;
                                0C  A7  DD 0004A  2$:       PUSHL   12(R7)                                ; 0903
                00000000G       00  01  FB 0004D        CALLS   #1, DBG$STA_SETCONTEXT                    ;
                                18  AE  7C 00054        CLRQ    BIT_OFFSET                                ; 0904
0C              00              6E  00  2C 00057        MOVC5   #0, (SP), #0, #12, RESULT_DESC            ; 0905
                                20  AE      0005C
                        53      14  A7  D0 0005E        MOVL    20(R7), PRIM_SUBNODE                      ; 0914
                        56      18  A7  D0 00062        MOVL    24(R7), DATA_SUBNODE                      ; 0915
                        56          53  D1 00066  3$:   CMPL    PRIM_SUBNODE, DATA_SUBNODE                ; 0916
                                    03  12 00069        BNEQ    4$                                        ;
                                01E5    31 0006B        BRW     25$                                       ;
                            24  AE  14  A3  C0 0006E  4$:   ADDL2   20(PRIM_SUBNODE), RESULT_DESC+4      ; 0918
                                52  10  A3  D0 00073        MOVL    16(PRIM_SUBNODE), R2                  ; 0925
```

```
                                63 13 00077          BEQL    8$                                                      ;
                             5E 05 E0 00079          BBS     #5, 10(PRIM_SUBNODE), 8$                                ; 0926
                          10    AE 9F 0007E          PUSHAB  ADR_KIND                                                ; 0929
                          30    AE 9F 00081          PUSHAB  ADR_PTRS                                                ;
                                52 DD 00084          PUSHL   R2                                                      ;
                   00000000G    00 FB 00086          CALLS   #3, DBG$STA_SYMVALUE                                    ;
                          50    10 AE D0 0008D          MOVL    ADR_KIND, R0                                            ; 0930
                          02       50 D1 00091          CMPL    R0, #2                                                  ; 0932
                                   0C 12 00094          BNEQ    5$                                                      ;
                             24 AE    2C AE C0 00096          ADDL2   ADR_PTRS, RESULT_DESC+4                         ; 0934
                             18 AE    30 AE C0 0009B          ADDL2   ADR_PTRS+4, BIT_OFFSET                          ; 0935
                                      3A 11 000A0          BRB     8$                                                      ; 0930
                                03       50 D1 000A2 5$:         CMPL    R0, #3                                          ; 0937
                                         0B 12 000A5          BNEQ    6$                                                      ;
                                50    2C AE D0 000A7          MOVL    ADR_DESC, R0                                            ; 0940
                             24 AE    04 A0 C0 000AB          ADDL2   4(R0), RESULT_DESC+4                            ;
                                      2A 11 000B0          BRB     8$                                                      ; 0930
                                04       50 D1 000B2 6$:         CMPL    R0, #4                                          ; 0942
                                         1C 12 000B5          BNEQ    7$                                                      ;
                                   04 AE 9F 000B7          PUSHAB  NAME                                                    ; 0945
                                      52 DD 000BA          PUSHL   R2                                                      ;
                   00000000G    00    02 FB 000BC          CALLS   #2, DBG$STA_SYMNAME                                     ;
                                   04 AE DD 000C3          PUSHL   NAME                                                    ; 0946
                                      01 DD 000C6          PUSHL   #1                                                      ;
                             00028170    8F DD 000C8          PUSHL   #164208                                                 ;
                                6B       03 FB 000CE          CALLS   #3, LIB$SIGNAL                                          ;
                                         09 11 000D1          BRB     8$                                                      ; 0930
                             000287F8    8F DD 000D3 7$:         PUSHL   #165880                                         ; 0949
                                6B       01 FB 000D9          CALLS   #1, LIB$SIGNAL                                          ;
                                50    09 A3 9A 000DC 8$:      MOVZBL  9(PRIM_SUBNODE), R0                             ; 0952
                                01       50 91 000E0          CMPB    R0, #1                                          ; 0954
                                         5D 12 000E3          BNEQ    14$                                                     ;
                                58    20 A3 D0 000E5          MOVL    32(PRIM_SUBNODE), ADDR_OFFSET                           ; 0961
                                52    1B A3 9A 000E9          MOVZBL  27(PRIM_SUBNODE), INDEX                         ; 0965
                                      3E 11 000ED          BRB     11$                                                     ;
                          54       52    14 C5 000EF 9$:         MULL3   #20, INDEX, R4                                  ; 0967
                                   28 A344 9F 000F3          PUSHAB  40(PRIM_SUBNODE)[R4]                                    ;
                                59       9E D0 000F7          MOVL    @(SP)+, S_VALUE                                         ;
                                   38 A344 9F 000FA          PUSHAB  56(PRIM_SUBNODE)[R4]                            ; 0968
                                55       9E D0 000FE          MOVL    @(SP)+, TYPEID                                          ;
                                09 00000000G    00 91 00101          CMPB    DBG$GB_LANGUAGE, #9                             ; 0975
                                         18 12 00108          BNEQ    10$                                                     ;
                                         55 D5 0010A          TSTL    TYPEID                                          ; 0976
                                         14 13 0010C          BEQL    10$                                                     ;
                                04       18 A5 91 0010E          CMPB    24(TYPEID), #4                                  ; 0978
                                         0E 12 00112          BNEQ    10$                                                     ;
                                   0220    8F BB 00114          PUSHR   #^M<R5,R9>                                      ; 0980
                   00000000G    00       02 FB 00118          CALLS   #2, DBG$ENUM_POS                                        ;
                                59       50 D0 0011F          MOVL    R0, S_VALUE                                             ;
                                   2C A344 9F 00122 10$:       PUSHAB  44(PRIM_SUBNODE)[R4]                            ; 0982
                          50       59    9E C5 00126          MULL3   @(SP)+, S_VALUE, R0                                     ;
                                58       50 C0 0012A          ADDL2   R0, ADDR_OFFSET                                         ;
                                BF       52 F4 0012D 11$:        SOBGEQ  INDEX, 9$                                       ; 0965
                          06    0A A3    02 E1 00130          BBC     #2, 10(PRIM_SUBNODE), 12$                       ; 0985
                                18 AE       58 C0 00135          ADDL2   ADDR_OFFSET, BIT_OFFSET                         ; 0986
                                         04 11 00139          BRB     13$                                                     ;
                                24 AE       58 C0 0013B 12$:        ADDL2   ADDR_OFFSET, RESULT_DESC+4                      ; 0987
```

```
                                    010B  31  0013F 13$:     BRW     24$                                             ; 0952
                              06          50  91  00142 14$:     CMPB    R0, #6                                          ; 0990
                                          05  13  00145         BEQL    15$                                             ;
                              10          50  91  00147         CMPB    R0, #16                                         ;
                                          20  12  0014A         BNEQ    17$                                             ;
                52      18    AE      FD  8F  78  0014C 15$:     ASHL    #-3, BIT_OFFSET, ADDR                           ; 0997
                              52      24  AE  C0  00152         ADDL2   RESULT_DESC+4, ADDR                             ;
                62            05          00  0C  00156         PROBER  #0, #5, (ADDR)                                  ; 0998
                                          0D  12  0015A         BNEQ    16$                                             ;
                                          52  DD  0015C         PUSHL   ADDR                                            ; 0999
                                          01  DD  0015E         PUSHL   #1                                              ;
                                  00028228    8F  DD  00160         PUSHL   #164392                                         ;
                              6B          03  FB  00166         CALLS   #3, LIB$SIGNAL                                  ;
                                      00CC    31  00169 16$:     BRW     22$                                             ; 1003
                              07          50  91  0016C 17$:     CMPB    R0, #7                                          ; 1007
                                          CE  13  0016F         BEQL    13$                                             ;
                              13          50  91  00171         CMPB    R0, #19                                         ; 1010
                                          03  13  00174         BEQL    18$                                             ;
                                      0088    31  00176         BRW     20$                                             ;
                C1      0A    A3          04  E0  00179 18$:     BBS     #4, 10(PRIM_SUBNODE), 13$                       ; 1011
                        0A    A3          10  88  0017E         BISB2   #16, 10(PRIM_SUBNODE)                           ; 1014
                              52      1C  A3  D0  00182         MOVL    28(PRIM_SUBNODE), R2                            ; 1015
                                          B7  13  00186         BEQL    13$                                             ;
                                      08  AE  9F  00188         PUSHAB  TAG_NAME                                        ; 1018
                                          52  DD  0018B         PUSHL   R2                                              ;
                    00000000G     00        02  FB  0018D         CALLS   #2, DBG$STA_SYMNAME                             ;
                                      0C  AE  9F  00194         PUSHAB  TAG_SIZE                                        ; 1019
                                          52  DD  00197         PUSHL   R2                                              ;
                    00000000G     00        02  FB  00199         CALLS   #2, DBG$STA_SYMSIZE                             ;
                                      0C  AE  D5  001A0         TSTL    TAG_SIZE                                        ; 1020
                                          9A  13  001A3         BEQL    13$                                             ;
                                      08  BE  95  001A5         TSTB    @TAG_NAME                                       ;
                                          95  13  001A8         BEQL    13$                                             ;
                                      10  AE  9F  001AA         PUSHAB  ADR_KIND                                        ; 1022
                                      30  AE  9F  001AD         PUSHAB  ADR_PTRS                                        ;
                                          52  DD  001B0         PUSHL   R2                                              ;
                        00000000G     00        03  FB  001B2         CALLS   #3, DBG$STA_SYMVALUE                            ;
                                  2C  AE      24  AE  C0  001B9         ADDL2   RESULT_DESC+4, ADR_PTRS                         ; 1023
                                  30  AE      18  AE  C0  001BE         ADDL2   BIT_OFFSET, ADR_PTRS+4                          ; 1024
                    2C  BE                04        00  0C  001C3         PROBER  #0, #4, @ADR_PTRS                               ; 1029
                                              0E  12  001C8         BNEQ    19$                                             ;
                                          2C  AE  DD  001CA         PUSHL   ADR_PTRS                                        ; 1031
                                          01  DD  001CD         PUSHL   #1                                              ;
                                  00028228    8F  DD  001CF         PUSHL   #164392                                         ;
                                      6B          03  FB  001D5         CALLS   #3, LIB$SIGNAL                                  ;
                52  2C  BE        0C  AE      30  AE  EF  001D8 19$:     EXTZV   ADR_PTRS+4, TAG_SIZE, @ADR_PTRS, TAG_VALUE      ; 1033
                                      24  A3  DD  001E0         PUSHL   36(PRIM_SUBNODE)                                ; 1034
                                          52  DD  001E3         PUSHL   TAG_VALUE                                       ;
                        00000000G     00        02  FB  001E5         CALLS   #2, DBG$STA_VARIANT_VALUE                       ;
                                          5E      50  E8  001EC         BLBS    R0, 24$                                         ;
                                      08  AE  DD  001EF         PUSHL   TAG_NAME                                        ; 1035
                                          52  DD  001F2         PUSHL   TAG_VALUE                                       ;
                                          02  DD  001F4         PUSHL   #2                                              ;
                                  0002871B    8F  DD  001F6         PUSHL   #165659                                         ;
                                      6B          04  FB  001FC         CALLS   #4, LIB$SIGNAL                                  ;
                                                  4C  11  001FF         BRB     24$                                             ; 1011
                              0F          50  91  00201 20$:     CMPB    R0, #15                                         ; 1040
```

```
                                                3E  12 00204          BNEQ    23$                                                     ;
                        52      18      AE      FD  8F 78 00206          ASHL    #-3, BIT_OFFSET, ADDR                                   ; 1062
                                52      24      AE  C0 0020C          ADDL2   RESULT_DESC+4, ADDR                                     ;
    18  AE          18  AE              03          00 EF 00210          EXTZV   #0, #3, BIT_OFFSET, BIT_OFFSET                          ; 1063
                    62                  08          00 0C 00217          PROBER  #0, #8, (ADDR)                                          ; 1064
                                                    09 12 0021B          BNEQ    21$                                                     ;
                                    00028F40        8F DD 0021D          PUSHL   #167744                                                 ;
                                6B                  01 FB 00223          CALLS   #1, LIB$SIGNAL                                          ;
                        50      18      AE          30 C1 00226 21$:     ADDL3   #48, BIT_OFFSET, R0                                     ; 1068
                        09              62          50 E0 0022B          BBS     R0, (ADDR), 22$                                         ;
                                    00028F40        8F DD 0022F          PUSHL   #167744                                                 ;
                                6B                  01 FB 00235          CALLS   #1, LIB$SIGNAL                                          ;
    24  AE              62              20      18  AE EF 00238 22$:     EXTZV   BIT_OFFSET, #32, (ADDR), RESULT_DESC+4                  ; 1072
                                                18  AE D4 0023F          CLRL    BIT_OFFSET                                              ; 1073
                                                    09 11 00242          BRB     24$                                                     ; 0952
                                    000287D8        8F DD 00244 23$:     PUSHL   #165848                                                 ; 1077
                                6B                  01 FB 0024A          CALLS   #1, LIB$SIGNAL                                          ;
                                53                  63 D0 0024D 24$:     MOVL    (PRIM_SUBNODE), PRIM_SUBNODE                            ; 1080
                                                FE13    31 00250          BRW     3$                                                      ; 0916
                                52      10          A6 D0 00253 25$:     MOVL    16(DATA_SUBNODE), R2                                    ; 1087
                                                    74 13 00257          BEQL    30$                                                     ;
                        6F      0A      A6          05 E0 00259          BBS     #5, 10(DATA_SUBNODE), 30$                               ; 1088
                                        10  AE      9F 0025E          PUSHAB  ADR_KIND                                                ; 1091
                                        30  AE      9F 00261          PUSHAB  ADR_PTRS                                                ;
                                                    52 DD 00264          PUSHL   R2                                                      ;
                    00000000G   00                  03 FB 00266          CALLS   #3, DBG$STA_SYMVALUE                                    ;
                                50      10          AE D0 0026D          MOVL    ADR_KIND, R0                                            ; 1092
                                02                  50 D1 00271          CMPL    R0, #2                                                  ; 1094
                                                    0C 12 00274          BNEQ    26$                                                     ;
                        24      AE          2C      AE C0 00276          ADDL2   ADR_PTRS, RESULT_DESC+4                                 ; 1096
                        18      AE          30      AE C0 0027B          ADDL2   ADR_PTRS+4, BIT_OFFSET                                  ; 1097
                                                    4B 11 00280          BRB     30$                                                     ; 1092
                                03                  50 D1 00282 26$:     CMPL    R0, #3                                                  ; 1099
                                                    0B 12 00285          BNEQ    27$                                                     ;
                                50          2C      AE D0 00287          MOVL    ADR_DESC, R0                                            ; 1102
                        24      AE          04      A0 C0 0028B          ADDL2   4(R0), RESULT_DESC+4                                    ;
                                                    3B 11 00290          BRB     30$                                                     ; 1092
                                01                  50 D1 00292 27$:     CMPL    R0, #1                                                  ; 1104
                                                    0C 12 00295          BNEQ    28$                                                     ;
                        24      AE          2C      AE D0 00297          MOVL    ADR_PTRS, RESULT_DESC+4                                 ; 1106
                        18      AE          30      AE D0 0029C          MOVL    ADR_PTRS+4, BIT_OFFSET                                  ; 1107
                                                    2A 11 002A1          BRB     30$                                                     ; 1092
                                04                  50 D1 002A3 28$:     CMPL    R0, #4                                                  ; 1109
                                                    1C 12 002A6          BNEQ    29$                                                     ;
                                        14      AE      9F 002A8          PUSHAB  NAME                                                    ; 1112
                                                    52 DD 002AB          PUSHL   R2                                                      ;
                    00000000G   00                  02 FB 002AD          CALLS   #2, DBG$STA_SYMNAME                                     ;
                                        14      AE  DD 002B4          PUSHL   NAME                                                    ; 1113
                                                    01 DD 002B7          PUSHL   #1                                                      ;
                                    00028170        8F DD 002B9          PUSHL   #164208                                                 ;
                                6B                  03 FB 002BF          CALLS   #3, LIB$SIGNAL                                          ;
                                                    09 11 002C2          BRB     30$                                                     ; 1092
                                    000287F8        8F DD 002C4 29$:     PUSHL   #165880                                                 ; 1116
                                6B                  01 FB 002CA          CALLS   #1, LIB$SIGNAL                                          ;
                    0D              00      08      A6 8F 002CD 30$:     CASEB   8(DATA_SUBNODE), #0, #13                                ; 1125
0027                0027                001C            001C    002D2 31$:     .WORD   32$-31$,-                                       ;
001C                0098                0027            0027    002DA          32$-31$,-                                               ;
```

```
     001C       0098        001C        0027   002E2                  33$-31$,-                            ;
                            001C        001C   002EA                  33$-31$,-                            ;
                                                                      33$-31$,-                            ;
                                                                      33$-31$,-                            ;
                                                                      39$-31$,-                            ;
                                                                      32$-31$,-                            ;
                                                                      33$-31$,-                            ;
                                                                      32$-31$,-                            ;
                                                                      39$-31$,-                            ;
                                                                      32$-31$,-                            ;
                                                                      32$-31$,-                            ;
                                                                      32$-31$                              ;
                                  00028800   8F   DD 002EE 32$:   PUSHL    #165888                         ; 1281
                            6B               01   FB 002F4        CALLS    #1, LIB$SIGNAL                  ;
                                             6E   11 002F7        BRB      38$                             ;
                                        23   AE   94 002F9 33$:   CLRB     RESULT_DESC+3                   ; 1143
                            04          08   A6   91 002FC        CMPB     8(DATA_SUBNODE), #4             ; 1135
                                             27   12 00300        BNEQ     34$                             ;
                                        03   A7   95 00302        TSTB     3(R7)                           ; 1136
                                             22   12 00305        BNEQ     34$                             ;
                                             6A   95 00307        TSTB     (R10)                           ; 1137
                                             1E   19 00309        BLSS     34$                             ;
                            16 00000000G     00   D1 0030B        CMPL     DBG$GL_DFLTTYP, #22             ; 1138
                                             15   13 00312        BEQL     34$                             ;
                       22   AE 00000000G     00   90 00314        MOVB     DBG$GL_DFLTTYP, RESULT_DESC+2   ; 1144
                            50 00000000G     00   3C 0031C        MOVZWL   DBG$GW_DFLTLENG, R0             ; 1145
                       20   AE               50   B0 00323        MOVW     R0, RESULT_DESC                 ;
                                             39   11 00327        BRB      37$                             ; 1146
                                        18   AE   D5 00329 34$:   TSTL     BIT_OFFSET                      ; 1155
                                             09   13 0032C        BEQL     35$                             ;
                                  00028800   8F   DD 0032E        PUSHL    #165888                         ;
                            6B               01   FB 00334        CALLS    #1, LIB$SIGNAL                  ;
                                        24   AE   DD 00337 35$:   PUSHL    RESULT_DESC+4                   ; 1157
                  00000000G 00               01   FB 0033A        CALLS    #1, DBG$IS_IT_ENTRY             ;
                            0A               50   E9 00341        BLBC     R0, 36$                         ;
                       22   AE               17   90 00344        MOVB     #23, RESULT_DESC+2              ; 1160
                       1C   AE               10   D0 00348        MOVL     #16, BIT_LENGTH                 ; 1161
                                             19   11 0034C        BRB      38$                             ; 1157
                       22   AE               16   90 0034E 36$:   MOVB     #22, RESULT_DESC+2              ; 1165
                                             7E   7C 00352        CLRQ     -(SP)                           ; 1166
                                        2C   AE   DD 00354        PUSHL    RESULT_DESC+4                   ;
                  00000000G 00               03   FB 00357        CALLS    #3, DBG$INS_DECODE              ;
                            50          24   AE   C2 0035E        SUBL2    RESULT_DESC+4, R0               ; 1167
           1C   AE          50               03   78 00362 37$:   ASHL     #3, R0, BIT_LENGTH              ; 1166
                                           00DF   31 00367 38$:   BRW      53$                             ; 1125
                            01          09   A6   91 0036A 39$:   CMPB     9(DATA_SUBNODE), #1             ; 1179
                                             03   13 0036E        BEQL     40$                             ;
                                           00AE   31 00370        BRW      51$                             ;
                            58          20   A6   D0 00373 40$:   MOVL     32(DATA_SUBNODE), ADDR_OFFSET   ; 1183
                            54          18   A6   9E 00377        MOVAB    24(DATA_SUBNODE), R4            ; 1187
                            52          03   A4   9A 0037B        MOVZBL   3(R4), INDEX                    ;
                                             3E   11 0037F        BRB      43$                             ;
                53          52               14   C5 00381 41$:   MULL3    #20, INDEX, R3                  ; 1189
                                   28 A643   9F   9F 00385        PUSHAB   40(DATA_SUBNODE)[R3]            ;
                            59               9E   D0 00389        MOVL     @(SP)+, S_VALUE                 ;
                                   38 A643   9F   9F 0038C        PUSHAB   56(DATA_SUBNODE)[R3]            ; 1190
                            55               9E   D0 00390        MOVL     @(SP)+, TYPEID                  ;
```

```
                        09 00000000G   00   91 00393          CMPB    DBG$GB_LANGUAGE, #9                         ; 1197
                                       18   12 0039A          BNEQ    42$                                         ;
                                       55   D5 0039C          TSTL    TYPEID                                      ; 1198
                                       14   13 0039E          BEQL    42$                                         ;
                        04        18   A5   91 003A0          CMPB    24(TYPEID), #4                              ; 1200
                                       0E   12 003A4          BNEQ    42$                                         ;
                              0220     8F   BB 003A6          PUSHR   #^M<R5,R9>                                  ; 1202
                00000000G     00            02 FB 003AA          CALLS   #2, DBG$ENUM_POS                            ;
                              59            50 D0 003B1          MOVL    R0, S_VALUE                                 ;
                                 2C A643    9F 003B4 42$:        PUSHAB  44(DATA_SUBNODE)[R3]                        ; 1204
                50            59            9E C5 003B8          MULL3   @(SP)+, S_VALUE, R0                         ;
                              58            50 C0 003BC          ADDL2   R0, ADDR_OFFSET                             ;
                              BF            52 F4 003BF 43$:     SOBGEQ  INDEX, 4T$                                  ; 1187
                06        0A  A6            02 E1 003C2          BBC     #2, 10(DATA_SUBNODE), 44$                   ; 1207
                          18  AE            58 C0 003C7          ADDL2   ADDR_OFFSET, BIT_OFFSET                     ; 1208
                                             04 11 003CB          BRB     45$                                         ;
                          24  AE            58 C0 003CD 44$:     ADDL2   ADDR_OFFSET, RESULT_DESC+4                  ; 1209
                              25        02  A4   91 003D1 45$:     CMPB    2(R4), #37                                  ; 1218
                                             06 12 003D5          BNEQ    46$                                         ;
                          23  AE            0B 90 003D7          MOVB    #11, RESULT_DESC+3                          ; 1220
                                             17 11 003DB          BRB     49$                                         ;
                                       01    A4   95 003DD 46$:     TSTB    1(R4)                                       ; 1223
                                             04 12 003E0          BNEQ    47$                                         ;
                                             64 95 003E2          TSTB    (R4)                                        ; 1224
                                             0A 13 003E4          BEQL    48$                                         ;
                          23  AE            09 90 003E6 47$:     MOVB    #9, RESULT_DESC+3                           ; 1227
                          28  AE            64 B0 003EA          MOVW    (R4), RESULT_DESC+8                         ; 1229
                                             04 11 003EE          BRB     49$                                         ; 1223
                          23  AE            01 90 003F0 48$:     MOVB    #1, RESULT_DESC+3                           ; 1232
                          22  AE        02  A4   90 003F4 49$:     MOVB    2(R4), RESULT_DESC+2                        ; 1239
                          20  AE        1C  A6   B0 003F9          MOVW    28(DATA_SUBNODE), RESULT_DESC               ; 1240
                          9E  8F        22  AE   91 003FE          CMPB    RESULT_DESC+2, #158                         ; 1244
                                             0E 12 00403          BNEQ    50$                                         ;
                          1C  AE            01 D0 00405          MOVL    #1, BIT_LENGTH                              ; 1252
                          20  AE  01280001   8F D0 00409          MOVL    #19398657, RESULT_DESC                      ;
                                             26 11 00411          BRB     52$                                         ; 1244
                                       20    AE   9F 00413 50$:     PUSHAB  RESULT_DESC                                 ; 1255
                              F69F     CF        01 FB 00416          CALLS   #1, DBG$DATA_LENGTH                         ;
                                1C     AE        50 D0 0041B          MOVL    R0, BIT_LENGTH                              ;
                                             18 11 0041F          BRB     52$                                         ; 1179
                                       18    AE   9F 00421 51$:     PUSHAB  BIT_OFFSET                                  ; 1263
                                       20    AE   9F 00424          PUSHAB  BIT_LENGTH                                  ;
                                       28    AE   9F 00427          PUSHAB  RESULT_DESC                                 ;
                                       0C    A7   DD 0042A          PUSHL   12(R7)                                      ; 1265
                                       0C    A6   DD 0042D          PUSHL   12(DATA_SUBNODE)                            ; 1264
                                 7E        09    A6   9A 00430          MOVZBL  9(DATA_SUBNODE), -(SP)                      ; 1263
                              F970     CF        06 FB 00434          CALLS   #6, DBG$FILL_IN_VMS_DESC                    ;
                                       0B    23   AE   91 00439 52$:     CMPB    RESULT_DESC+3, #11                          ; 1272
                                             0A 12 0043D          BNEQ    53$                                         ;
                                       0E    22   AE   91 0043F          CMPB    RESULT_DESC+2, #14                          ; 1273
                                             04 12 00443          BNEQ    53$                                         ;
                                 22    AE        25 90 00445          MOVB    #37, RESULT_DESC+2                          ; 1274
                                       18    22   AE   91 00449 53$:     CMPB    RESULT_DESC+2, #24                          ; 1287
                                             1C 12 0044D          BNEQ    55$                                         ;
                24  BE                 08        00 0C 0044F          PROBER  #0, #8, @RESULT_DESC+4                      ; 1290
                                             0E 12 00454          BNEQ    54$                                         ;
                                       24    AE   DD 00456          PUSHL   RESULT_DESC+4                               ; 1292
```

```
                                    01 DD 00459          PUSHL   #1                                                                  ;
                          00028228  8F DD 00045B         PUSHL   #164392                                                             ;
                         6B         03 FB 00461          CALLS   #3, LIB$SIGNAL                                                      ;
20 AE       24 BE          20 AE   28 00464 54$:        MOVC3   RESULT_DESC, @RESULT_DESC+4, RESULT_DESC                            ; 1293
                21          22 AE   91 0046B 55$:        CMPB    RESULT_DESC+2, #33                                                  ; 1300
                                    06 13 0046F          BEQL    56$                                                                 ;
                20          22 AE   91 00471             CMPB    RESULT_DESC+2, #32                                                  ; 1301
                                    2E 12 00475          BNEQ    59$                                                                 ;
24 BE           08                  00 0C 00477 56$:     PROBER  #0, #8, @RESULT_DESC+4                                              ; 1304
                                    0E 12 0047C          BNEQ    57$                                                                 ;
                          24 AE     DD 0047E             PUSHL   RESULT_DESC+4                                                       ; 1306
                                    01 DD 00481          PUSHL   #1                                                                  ;
                          00028228  8F DD 00483          PUSHL   #164392                                                             ;
                         6B         03 FB 00489          CALLS   #3, LIB$SIGNAL                                                      ;
                24 AE       24 BE   D0 0048C 57$:        MOVL    @RESULT_DESC+4, RESULT_DESC+4                                       ; 1307
                21          22 AE   91 00491             CMPB    RESULT_DESC+2, #33                                                  ; 1308
                                    04 12 00495          BNEQ    58$                                                                 ;
                22 AE               16 90 00497          MOVB    #22, RESULT_DESC+2                                                  ; 1310
                20          22 AE   91 0049B 58$:        CMPB    RESULT_DESC+2, #32                                                  ; 1311
                                    04 12 0049F          BNEQ    59$                                                                 ;
                22 AE               17 90 004A1          MOVB    #23, RESULT_DESC+2                                                  ; 1313
                24 AE       14 A6   C0 004A5 59$:        ADDL2   20(DATA_SUBNODE), RESULT_DESC+4                                     ; 1317
         59                 6A      01 E1 004AA          BBC     #1, (R10), 66$                                                      ; 1319
         57                 6A      02 E1 004AE          BBC     #2, (R10), 67$                                                      ; 1325
                50          10 A7   32 004B2             CVTWL   16(R7), R0                                                          ; 1328
                18 AE       50 C0   004B6                ADDL2   R0, BIT_OFFSET                                                      ;
                1C AE       12 A7   3C 004BA             MOVZWL  18(R7), BIT_LENGTH                                                  ; 1329
                          20 AE     D4 004BF             CLRL    RESULT_DESC                                                         ; 1332
                07          18 AE   93 004C2             BITB    BIT_OFFSET, #7                                                      ; 1333
                                    78 12 004C6          BNEQ    70$                                                                 ;
                50          1C AE   D0 004C8             MOVL    BIT_LENGTH, R0                                                      ; 1334
                08                  50 D1 004CC          CMPL    R0, #8                                                              ; 1337
                                    0E 12 004CF          BNEQ    61$                                                                 ;
         05                 6A      03 E1 004D1          BBC     #3, (R10), 60$                                                      ; 1338
                50                  06 D0 004D5          MOVL    #6, R0                                                              ;
                                    29 11 004D8          BRB     65$                                                                 ;
                50                  02 D0 004DA 60$:     MOVL    #2, R0                                                              ;
                                    24 11 004DD          BRB     65$                                                                 ;
                10                  50 D1 004DF 61$:     CMPL    R0, #16                                                             ; 1340
                                    0E 12 004E2          BNEQ    63$                                                                 ;
         05                 6A      03 E1 004E4          BBC     #3, (R10), 62$                                                      ; 1341
                50                  07 D0 004E8          MOVL    #7, R0                                                              ;
                                    16 11 004EB          BRB     65$                                                                 ;
                50                  03 D0 004ED 62$:     MOVL    #3, R0                                                              ;
                                    11 11 004F0          BRB     65$                                                                 ;
                20                  50 D1 004F2 63$:     CMPL    R0, #32                                                             ; 1343
                                    49 12 004F5          BNEQ    70$                                                                 ;
         05                 6A      03 E1 004F7          BBC     #3, (R10), 64$                                                      ; 1344
                50                  08 D0 004FB          MOVL    #8, R0                                                              ;
                                    03 11 004FE          BRB     65$                                                                 ;
                50                  04 D0 00500 64$:     MOVL    #4, R0                                                              ;
          22 AE                     50 90 00503 65$:     MOVB    R0, RESULT_DESC+2                                                   ;
                                    37 11 00507 66$:     BRB     70$                                                                 ;
                25          22 AE   91 00509 67$:        CMPB    RESULT_DESC+2, #37                                                  ; 1355
                                    0A 12 0050D          BNEQ    68$                                                                 ;
          22 AE       010E          8F B0 0050F          MOVW    #270, RESULT_DESC+2                                                 ; 1359
          24 AE                     02 C0 00515          ADDL2   #2, RESULT_DESC+4                                                   ; 1360
```

```
                        50      10  A7  32 00519 68$:    CVTWL   16(R7), R0                          ; 1362
                    24  AE      50  C0  00 0051D         ADDL2   R0, RESULT_DESC+4
                        50      12  A7  3C 00521         MOVZWL  18(R7), R0                          ; 1363
                1C  AE          50  03  78 00525         ASHL    #3, R0, BIT_LENGTH                  ; 1373
                        0E      22  AE  91 00528         CMPB    RESULT_DESC+2, #14
                                    0D  1F 0052E         BLSSU   69$
                        14      22  AE  91 00530         CMPB    RESULT_DESC+2, #20                  ; 1374
                                    07  1A 00534         BGTRU   69$
                    20  AE      12  A7  B0 00536         MOVW    18(R7), RESULT_DESC                 ; 1376
                                    03  11 0053B         BRB     70$
                        20      AE  D4  00 0053D 69$:    CLRL    RESULT_DESC                         ; 1381
                        07      18  AE  93 00540 70$:    BITB    BIT_OFFSET, #7                      ; 1390
                                    42  13 00544         BEQL    75$
                    23  AE          0D  90 00546         MOVB    #13, RESULT_DESC+3                  ; 1396
                            22  AE  95 0054A             TSTB    RESULT_DESC+2                       ; 1397
                                    04  12 0054D         BNEQ    71$
                    22  AE          22  90 0054F         MOVB    #34, RESULT_DESC+2                  ; 1398
                        50      22  AE  9A 00553 71$:    MOVZBL  RESULT_DESC+2, R0                   ; 1404
                        22          50  91 00557         CMPB    R0, #34
                                    14  13 0055A         BEQL    72$
                        01          50  91 0055C         CMPB    R0, #1                              ; 1405
                                    0F  13 0055F         BEQL    72$
                        2A          50  91 00561         CMPB    R0, #42                             ; 1406
                                    0A  13 00564         BEQL    72$
                        29          50  91 00566         CMPB    R0, #41                             ; 1407
                                    05  13 00569         BEQL    72$
                        28          50  91 0056B         CMPB    R0, #40                             ; 1408
                                    05  12 0056E         BNEQ    73$
                        50          01  D0  00 00570 72$:    MOVL    #1, R0                          ; 1404
                                    03  11 00573         BRB     74$
                        50          08  D0  00 00575 73$:    MOVL    #8, R0
            51          1C  AE          50  C7  00 00578 74$:    DIVL3   R0, BIT_LENGTH, R1
                        20  AE          51  B0  00 0057D         MOVW    R1, RESULT_DESC
                        28  AE      18  AE  D0  00 00581         MOVL    BIT_OFFSET, RESULT_DESC+8   ; 1410
                                    3E  11 00586         BRB     78$                                 ; 1390
            50          18  AE          08  C7  00 00588 75$:    DIVL3   #8, BIT_OFFSET, R0          ; 1419
                        24  AE          50  C0  00 0058D         ADDL2   R0, RESULT_DESC+4
                                23  AE  95  00 00591             TSTB    RESULT_DESC+3               ; 1420
                                    30  12 00594         BNEQ    78$
                        50      1C  AE  D0  00 00596         MOVL    BIT_LENGTH, R0                  ; 1423
                        07          50  93  00 0059A         BITB    R0, #7
                                    0A  12 0059D         BNEQ    76$
            51          50          08  C7  00 0059F         DIVL3   #8, R0, R1                      ; 1429
                    20  AE          51  B0  00 005A3         MOVW    R1, RESULT_DESC
                                    1D  11 005A7         BRB     78$
                    22  AE          01  90  00 005A9 76$:    MOVB    #1, RESULT_DESC+2               ; 1436
                00010000    8F      50  D1  00 005AD         CMPL    R0, #65536                      ; 1437
                                    09  1E 005B4         BGEQU   77$
                    20  AE          50  B0  00 005B6         MOVW    R0, RESULT_DESC                 ; 1440
                                23  AE  D4  00 005BA         CLRL    RESULT_DESC+8                   ; 1441
                                    07  11 005BD         BRB     78$                                 ; 1437
                                20  AE  B4  00 005BF 77$:    CLRW    RESULT_DESC                     ; 1429
                    28  AE          50  D0  00 005C2         MOVL    R0, RESULT_DESC+8               ; 1450
        08  BC      20  AE          0C  28  00 005C6 78$:    MOVC3   #12, RESULT_DESC, @VMS_DESC     ; 1456
                        50          01  D0  00 005CC         MOVL    #1, R0                          ; 1458
                                    04  00 005CF             RET                                     ; 1460
```

```
; Routine Size:  1488 bytes,    Routine Base:  DBG$CODE + 0546
```

```
;  1345          1461  1 GLOBAL ROUTINE DBG$PRIM_TO_VAL (prm_desc,target_type,val_desc) =
;  1346          1462  1 !++
;  1347          1463  1 ! FUNCTIONAL DESCRIPTION:
;  1348          1464  1 !
;  1349          1465  1 !       Translates language independent descriptors to language independent
;  1350          1466  1 !       value descriptors.  Normally the input descriptor will be a primary
;  1351          1467  1 !       descriptor (a symbolic address reference),  but it is also possible
;  1352          1468  1 !       to call this routine passing it a 'volatile value descriptor' (used
;  1353          1469  1 !       for absolute addressing) or a normal value descriptor (indirection).
;  1354          1470  1 !       This routine should be able to use the symbol table access routines
;  1355          1471  1 !       and the information contained within the primary descriptor to make
;  1356          1472  1 !       a descriptor representing a 'value  materialization' for the object
;  1357          1473  1 !       described by the input descriptor.
;  1358          1474  1 !
;  1359          1475  1 !       Note that this routine must be able to use life-time, invocation, and
;  1360          1476  1 !       generation information to produce an accurate value descriptor of the
;  1361          1477  1 !       input object, or to decide when the value of an object cannot be
;  1362          1478  1 !       materialized (such as when the user's PC is not within the scope of
;  1363          1479  1 !       a dynamic variable).
;  1364          1480  1 !
;  1365          1481  1 !       Value descriptors produced by this routine must be marked (within the
;  1366          1482  1 !       type field of the language independent header block) as to whether
;  1367          1483  1 !       they are non-volatile (dbg$k_value_desc) or volatile (dbg$k_v_value_desc).
;  1368          1484  1 !       Volatile value descriptors will NOT be stored to represent '\', 'last value'.
;  1369          1485  1 !
;  1370          1486  1 !       Since value descriptors may be used as target descriptors ( as input to
;  1371          1487  1 !       dbg$ncob_type_conv ), some provision must be made for incorporating
;  1372          1488  1 !       a value pointer field within the value descriptor. This type of value
;  1373          1489  1 !       descriptor is loosely defined as a volatile type.
;  1374          1490  1 !
;  1375          1491  1 ! FORMAL PARAMETERS:
;  1376          1492  1 !
;  1377          1493  1 !       prm_desc        - A longword containing the address of a primary descriptor
;  1378          1494  1 !
;  1379          1495  1 !       target_type     - A longword containing the type of value descriptor
;  1380          1496  1 !                         (dbg$k_value_desc or dbg$k_v_value_desc).
;  1381          1497  1 !
;  1382          1498  1 !       val_desc        - The address of a longword to contain the address of the
;  1383          1499  1 !                         resulting value descriptor
;  1384          1500  1 !
```

```
;  1386          1501  1 ! IMPLICIT INPUTS:
;  1387          1502  1 !
;  1388          1503  1 !       NONE
;  1389          1504  1 !
;  1390          1505  1 ! IMPLICIT OUTPUTS:
;  1391          1506  1 !
;  1392          1507  1 !       In case of a success return, the resulting value descriptor must be
;  1393          1508  1 !       constructed from dynamic storage and returned.
;  1394          1509  1 !
;  1395          1510  1 ! ROUTINE VALUE:
;  1396          1511  1 !
;  1397          1512  1 !       An unsigned integer longword completion code
;  1398          1513  1 !
;  1399          1514  1 ! COMPLETION CODES:
;  1400          1515  1 !
;  1401          1516  1 !       STS$K_SUCCESS (1) - Success. Value descriptor constructed and returned.
;  1402          1517  1 !
;  1403          1518  1 !       STS$K_ERROR   (2) - Failure. Data-type is not known to DEBUG kernel.
;  1404          1519  1 !
;  1405          1520  1 ! SIDE EFFECTS:
;  1406          1521  1 !
;  1407          1522  1 !       In case of a severe error, this routine will SIGNAL the error.
;  1408          1523  1 !
;  1409          1524  1 !--
```

```
;  1411     1525  1 !DBG$PRIM_TO_VAL(prm_desc,val_desc,target_type)
;  1412     1526  2     BEGIN
;  1413     1527  2     !
;  1414     1528  2     ! Describe formal parameters that are structures
;  1415     1529  2     !
;  1416     1530  2     MAP
;  1417     1531  2         prm_desc        : REF dbg$primary;
;  1418     1532  2     !
;  1419     1533  2     !
;  1420     1534  2     ! Declare routine-level local variables
;  1421     1535  2     !
;  1422     1536  2     LOCAL
;  1423     1537  2         result_desc     : REF dbg$valdesc,
;  1424     1538  2         data_desc       : BLOCK [dbg$k_valdesc_base_size+1,LONG]
;  1425     1539  2                           FIELD(dbg$dhdr_fields,dbg$value_fields);
;  1426     1540  2
;  1427     1541  2     BIND vms_desc = data_desc[dbg$a_value_vmsdesc] : dbg$stg_desc;
;  1428     1542  2
;  1429     1543  2     dbg$gl_current_primary = .prm_desc;                                    ! A003
;  1430     1544  2
;  1431     1545  2     ! It is illegal to call DBG$PRIM_TO_VAL with a type.
;  1432     1546  2     !
;  1433     1547  2     IF .prm_desc[dbg$b_dhdr_kind] EQL rst$k_type
;  1434     1548  2     THEN
;  1435     1549  3         BEGIN
;  1436     1550  3         LOCAL
;  1437     1551  3             name;
;  1438     1552  3         IF .prm_desc[dbg$l_dhdr_symid0] NEQ 0
;  1439     1553  3         THEN
;  1440     1554  4             BEGIN
;  1441     1555  4             dbg$sta_symname(.prm_desc[dbg$l_dhdr_symid0], name);
;  1442     1556  4             SIGNAL (dbg$_novaltyp, 1, .name);
;  1443     1557  4             END
;  1444     1558  3         ELSE
;  1445     1559  3             SIGNAL (dbg$_novalue);
;  1446     1560  2         END;
;  1447     1561  2
;  1448     1562  2     !+
;  1449     1563  2     ! First construct a VAX/VMS descriptor that for the desired value
;  1450     1564  2     !-
;  1451     1565  2     IF .prm_desc[dbg$l_dhdr_symid0] NEQ 0 THEN dbg$sta_setcontext(.prm_desc[dbg$l_dhdr_symid0]);
;  1452     1566  2     dbg$collect(.prm_desc);
;  1453     1567  2
;  1454     1568  2     SELECTONE .prm_desc[dbg$b_dhdr_type] OF
;  1455     1569  2         SET
;  1456     1570  2         [dbg$k_primary_desc]:
;  1457     1571  2             IF NOT dbg$make_vms_desc(.prm_desc,vms_desc) THEN RETURN sts$k_error;
;  1458     1572  2         [dbg$k_v_value_desc]:
;  1459     1573  2             ch$move(12,prm_desc[dbg$a_value_vmsdesc],vms_desc);
;  1460     1574  2         [dbg$k_value_desc]:
;  1461     1575  3             BEGIN
;  1462     1576  3             ch$fill(0,(dbg$k_valdesc_base_size+1)*%UPVAL,data_desc);
;  1463     1577  3             data_desc[dbg$b_dhdr_lang]   = .prm_desc[dbg$b_dhdr_lang];
;  1464     1578  3             data_desc[dbg$b_dhdr_type]   = .prm_desc[dbg$b_dhdr_type];
;  1465     1579  3             data_desc[dbg$l_dhdr_symid0] = .prm_desc[dbg$l_dhdr_symid0];
;  1466     1580  3             data_desc[dbg$w_dhdr_length] = dbg$k_valdesc_base_size*%UPVAL;
;  1467     1581  3             data_desc[dbg$b_dhdr_kind]   = rst$k_data;
```

```
;  1468          1582  3              data_desc[dbg$b_dhdr_fcode]  = rst$k_type_descr;
;  1469          1583  3
;  1470          1584  3              !+
;  1471          1585  3              ! Passing a pointer value into DBG$PRIM_TO_VAL results in a
;  1472          1586  3              ! value which is obtained by dereferencing the pointer.
;  1473          1587  3              !-
;  1474          1588  3              IF .prm_desc[dbg$b_dhdr_fcode] EQL rst$k_type_tptr
;  1475          1589  3              THEN
;  1476          1590  4                  BEGIN
;  1477          1591  4                  LOCAL bit_length,bit_offset;
;  1478          1592  4                  dbg$sta_typ_typedptr(.prm_desc[dbg$l_dhdr_typeid],data_desc[dbg$l_dhdr_typeid]);
;  1479          1593  4                  data_desc[dbg$b_dhdr_fcode] = dbg$sta_typefcode(data_desc[dbg$l_dhdr_typeid]);
;  1480          1594  4                  IF .data_desc[dbg$b_dhdr_fcode] EQL rst$k_type_array THEN RETURN 0;
;  1481          1595  4                  bit_length = 0;
;  1482          1596  4                  bit_offset = 0;
;  1483          1597  4                  vms_desc[dsc$b_class]  = 0;
;  1484          1598  4                  vms_desc[dsc$b_dtype]  = 0;
;  1485          1599  4                  vms_desc[dsc$w_length] = 0;
;  1486          1600  4                  vms_desc[dsc$a_pointer] = .(.prm_desc[dbg$l_value_pointer]);
;  1487          1601  4                  dbg$fill_in_vms_desc(.data_desc[dbg$b_dhdr_fcode],
;  1488          1602  4                                       .data_desc[dbg$l_dhdr_typeid],
;  1489          1603  4                                       .prm_desc[dbg$l_dhdr_symid0],
;  1490          1604  4                                       vms_desc,bit_length,bit_offset);
;  1491          1605  3                  END;
;  1492          1606  2              END;
;  1493          1607  2          [OTHERWISE]:
;  1494          1608  2              SIGNAL(dbg$_illtype);
;  1495          1609  2          TES;
;  1496          1610  2
;  1497          1611  2      result_desc = dbg$make_val_desc(vms_desc,.target_type);
;  1498          1612  2
;  1499          1613  2      result_desc[dbg$b_dhdr_lang]   = .prm_desc[dbg$b_dhdr_lang];
;  1500          1614  2      result_desc[dbg$b_dhdr_kind]   = .prm_desc[dbg$b_dhdr_kind];
;  1501          1615  2      result_desc[dbg$b_dhdr_fcode]  = .prm_desc[dbg$b_dhdr_fcode];
;  1502          1616  2      result_desc[dbg$l_dhdr_typeid] = .prm_desc[dbg$l_dhdr_typeid];
;  1503          1617  2      result_desc[dbg$l_dhdr_symid0] = .prm_desc[dbg$l_dhdr_symid0];
;  1504          1618  2      result_desc[dbg$v_dhdr_override] = .prm_desc[dbg$v_dhdr_override];
;  1505          1619  2
;  1506          1620  2
;  1507          1621  2      !+
;  1508          1622  2      ! Special case in COBOL.  Treat the cobol record as text string.
;  1509          1623  2      !-
;  1510          1624  2      IF .result_desc[dbg$b_dhdr_fcode] EQL rst$k_type_record
;  1511          1625  2      THEN
;  1512          1626  3          BEGIN
;  1513          1627  3          LOCAL tmp_symid: REF rst$entry;
;  1514          1628  3
;  1515          1629  3          tmp_symid = .result_desc[dbg$l_dhdr_symid0];
;  1516          1630  3          WHILE .tmp_symid[rst$b_kind] NEQ rst$k_module DO
;  1517          1631  3              tmp_symid = .tmp_symid[rst$l_upscopeptr];
;  1518          1632  3          IF .tmp_symid[rst$b_language] EQL dbg$k_cobol
;  1519          1633  3          THEN
;  1520          1634  4              BEGIN
;  1521          1635  4              result_desc[dbg$b_dhdr_fcode] = rst$k_type_descr;
;  1522          1636  4              result_desc[dbg$b_value_class] = dsc$k_class_s;
;  1523          1637  4              result_desc[dbg$b_value_dtype] = dsc$k_dtype_t;
;  1524          1638  3              END;
```

```
;  1525          1639  2          END;
;  1526          1640  2
;  1527          1641  2      !+
;  1528          1642  2      ! Special case for subrange - if we are turning a subrange primary into
;  1529          1643  2      ! a value descriptor, then change the type to reflect the parent type.
;  1530          1644  2      ! This is because, in all operations, a subrange is treated the same
;  1531          1645  2      ! as its parent type. This thus simplifies the operator tables.
;  1532          1646  2      !-
;  1533          1647  2      IF .target_type EQL dbg$k_value_desc THEN
;  1534          1648  2        WHILE .result_desc[dbg$b_dhdr_fcode] EQL rst$k_type_subrng DO
;  1535          1649  2          BEGIN
;  1536          1650  2          LOCAL parent,fcode,typeid,bit_length,bit_offset,dummy;
;  1537          1651  2          dbg$sta_typ_subrng(.result_desc[dbg$l_dhdr_typeid],parent,dummy,dummy,bit_length);
;  1538          1652  2          bit_offset = 0;
;  1539          1653  2          dbg$sta_symtype(.parent,fcode,typeid);
;  1540          1654  2          result_desc[dbg$b_dhdr_fcode]  = .fcode;
;  1541          1655  2          result_desc[dbg$l_dhdr_typeid] = .typeid;
;  1542          1656  2          result_desc[dbg$b_value_class]  = 0;
;  1543          1657  2          result_desc[dbg$b_value_dtype]  = 0;
;  1544          1658  2          result_desc[dbg$w_value_length] = 0;
;  1545          1659  2          dbg$fill_in_vms_desc(.fcode,.typeid,.prm_desc[dbg$l_dhdr_symid0],
;  1546          1660  2                               result_desc[dbg$a_value_vmsdesc],bit_length,bit_offset);
;  1547          1661  2          END;
;  1548          1662  2
;  1549          1663  2      .val_desc = .result_desc;
;  1550          1664  2      RETURN sts$k_success;
;  1551          1665  2
;  1552          1666  1      END;                                    ! End of dbg$prim_to_val


                                01FC 00000          .ENTRY  DBG$PRIM_TO_VAL, Save R2,R3,R4,R5,R6,R7,R8 ; 1461
                    58 00000000G  00   9E 00002     MOVAB   LIB$SIGNAL, R8
                    5E       B8   AE   9E 00009     MOVAB   -72(SP), SP
                    56       04   AC   D0 0000D     MOVL    PRM_DESC, R6                               ; 1543
    00000000G       00            56   D0 00011     MOVL    R6, DBG$GL_CURRENT_PRIMARY
                    57       04   A6   9E 00018     MOVAB   4(R6), R7                                  ; 1547
                    07       03   A7   91 0001C     CMPB    3(R7), #7
                                  29   12 00020     BNEQ    2$
                             0C   A6   D5 00022     TSTL    12(R6)                                     ; 1552
                                  1B   13 00025     BEQL    1$
                                  5E   DD 00027     PUSHL   SP                                         ; 1555
                             0C   A6   DD 00029     PUSHL   12(R6)
    00000000G       00            02   FB 0002C     CALLS   #2, DBG$STA_SYMNAME
                                  6E   DD 00033     PUSHL   NAME                                       ; 1556
                                  01   DD 00035     PUSHL   #1
                    00028168      8F   DD 00037     PUSHL   #164200
                    68            03   FB 0003D     CALLS   #3, LIB$SIGNAL
                                  09   11 00040     BRB     2$                                         ; 1552
                    000287F8      8F   DD 00042 1$: PUSHL   #165880                                    ; 1559
                    68            01   FB 00048     CALLS   #1, LIB$SIGNAL
                             0C   A6   D5 0004B 2$: TSTL    12(R6)                                     ; 1565
                                  0A   13 0004E     BEQL    3$
                             0C   A6   DD 00050     PUSHL   12(R6)
    00000000G       00            01   FB 00053     CALLS   #1, DBG$STA_SETCONTEXT
```

```
                                                    56  DD  0005A 3$:     PUSHL   R6                                      ; 1566
                          00000000G   00            01  FB  0005C         CALLS   #1, DBG$COLLECT
                                79    8F      02    A6  91  00063         CMPB    2(R6), #121                             ; 1570
                                                    11  12  00068         BNEQ    4$
                                                38  AE  9F  0006A         PUSHAB  VMS_DESC                                ; 1571
                                                    56  DD  0006D         PUSHL   R6
                              F9BC  CF              02  FB  0006F         CALLS   #2, DBG$MAKE_VMS_DESC
                                    11              50  E8  00074         BLBS    R0, 5$
                                    50              02  D0  00077         MOVL    #2, R0
                                                        04  0007A         RET
                                83  8F      02    A6  91  0007B 4$:     CMPB    2(R6), #131                             ; 1572
                                                    08  12  00080         BNEQ    6$
                    38  AE        14    A6          0C  28  00082         MOVC3   #12, 20(R6), VMS_DESC                   ; 1573
                                                    7A  11  00088 5$:     BRB     9$
                                7A  8F      02    A6  91  0008A 6$:     CMPB    2(R6), #122                             ; 1574
                                                    6A  12  0008F         BNEQ    8$
                24                  00          6E  00  2C  00091         MOVC5   #0, (SP), #0, #36, DATA_DESC            ; 1576
                                                    24  AE      00096
                                26  AE      02    A6  B0  00098         MOVW    2(R6), DATA_DESC+2                      ; 1578
                                30  AE      0C    A6  D0  0009D         MOVL    12(R6), DATA_DESC+12                    ; 1579
                                24  AE            20  B0  000A2         MOVW    #32, DATA_DESC                          ; 1580
                                2A  AE    0603    8F  B0  000A6         MOVW    #1539, DATA_DESC+6                      ; 1582
                                    06      02    A7  91  000AC         CMPB    2(R7), #6                               ; 1588
                                                    52  12  000B0         BNEQ    9$
                                                2C  AE  9F  000B2         PUSHAB  DATA_DESC+8                             ; 1592
                                                08  A6  DD  000B5         PUSHL   8(R6)
                          00000000G   00            02  FB  000B8         CALLS   #2, DBG$STA_TYP_TYPEDPTR
                                                2C  AE  9F  000BF         PUSHAB  DATA_DESC+8                             ; 1593
                          00000000G   00            01  FB  000C2         CALLS   #1, DBG$STA_TYPEFCODE
                                2A  AE              50  90  000C9         MOVB    R0, DATA_DESC+6
                                    01      2A    AE  91  000CD         CMPB    DATA_DESC+6, #1                         ; 1594
                                                    03  12  000D1         BNEQ    7$
                                                00E6    31  000D3         BRW     15$
                                                04  AE  7C  000D6 7$:     CLRQ    BIT_OFFSET                              ; 1596
                                                38  AE  D4  000D9         CLRL    VMS_DESC                                ; 1599
                                3C  AE          18  B6  D0  000DC         MOVL    @24(R6), VMS_DESC+4                     ; 1600
                                                04  AE  9F  000E1         PUSHAB  BIT_OFFSET                              ; 1601
                                                0C  AE  9F  000E4         PUSHAB  BIT_LENGTH
                                                40  AE  9F  000E7         PUSHAB  VMS_DESC
                                                0C  A6  DD  000EA         PUSHL   12(R6)                                  ; 1603
                                                3C  AE  DD  000ED         PUSHL   DATA_DESC+8                             ; 1602
                                    7E          3E  AE  9A  000F0         MOVZBL  DATA_DESC+6, -(SP)                      ; 1601
                              F6E0  CF              06  FB  000F4         CALLS   #6, DBG$FILL_IN_VMS_DESC
                                                    09  11  000F9         BRB     9$                                      ; 1568
                                    000287D8        8F  DD  000FB 8$:     PUSHL   #165848                                 ; 1608
                                    68              01  FB  00101         CALLS   #1, LIB$SIGNAL
                                                08  AC  DD  00104 9$:     PUSHL   TARGET_TYPE                             ; 1611
                                                3C  AE  9F  00107         PUSHAB  VMS_DESC
                              F51E  CF              02  FB  0010A         CALLS   #2, DBG$MAKE_VAL_DESC
                                    52              50  D0  0010F         MOVL    R0, RESULT_DESC
                                03  A2      03    A6  90  00112         MOVB    3(R6), 3(RESULT_DESC)                   ; 1613
                                    53          04  A2  9E  00117         MOVAB   4(RESULT_DESC), R3                      ; 1614
                                02  A3      02    A7  B0  0011B         MOVW    2(R7), 2(R3)                            ; 1615
                                08  A2      08    A6  7D  00120         MOVQ    8(R6), 8(RESULT_DESC)                   ; 1616
                50      67              01          07  EF  00125         EXTZV   #7, #1, (R7), R0                        ; 1618
                63      01              07          50  F0  0012A         INSV    R0, #7, #1, (R3)
                                        07      02  A3  91  0012F         CMPB    2(R3), #7                               ; 1624
```

```
                                 20  12 00133          BNEQ    12$                                                     ;
                           50    0C  A2  D0 00135          MOVL    12(RESULT_DESC), TMP_SYMID                              ; 1629
                           01    14  A0  91 00139 10$:     CMPB    20(TMP_SYMID), #1                                       ; 1630
                                 06  13 0013D          BEQL    11$
                           50    10  A0  D0 0013F          MOVL    16(TMP_SYMID), TMP_SYMID                                ; 1631
                                 F4  11 00143          BRB     10$
                           03    29  A0  91 00145 11$:     CMPB    41(TMP_SYMID), #3                                       ; 1632
                                 0A  12 00149          BNEQ    12$
                        02 A3        03  90 0014B          MOVB    #3, 2(R3)                                               ; 1635
                        16 A2  010E  8F  B0 0014F          MOVW    #270, 22(RESULT_DESC)                                   ; 1637
                0000007A 8F    08    AC  D1 00155 12$:     CMPL    TARGET_TYPE, #122                                       ; 1647
                                 55  12 0015D          BNEQ    14$
                           09    02  A3  91 0015F 13$:     CMPB    2(R3), #9                                               ; 1648
                                 4F  12 00163          BNEQ    14$
                                 20  AE  9F 00165          PUSHAB  BIT_LENGTH                                              ; 1651
                                 10  AE  9F 00168          PUSHAB  DUMMY
                                 14  AE  9F 0016B          PUSHAB  DUMMY
                                 1C  AE  9F 0016E          PUSHAB  PARENT
                                 08  A2  DD 00171          PUSHL   8(RESULT_DESC)
                00000000G  00          05  FB 00174          CALLS   #5, DBG$STA_TYP_SUBRNG
                                 1C  AE  D4 0017B          CLRL    BIT_OFFSET                                              ; 1652
                                 14  AE  9F 0017E          PUSHAB  TYPEID                                                  ; 1653
                                 1C  AE  9F 00181          PUSHAB  FCODE
                                 18  AE  DD 00184          PUSHL   PARENT
                00000000G  00          03  FB 00187          CALLS   #3, DBG$STA_SYMTYPE
                        02 A3    18  AE  90 0018E          MOVB    FCODE, 2(R3)                                            ; 1654
                        08 A2    14  AE  D0 00193          MOVL    TYPEID, 8(RESULT_DESC)                                  ; 1655
                                 14  A2  D4 00198          CLRL    20(RESULT_DESC)                                         ; 1658
                                 1C  AE  9F 0019B          PUSHAB  BIT_OFFSET                                              ; 1660
                                 24  AE  9F 0019E          PUSHAB  BIT_LENGTH
                                 14  A2  9F 001A1          PUSHAB  20(RESULT_DESC)
                                 0C  A6  DD 001A4          PUSHL   12(R6)
                                 24  AE  DD 001A7          PUSHL   TYPEID
                                 2C  AE  DD 001AA          PUSHL   FCODE
                      F627 CF          06  FB 001AD          CALLS   #6, DBG$FILL_IN_VMS_DESC
                                 AB  11 001B2          BRB     13$                                                     ; 1648
                        0C BC          52  D0 001B4 14$:     MOVL    RESULT_DESC, @VAL_DESC                          ; 1663
                           50          01  D0 001B8          MOVL    #1, R0                                          ; 1664
                                       04 001BB          RET
                                 50  D4  001BC 15$:     CLRL    R0                                                      ; 1666
                                       04 001BE          RET

; Routine Size:  447 bytes,    Routine Base:  DBG$CODE + 0B16
```

```
;  1554     1667  1 GLOBAL ROUTINE DBG$PRINT_AGGREGATE(prm_desc,radix) : NOVALUE =
;  1555     1668  2     BEGIN
;  1556     1669  2     MAP prm_desc        : REF dbg$primary;
;  1557     1670  2     BUILTIN REMQUE;
;  1558     1671  2     LOCAL
;  1559     1672  2         subnode         : REF dbg$prim_node,
;  1560     1673  2         val_desc        : REF dbg$valdesc,
;  1561     1674  2         symid,kind,fcode,typeid,dummy,mark_one,mark_two;
;  1562     1675  2
;  1563     1676  2     dbg$gl_current_primary = .prm_desc;                                    ! A003
;  1564     1677  2
;  1565     1678  2     dbg$newline();
;  1566     1679  2     dbg$print_control(dbg$k_prtset_rlmargin,+4);        ! Indent by +4
;  1567     1680  2     subnode = .prm_desc[dbg$l_prim_blink];
;  1568     1681  2     subnode[dbg$v_pnode_eval] = true;
;  1569     1682  2     SELECTONE .subnode[dbg$b_pnode_fcode] OF
;  1570     1683  2         SET
;  1571     1684  2         [rst$k_type_array]:
;  1572     1685  3             BEGIN
;  1573     1686  3             LABEL cell;
;  1574     1687  3             LOCAL s_vector      : REF dbg$prim_node_subs;
;  1575     1688  3
;  1576     1689  3             s_vector = subnode[dbg$a_pnarr_svector];
;  1577     1690  3
;  1578     1691  3             ! Check for the array being empty (i.e., if any of the
;  1579     1692  3             ! dimensions has zero or negative length). If this is the
;  1580     1693  3             ! case, indicate that this is an empty array,
;  1581     1694  3             ! and then clean up and return.
;  1582     1695  3             !
;  1583     1696  3             INCR i FROM 0 to .subnode[dbg$b_pnarr_dimcnt]-1 DO
;  1584     1697  4                 BEGIN
;  1585     1698  4                 IF .s_vector[.i,dbg$l_pnsub_lbound] GTR
;  1586     1699  4                    .s_vector[.i,dbg$l_pnsub_ubound]
;  1587     1700  4                 THEN
;  1588     1701  5                     BEGIN
;  1589     1702  5                     dbg$print(UPLIT (%ASCIC '[empty array]'));
;  1590     1703  5                     dbg$newline();
;  1591     1704  5                     subnode[dbg$v_pnode_eval] = false;
;  1592     1705  5                     dbg$print_control(dbg$k_prtset_rlmargin,-4);
;  1593     1706  5                     RETURN;
;  1594     1707  4                     END;
;  1595     1708  3                 END;
;  1596     1709  3
;  1597     1710  3             mark_one = dbg$push_tempmem();
;  1598     1711  3             dbg$sta_symtype(.subnode[dbg$l_pnarr_celltype],fcode,typeid);
;  1599     1712  3             dbg$build_primary_subnode(.prm_desc,rst$k_data,0,,fcode,,typeid,0);
;  1600     1713  3             dbg$collect(.prm_desc);
;  1601     1714  3             WHILE NOT .dbg$gv_control[dbg$v_control_stop] DO
;  1602     1715  4 cell:             BEGIN
;  1603     1716  4                 mark_two = dbg$push_tempmem();
;  1604     1717  4                 dbg$print_identifier(.prm_desc,0);
;  1605     1718  4                 IF .prm_desc[dbg$v_dhdr_aggr]
;  1606     1719  4                   THEN dbg$print_aggregate(.prm_desc,,radix)
;  1607     1720  4                   ELSE
;  1608     1721  5                     BEGIN
;  1609     1722  5                     dbg$print(UPLIT BYTE(%ASCIC '!AD!_'),1,UPLIT BYTE(':'));
;  1610     1723  5
```

```
;  1611     1724  5                    ! If you examine a label array in PLI then you see the
;  1612     1725  5                    ! instructions at the labels. Instructions must always
;  1613     1726  5                    ! be represented as volatile value descriptors (since
;  1614     1727  5                    ! if you copy the bits, then the operands may change).
;  1615     1728  5                    !
;  1616     1729  5                    IF .fcode EQL rst$k_type_self_rel_lab
;  1617     1730  5                    THEN
;  1618     1731  5                        dbg$prim_to_val(.prm_desc,dbg$k_v_value_desc,val_desc)
;  1619     1732  5                    ELSE
;  1620     1733  5                        dbg$prim_to_val(.prm_desc,dbg$k_value_desc,val_desc);
;  1621     1734  5                    dbg$print_value(.val_desc,.radix, .dbg$gl_sign_flag);
;  1622     1735  5                    dbg$newline();
;  1623     1736  4                    END;
;  1624     1737  4                dbg$pop_tempmem(.mark_two);
;  1625     1738  4                INCR dimension FROM 1 TO .subnode[dbg$b_pnarr_dimcnt] DO
;  1626     1739  5                    BEGIN
;  1627     1740  5                    LOCAL s,typeid: REF rst$entry;
;  1628     1741  6                    s = (IF .subnode[dbg$v_pnarr_column]
;  1629     1742  6                        THEN .dimension - 1
;  1630     1743  5                        ELSE .subnode[dbg$b_pnarr_dimcnt] - .dimension);
;  1631     1744  5
;  1632     1745  5                    typeid  = .s_vector[.s,dbg$l_pnsub_typeid];
;  1633     1746  6                    IF  .s_vector[.s,dbg$l_pnsub_svalue] GEQ (
;  1634     1747  6                        IF (.dbg$gb_language EQL dbg$k_ada) AND
;  1635     1748  7                           (.typeid NEQ 0)
;  1636     1749  6                        THEN
;  1637     1750  7                            IF (.typeid[rst$b_fcode] EQL rst$k_type_enum)
;  1638     1751  6                            THEN
;  1639     1752  6                                dbg$enum_val(.typeid, .s_vector[.s,dbg$l_pnsub_ubound])
;  1640     1753  6                            ELSE
;  1641     1754  6                                .s_vector[.s,dbg$l_pnsub_ubound]
;  1642     1755  6                        ELSE
;  1643     1756  6                            .s_vector[.s,dbg$l_pnsub_ubound])
;  1644     1757  5                    THEN
;  1645     1758  6                        BEGIN
;  1646     1759  6
;  1647     1760  6                        ! For arrays indexed by enumeration types in ADA, the lower bound field gives
;  1648     1761  6                        ! the position, which we need to translate into a value.
;  1649     1762  6                        !
;  1650     1763  6                        typeid  = .s_vector[.s,dbg$l_pnsub_typeid];
;  1651     1764  6                        IF (.dbg$gb_language EQL dbg$k_ada) AND
;  1652     1765  7                           (.typeid NEQ 0)
;  1653     1766  6                        THEN
;  1654     1767  7                            IF (.typeid[rst$b_fcode] EQL rst$k_type_enum)
;  1655     1768  6                            THEN
;  1656     1769  6                                s_vector[.s,dbg$l_pnsub_svalue] = dbg$enum_val(.typeid, .s_vector[.s,dbg$l_p
;  1657     1770  6                            ELSE
;  1658     1771  6                                s_vector[.s,dbg$l_pnsub_svalue]  =  .s_vector[.s,dbg$l_pnsub_lbound]
;  1659     1772  6                        ELSE
;  1660     1773  6                            s_vector[.s,dbg$l_pnsub_svalue]  =  .s_vector[.s,dbg$l_pnsub_lbound];
;  1661     1774  6                        END
;  1662     1775  5                    ELSE
;  1663     1776  6                        BEGIN
;  1664     1777  6                        LOCAL
;  1665     1778  6                            s_value;
;  1666     1779  6
;  1667     1780  6                        ! For arrays indexed by enumeration types in ADA, we need to use ENUM_SUCC to get
```

```
;  1668          1781  6                    ! the successor subscript. In all other cases, we can just add one.
;  1669          1782  6                    !
;  1670          1783  6                    s_value = .s_vector[.s,dbg$l_pnsub_svalue];
;  1671          1784  6                    typeid  = .s_vector[.s,dbg$l_pnsub_typeid];
;  1672          1785  6                    IF (.dbg$gb_language EQL dbg$k_ada) AND
;  1673          1786  7                       (.typeid NEQ 0)
;  1674          1787  6                    THEN
;  1675          1788  7                        IF (.typeid[rst$b_fcode] EQL rst$k_type_enum)
;  1676          1789  6                        THEN
;  1677          1790  6                            s_vector[.s,dbg$l_pnsub_svalue] = dbg$enum_succ(.typeid, .s_value)
;  1678          1791  6                        ELSE
;  1679          1792  6                            s_vector[.s,dbg$l_pnsub_svalue]  =  .s_vector[.s,dbg$l_pnsub_svalue] + 1
;  1680          1793  6                    ELSE
;  1681          1794  6                        s_vector[.s,dbg$l_pnsub_svalue]  =  .s_vector[.s,dbg$l_pnsub_svalue] + 1;
;  1682          1795  6                    LEAVE cell;
;  1683          1796  5                    END;
;  1684          1797  4                END;
;  1685          1798  4            EXITLOOP;
;  1686          1799  3            END;                    ! End of block 'cell'
;  1687          1800  3        REMQUE(.prm_desc[dbg$l_prim_blink],dummy);
;  1688          1801  3        dbg$pop_tempmem(.mark_one);
;  1689          1802  2        END;
```

```
;  1691          1803  2          [rst$k_type_record,rst$k_type_variant]:
;  1692          1804  3              BEGIN
;  1693          1805  3              LOCAL n_comps,s_vector : REF VECTOR [,LONG];
;  1694          1806  3              IF .subnode[dbg$b_pnode_fcode] EQL rst$k_type_record
;  1695          1807  3                THEN
;  1696          1808  3                  dbg$sta_typ_record(.subnode[dbg$l_pnode_typeid],n_comps,s_vector,dummy)
;  1697          1809  3                ELSE
;  1698          1810  4                  BEGIN
;  1699          1811  4                  n_comps  = .subnode[dbg$w_pnvar_ncomps];
;  1700          1812  4                  s_vector = .subnode[dbg$l_pnvar_complst];
;  1701          1813  3                  END;
;  1702          1814  3              INCR component FROM 0 TO .n_comps-1 DO
;  1703          1815  3                IF (symid = .s_vector[.component]) NEQ 0 THEN
;  1704          1816  4                  BEGIN
;  1705          1817  4                  IF .dbg$gv_control[dbg$v_control_stop] THEN EXITLOOP;
;  1706          1818  4                  IF .prm_desc[dbg$v_dhdr_tmpref] THEN
;  1707          1819  5                      BEGIN
;  1708          1820  5                      prm_desc[dbg$v_dhdr_tmpref] = false;
;  1709          1821  5                      prm_desc[dbg$v_dhdr_subref] = false;
;  1710          1822  5                      prm_desc[dbg$w_prim_offset] = 0;
;  1711          1823  5                      prm_desc[dbg$w_prim_length] = 0;
;  1712          1824  4                      END;
;  1713          1825  4                  mark_one = dbg$push_tempmem();
;  1714          1826  4                  dbg$sta_symkind(.symid,kind);
;  1715          1827  4                  IF .kind EQL rst$k_variant
;  1716          1828  4                    THEN
;  1717          1829  5                      BEGIN
;  1718          1830  5                      LOCAL
;  1719          1831  5                          tagid,tag_val,
;  1720          1832  5                          tag_name        : REF VECTOR[,BYTE],
;  1721          1833  5                          variant         : REF rst$var_entry;
;  1722          1834  5                      MAP symid           : REF rst$entry;
;  1723          1835  5
;  1724          1836  5                      variant = 0;
;  1725          1837  5                      IF (tagid = .symid[rst$l_vartagptr]) NEQ 0 THEN
;  1726          1838  6                          BEGIN
;  1727          1839  6                          dbg$sta_symname(.tagid,tag_name);
;  1728          1840  6                          IF .tag_name[0] NEQ 0 THEN
;  1729          1841  7                              BEGIN
;  1730          1842  7                              dbg$sta_symtype(.tagid,fcode,typeid);
;  1731          1843  7                              dbg$build_primary_subnode(.prm_desc,rst$k_data,.tagid,.fcode,.typeid,0);
;  1732          1844  7                              dbg$prim_to_val(.prm_desc,dbg$k_value_desc,val_desc);
;  1733          1845  7                              tag_val = .val_desc[dbg$l_value_value0];
;  1734          1846  7                              variant = dbg$sta_variant_select(.tag_val,.symid);
;  1735          1847  7                              REMQUE(.prm_desc[dbg$l_prim_blink],dummy);
;  1736          1848  6                              END;
;  1737          1849  5                          END;
;  1738          1850  5                      IF .variant EQL 0
;  1739          1851  5                        THEN
;  1740          1852  6                          BEGIN
;  1741          1853  6                          dbg$print(UPLIT(%ASCIC '!AD'),52,UPLIT BYTE('[Variant Record omitted - null or illeg
;  1742          1854  6                          dbg$newline();
;  1743          1855  6                          END
;  1744          1856  5                        ELSE
;  1745          1857  6                          BEGIN
;  1746          1858  6                          dbg$build_primary_subnode(.prm_desc,rst$k_variant,0,rst$k_type_variant,0,0);
;  1747          1859  6                          subnode = .prm_desc[dbg$l_prim_blink];
```

```
;  1748          1860  6                    subnode[dbg$l_pnvar_tagid]   = .tagid;
;  1749          1861  6                    subnode[dbg$w_pnvar_index]   = 1;
;  1750          1862  6                    subnode[dbg$v_pnvar_valid]   = true;
;  1751          1863  6                    subnode[dbg$w_pnvar_ncomps]  = .variant[rst$l_var_compcnt];
;  1752          1864  6                    subnode[dbg$l_pnvar_complst] =  variant[rst$a_var_complst];
;  1753          1865  6                    subnode[dbg$l_pnvar_dstptr]  = .variant[rst$l_var_dstptr];
;  1754          1866  6                    dbg$print(UPLIT(%ASCIC '!AD'),30,UPLIT BYTE('Variant Record with Tag Value '));
;  1755          1867  6                    dbg$print_value(.val_desc,.radix, .dbg$gl_sign_flag);
;  1756          1868  6                    dbg$print_aggregate(.prm_desc,.radix);
;  1757          1869  6                    REMQUE(.prm_desc[dbg$l_prim_blink],dummy);
;  1758          1870  5                    END;
;  1759          1871  5                END
;  1760          1872  4            ELSE
;  1761          1873  5                BEGIN
;  1762          1874  5                dbg$sta_symtype(.symid,fcode,typeid);
;  1763          1875  5                dbg$build_primary_subnode(.prm_desc,.kind,.symid,.fcode,.typeid,0);
;  1764          1876  5                dbg$collect(.prm_desc);
;  1765          1877  5                IF .prm_desc[dbg$v_dhdr_aggr]
;  1766          1878  5                  THEN
;  1767          1879  6                    BEGIN
;  1768          1880  6                    dbg$print_identifier(.prm_desc,0);
;  1769          1881  6                    dbg$print_aggregate(.prm_desc,.radix);
;  1770          1882  6                    END
;  1771          1883  5                  ELSE
;  1772          1884  6                    BEGIN
;  1773          1885  6                    LOCAL name : REF VECTOR[,BYTE];
;  1774          1886  6                    dbg$sta_symname(.symid,name);
;  1775          1887  6                    IF .name[0] NEQ 0 THEN
;  1776          1888  7                        BEGIN
;  1777          1889  7                        dbg$print_identifier(.prm_desc,0);
;  1778          1890  7                        dbg$print(UPLIT BYTE(%ASCIC '!AD!_'),1,UPLIT BYTE(':'));
;  1779          1891  7                        dbg$prim_to_val(.prm_desc,dbg$k_value_desc,val_desc);
;  1780          1892  7                        dbg$print_value(.val_desc,.radix, .dbg$gl_sign_flag);
;  1781          1893  7                        dbg$newline();
;  1782          1894  6                        END;
;  1783          1895  5                    END;
;  1784          1896  5                REMQUE(.prm_desc[dbg$l_prim_blink],dummy);
;  1785          1897  4                END;
;  1786          1898  4            dbg$pop_tempmem(.mark_one);
;  1787          1899  3            END;
;  1788          1900  2        END;
;  1789          1901  2
;  1790          1902  2        [OTHERWISE]:
;  1791          1903  2            SIGNAL(dbg$_illtype);
;  1792          1904  2
;  1793          1905  2        TES;
;  1794          1906  2
;  1795          1907  2    subnode[dbg$v_pnode_eval]   = false;
;  1796          1908  2    prm_desc[dbg$v_dhdr_aggr]   = true;
;  1797          1909  2    prm_desc[dbg$b_dhdr_kind]   = .subnode[dbg$b_pnode_kind];
;  1798          1910  2    prm_desc[dbg$b_dhdr_fcode]  = .subnode[dbg$b_pnode_fcode];
;  1799          1911  2    prm_desc[dbg$l_dhdr_typeid] = .subnode[dbg$l_pnode_typeid];
;  1800          1912  2    dbg$print_control(dbg$k_prtset_rlmargin,-4);        ! Reset indentation
;  1801          1913  1    END;                                    ! End of dbg$print_aggregate
```

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

00  5D  79  61  72  72  61  20  79  74  70  6D  65  5B  0D  00008 P.AAC:  .ASCII  <13>\[empty array]\<0><0>                    ;
                                                          00  00017                                                          ;
                                    .   5F  21  44  41  21  05  00018 P.AAD:  .ASCII  <5>\!AD!_\                           ;
                                                          3A  0001E P.AAE:  .ASCII  \:\                                    ;
                                                              0001F         .BLKB   1                                      ;
                                                      44  41  21  03  00020 P.AAF:  .ASCII  <3>\!AD\                       ;
64  72  6F  63  65  52  20  74  6E  61  69  72  61  56  5B  00024 P.AAG:  .ASCII  \[Variant Record omitted - null or illega\ ;
6C  6C  75  6E  20  2D  20  64  65  74  74  69  6D  6F  20  00033                                                          ;
                    61  67  65  6C  6C  69  20  72  6F  20  00042                                                          ;
            5D  65  75  6C  61  56  20  67  61  54  20  6C  0004C         .ASCII  \l Tag Value]\                           ;
                                                      44  41  21  03  00058 P.AAH:  .ASCII  <3>\!AD\                       ;
20  64  72  6F  63  65  52  20  74  6E  61  69  72  61  56  0005C P.AAI:  .ASCII  \Variant Record with Tag Value \         ;
20  65  75  6C  61  56  20  67  61  54  20  68  74  69  77  0006B                                                          ;
                            5F  21  44  41  21  05  0007A P.AAJ:  .ASCII  <5>\!AD!_\                                     ;
                                                          3A  00080 P.AAK:  .ASCII  \:\                                    ;


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

                                                          0FFC 00000          .ENTRY  DBG$PRINT_AGGREGATE, Save R2,R3,R4,R5,R6,- ; 1667
                                                                                      R7,R8,R9,R10,R11                       ;
                                                  5E      28  C2  00002       SUBL2   #40, SP                                ;
                                                  58      04  AC  D0  00005   MOVL    PRM_DESC, R8                           ; 1676
                                  00000000G       00      58  D0  00009       MOVL    R8, DBG$GL_CURRENT_PRIMARY             ;
                                  00000000G       00          00  FB  00010   CALLS   #0, DBG$NEWLINE                      ; 1678
                                                              04  DD  00017   PUSHL   #4                                    ; 1679
                                                              02  DD  00019   PUSHL   #2                                    ;
                                  00000000G       00          02  FB  0001B   CALLS   #2, DBG$PRINT_CONTROL                 ;
                                                  52      18  A8  D0  00022   MOVL    24(R8), SUBNODE                        ; 1680
                                        0A        A2          01  88  00026   BISB2   #1, 10(SUBNODE)                       ; 1681
                                                  50      09  A2  9A  0002A   MOVZBL  9(SUBNODE), R0                        ; 1682
                                                  01          50  91  0002E   CMPB    R0, #1                                 ; 1684
                                                              03  13  00031   BEQL    1$                                     ;
                                                          01C2  31  00033     BRW     23$                                    ;
                                                  53      28  A2  9E  00036 1$: MOVAB   40(R2), S_VECTOR                    ; 1689
                                                  55      1B  A2  9A  0003A   MOVZBL  27(SUBNODE), R5                      ; 1696
                                                  54          01  CE  0003E   MNEGL   #1, I                                 ; 1698
                                                              2C  11  00041   BRB     3$                                     ;
50                                                54          14  C5  00043 2$: MULL3   #20, I, R0                          ;
                                                          0C  A043  9F  00047   PUSHAB  12(R0)[S_VECTOR]                     ; 1699
                                                          08  A043  9F  0004B   PUSHAB  8(R0)[S_VECTOR]                      ;
                                                  9E              9E  D1  0004F   CMPL    @(SP)+, @(SP)+                     ;
                                                                  1B  15  00052   BLEQ    3$                                  ;
                                              00000000'           EF  9F  00054   PUSHAB  P.AAC                               ; 1702
                                  00000000G       00              01  FB  0005A   CALLS   #1, DBG$PRINT                     ;
                                  00000000G       00              00  FB  00061   CALLS   #0, DBG$NEWLINE                   ; 1703
                                        0A        A2              01  8A  00068   BICB2   #1, 10(SUBNODE)                    ; 1704
                                                              03C5  31  0006C     BRW     42$                                    ; 1705
D0                                                54              55  F2  0006F 3$: AOBLSS  R5, I, 2$                        ; 1696
                                  00000000G       00              00  FB  00073   CALLS   #0, DBG$PUSH_TEMPMEM                ; 1710
                                                  6E              50  D0  0007A   MOVL    R0, MARK_ONE                       ;
                                                          18      AE  9F  0007D   PUSHAB  TYPEID                              ; 1711
                                                          20      AE  9F  00080   PUSHAB  FCODE                               ;
                                                          24      A2  DD  00083   PUSHL   36(SUBNODE)                        ;
```

```
                00000000G  00          03  FB 00086          CALLS   #3, DBG$STA_SYMTYPE                          ;
                           7E           D4  0008D          CLRL    -(SP)                                       ; 1712
                       1C  AE           DD  0008F          PUSHL   TYPEID                                      ;
                       24  AE           DD  00092          PUSHL   FCODE                                       ;
                     7E    06           7D  00095          MOVQ    #6, -(SP)                                   ;
                           58           DD  00098          PUSHL   R8                                          ;
                00000000G  00           06  FB 0009A          CALLS   #6, DBG$BUILD_PRIMARY_SUBNODE               ;
                           58           DD  000A1          PUSHL   R8                                          ; 1713
                00000000G  00           01  FB 000A3          CALLS   #1, DBG$COLLECT                             ;
03              00000000G  00           01  E1 000AA 4$:       BBC     #1, DBG$GV_CONTROL+1, 5$                    ; 1714
                                  0132  31  000B2          BRW     21$                                         ;
                00000000G  00           00  FB 000B5 5$:       CALLS   #0, DBG$PUSH_TEMPMEM                        ; 1716
                           5B           50  D0 000BC          MOVL    R0, MARK_TWO                                ;
                           7E           D4  000BF          CLRL    -(SP)                                       ; 1717
                           58           DD  000C1          PUSHL   R8                                          ;
                00000000G  00           02  FB 000C3          CALLS   #2, DBG$PRINT_IDENTIFIER                    ;
                           0C       04  A8  E9 000CA          BLBC    4(R8), 6$                                   ; 1718
                           08       AC  DD  000CE          PUSHL   RADIX                                       ; 1719
                           58           DD  000D1          PUSHL   R8                                          ;
                     FF28  CF           02  FB 000D3          CALLS   #2, DBG$PRINT_AGGREGATE                     ;
                           4A           11  000D8          BRB     9$                                          ;
                    00000000'           EF  9F 000DA 6$:       PUSHAB  P.AAE                                       ; 1722
                           01           DD  000E0          PUSHL   #1                                          ;
                    00000000'           EF  9F 000E2          PUSHAB  P.AAD                                       ;
                00000000G  00           03  FB 000E8          CALLS   #3, DBG$PRINT                               ;
                           15       1C  AE  D1 000EF          CMPL    FCODE, #21                                  ; 1729
                           09           12  000F3          BNEQ    7$                                          ;
                           24  AE           9F 000F5          PUSHAB  VAL_DESC                                    ; 1731
                     7E    83  8F           9A 000F8          MOVZBL  #13T, -(SP)                                 ;
                           07           11  000FC          BRB     8$                                          ;
                           24  AE           9F 000FE 7$:       PUSHAB  VAL_DESC                                    ; 1733
                     7E    7A  8F           9A 00101          MOVZBL  #122, -(SP)                                 ;
                           58           DD  00105 8$:       PUSHL   R8                                          ;
                 FD35  CF                  03  FB 00107          CALLS   #3, DBG$PRIM_TO_VAL                         ;
                    00000000G              00  DD 0010C          PUSHL   DBG$GL_SIGN_FLAG                            ; 1734
                           08           AC  DD  00112          PUSHL   RADIX                                       ;
                           2C           AE  DD  00115          PUSHL   VAL_DESC                                    ;
                  0000V    CF               03  FB 00118          CALLS   #3, DBG$PRINT_VALUE                         ;
                00000000G  00               00  FB 0011D          CALLS   #0, DBG$NEWLINE                             ; 1735
                           5B               DD  00124 9$:       PUSHL   MARK_TWO                                    ; 1737
                00000000G  00               01  FB 00126          CALLS   #1, DBG$POP_TEMPMEM                         ;
                           5A           1B  A2  9A 0012D          MOVZBL  27(SUBNODE), R10                            ; 1738
                                            54  D4 00131          CLRL    DIMENSION                                   ;
                                            7C  11 00133          BRB     16$                                         ;
06                    0A   A2                01  E1 00135 10$:      BBC     #1, 10(SUBNODE), 11$                        ; 1741
                           50           FF  A4  9E 0013A          MOVAB   -1(R4), S                                   ; 1742
                                            04  11 0013E          BRB     12$                                         ;
50                         5A               54  C3 00140 11$:      SUBL3   DIMENSION, R10, S                           ; 1743
56                         50               14  C5 00144 12$:      MULL3   #20, S, R6                                  ; 1745
                                       10 A643  9F 00148          PUSHAB  16(R6)[S_VECTOR]                            ;
                           59               9E  D0 0014C          MOVL    @(SP)+, R9                                  ;
                           55               59  D0 0014F          MOVL    R9, TYPEID                                  ;
57                         53               56  C1 00152          ADDL3   R6, S_VECTOR, R7                            ; 1746
                           50           0C A643  9E 00156          MOVAB   12(R6)[S_VECTOR], R0                        ; 1752
                           09 00000000G        91  0015B          CMPB    DBG$GB_LANGUAGE, #9                         ; 1747
                                               12  00162          BNEQ    13$                                         ;
                                               D5  00164          TSTL    TYPEID                                      ; 1748
```

```
                                13  13 00166          BEQL    13$                                     ;
                          04        18  A5  91 00168          CMPB    24(TYPEID), #4                  ; 1750
                                    0D  12  0016C          BNEQ    13$                             ;
                                    60  DD  0016E          PUSHL   (R0)                            ; 1752
                                    55  DD  00170          PUSHL   TYPEID                          ;
                00000000G   00          02  FB  00172          CALLS   #2, DBG$ENUM_VAL            ;
                                    03  11  00179          BRB     14$                             ;
                          50            60  D0  0017B 13$:     MOVL    (R0), R0                    ; 1756
                          50            67  D1  0017E 14$:     CMPL    (R7), R0                    ; 1746
                                    30  19  00181          BLSS    17$                             ;
                          55            59  D0  00183          MOVL    R9, TYPEID                  ; 1763
                          50      08 A643  9E  00186          MOVAB   8(R6)[S_VECTOR], R0         ; 1769
                          09 00000000G  00  91  0018B          CMPB    DBG$GB_LANGUAGE, #9         ; 1764
                                    1A  12  00192          BNEQ    15$                             ;
                                    55  D5  00194          TSTL    TYPEID                          ; 1765
                                    16  13  00196          BEQL    15$                             ;
                          04        18  A5  91 00198          CMPB    24(TYPEID), #4              ; 1767
                                    10  12  0019C          BNEQ    15$                             ;
                                    60  DD  0019E          PUSHL   (R0)                            ; 1769
                                    55  DD  001A0          PUSHL   TYPEID                          ;
                00000000G   00          02  FB  001A2          CALLS   #2, DBG$ENUM_VAL            ;
                          67            50  D0  001A9          MOVL    R0, (R7)                    ;
                                    33  11  001AC          BRB     20$                             ;
                          67            60  D0  001AE 15$:     MOVL    (R0), (R7)                  ; 1773
                                    2E  11  001B1 16$:     BRB     20$                             ; 1746
                          50            67  D0  001B3 17$:     MOVL    (R7), S_VALUE               ; 1783
                          55            59  D0  001B6          MOVL    R9, TYPEID                  ; 1784
                          09 00000000G  00  91  001B9          CMPB    DBG$GB_LANGUAGE, #9         ; 1785
                                    1A  12  001C0          BNEQ    18$                             ;
                                    55  D5  001C2          TSTL    TYPEID                          ; 1786
                                    16  13  001C4          BEQL    18$                             ;
                                04        18  A5  91 001C6          CMPB    24(TYPEID), #4          ; 1788
                                    10  12  001CA          BNEQ    18$                             ;
                                    50  DD  001CC          PUSHL   S_VALUE                         ; 1790
                                    55  DD  001CE          PUSHL   TYPEID                          ;
                            00000000G   00          02  FB  001D0          CALLS   #2, DBG$ENUM_SUCC   ;
                                67            50  D0  001D7          MOVL    R0, (R7)                ;
                                    02  11  001DA          BRB     19$                             ;
                                    67  D6  001DC 18$:     INCL    (R7)                            ; 1794
                                FEC9    31  001DE 19$:     BRW     4$                              ; 1795
                FF4E        54          01      5A  F1  001E1 20$:     ACBL    R10, #1, DIMENSION, 10$ ; 1738
                                  04    AE        18  B8  0F  001E7 21$:     REMQUE  @24(R8), DUMMY  ; 1800
                                    6E  DD  001EC          PUSHL   MARK_ONE                        ; 1801
                            00000000G   00          01  FB  001EE          CALLS   #1, DBG$POP_TEMPMEM ;
                                  0221      31  001F5 22$:     BRW     41$                             ; 1682
                                07            50  91  001F8 23$:     CMPB    R0, #7                  ; 1803
                                    08  13  001FB          BEQL    24$                             ;
                                13            50  91  001FD          CMPB    R0, #19                 ;
                                    03  13  00200          BEQL    24$                             ;
                                  0207      31  00202          BRW     40$                             ;
                                07            50  91  00205 24$:     CMPB    R0, #7                  ; 1806
                                    15  12  00208          BNEQ    25$                             ;
                                04    AE  9F  0020A          PUSHAB  DUMMY                           ; 1808
                                0C    AE  9F  0020D          PUSHAB  S_VECTOR                        ;
                                14    AE  9F  00210          PUSHAB  N_COMPS                         ;
                                0C    A2  DD  00213          PUSHL   12(SUBNODE)                     ;
                            00000000G   00          04  FB  00216          CALLS   #4, DBG$STA_TYP_RECORD ;
```

```
                                    0A   11 0021D                BRB     26$                                     ;
                              0C    AE      1A    A2  3C 0021F 25$:     MOVZWL  26(SUBNODE), N_COMPS            ; 1811
                              08    AE      20    A2  D0 00224         MOVL    32(SUBNODE), S_VECTOR           ; 1812
                                    53      0C    AE  D0 00229 26$:     MOVL    N_COMPS, R3                     ; 1814
                                    54            01  CE 0022D         MNEGL   #1, COMPONENT                   ;
                                                01D0  31 00230 27$:     BRW     38$                             ;
                                    57      08 BE44   D0 00233 28$:     MOVL    @S_VECTOR[COMPONENT], SYMID     ; 1815
                                                F6    13 00238         BEQL    27$                             ;
                        B3 00000000G    00            01  E0 0023A         BBS     #1, DBG$GV_CONTROL+1, 22$       ; 1817
                                    55      04    AC  D0 00242         MOVL    PRM_DESC, R5                    ; 1818
                                    09      05    A5  E9 00246         BLBC    5(R5), 29$                      ;
                              04    A5    0102    8F  AA 0024A         BICW2   #258, 4(R5)                     ; 1820
                                    10      A5    D4 00250             CLRL    16(R5)                          ; 1822
                           00000000G    00            00  FB 00253 29$:     CALLS   #0, DBG$PUSH_TEMPMEM            ; 1825
                                    6E            50  D0 0025A         MOVL    R0, MARK_ONE                    ;
                                          10    AE  9F 0025D         PUSHAB  KIND                            ; 1826
                                                57  DD 00260         PUSHL   SYMID                           ;
                           00000000G    00            02  FB 00262         CALLS   #2, DBG$STA_SYMKIND             ;
                                    0B          10    AE  D1 00269         CMPL    KIND, #11                       ; 1827
                                                03  13 0026D         BEQL    30$                             ;
                                              00E3  31 0026F         BRW     33$                             ;
                                                5A  D4 00272 30$:     CLRL    VARIANT                         ; 1836
                                    56          10    A7  D0 00274         MOVL    16(SYMID), TAGID                ; 1837
                                                5E  13 00278         BEQL    31$                             ;
                                          14    AE  9F 0027A         PUSHAB  TAG_NAME                        ; 1839
                                                56  DD 0027D         PUSHL   TAGID                           ;
                           00000000G    00            02  FB 0027F         CALLS   #2, DBG$STA_SYMNAME             ;
                                          14    BE  95 00286         TSTB    @TAG_NAME                       ; 1840
                                                4D  13 00289         BEQL    31$                             ;
                                          18    AE  9F 0028B         PUSHAB  TYPEID                          ; 1842
                                          20    AE  9F 0028E         PUSHAB  FCODE                           ;
                                                56  DD 00291         PUSHL   TAGID                           ;
                           00000000G    00            03  FB 00293         CALLS   #3, DBG$STA_SYMTYPE             ;
                                                7E  D4 0029A         CLRL    -(SP)                           ; 1843
                                          1C    AE  DD 0029C         PUSHL   TYPEID                          ;
                                          24    AE  DD 0029F         PUSHL   FCODE                           ;
                                                56  DD 002A2         PUSHL   TAGID                           ;
                                                06  DD 002A4         PUSHL   #6                              ;
                                                55  DD 002A6         PUSHL   R5                              ;
                           00000000G    00            06  FB 002A8         CALLS   #6, DBG$BUILD_PRIMARY_SUBNODE   ;
                                          24    AE  9F 002AF         PUSHAB  VAL_DESC                        ; 1844
                                    7E          7A    8F  9A 002B2         MOVZBL  #122, -(SP)                     ;
                                                55  DD 002B6         PUSHL   R5                              ;
                                 FB84       CF            03  FB 002B8         CALLS   #3, DBG$PRIM_TO_VAL             ;
                                    50          24    AE  D0 002BD         MOVL    VAL_DESC, R0                    ; 1845
                                    50          20    A0  D0 002C1         MOVL    32(R0), TAG_VAL                 ;
                                              0081    8F  BB 002C5         PUSHR   #^M<R0,R7>                      ; 1846
                           00000000G    00            02  FB 002C9         CALLS   #2, DBG$STA_VARIANT_SELECT      ;
                                    5A                  50  D0 002D0         MOVL    R0, VARIANT                     ;
                                    04    AE      18    B5  0F 002D3         REMQUE  @24(R5), DUMMY                  ; 1847
                                                5A  D5 002D8 31$:     TSTL    VARIANT                         ; 1850
                                                1F  12 002DA         BNEQ    32$                             ;
                                       00000000'    EF  9F 002DC         PUSHAB  P.AAG                           ; 1853
                                                34  DD 002E2         PUSHL   #52                             ;
                                       00000000'    EF  9F 002E4         PUSHAB  P.AAF                           ;
                           00000000G    00            03  FB 002EA         CALLS   #3, DBG$PRINT                   ;
                           00000000G    00            00  FB 002F1         CALLS   #0, DBG$NEWLINE                 ; 1854
```

```
                                00FF  31  002F8          BRW     37$                                             ; 1850
                                   7E  7C  002FB 32$:    CLRQ    -(SP)                                           ; 1858
                                   13  DD  002FD          PUSHL   #19
                             7E    0B  7D  002FF          MOVQ    #11, -(SP)
                             55    04  AC  D0  00302      MOVL    PRM_DESC, R5
                                   55  DD  00306          PUSHL   R5
                00000000G    00      06  FB  00308      CALLS   #6, DBG$BUILD_PRIMARY_SUBNODE
                             52    18  A5  D0  0030F      MOVL    24(R5), SUBNODE                                 ; 1859
                          1C 52        56  D0  00313      MOVL    TAGID, 28(SUBNODE)                              ; 1860
                          18 A2        01  B0  00317      MOVW    #1, 24(SUBNODE)                                 ; 1861
                          0A A2        10  88  0031B      BISB2   #16, 10(SUBNODE)                                ; 1862
                          1A A2  04    AA  B0  0031F      MOVW    4(VARIANT), 26(SUBNODE)                         ; 1863
                          20 A2  08    AA  9E  00324      MOVAB   8(R10), 32(SUBNODE)                             ; 1864
                          24 A2        6A  D0  00329      MOVL    (VARIANT), 36(SUBNODE)                          ; 1865
                       00000000'       EF  9F  0032D      PUSHAB  P.AAI                                           ; 1866
                                   1E  DD  00333          PUSHL   #30
                       00000000'       EF  9F  00335      PUSHAB  P.AAH
                00000000G    00      03  FB  0033B      CALLS   #3, DBG$PRINT
                          00000000G    0C  DD  00342      PUSHL   DBG$GL_SIGN_FLAG                                ; 1867
                                08     AC  DD  00348      PUSHL   RADIX
                                2C     AE  DD  0034B      PUSHL   VAL_DESC
                          0000V CF         03  FB  0034E      CALLS   #3, DBG$PRINT_VALUE
                                   3D  11  00353          BRB     34$                                             ; 1868
                                18     AE  9F  00355 33$:    PUSHAB  TYPEID                                          ; 1874
                                20     AE  9F  00358          PUSHAB  FCODE
                                   57  DD  0035B          PUSHL   SYMID
                00000000G    00      03  FB  0035D      CALLS   #3, DBG$STA_SYMTYPE
                                   7E  D4  00364          CLRL    -(SP)                                           ; 1875
                                1C     AE  DD  00366          PUSHL   TYPEID
                                24     AE  DD  00369          PUSHL   FCODE
                                   57  DD  0036C          PUSHL   SYMID
                                20     AE  DD  0036E          PUSHL   KIND
                                   55  DD  00371          PUSHL   R5
                00000000G    00      06  FB  00373      CALLS   #6, DBG$BUILD_PRIMARY_SUBNODE
                                   55  DD  0037A          PUSHL   R5                                              ; 1876
                00000000G    00      01  FB  0037C      CALLS   #1, DBG$COLLECT
                             17    04  A5  E9  00383      BLBC    4(R5), 35$                                      ; 1877
                                   7E  D4  00387          CLRL    -(SP)                                           ; 1880
                                   55  DD  00389          PUSHL   R5
                00000000G    00      02  FB  0038B      CALLS   #2, DBG$PRINT_IDENTIFIER
                                08     AC  DD  00392 34$:    PUSHL   RADIX                                           ; 1881
                                   55  DD  00395          PUSHL   R5
                             FC64  CF      02  FB  00397      CALLS   #2, DBG$PRINT_AGGREGATE
                                   57  11  0039C          BRB     36$                                             ; 1877
                                20     AE  9F  0039E 35$:    PUSHAB  NAME                                            ; 1886
                                   57  DD  003A1          PUSHL   SYMID
                00000000G    00      02  FB  003A3      CALLS   #2, DBG$STA_SYMNAME
                                20     BE  95  003AA          TSTB    @NAME                                           ; 1887
                                   46  13  003AD          BEQL    36$
                                   7E  D4  003AF          CLRL    -(SP)                                           ; 1889
                                   55  DD  003B1          PUSHL   R5
                00000000G    00      02  FB  003B3      CALLS   #2, DBG$PRINT_IDENTIFIER
                       00000000'       EF  9F  003BA      PUSHAB  P.AAK                                           ; 1890
                                   01  DD  003C0          PUSHL   #1
                       00000000'       EF  9F  003C2      PUSHAB  P.AAJ
                00000000G    00      03  FB  003C8      CALLS   #3, DBG$PRINT
                                24     AE  9F  003CF          PUSHAB  VAL_DESC                                        ; 1891
```

```
                    7E       7A   8F   9A 003D2          MOVZBL  #122, -(SP)                                    ;
                                  55   DD 003D6          PUSHL   R5                                              ;
                FA64     CF       03   FB 003D8          CALLS   #3, DBG$PRIM_TO_VAL                             ;
                         00000000G 00  DD 003DD          PUSHL   DBG$GL_SIGN_FLAG                                ; 1892
                              08   AC   DD 003E3          PUSHL   RADIX                                           ;
                              2C   AE   DD 003E6          PUSHL   VAL_DESC                                        ;
                0000V    CF       03   FB 003E9          CALLS   #3, DBG$PRINT_VALUE                             ;
            00000000G    00       00   FB 003EE          CALLS   #0, DBG$NEWLINE                                 ; 1893
                   04    AE       18   B5   0F 003F5 36$:  REMQUE  @24(R5), DUMMY                               ; 1896
                                  6E   DD 003FA 37$:  PUSHL   MARK_ONE                                        ; 1898
            00000000G    00       01   FB 003FC          CALLS   #1, DBG$POP_TEMPMEM                             ;
        02          54            53   F2 00403 38$:  AOBLSS  R3, COMPONENT, 39$                              ; 1815
                                  10   11 00407          BRB     41$                                             ; 1682
                         FE27     31   00409 39$:  BRW     28$                                             ; 1815
                         000287D8 8F   DD 0040C 40$:  PUSHL   #165848                                         ; 1903
            00000000G    00       01   FB 00412          CALLS   #1, LIB$SIGNAL                                  ;
                   0A    A2       01   8A 00419 41$:  BICB2   #1, 10(SUBNODE)                                ; 1907
                         50   04   AC   D0 0041D          MOVL    PRM_DESC, R0                                    ; 1908
                   04    A0       01   88 00421          BISB2   #1, 4(R0)                                       ;
                   07    A0   08   A2   90 00425          MOVB    8(SUBNODE), 7(R0)                               ; 1909
                   06    A0   09   A2   90 0042A          MOVB    9(SUBNODE), 6(R0)                               ; 1910
                   08    A0   0C   A2   D0 0042F          MOVL    12(SUBNODE), 8(R0)                              ; 1911
                         7E       04   CE 00434 42$:  MNEGL   #4, -(SP)                                       ; 1912
                                  02   DD 00437          PUSHL   #2                                              ;
            00000000G    00       02   FB 00439          CALLS   #2, DBG$PRINT_CONTROL                           ;
                                       04   00440          RET                                                     ; 1913

; Routine Size:  1089 bytes,    Routine Base:  DBG$CODE + 0CD5
```

```
;  1803          1914  1 GLOBAL ROUTINE DBG$PRINT_VALUE(val_desc: REF dbg$valdesc,radix) : NOVALUE =
;  1804          1915  2     BEGIN
;  1805          1916  2     BUILTIN ACTUALCOUNT,ACTUALPARAMETER;
;  1806          1917  2
;  1807          1918  2     LOCAL
;  1808          1919  2         sign_flag,
;  1809          1920  2         save_flag,
;  1810          1921  2         vms_desc        : dbg$stg_desc;
;  1811          1922  2
;  1812          1923  2     sign_flag = (actualcount() GTR 2 AND actualparameter(3));
;  1813          1924  2     save_flag = (actualcount() LSS 4 OR  actualparameter(4));
;  1814          1925  2
;  1815          1926  2     IF .save_flag THEN dbg$save_val(.val_desc);
;  1816          1927  2     ch$move(T2,val_desc[dbg$a_value_vmsdesc],vms_desc);
;  1817          1928  2
;  1818          1929  2     IF .val_desc[dbg$v_dhdr_format] NEQ 0 THEN
;  1819          1930  3         BEGIN
;  1820          1931  3         SELECTONE .val_desc[dbg$v_dhdr_format] OF
;  1821          1932  3             SET
;  1822          1933  4             [1]:BEGIN           ! Condition Value
;  1823          1934  4                 LOCAL
;  1824          1935  4                     msgbuffer   : VECTOR [256,BYTE],
;  1825          1936  4                     msg_desc    : dbg$stg_desc;
;  1826          1937  4
;  1827          1938  4                 msg_desc[dsc$b_class]   = dsc$k_class_s;
;  1828          1939  4                 msg_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1829          1940  4                 msg_desc[dsc$w_length]  = 256;
;  1830          1941  4                 msg_desc[dsc$a_pointer] = msgbuffer;
;  1831        P 1942  4                 $GETMSG(MSGID  = .val_desc[dbg$l_value_value0],
;  1832        P 1943  4                         MSGLEN =  msg_desc[dsc$w_length],
;  1833          1944  4                         BUFADR =  msg_desc);
;  1834          1945  4                 dbg$print(UPLIT BYTE(%ASCIC '!AS'),msg_desc);
;  1835          1946  3                 END;
;  1836          1947  3
;  1837          1948  3             [2,3]:
;  1838          1949  4                 BEGIN
;  1839          1950  4                 BIND format_tab = UPLIT BYTE(%ASCIC '! '),
;  1840          1951  4                      header_one = UPLIT BYTE(%ASCIC 'CMP TP FPD IS CURMOD PRVMOD IPL'),
;  1841          1952  4                      header_two = UPLIT BYTE(%ASCIC ' DV FU IV T N Z V C'),
;  1842          1953  4                      mode_names = UPLIT ('KRNL','EXEC','SUPR','USER') : VECTOR[4,LONG];
;  1843          1954  4
;  1844          1955  4
;  1845          1956  4                 dbg$newline();
;  1846          1957  4                 dbg$print(format_tab);
;  1847          1958  4                 IF NOT .val_desc[dbg$v_dhdr_format] THEN dbg$print(header_one);
;  1848          1959  4                 dbg$print(header_two);
;  1849          1960  4                 dbg$newline();
;  1850          1961  4                 dbg$print(format_tab);
;  1851          1962  4                 IF NOT .val_desc[dbg$v_dhdr_format]
;  1852          1963  4                   THEN dbg$print(UPLIT BYTE(%ASCIC '!2UL!4UL!3UL!4UL  !AD   !AD!5UL'),
;  1853          1964  4                                  .(val_desc[dbg$l_value_value0])<31,1,0>,
;  1854          1965  4                                  .(val_desc[dbg$l_value_value0])<30,1,0>,
;  1855          1966  4                                  .(val_desc[dbg$l_value_value0])<27,1,0>,
;  1856          1967  4                                  .(val_desc[dbg$l_value_value0])<26,1,0>,
;  1857          1968  4                                 4,mode_names[.(val_desc[dbg$l_value_value0])<24,2,0>],
;  1858          1969  4                                 4,mode_names[.(val_desc[dbg$l_value_value0])<22,2,0>],
;  1859          1970  4                                 .(val_desc[dbg$l_value_value0])<16,5,0>);
```

```
;  1860          1971  4                dbg$print(UPLIT BYTE(%ASCIC '!3(3UL)!5(2UL)'),
;  1861          1972  4                                  .(val_desc[dbg$l_value_value0])<7,1,0>,
;  1862          1973  4                                  .(val_desc[dbg$l_value_value0])<6,1,0>,
;  1863          1974  4                                  .(val_desc[dbg$l_value_value0])<5,1,0>,
;  1864          1975  4                                  .(val_desc[dbg$l_value_value0])<4,1,0>,
;  1865          1976  4                                  .(val_desc[dbg$l_value_value0])<3,1,0>,
;  1866          1977  4                                  .(val_desc[dbg$l_value_value0])<2,1,0>,
;  1867          1978  4                                  .(val_desc[dbg$l_value_value0])<1,1,0>,
;  1868          1979  4                                  .(val_desc[dbg$l_value_value0])<0,1,0>);
;  1869          1980  3                END;
;  1870          1981  3
;  1871          1982  3            [OTHERWISE]:        SIGNAL(dbg$_interr,1,UPLIT BYTE(
;  1872          1983  3                                %ASCIC 'DBGVALUES\DBG$PRINT_VALUE - unknown format code'));
;  1873          1984  3                TES;
;  1874          1985  3           RETURN;
;  1875          1986  2           END;
```

```
;  1877     1987  2      !+
;  1878     1988  2      ! Radix will come in as something other than default if a radix was
;  1879     1989  2      ! explicitly specified in the command as in EX/HEX or if a radix
;  1880     1990  2      ! override was specified as in SET RADIX/OVERRIDE.
;  1881     1991  2      !-
;  1882     1992  2      IF .radix NEQ dbg$k_default
;  1883     1993  2      THEN
;  1884     1994  2          dbg$print_value_as_integer(vms_desc,.radix)
;  1885     1995  2      ELSE
;  1886     1996  2          BEGIN
;  1887     1997  3          !+
;  1888     1998  3          ! Unless this is a "DEBUG" descriptor (created because a type override
;  1889     1999  3          ! switch has been given),  we first see if we can find any
;  1890     2000  3          ! language-specific formatting rules.
;  1891     2001  3          !-
;  1892     2002  4          IF (.val_desc[dbg$b_dhdr_fcode] NEQ rst$k_type_descr)
;  1893     2003  4           OR (.val_desc[dbg$b_value_class] NEQ dsc$k_class_z)
;  1894     2004  4            THEN IF dbg$language_format(.val_desc) THEN RETURN;
;  1895     2005  3          !+
;  1896     2006  3          ! We get here if there are no language-specific formatting
;  1897     2007  3          ! rules applicable to this data item,  either because this
;  1898     2008  3          ! is a "DEBUG"-built value descriptor or because there are
;  1899     2009  3          ! no applicable language-specific format exception entries.
;  1900     2010  3          !-
;  1901     2011  3          SELECTONE .val_desc[dbg$b_dhdr_fcode]
;  1902     2012  3           OF SET
;  1903     2013  3              [rst$k_type_enum]:
;  1904     2014  4                  BEGIN
;  1905     2015  4                  LOCAL
;  1906     2016  4                      size,n_elems,elem_vect : REF VECTOR[,LONG];
;  1907     2017  4
;  1908     2018  4                  dbg$sta_typ_enum(.val_desc[dbg$l_dhdr_typeid],n_elems,elem_vect,size);
;  1909     2019  4                  INCR e FROM 0 TO .n_elems-1 DO
;  1910     2020  5                      BEGIN
;  1911     2021  5                      LOCAL adr_kind,adr_ptrs : VECTOR[3,LONG];
;  1912     2022  5                      dbg$sta_symvalue(.elem_vect[.e],adr_ptrs,adr_kind);
;  1913     2023  5                      IF .adr_kind NEQ dbg$k_val_literal THEN SIGNAL(dbg$_unimplent);
;  1914     2024  5                      IF .(.adr_ptrs[0])<.adr_ptrs[1],.size,0> EQL .val_desc[dbg$l_value_value0] THEN
;  1915     2025  6                          BEGIN
;  1916     2026  6                          dbg$print_symbol_name(.elem_vect[.e]);
;  1917     2027  6                          RETURN;
;  1918     2028  5                          END;
;  1919     2029  4                      END;
;  1920     2030  4                  !+
;  1921     2031  4                  ! Warn value out of range for enumeration type.
;  1922     2032  4                  !-
;  1923     2033  4                  SIGNAL(dbg$_enumrange);
;  1924     2034  4                  dbg$print_value_as_integer(vms_desc);
;  1925     2035  3                  END;
;  1926     2036  3
;  1927     2037  3              [rst$k_type_blifld]:
;  1928     2038  4                  BEGIN
;  1929     2039  4                  LOCAL
;  1930     2040  4                      count,
;  1931     2041  4                      fields: REF VECTOR[,LONG];
;  1932     2042  4
;  1933     2043  4                  fields = .val_desc[dbg$l_value_pointer];
```

```
;  1934          2044  4                 count = .fields[0];
;  1935          2045  4                 dbg$print(UPLIT BYTE(%ASCIC '['));
;  1936          2046  4                 INCR e from 1 to .count-1 DO
;  1937          2047  5                     BEGIN
;  1938          2048  5                     dbg$print(UPLIT BYTE(%ASCIC '!UL'), .fields[.e]);
;  1939          2049  5                     dbg$print(UPLIT BYTE(%ASCIC ', '));
;  1940          2050  4                     END;
;  1941          2051  4
;  1942          2052  4                 dbg$print(UPLIT BYTE(%ASCIC '!UL'), .fields[.count]);
;  1943          2053  4                 dbg$print(UPLIT BYTE(%ASCIC ']'));
;  1944          2054  4                 END;
;  1945          2055
;  1946          2056              [rst$k_type_set]:
;  1947          2057                  dbg$print_set_value(.val_desc);
;  1948          2058
;  1949          2059              [rst$k_type_file]:
;  1950          2060                  !++
;  1951          2061                  ! Just print the information that this is a file pointer.
;  1952          2062                  !--
;  1953          2063  3               dbg$print(UPLIT BYTE(%ASCIC 'file variable'));
```

```
;  1955          2064  3          [OTHERWISE]:
;  1956          2065  4              BEGIN
;  1957          2066  4              !+
;  1958          2067  4              ! Here if there are no special formatting rules for this FCODE.
;  1959          2068  4              ! Just look at the DTYPE in the descriptor, and print the value
;  1960          2069  4              ! in a type-dependent format.  However, we first have to ensure
;  1961          2070  4              ! that we have a valid CLASS field for LIB$CVT_DX_DX.
;  1962          2071  4              !-
;  1963          2072  4              IF .vms_desc[dsc$b_class] EQL dsc$k_class_z THEN vms_desc[dsc$b_class] = dsc$k_class_s;
;  1964          2073  4              CASE .vms_desc[dsc$b_dtype] FROM dbg$k_minimum_dtype TO dbg$k_maximum_dtype OF
;  1965          2074  4                  SET
;  1966          2075  4                  !+
;  1967          2076  4                  ! The first few case entries are for the various types of
;  1968          2077  4                  ! ASCII text (ASCIC, ASCIZ, ASCIW).
;  1969          2078  4                  ! These all get translated to ASCII (DTYPE dsc$k_dtype_t).
;  1970          2079  4                  !-
;  1971          2080  4
;  1972          2081  4                  [dsc$k_dtype_vt]:
;  1973          2082  5                      BEGIN
;  1974          2083  5                      vms_desc[dsc$w_length]  = .(.vms_desc[dsc$a_pointer])<0,16,0>;
;  1975          2084  5                      vms_desc[dsc$a_pointer] = 2+.vms_desc[dsc$a_pointer];
;  1976          2085  5                      vms_desc[dsc$b_class]   = dsc$k_class_s;
;  1977          2086  5                      vms_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1978          2087  5                      dbg$print_vms_value(vms_desc);
;  1979          2088  4                      END;
;  1980          2089  4
;  1981          2090  4                  [dsc$k_dtype_ac]:
;  1982          2091  5                      BEGIN
;  1983          2092  5                      vms_desc[dsc$w_length]  = .(.vms_desc[dsc$a_pointer])<0,8,0>;
;  1984          2093  5                      vms_desc[dsc$a_pointer] = 1+.vms_desc[dsc$a_pointer];
;  1985          2094  5                      vms_desc[dsc$b_class]   = dsc$k_class_s;
;  1986          2095  5                      vms_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  1987          2096  5                      dbg$print_vms_value(vms_desc);
;  1988          2097  4                      END;
;  1989          2098  4
;  1990          2099  4                  [dsc$k_dtype_az]:
;  1991          2100  5                      BEGIN
;  1992          2101  5                      BUILTIN LOCC;
;  1993          2102  5                      LOCAL length;
;  1994          2103  5                      LOCC(%REF(0),vms_desc[dsc$w_length],.vms_desc[dsc$a_pointer];length);
;  1995          2104  5                      vms_desc[dsc$w_length] = .vms_desc[dsc$w_length] - .length;
;  1996          2105  5                      vms_desc[dsc$b_class]  = dsc$k_class_s;
;  1997          2106  5                      vms_desc[dsc$b_dtype]  = dsc$k_dtype_t;
;  1998          2107  5                      dbg$print_vms_value(vms_desc);
;  1999          2108  4                      END;
```

```
;  2001          2109  4                    [dsc$k_dtype_b,dsc$k_dtype_bu,dsc$k_dtype_w,dsc$k_dtype_wu,
;  2002          2110  4                     dsc$k_dtype_l,dsc$k_dtype_lu,dsc$k_dtype_q,dsc$k_dtype_qu,
;  2003          2111  4                     dsc$k_dtype_o,dsc$k_dtype_ou]:
;  2004          2112  4                        IF .dbg$gb_radix[dbg$b_radix_output] NEQ dbg$k_decimal
;  2005          2113  4                        THEN
;  2006          2114  4                            dbg$print_value_as_integer(vms_desc)
;  2007          2115  4                        ELSE
;  2008          2116  4                            dbg$print_vms_value(vms_desc, .sign_flag);
;  2009          2117  4
;  2010          2118  4                    [dsc$k_dtype_f ,dsc$k_dtype_d  ,dsc$k_dtype_g ,dsc$k_dtype_h  ,
;  2011          2119  4                     dsc$k_dtype_fc,dsc$k_dtype_dc ,dsc$k_dtype_gc,dsc$k_dtype_hc ,
;  2012          2120  4                     dsc$k_dtype_nl,dsc$k_dtype_nlo,dsc$k_dtype_nr,dsc$k_dtype_nro,
;  2013          2121  4                     dsc$k_dtype_nu,dsc$k_dtype_nz ,dsc$k_dtype_p,dsc$k_dtype_t]:
;  2014          2122  4                        dbg$print_vms_value(vms_desc, .sign_flag);
;  2015          2123  4
;  2016          2124  4                    [dsc$k_dtype_zi]:
;  2017          2125  4                        dbg$ins_decode(.vms_desc[dsc$a_pointer],true,false);
;  2018          2126  4
;  2019          2127  4                    [dsc$k_dtype_zem]:
;  2020          2128  4                        dbg$ins_decode(.vms_desc[dsc$a_pointer],true,true);
;  2021          2129  4
;  2022          2130  4                    [dsc$k_dtype_tf]:
;  2023          2131  6                        dbg$print(Format_AC,(IF .(.vms_desc[dsc$a_pointer])
;  2024          2132  5                                             THEN UPLIT BYTE(%ASCIC 'True')
;  2025          2133  4                                             ELSE UPLIT BYTE(%ASCIC 'False')));
;  2026          2134  4
;  2027          2135  4                    [dsc$k_dtype_adt]:                                          ! A002
;  2028          2136  4                        dbg$print_vms_value(vms_desc);                          ! A002
;  2029          2137  4
;  2030          2138  4 !                  [dsc$k_dtype_dsc]:
;  2031          2139  4
;  2032          2140  4                    [INRANGE,OUTRANGE]:
;  2033          2141  4                        dbg$print_value_as_integer(vms_desc);
;  2034          2142  4                    TES;            ! CASE .vms_desc[dsc$b_dtype]
;  2035          2143  3                END;
;  2036          2144  3            TES;                    ! SELECTONE .val_desc[dbg$b_dhdr_fcode]
;  2037          2145  2        END;
;  2038          2146  1    END;                            ! End of 'dbg$print_value'


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

                                                53  41  21  03  00081 P.AAL:  .ASCII  <3>\!AS\                                                    ;
                                                    5F  21  02  00085 P.AAM:  .ASCII  <2>\!_\                                                     ;
20  53  49  20  44  50  46  20  50  54  20  50  4D  43  1F  00088 P.AAN:  .ASCII  <31>\CMP TP FPD IS CURMOD PRVMOD IPL\                                 ;
49  20  44  4F  4D  56  52  50  20  44  4F  4D  52  55  43  00097                                                                                   ;
                                                        4C  50  000A6                                                                             ;
20  4E  20  54  20  56  49  20  55  46  20  56  44  20  13  000A8 P.AAO:  .ASCII  <19>\ DV FU IV T N Z V C\                                          ;
                                            43  20  56  20  5A  000B7                                                                             ;
                                                4C  4E  52  4B  000BC P.AAP:  .ASCII  \KRNL\                                                      ;
                                                43  45  58  45  000C0        .ASCII  \EXEC\                                                       ;
                                                52  50  55  53  000C4        .ASCII  \SUPR\                                                       ;
                                                52  45  53  55  000C8        .ASCII  \USER\                                                       ;
34  21  4C  55  33  21  4C  55  34  21  4C  55  32  21  1F  000CC P.AAQ:  .ASCII  <31>\!2UL!4UL!3UL!4UL  !AD   !AD!5UL\                             ;
35  21  44  41  21  20  20  20  44  41  21  20  20  4C  55  000DB                                                                                   ;
                                                    4C  55  000EA                                                                                 ;
```

```
29 4C 55 32 28 35 21 29 4C 55 33 28 33 21 0E 000EC P.AAR:  .ASCII  <14>\!3(3UL)!5(2UL)\                    ;
24 47 42 44 5C 53 45 55 4C 41 56 47 42 44 2F 000FB P.AAS:  .ASCII  \/DBGVALUES\<92>\DBG$PRINT_VALUE - unkno\  ;
75 20 2D 20 45 55 4C 41 56 5F 54 4E 49 52 50 0010A                                                          ;
                          6F 6E 6B 6E 00119                                                                 ;
   65 64 6F 63 20 74 61 6D 72 6F 66 20 6E 77 0011D          .ASCII  \wn format code\                         ;
                                       5B 01 0012B P.AAT:   .ASCII  <1>\[\                                  ;
                                 4C 55 21 03 0012D P.AAU:   .ASCII  <3>\!UL\                                ;
                                       20 2C 02 00131 P.AAV:   .ASCII  <2>\, \                              ;
                                 4C 55 21 03 00134 P.AAW:   .ASCII  <3>\!UL\                                ;
                                       5D 01 00138 P.AAX:   .ASCII  <1>\]\                                  ;
   65 6C 62 61 69 72 61 76 20 65 6C 69 66 0D 0013A P.AAY:   .ASCII  <13>\file variable\                       ;
                              65 75 72 54 04 00148 P.AAZ:   .ASCII  <4>\True\                               ;
                           65 73 6C 61 46 05 0014D P.ABA:   .ASCII  <5>\False\                              ;

                                                            FORMAT_TAB=         P.AAM                       ;
                                                            HEADER_ONE=         P.AAN                       ;
                                                            HEADER_TWO=         P.AAO                       ;
                                                            MODE_NAMES=         P.AAP                       ;
                                                                    .EXTRN  SYS$GETMSG                      ;

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

                                                  0FFC 00000        .ENTRY  DBG$PRINT_VALUE, Save R2,R3,R4,R5,R6,R7,R8,-; 1914
                                                                            R9,R10,R11                      ;
                                                  5B 00000000G   00  9E 00002        MOVAB   DBG$NEWLINE, R11                        ;
                                                  5A 00000000G   00  9E 00009        MOVAB   LIB$SIGNAL, R10                         ;
                                                  59 00000000G   00  9E 00010        MOVAB   DBG$PRINT, R9                           ;
                                                  58 00000000'   EF  9E 00017        MOVAB   FORMAT_TAB, R8                          ;
                                                  5E      FED8   CE  9E 0001E        MOVAB   -296(SP), SP                            ;
                                                                 50  D4 00023        CLRL    R0                                      ; 1923
                                                               02    6C  91 00025        CMPB    (AP), #2                                ;
                                                                     02  1B 00028        BLEQU   1$                                      ;
                                                                 50  D6 0002A        INCL    R0                                      ;
                                                               57        0C    AC  D2 0002C 1$:    MCOML   12(AP), SIGN_FLAG               ;
                                                     57          50        57  CB 00030        BICL3   SIGN_FLAG, R0, SIGN_FLAG        ;
                                                                 50  D4 00034        CLRL    R0                                      ; 1924
                                                               04    6C  91 00036        CMPB    (AP), #4                                ;
                                                                     02  1E 00039        BGEQU   2$                                      ;
                                                                 50  D6 0003B        INCL    R0                                      ;
                                                               50        10    AC  C8 0003D 2$:    BISL2   16(AP), SAVE_FLAG               ;
                                                               0A        50  E9 00041        BLBC    SAVE_FLAG, 3$                   ; 1926
                                                                     04    AC  DD 00044        PUSHL   VAL_DESC                        ;
                                                00000000G      00          01  FB 00047        CALLS   #1, DBG$SAVE_VAL                ;
                                                               56        04    AC  D0 0004E 3$:    MOVL    VAL_DESC, R6            ; 1927
                                           F4  AD      14      A6          0C  28 00052        MOVC3   #12, 20(R6), VMS_DESC           ;
                                        52 05  A6              04          04  EF 00058        EXTZV   #4, #4, 5(R6), R2               ; 1929
                                                                     03  12 0005E        BNEQ    4$                                      ;
                                                                         00DA 31 00060        BRW     11$                             ;
                                                               01          52  D1 00063 4$:    CMPL    R2, #1                  ; 1933
                                                                     29  12 00066        BNEQ    5$                                      ;
                                                     10    AE 010E0100    8F  D0 00068        MOVL    #17694976, MSG_DESC             ; 1940
                                                     14    AE      1C    AE  9E 00070        MOVAB   MSGBUFFER, MSG_DESC+4           ; 1941
                                                     7E            0F  7D 00075        MOVQ    #15, -(SP)                      ; 1944
                                                                 18    AE  9F 00078        PUSHAB  MSG_DESC                        ;
                                                                 1C    AE  9F 0007B        PUSHAB  MSG_DESC                        ;
                                                                 20    A6  DD 0007E        PUSHL   32(R6)                          ;
                                                00000000G      00          05  FB 00081        CALLS   #5, SYS$GETMSG                  ;
```

```
                                                10   AE  9F 00088         PUSHAB  MSG_DESC                                ; 1945
                                                FC   A8  9F 0008B         PUSHAB  P.AAL                                   ;
                                                   026D  31 0008E         BRW     40$                                     ;
                                          02           52  D1 00091 5$:    CMPL    R2, #2                                  ; 1948
                                                       03  18 00094         BGEQ    7$                                      ;
                                                   0095  31 00096 6$:      BRW     10$                                     ;
                                          03           52  D1 00099 7$:    CMPL    R2, #3                                  ;
                                                       F8  14 0009C         BGTR    6$                                      ;
                                          6B           00  FB 0009E         CALLS   #0, DBG$NEWLINE                         ; 1956
                                                       58  DD 000A1         PUSHL   R8                                      ; 1957
                                          69           01  FB 000A3         CALLS   #1, DBG$PRINT                           ;
                                          06           52  E8 000A6         BLBS    R2, 8$                                  ; 1958
                                                03     A8  9F 000A9         PUSHAB  HEADER_ONE                              ;
                                          69           01  FB 000AC         CALLS   #1, DBG$PRINT                           ;
                                                23     A8  9F 000AF 8$:      PUSHAB  HEADER_TWO                              ; 1959
                                          69           01  FB 000B2         CALLS   #1, DBG$PRINT                           ;
                                          6B           00  FB 000B5         CALLS   #0, DBG$NEWLINE                         ; 1960
                                                       58  DD 000B8         PUSHL   R8                                      ; 1961
                                          69           01  FB 000BA         CALLS   #1, DBG$PRINT                           ;
                                          3B           52  E8 000BD         BLBS    R2, 9$                                  ; 1962
                                          51       20  A6  9E 000C0         MOVAB   32(R6), R1                              ; 1970
                7E        02    A1        05           00  EF 000C4         EXTZV   #0, #5, 2(R1), -(SP)                    ;
                50              61        02           16  EF 000CA         EXTZV   #22, #2, (R1), R0                       ; 1969
                                                37 A840  DF 000CF         PUSHAL  MODE_NAMES[R0]                          ;
                                                       04  DD 000D3         PUSHL   #4                                      ;
                50        03    A1        02           00  EF 000D5         EXTZV   #0, #2, 3(R1), R0                       ; 1968
                                                37 A840  DF 000DB         PUSHAL  MODE_NAMES[R0]                          ;
                                                       04  DD 000DF         PUSHL   #4                                      ; 1969
                7E              61        01           1A  EF 000E1         EXTZV   #26, #1, (R1), -(SP)                    ;
                7E              61        01           1B  EF 000E6         EXTZV   #27, #1, (R1), -(SP)                    ;
                7E              61        01           1E  EF 000EB         EXTZV   #30, #1, (R1), -(SP)                    ;
                7E              61        01           1F  EF 000F0         EXTZV   #31, #1, (R1), -(SP)                    ;
                                                47     A8  9F 000F5         PUSHAB  P.AAQ                                   ; 1963
                                          69           0A  FB 000F8         CALLS   #10, DBG$PRINT                          ; 1969
                                          50       20  A6  9E 000FB 9$:      MOVAB   32(R6), R0                              ; 1979
                7E              60        01           00  EF 000FF         EXTZV   #0, #1, (R0), -(SP)                     ;
                7E              60        01           01  EF 00104         EXTZV   #1, #1, (R0), -(SP)                     ; 1978
                7E              60        01           02  EF 00109         EXTZV   #2, #1, (R0), -(SP)                     ; 1977
                7E              60        01           03  EF 0010E         EXTZV   #3, #1, (R0), -(SP)                     ; 1976
                7E              60        01           04  EF 00113         EXTZV   #4, #1, (R0), -(SP)                     ; 1975
                7E              60        01           05  EF 00118         EXTZV   #5, #1, (R0), -(SP)                     ; 1974
                7E              60        01           06  EF 0011D         EXTZV   #6, #1, (R0), -(SP)                     ; 1973
                7E              60        01           07  EF 00122         EXTZV   #7, #1, (R0), -(SP)                     ; 1972
                                                67     A8  9F 00127         PUSHAB  P.AAR                                   ; 1971
                                          69           09  FB 0012A         CALLS   #9, DBG$PRINT                           ;
                                                       04  0012D         RET                                             ; 1931
                                                76     A8  9F 0012E 10$:     PUSHAB  P.AAS                                   ; 1982
                                                       01  DD 00131         PUSHL   #1                                      ;
                                             00028362  8F  DD 00133         PUSHL   #164706                                 ;
                                          6A           03  FB 00139         CALLS   #3, LIB$SIGNAL                          ;
                                                       04  0013C         RET                                             ; 1930
                                          01       08  AC  D1 0013D 11$:     CMPL    RADIX, #1                               ; 1992
                                                   0C  13 00141         BEQL    12$                                     ;
                                                08     AC  DD 00143         PUSHL   RADIX                                   ; 1994
                                                F4     AD  9F 00146         PUSHAB  VMS_DESC                                ;
                        0000V   CF              02  FB 00149         CALLS   #2, DBG$PRINT_VALUE_AS_INTEGER          ;
                                                   04  0014E         RET                                             ;
```

```
                                03        06    A6  91 0014F 12$:    CMPB    6(R6), #3                               ; 2002
                                          05    12  00153           BNEQ    13$                                     ;
                                   17     A6    95  00155           TSTB    23(R6)                                  ; 2003
                                          0D    13  00158           BEQL    14$                                     ;
                                          56    DD  0015A 13$:      PUSHL   R6                                      ; 2004
                00000000G         00            01  FB  0015C       CALLS   #1, DBG$LANGUAGE_FORMAT                 ;
                                  01            50  E9  00163       BLBC    R0, 14$                                 ;
                                                04  00166           RET                                             ;
                                  50        06  A6  9A  00167 14$:   MOVZBL  6(R6), R0                              ; 2011
                                  04            50  91  0016B       CMPB    R0, #4                                  ; 2013
                                                61  12  0016E       BNEQ    18$                                     ;
                                                5E  DD  00170       PUSHL   SP                                      ; 2018
                                          08    AE  9F  00172       PUSHAB  ELEM_VECT                               ;
                                          10    AE  9F  00175       PUSHAB  N_ELEMS                                 ;
                                          08    A6  DD  00178       PUSHL   8(R6)                                   ;
                00000000G         00            04  FB  0017B       CALLS   #4, DBG$STA_TYP_ENUM                    ;
                                  52            01  CE  00182       MNEGL   #1, E                                   ; 2024
                                                39  11  00185       BRB     17$                                     ;
                                          0C    AE  9F  00187 15$:   PUSHAB  ADR_KIND                                ; 2022
                                          E8    AD  9F  0018A       PUSHAB  ADR_PTRS                                ;
                                          0C BE42   DD  0018D       PUSHL   @ELEM_VECT[E]                           ;
                00000000G         00            03  FB  00191       CALLS   #3, DBG$STA_SYMVALUE                    ;
                                  01        0C    AE  D1  00198       CMPL    ADR_KIND, #1                            ; 2023
                                                09  13  0019C       BEQL    16$                                     ;
                                     00028800   8F  DD  0019E       PUSHL   #165888                                 ;
                                  6A            01  FB  001A4       CALLS   #1, LIB$SIGNAL                          ;
        50      E8  BD                    6E    EC  AD  EF  001A7 16$:   EXTZV   ADR_PTRS+4, SIZE, @ADR_PTRS, R0     ; 2024
                                     20     A6        50  D1  001AE       CMPL    R0, 32(R6)                              ;
                                                      0C  12  001B2       BNEQ    17$                                     ;
                                                   04 BE42   DD  001B4       PUSHL   @ELEM_VECT[E]                           ; 2026
                    00000000G     00                01  FB  001B8       CALLS   #1, DBG$PRINT_SYMBOL_NAME               ;
                                                      04  001BF           RET                                             ; 2025
                C2                52           08    AE  F2  001C0 17$:   AOBLSS  N_ELEMS, E, 15$                        ; 2019
                                        000286CB    8F  DD  001C5       PUSHL   #T65579                                 ; 2033
                                  6A                01  FB  001CB       CALLS   #1, LIB$SIGNAL                          ;
                                                 00F0   31  001CE       BRW     32$                                     ; 2034
                                  0E                50  91  001D1 18$:   CMPB    R0, #14                                 ; 2037
                                                    37  12  001D4       BNEQ    21$                                     ;
                                  54           18    A6  D0  001D6       MOVL    24(R6), FIELDS                          ; 2043
                                  53                64  D0  001DA       MOVL    (FIELDS), COUNT                         ; 2044
                                            00A6    C8  9F  001DD       PUSHAB  P.AAT                                   ; 2045
                                  69                01  FB  001E1       CALLS   #1, DBG$PRINT                           ;
                                                    52  D4  001E4       CLRL    E                                       ; 2046
                                                    11  11  001E6       BRB     20$                                     ;
                                                  6442  DD  001E8 19$:   PUSHL   (FIELDS)[E]                           ; 2048
                                            00A8    C8  9F  001EB       PUSHAB  P.AAU                                   ;
                                  69                02  FB  001EF       CALLS   #2, DBG$PRINT                           ;
                                            00AC    C8  9F  001F2       PUSHAB  P.AAV                                   ; 2049
                                  69                01  FB  001F6       CALLS   #1, DBG$PRINT                           ;
                EB                52                53  F2  001F9 20$:   AOBLSS  COUNT, E, 19$                        ; 2046
                                                  6443  DD  001FD       PUSHL   (FIELDS)[COUNT]                         ; 2052
                                            00AF    C8  9F  00200       PUSHAB  P.AAW                                   ;
                                  69                02  FB  00204       CALLS   #2, DBG$PRINT                           ;
                                            00B3    C8  9F  00207       PUSHAB  P.AAX                                   ; 2053
                                                    18  11  0020B       BRB     23$                                     ;
                                  08                50  91  0020D 21$:   CMPB    R0, #8                                  ; 2056
                                                    0A  12  00210       BNEQ    22$                                     ;
```

```
                                   56  DD 00212        PUSHL   R6                                      ; 2057
            00000000G  00          01  FB 00214        CALLS   #1, DBG$PRINT_SET_VALUE
                                       04 0021B        RET
                       0F          50  91 0021C 22$:   CMPB    R0, #15                                 ; 2059
                                   08  12 0021F        BNEQ    24$
                              00B5 C8  9F 00221        PUSHAB  P.AAY                                   ; 2063
                       69          01  FB 00225 23$:   CALLS   #1, DBG$PRINT
                                       04 00228        RET
                               F7  AD  95 00229 24$:   TSTB    VMS_DESC+3                              ; 2072
                                   04  12 0022C        BNEQ    25$
                   F7  AD          01  90 0022E        MOVB    #1, VMS_DESC+3
             2B        00      F6  AD  8F 00232 25$:   CASEB   VMS_DESC+2, #0, #43                     ; 2073
 0081        0081      008A        008A     00237 26$: .WORD   32$=26$,-
 0081        0081      0081        0081     0023F              32$-26$,-
 0093        0093      0081        0081     00247              31$-26$,-
 0093        0093      0093        0093     0024F              31$-26$,-
 0093        0093      0093        0093     00257              31$-26$,-
 00A2        009E      0093        0093     0025F              31$-26$,-
 0093        0081      0081        008A     00267              31$-26$,-
 008A        0093      0093        0093     0026F              31$-26$,-
 00CB        008A      008A        008A     00277              31$-26$,-
 006F        0065      005A        008A     0027F              31$-26$,-
 008A        008A      008A        00B1     00287              33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               34$-26$,-
                                                               35$-26$,-
                                                               32$-26$,-
                                                               31$-26$,-
                                                               31$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               33$-26$,-
                                                               32$-26$,-
                                                               32$-26$,-
                                                               32$-26$,-
                                                               32$-26$,-
                                                               41$-26$,-
                                                               32$-26$,-
                                                               27$-26$,-
                                                               28$-26$,-
                                                               29$-26$,-
                                                               37$-26$,-
                                                               32$-26$,-
                                                               32$-26$,-
                                                               32$-26$
                                   30  11 0028F        BRB     32$                                     ; 2141
```

```
                F4   AD        F8   BD   B0 00291 27$:    MOVW    @VMS_DESC+4, VMS_DESC                   ;2083
                F8   AD             02   C0 00296         ADDL2   #2, VMS_DESC+4                          ;2084
                                    14   11 0029A         BRB     30$                                     ;2085
                F4   AD        F8   BD   9B 0029C 28$:    MOVZBW  @VMS_DESC+4, VMS_DESC                   ;2092
                               F8   AD   D6 002A1         INCL    VMS_DESC+4                              ;2093
                                    0A   11 002A4         BRB     30$                                     ;2094
F8  BD          F4   AD             00   3A 002A6 29$:    LOCC    #0, VMS_DESC, @VMS_DESC+4               ;2103
                F4   AD             50   A2 002AC         SUBW2   LENGTH, VMS_DESC                        ;2104
                F6   AD      010E   8F   B0 002B0 30$:    MOVW    #270, VMS_DESC+2                        ;2106
                                    4A   11 002B6         BRB     41$                                     ;2107
                     0A 00000000G   00   91 002B8 31$:    CMPB    DBG$GB_RADIX+1, #10                     ;2112
                                    09   13 002BF         BEQL    33$                                     ;
                               F4   AD   9F 002C1 32$:    PUSHAB  VMS_DESC                                ;2114
                0000V  CF           01   FB 002C4         CALLS   #1, DBG$PRINT_VALUE_AS_INTEGER          ;
                                    04   002C9            RET                                             ;
                                    57   DD 002CA 33$:    PUSHL   SIGN_FLAG                               ;2122
                               F4   AD   9F 002CC         PUSHAB  VMS_DESC                                ;
                0000V  CF           02   FB 002CF         CALLS   #2, DBG$PRINT_VMS_VALUE                 ;
                                    04   002D4            RET                                             ;
                                    7E   D4 002D5 34$:    CLRL    -(SP)                                   ;2125
                                    02   11 002D7         BRB     36$                                     ;
                                    01   DD 002D9 35$:    PUSHL   #1                                      ;2128
                                    01   DD 002DB 36$:    PUSHL   #1                                      ;
                               F8   AD   DD 002DD         PUSHL   VMS_DESC+4                              ;
          00000000G  00             03   FB 002E0         CALLS   #3, DBG$INS_DECODE                      ;
                                    04   002E7            RET                                             ;
                     07        F8   BD   E9 002E8 37$:    BLBC    @VMS_DESC+4, 38$                        ;2131
                     50   00C3   C8   9E 002EC         MOVAB   P.AAZ, R0                               ;2132
                                    05   11 002F1         BRB     39$                                     ;
                     50   00C8   C8   9E 002F3 38$:    MOVAB   P.ABA, R0                               ;2133
                                    50   DD 002F8 39$:    PUSHL   R0                                      ;
                          FF7B   C8   9F 002FA         PUSHAB  FORMAT_AC                               ;2131
                     69             02   FB 002FE 40$:    CALLS   #2, DBG$PRINT                           ;
                                    04   00301            RET                                             ;
                               F4   AD   9F 00302 41$:    PUSHAB  VMS_DESC                                ;2136
                0000V  CF           01   FB 00305         CALLS   #1, DBG$PRINT_VMS_VALUE                 ;
                                    04   0030A            RET                                             ;2146

; Routine Size:  779 bytes,    Routine Base:  DBG$CODE + 1116
```

```
;  2040          2147  1 GLOBAL ROUTINE DBG$PRINT_VALUE_AS_INTEGER(vms_desc: REF dbg$stg_desc) : NOVALUE =
;  2041          2148  2     BEGIN
;  2042          2149  2     BUILTIN ACTUALCOUNT,ACTUALPARAMETER,MOVC5;
;  2043          2150  2     LOCAL
;  2044          2151  2         radix,
;  2045          2152  2         radix_override_flag,    ! TRUE if radix override was applied
;  2046          2153  2         data_bytes,
;  2047          2154  2         byte_size,
;  2048          2155  2         data_size,
;  2049          2156  2         data_addr,
;  2050          2157  2         data_buff       : VECTOR [512+4,BYTE],
;  2051          2158  2         digit_count,
;  2052          2159  2         digit_value     : BYTE,
;  2053          2160  2         text_index,
;  2054          2161  2         text_buff       : VECTOR [512*9,BYTE];
;  2055          2162  2
;  2056          2163  2     BIND digit = UPLIT BYTE('0123456789ABCDEF') : VECTOR [16,BYTE];
;  2057          2164  2
;  2058          2165  2     radix_override_flag = FALSE;
;  2059          2166  2     IF actualcount() GTR 1
;  2060          2167  2     THEN
;  2061          2168  3         BEGIN
;  2062          2169  3         radix = actualparameter(2);
;  2063          2170  3         IF .radix EQL dbg$k_default
;  2064          2171  3         THEN
;  2065          2172  3             radix = dbg$nget_radix()
;  2066          2173  3         ELSE
;  2067          2174  3             radix_override_flag = TRUE;
;  2068          2175  3         END
;  2069          2176  2     ELSE
;  2070          2177  2         radix = dbg$nget_radix();
;  2071          2178  2
;  2072          2179  2     text_index = 512*9;
;  2073          2180  2
;  2074          2181  2     data_addr = .vms_desc[dsc$a_pointer];
;  2075          2182  2     IF (data_size = dbg$data_length(.vms_desc)) GTR 512*%BPUNIT THEN
;  2076          2183  3         BEGIN
;  2077          2184  3         ! ****   SIGNAL(truncation)     ****************************
;  2078          2185  3         data_size = 512*%BPUNIT;
;  2079          2186  2         END;
;  2080          2187  2     data_bytes = (.data_size + (%BPUNIT-1))/%BPUNIT;
;  2081          2188  2     MOVC5(data_bytes,.data_addr,%REF(0),%REF(512+4),data_buff);
;  2082          2189  2     IF (.data_size AND (%BPUNIT-1)) NEQ 0
;  2083          2190  2       THEN data_buff[.data_bytes-1] =
;  2084          2191  2           .data_buff[.data_bytes-1] AND NOT (-1^(.data_size AND (%BPUNIT-1)));
;  2085          2192  2
;  2086          2193  2     SELECTONE .radix OF
;  2087          2194  2         SET
;  2088          2195  2         [dbg$k_decimal]:
;  2089          2196  3             BEGIN
;  2090          2197  3             SELECTONE .vms_desc[dsc$b_dtype] OF
;  2091          2198  3                 SET
;  2092          2199  3                 [dsc$k_dtype_bu,dsc$k_dtype_wu,
;  2093          2200  3                  dsc$k_dtype_lu,dsc$k_dtype_qu,
;  2094          2201  3                  dsc$k_dtype_ou,dsc$k_dtype_z ,
;  2095          2202  3                  dsc$k_dtype_v,dsc$k_dtype_vu]:
;  2096          2203  3                     IF .radix_override_flag
```

```
;  2097          2204  3                THEN
;  2098          2205  3                  IF .(data_buff)<.data_size-1,1,0> THEN
;  2099          2206  4                    BEGIN
;  2100          2207  4                    dbg$print(Format_AD,1,UPLIT BYTE('-'));
;  2101          2208  4                    INCR m FROM 0 TO .data_bytes-1 DO
;  2102          2209  4                      IF .data_buff[.m] NEQ 0 THEN
;  2103          2210  5                        BEGIN
;  2104          2211  5                        data_buff[.m] = -.data_buff[.m];
;  2105          2212  5                        INCR n FROM .m+1 TO .data_bytes-1 DO
;  2106          2213  5                          data_buff[.n] = NOT .data_buff[.n];
;  2107          2214  5                        EXITLOOP;
;  2108          2215  4                        END;
;  2109          2216  4                    IF (.data_size AND (%BPUNIT-1)) NEQ 0
;  2110          2217  4                      THEN data_buff[.data_bytes-1] =
;  2111          2218  4                        .data_buff[.data_bytes-1] AND NOT (-1^(.data_size AND (%BPUNIT-1)));
;  2112          2219  3                    END;
;  2113          2220  3
;  2114          2221  3              [OTHERWISE]:
;  2115          2222  3                  IF .(data_buff)<.data_size-1,1,0> THEN
;  2116          2223  4                    BEGIN
;  2117          2224  4                    dbg$print(Format_AD,1,UPLIT BYTE('-'));
;  2118          2225  4                    INCR m FROM 0 TO .data_bytes-1 DO
;  2119          2226  4                      IF .data_buff[.m] NEQ 0 THEN
;  2120          2227  5                        BEGIN
;  2121          2228  5                        data_buff[.m] = -.data_buff[.m];
;  2122          2229  5                        INCR n FROM .m+1 TO .data_bytes-1 DO
;  2123          2230  5                          data_buff[.n] = NOT .data_buff[.n];
;  2124          2231  5                        EXITLOOP;
;  2125          2232  4                        END;
;  2126          2233  4                    IF (.data_size AND (%BPUNIT-1)) NEQ 0
;  2127          2234  4                      THEN data_buff[.data_bytes-1] =
;  2128          2235  4                        .data_buff[.data_bytes-1] AND NOT (-1^(.data_size AND (%BPUNIT-1)));
;  2129          2236  3                    END;
;  2130          2237  3                TES;
;  2131          2238  3
;  2132          2239  3              WHILE (data_size = .data_bytes) GTR 0 DO
;  2133          2240  4                  BEGIN
;  2134          2241  4                  LOCAL digit_val;
;  2135          2242  4                  data_bytes = 0;
;  2136          2243  4                  digit_val  = 0;
;  2137          2244  4                  DECR d FROM .data_size-1 TO 0 DO
;  2138          2245  5                      BEGIN
;  2139          2246  5                      digit_val = (.digit_val^8)+.data_buff[.d];
;  2140          2247  5                      IF (data_buff[.d] = .digit_val/10) NEQ 0
;  2141          2248  5                        THEN IF .data_bytes EQL 0 THEN data_bytes = .d+1;
;  2142          2249  5                      digit_val = .digit_val - 10*(.digit_val/10);
;  2143          2250  4                      END;
;  2144          2251  4                  text_buff[(text_index=.text_index-1)] = .digit_val<0,8,0>+'0';
;  2145          2252  3                  END;
;  2146          2253  3              dbg$print(Format_AD,512*9-.text_index,text_buff[.text_index]);
;  2147          2254  3              RETURN;
;  2148          2255  2              END;
;  2149          2256  2
;  2150          2257  2          [dbg$k_binary]: byte_size = 1;
;  2151          2258  2          [dbg$k_octal]:  byte_size = 3;
;  2152          2259  2          [dbg$k_hex]:    byte_size = 4;
;  2153          2260  2 !        [OTHERWISE]:
```

```
;  2154          2261  2          TES;
;  2155          2262  2
;  2156          2263  2      digit_count = (.data_size + (.byte_size-1))/.byte_size;
;  2157          2264  2
;  2158          2265  2      INCR index FROM 0 TO .digit_count-1 DO
;  2159          2266  3          BEGIN
;  2160          2267  4          IF (.byte_size NEQ 3) AND ((.index AND 7) EQL 0) AND (.index NEQ 0)
;  2161          2268  3            THEN text_buff[(text_index = .text_index-1)] = ' ';
;  2162          2269  3          digit_value = .digit[.(data_buff)<.index*.byte_size,.byte_size,0>];
;  2163          2270  3          text_buff[(text_index = .text_index-1)] = .digit_value;
;  2164          2271  2          END;
;  2165          2272  2
;  2166          2273  2      IF .digit_value GTRU '9' THEN text_buff[(text_index = .text_index-1)] = '0';
;  2167          2274  2
;  2168          2275  2      dbg$print(Format_AD,512*9-.text_index,text_buff[.text_index]);
;  2169          2276  1      END;                            ! End of routine 'dbg$print_value_as_integer'


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

45  44  43  42  41  39  38  37  36  35  34  33  32  31  30  00153 P.ABB:  .ASCII  \0123456789ABCDEF\                          ;
                                                        46  00162                                                           ;
                                                        2D  00163 P.ABC:  .ASCII  \-\                                       ;
                                                        2D  00164 P.ABD:  .ASCII  \-\                                       ;

                                                                  DIGIT=            P.ABB


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

                                                                OFFC 00000         .ENTRY  DBG$PRINT_VALUE_AS_INTEGER, Save R2,R3,R4,- ; 2147
                                                                                           R5,R6,R7,R8,R9,R10,R11
                                                            5E      EBFC    CE  9E 00002       MOVAB   -5124(SP), SP
                                                                            5B  D4 00007       CLRL    RADIX_OVERRIDE_FLAG                      ; 2165
                                                            01              6C  91 00009       CMPB    (AP), #1                                 ; 2166
                                                                            0E  1B 0000C       BLEQU   1$
                                                            5A      08      AC  D0 0000E       MOVL    8(AP), RADIX                             ; 2169
                                                            01              5A  D1 00012       CMPL    RADIX, #1                                ; 2170
                                                                            05  13 00015       BEQL    1$
                                                            5B              01  D0 00017       MOVL    #1, RADIX_OVERRIDE_FLAG                  ; 2174
                                                                            0A  11 0001A       BRB     2$                                       ; 2166
                                                    00000000G   00          00  FB 0001C 1$:   CALLS   #0, DBG$NGET_RADIX                       ; 2177
                                                            5A              50  D0 00023       MOVL    R0, RADIX
                                                            57      1200    8F  3C 00026 2$:   MOVZWL  #4608, TEXT_INDEX                        ; 2179
                                                            5B        04    AC  D0 0002B       MOVL    VMS_DESC, R8                             ; 2181
                                                            52        04    A8  D0 0002F       MOVL    4(R8), DATA_ADDR
                                                                            58  DD 00033       PUSHL   R8                                       ; 2182
                                                        EBA5    CF          01  FB 00035       CALLS   #1, DBG$DATA_LENGTH
                                                            59              50  D0 0003A       MOVL    R0, DATA_SIZE
                                                    00001000    8F          59  D1 0003D       CMPL    DATA_SIZE, #4096
                                                                            05  15 00044       BLEQ    3$
                                                            59      1000    8F  3C 00046       MOVZWL  #4096, DATA_SIZE                         ; 2185
                                                            50        07    A9  9E 0004B 3$:   MOVAB   7(R9), R0                                ; 2187
                                                56          50              08  C7 0004F       DIVL3   #8, R0, DATA_BYTES
0204    8F              00  62                              56  2C 00053       MOVC5   DATA_BYTES, (DATA_ADDR), #0, #516, -             ; 2188
                                                                FDFC    CD      0005A                  DATA_BUFF
```

```
                                                52  D4 0005D          CLRL    R2                                          ; 2189
                                          07    59  93 0005F          BITB    DATA_SIZE, #7
                                                15  13 00062          BEQL    4$
                                                52  D6 00064          INCL    R2
          50                    59          03  00  EF 00066          EXTZV   #0, #3, DATA_SIZE, R0                       ; 2191
                                50 FFFFFFFF   8F  50  78 0006B          ASHL    R0, #-1, R0
                                      FDFB CD46   50  8A 00073          BICB2   R0, DATA_BUFF-1[DATA_BYTES]
                                          0A      5A  D1 00079 4$:      CMPL    RADIX, #T0                                  ; 2195
                                                03  13 0007C          BEQL    5$
                                              0109  31 0007E          BRW     24$
                                          50      02  A8  9A 00081 5$:      MOVZBL  2(R8), R0                           ; 2197
                                          05          50  91 00085          CMPB    R0, #5                              ; 2199
                                                0A    1B 00088          BLEQU   6$
                                          19          50  91 0008A          CMPB    R0, #25
                                                05    13 0008D          BEQL    6$
                                          22          50  91 0008F          CMPB    R0, #34
                                                52    12 00092          BNEQ    13$
                                          03          5B  E8 00094 6$:      BLBS    RADIX_OVERRIDE_FLAG, 8$             ; 2203
                                              00AC  31 00097 7$:      BRW     19$
                                          50      FF  A9  9E 0009A 8$:      MOVAB   -1(R9), R0                          ; 2205
                                F3      FDFC    CD  50  E1 0009E          BBC     R0, DATA_BUFF, 7$
                                      00000000'     EF  9F 000A4          PUSHAB  P.ABC                               ; 2207
                                                01  DD 000AA          PUSHL   #1
                                      00000000'     EF  9F 000AC          PUSHAB  FORMAT_AD
                             00000000G  00          03  FB 000B2          CALLS   #3, DBG$PRINT
                                          51        01  CE 000B9          MNEGL   #1, M                               ; 2208
                                                22  11 000BC          BRB     12$
                                          50    FDFC CD41   9A 000BE 9$:      MOVZBL  DATA_BUFF[M], R0                    ; 2209
                                                1A  13 000C4          BEQL    12$
                                      FDFC CD41       50  8E 000C6          MNEGB   R0, DATA_BUFF[M]                    ; 2211
                                          50          51  D0 000CC          MOVL    M, N                                ; 2212
                                                09  11 000CF          BRB     11$
                                      FDFC CD40   FDFC CD40   92 000D1 10$:      MCOMB   DATA_BUFF[N], DATA_BUFF[N]      ; 2213
                                F3        50          56  F2 000DA 11$:      AOBLSS  DATA_BYTES, N, 10$
                                                50  11 000DE          BRB     18$                                         ; 2210
                                DA        51          56  F2 000E0 12$:      AOBLSS  DATA_BYTES, M, 9$                   ; 2209
                                                4A  11 000E4          BRB     18$                                         ; 2216
                                          50      FF  A9  9E 000E6 13$:      MOVAB   -1(R9), R0                          ; 2222
                                56      FDFC    CD  50  E1 000EA          BBC     R0, DATA_BUFF, 19$
                                      00000000'     EF  9F 000F0          PUSHAB  P.ABD                               ; 2224
                                                01  DD 000F6          PUSHL   #1
                                      00000000'     EF  9F 000F8          PUSHAB  FORMAT_AD
                             00000000G  00          03  FB 000FE          CALLS   #3, DBG$PRINT
                                          51        01  CE 00105          MNEGL   #1, M                               ; 2225
                                                22  11 00108          BRB     17$
                                          50    FDFC CD41   9A 0010A 14$:      MOVZBL  DATA_BUFF[M], R0                    ; 2226
                                                1A  13 00110          BEQL    17$
                                      FDFC CD41       50  8E 00112          MNEGB   R0, DATA_BUFF[M]                    ; 2228
                                          50          51  D0 00118          MOVL    M, N                                ; 2229
                                                09  11 0011B          BRB     16$
                                      FDFC CD40   FDFC CD40   92 0011D 15$:      MCOMB   DATA_BUFF[N], DATA_BUFF[N]      ; 2230
                                F3        50          56  F2 00126 16$:      AOBLSS  DATA_BYTES, N, 15$
                                                04  11 0012A          BRB     18$                                         ; 2227
                                DA        51          56  F2 0012C 17$:      AOBLSS  DATA_BYTES, M, 14$                  ; 2226
                                          13          52  E9 00130 18$:      BLBC    R2, T9$                             ; 2233
          50                    59          03  00  EF 00133          EXTZV   #0, #3, DATA_SIZE, R0                       ; 2235
                                50 FFFFFFFF   8F  50  78 00138          ASHL    R0, #-1, R0
```

```
                FDFB CD46          50  8A 00140          BICB2   R0, DATA_BUFF-1[DATA_BYTES]                    ;
                      59           56  D0 00146 19$:      MOVL    DATA_BYTES, DATA_SIZE                          ; 2239
                                   03  14 00149          BGTR    20$                                            ;
                                 009C  31 0014B          BRW     31$                                            ;
                                   56  D4 0014E 20$:      CLRL    DATA_BYTES                                     ; 2242
                                   51  D4 00150          CLRL    DIGIT_VAL                                      ; 2243
                      50           59  D0 00152          MOVL    DATA_SIZE, D                                   ; 2244
                                   29  11 00155          BRB     23$                                            ;
                52    51           08  78 00157 21$:      ASHL    #8, DIGIT_VAL, R2                              ; 2246
                      51     FDFC CD40  9A 0015B          MOVZBL  DATA_BUFF[D], DIGIT_VAL                        ;
                      51           52  C0 00161          ADDL2   R2, DIGIT_VAL                                  ;
                52    51           0A  C7 00164          DIVL3   #10, DIGIT_VAL, R2                             ; 2247
                      FDFC CD40    52  90 00168          MOVB    R2, DATA_BUFF[D]                               ;
                                   52  D5 0016E          TSTL    R2                                             ;
                                   08  13 00170          BEQL    22$                                            ;
                                   56  D5 00172          TSTL    DATA_BYTES                                     ; 2248
                                   04  12 00174          BNEQ    22$                                            ;
                      56     01    A0  9E 00176          MOVAB   1(R0), DATA_BYTES                              ;
                      52           0A  C4 0017A 22$:      MULL2   #10, R2                                        ; 2249
                      51           52  C2 0017D          SUBL2   R2, DIGIT_VAL                                  ;
                      D4           50  F4 00180 23$:      SOBGEQ  D, 21$                                         ; 2244
                774E  51           30  81 00183          ADDB3   #48, DIGIT_VAL, TEXT_BUFF[SP]                  ; 2251
                                   BC  11 00188          BRB     19$                                            ; 2239
                      02           5A  D1 0018A 24$:      CMPL    RADIX, #2                                      ; 2257
                                   05  12 0018D          BNEQ    25$                                            ;
                      53           01  D0 0018F          MOVL    #1, BYTE_SIZE                                  ;
                                   12  11 00192          BRB     27$                                            ;
                      08           5A  D1 00194 25$:      CMPL    RADIX, #8                                      ; 2258
                                   05  12 00197          BNEQ    26$                                            ;
                      53           03  D0 00199          MOVL    #3, BYTE_SIZE                                  ;
                                   08  11 0019C          BRB     27$                                            ;
                      10           5A  D1 0019E 26$:      CMPL    RADIX, #16                                     ; 2259
                                   03  12 001A1          BNEQ    27$                                            ;
                      53           04  D0 001A3          MOVL    #4, BYTE_SIZE                                  ;
                      50     FF A349   9E 001A6 27$:      MOVAB   -1(BYTE_SIZE)[DATA_SIZE], R0                   ; 2263
                55    50           53  C7 001AB          DIVL3   BYTE_SIZE, R0, DIGIT_COUNT                     ;
                      50           01  CE 001AF          MNEGL   #1, INDEX                                      ; 2265
                                   29  11 001B2          BRB     30$                                            ;
                      03           53  D1 001B4 28$:      CMPL    BYTE_SIZE, #3                                  ; 2267
                                   0D  13 001B7          BEQL    29$                                            ;
                      07           50  93 001B9          BITB    INDEX, #7                                      ;
                                   08  12 001BC          BNEQ    29$                                            ;
                                   50  D5 001BE          TSTL    INDEX                                          ;
                                   04  13 001C0          BEQL    29$                                            ;
                774E               20  90 001C2          MOVB    #32, TEXT_BUFF[SP]                             ; 2268
                52    50           53  C5 001C6 29$:      MULL3   BYTE_SIZE, INDEX, R2                           ; 2269
51      FDFC    CD    53           52  EF 001CA          EXTZV   R2, BYTE_SIZE, DATA_BUFF, R1                   ;
                      54 00000000'EF41  90 001D1          MOVB    DIGIT[R1], DIGIT_VALUE                         ;
                774E  54           54  90 001D9          MOVB    DIGIT_VALUE, TEXT_BUFF[SP]                     ; 2270
                D3    50           55  F2 001DD 30$:      AOBLSS  DIGIT_COUNT, INDEX, 28$                        ; 2265
                      39           54  91 001E1          CMPB    DIGIT_VALUE, #57                               ; 2273
                                   04  1B 001E4          BLEQU   31$                                            ;
                774E               30  90 001E6          MOVB    #48, TEXT_BUFF[SP]                             ;
                             6E47  9F  9F 001EA 31$:      PUSHAB  TEXT_BUFF[TEXT_INDEX]                          ; 2275
                           EE00    C7  9F 001ED          PUSHAB  -4608(TEXT_INDEX)                              ;
                      6E           6E  CE 001F1          MNEGL   (SP), (SP)                                     ;
                        00000000'  EF  9F 001F4          PUSHAB  FORMAT_AD                                      ;
```

```
                    00000000G  00          03  FB 001FA          CALLS   #3, DBG$PRINT                    ;
                                                04 00201          RET                                      ; 2276

; Routine Size:  514 bytes,    Routine Base:  DBG$CODE + 1421
```

```
;  2171          2277  1 ROUTINE DBG$PRINT_VMS_VALUE(vms_desc: REF dbg$stg_desc) : NOVALUE =
;  2172          2278  2     BEGIN
;  2173          2279  2     BUILTIN ACTUALCOUNT,ACTUALPARAMETER;
;  2174          2280  2     BIND exp_zero = UPLIT BYTE('E+0000');       ! Hack to fix FOR$CVT bug
;  2175          2281  2     LOCAL
;  2176          2282  2         local_desc      : dbg$stg_desc,
;  2177          2283  2         buffer_desc     : dbg$stg_desc,
;  2178          2284  2         status,
;  2179          2285  2         text_buffer     : VECTOR [64,BYTE],
;  2180          2286  2         text_length     : WORD;
;  2181          2287  2
;  2182          2288  2     ch$move(12,,vms_desc,local_desc);
;  2183          2289  2     buffer_desc[dsc$b_class]   = dsc$k_class_s;
;  2184          2290  2     buffer_desc[dsc$b_dtype]   = dsc$k_dtype_t;
;  2185          2291  2     buffer_desc[dsc$w_length]  = 64;
;  2186          2292  2     buffer_desc[dsc$a_pointer] = text_buffer;
;  2187          2293  2
;  2188          2294  2     SELECTONE .vms_desc[dsc$b_dtype] OF
;  2189          2295  2         SET
;  2190          2296  2         [dsc$k_dtype_fc,dsc$k_dtype_dc,dsc$k_dtype_gc,dsc$k_dtype_hc]:
;  2191          2297  3             BEGIN
;  2192          2298  3             local_desc[dsc$w_length] = .local_desc[dsc$w_length]/2;
;  2193          2299  3             local_desc[dsc$b_dtype]  = .local_desc[dsc$b_dtype] -2;
;  2194          2300  3             local_desc[dsc$b_class]  =  dsc$k_class_s;
;  2195          2301  3             dbg$print(FORMAT_AD,1,UPLIT BYTE('('));
;  2196          2302  3             dbg$print_vms_value(local_desc);
;  2197          2303  3             dbg$print(FORMAT_AD,1,UPLIT BYTE(','));
;  2198          2304  3             local_desc[dsc$a_pointer] = .local_desc[dsc$a_pointer] + .local_desc[dsc$w_length];
;  2199          2305  3             dbg$print_vms_value(local_desc);
;  2200          2306  3             dbg$print(FORMAT_AD,1,UPLIT BYTE(')'));
;  2201          2307  3             END;
;  2202          2308  2
;  2203          2309  2         [dsc$k_dtype_f]:
;  2204          2310  3             BEGIN
;  2205          2311  3             BUILTIN CVTFD;
;  2206          2312  3             LOCAL dvalue : BLOCK[8,BYTE];
;  2207          2313  3             LOCAL digits,spaces,length;
;  2208          2314  3             BUILTIN SKPC,LOCC;
;  2209          2315  3
;  2210          2316  3             ! Use the FORTRAN "G" format routine, which only prints the
;  2211          2317  3             ! answer in exponential form if it has to.
;  2212          2318  3             ! Since there is not FOR$CVT_F_TG routine, convert the source
;  2213          2319  3             ! to d_float.
;  2214          2320  3             !
;  2215          2321  3             CVTFD(.local_desc[dsc$a_pointer], dvalue);
;  2216          2322  3             IF NOT for$cvt_d_tg( dvalue,
;  2217          2323  3                                  buffer_desc,
;  2218          2324  3                                  7,              ! Significant digits
;  2219          2325  3                                  0,              ! Scale factor
;  2220          2326  3                                  1,              ! Digits before decimal point
;  2221          2327  3                                                  ! in exponential form
;  2222          2328  3                                  2)              ! Digits after "E"
;  2223          2329  3                                                  ! in exponential form
;  2224          2330  3             THEN
;  2225          2331  3                 $DBG_ERROR('DBGVALUES\DBG$PRINT_VMS_VALUE');
;  2226          2332  3
;  2227          2333  3             ! The result is right-justified. Find the position where
```

```
;  2228          2334  3          ! the result begins (the first non-blank character in the
;  2229          2335  3          ! buffer) and print the result.
;  2230          2336  3          !
;  2231          2337  3          SKPC(%REF(' '),%REF(64),text_buffer;length,digits);
;  2232          2338  3          LOCC(%REF(' '),length,.digits;,spaces);
;  2233          2339  3          IF ch$eql(4,.spaces-4,4,exp_zero) THEN spaces = .spaces-4;
;  2234          2340  4          IF (actualcount() GTR 1 AND actualparameter(2))
;  2235          2341  4            AND (.(.digits)<0,8,0> NEQ '-')
;  2236          2342  3              THEN DBG$PRINT(Format_AD,1,UPLIT BYTE('+'));
;  2237          2343  3          dbg$print(Format_AD, .spaces - .digits, .digits);
;  2238          2344  2          END;
;  2239          2345  2
;  2240          2346  2      [dsc$k_dtype_d]:
;  2241          2347  2          BEGIN
;  2242          2348  2          LOCAL digits,spaces,length;
;  2243          2349  2          BUILTIN SKPC,LOCC;
;  2244          2350  2
;  2245          2351  2          ! Use the FORTRAN "G" format routine, which only prints the
;  2246          2352  2          ! answer in exponential form if it has to.
;  2247          2353  2          !
;  2248          2354  2          IF NOT for$cvt_d_tg(.local_desc[dsc$a_pointer],
;  2249          2355  2                              buffer_desc,
;  2250          2356  2                              16,             ! Significant digits
;  2251          2357  2                              0,              ! Scale factor
;  2252          2358  2                              1,              ! Digits before decimal point
;  2253          2359  2                                              ! in exponential form
;  2254          2360  2                              2)              ! Digits after "E"
;  2255          2361  2                                              ! in exponential form
;  2256          2362  2          THEN
;  2257          2363  2              $DBG_ERROR('DBGVALUES\DBG$PRINT_VMS_VALUE');
;  2258          2364  2
;  2259          2365  2          ! The result is right-justified. Find the position where
;  2260          2366  2          ! the result begins (the first non-blank character in the
;  2261          2367  2          ! buffer) and print the result.
;  2262          2368  2          !
;  2263          2369  2          SKPC(%REF(' '),%REF(64),text_buffer;length,digits);
;  2264          2370  2          LOCC(%REF(' '),length,.digits;,spaces);
;  2265          2371  2          IF ch$eql(4,.spaces-4,4,exp_zero) THEN spaces = .spaces-4;
;  2266          2372  3          IF (actualcount() GTR 1 AND actualparameter(2))
;  2267          2373  3            AND (.(.digits)<0,8,0> NEQ '-')
;  2268          2374  2              THEN DBG$PRINT(Format_AD,1,UPLIT BYTE('+'));
;  2269          2375  2          dbg$print(Format_AD, .spaces - .digits, .digits);
;  2270          2376  1          END;
;  2271          2377  1
;  2272          2378  1      [dsc$k_dtype_g]:
;  2273          2379  1          BEGIN
;  2274          2380  1          LOCAL digits,spaces,length;
;  2275          2381  1          BUILTIN SKPC,LOCC;
;  2276          2382  1
;  2277          2383  1          ! Use the FORTRAN "G" format routine, which only prints the
;  2278          2384  1          ! answer in exponential form if it has to.
;  2279          2385  1          !
;  2280          2386  1          IF NOT for$cvt_g_tg(.local_desc[dsc$a_pointer],
;  2281          2387  1                              buffer_desc,
;  2282          2388  1                              15,             ! Significant digits
;  2283          2389  1                              0,              ! Scale factor
;  2284          2390  1                              1,              ! Digits before decimal point
```

```
;  2285          2391  3                                    ! in exponential form
;  2286          2392  3                              3)    ! Digits after "E"
;  2287          2393  3                                    ! in exponential form
;  2288          2394  3                THEN
;  2289          2395  3                    $DBG_ERROR('DBGVALUES\DBG$PRINT_VMS_VALUE');
;  2290          2396  3
;  2291          2397  3                ! The result is right-justified. Find the position where
;  2292          2398  3                ! the result begins (the first non-blank character in the
;  2293          2399  3                ! buffer) and print the result.
;  2294          2400  3                !
;  2295          2401  3                SKPC(%REF(' '),%REF(64),text_buffer;length,digits);
;  2296          2402  3                LOCC(%REF(' '),length,,digits;,spaces);
;  2297          2403  3                IF ch$eql(5,.spaces-5,5,exp_zero) THEN spaces = .spaces-5;
;  2298          2404  4                IF (actualcount() GTR 1 AND actualparameter(2))
;  2299          2405  4                  AND (.(.digits)<0,8,0> NEQ '-')
;  2300          2406  3                    THEN DBG$PRINT(Format_AD,1,UPLIT BYTE('+'));
;  2301          2407  3                dbg$print(Format_AD, .spaces - .digits, .digits);
;  2302          2408  2                END;
;  2303          2409  2
;  2304          2410  2            [dsc$k_dtype_h]:
;  2305          2411  3                BEGIN
;  2306          2412  3                LOCAL digits,spaces,length;
;  2307          2413  3                BUILTIN SKPC,LOCC;
;  2308          2414  3
;  2309          2415  3                ! Use the FORTRAN "G" format routine, which only prints the
;  2310          2416  3                ! answer in exponential form if it has to.
;  2311          2417  3                !
;  2312          2418  3                IF NOT for$cvt_h_tg(.local_desc[dsc$a_pointer],
;  2313          2419  3                                    buffer_desc,
;  2314          2420  3                                    33,             ! Significant digits
;  2315          2421  3                                    0,              ! Scale factor
;  2316          2422  3                                    1,              ! Digits before decimal point
;  2317          2423  3                                                    ! in exponential form
;  2318          2424  3                                    4)              ! Digits after "E"
;  2319          2425  3                                                    ! in exponential form
;  2320          2426  3                THEN
;  2321          2427  3                    $DBG_ERROR('DBGVALUES\DBG$PRINT_VMS_VALUE');
;  2322          2428  3
;  2323          2429  3                ! The result is right-justified. Find the position where
;  2324          2430  3                ! the result begins (the first non-blank character in the
;  2325          2431  3                ! buffer) and print the result.
;  2326          2432  3                !
;  2327          2433  3                SKPC(%REF(' '),%REF(64),text_buffer;length,digits);
;  2328          2434  3                LOCC(%REF(' '),length,,digits;,spaces);
;  2329          2435  3                IF ch$eql(6,.spaces-6,6,exp_zero) THEN spaces = .spaces-6;
;  2330          2436  4                IF (actualcount() GTR 1 AND actualparameter(2))
;  2331          2437  4                  AND (.(.digits)<0,8,0> NEQ '-')
;  2332          2438  3                    THEN DBG$PRINT(Format_AD,1,UPLIT BYTE('+'));
;  2333          2439  3                dbg$print(Format_AD, .spaces - .digits, .digits);
;  2334          2440  2                END;
;  2335          2441  2
;  2336          2442  2            [dsc$k_dtype_t]:
;  2337          2443  3                BEGIN
;  2338          2444  3                BUILTIN
;  2339          2445  3                    PROBER;
;  2340          2446  3                LOCAL
;  2341          2447  3                    addr,
```

```
;  2342          2448  3                bytes;
;  2343          2449  3
;  2344          2450  3            local_desc[dsc$w_length] = MIN(.local_desc[dsc$w_length],2048);
;  2345          2451  3
;  2346          2452  3            ! Check for read access to the address. There are certain
;  2347          2453  3            ! cases where we can get here with the address in the descriptor
;  2348          2454  3            ! not readable. One example is EXAMINE/ASCID X, where X points
;  2349          2455  3            ! to the descriptor
;  2350          2456  3            !     0000FFFF
;  2351          2457  3            !     00000000
;  2352          2458  3            ! In this case the VMS descriptor came from a volatile value
;  2353          2459  3            ! descriptor, and the address "0" in the descriptor was never
;  2354          2460  3            ! checked for readability.
;  2355          2461  3            !
;  2356          2462  3            addr = .local_desc[dsc$a_pointer];
;  2357          2463  3            bytes = .local_desc[dsc$w_length];
;  2358          2464  3            IF NOT PROBER(%REF(0),bytes,.addr)
;  2359          2465  3            THEN
;  2360          2466  3                SIGNAL(dbg$_noaccessr,1,.addr);
;  2361          2467  3
;  2362          2468  3            dbg$print(UPLIT BYTE(%ASCIC '''!AF'''),.local_desc[dsc$w_length],
;  2363          2469  3                                                  .local_desc[dsc$a_pointer]);
;  2364          2470  3            END;
;  2365          2471  3
;  2366          2472  2        [OTHERWISE]:
;  2367          2473  2            BEGIN
;  2368          2474  2
;  2369          2475  2            !+
;  2370          2476  2            ! Somewhat of a hack for FORTRAN - the FORTRAN compiler gives
;  2371          2477  2            ! us types BU, WU, LU for LOGICAL variables even though they
;  2372          2478  2            ! are really treated as signed integers. Change the dtype here
;  2373          2479  2            ! so we print them right.
;  2374          2480  2            !-
;  2375          2481  2            IF .dbg$gb_language EQL dbg$k_fortran
;  2376          2482  2            THEN
;  2377          2483  2                IF .local_desc[dsc$b_dtype] EQL dsc$k_dtype_bu
;  2378          2484  2                THEN local_desc[dsc$b_dtype] = dsc$k_dtype_b
;  2379          2485  2                ELSE IF .local_desc[dsc$b_dtype] EQL dsc$k_dtype_wu
;  2380          2486  2                THEN local_desc[dsc$b_dtype] = dsc$k_dtype_w
;  2381          2487  2                ELSE IF .local_desc[dsc$b_dtype] EQL dsc$k_dtype_lu
;  2382          2488  2                THEN local_desc[dsc$b_dtype] = dsc$k_dtype_l;
;  2383          2489  2
;  2384          2490  2            dbg$cvt_dx_dx(local_desc,buffer_desc,text_length);
;  2385          2491  2
;  2386          2492  2            IF .signed_dtype[.local_desc[dsc$b_dtype]] AND
;  2387          2493  2             (actualcount() GTR 1 AND actualparameter(2)) AND
;  2388          2494  3             (.text_buffer[0] NEQ '-') AND (.text_buffer[0] NEQ '+') THEN
;  2389          2495  4                BEGIN
;  2390          2496  4                ch$move(.text_length,text_buffer[0],text_buffer[1]);
;  2391          2497  4                text_buffer[0] = '+';
;  2392          2498  4                text_length = .text_length + 1;
;  2393          2499  3                END;
;  2394          2500  2
;  2395          2501  2            dbg$print(Format_AD,.text_length,text_buffer);
;  2396          2502  2            END;
;  2397          2503  2        TES;
;  2398          2504  1    END;                            ! End of dbg$print_vms_value
```

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

                                                30  30  30  30  2B  45  00165 P.ABE:   .ASCII  \E+0000\                                    ;
                                                                    28  0016B P.ABF:   .ASCII  \(\                                       ;
                                                                    2C  0016C P.ABG:   .ASCII  \,\                                      ;
                                                                    29  0016D P.ABH:   .ASCII  \)\                                      ;
24  47  42  44  5C  53  45  55  4C  41  56  47  42  44             1D  0016E P.ABI:   .ASCII  <29>\DBGVALUES\<92>\DBG$PRINT_VMS_VALU\   ;
    55  4C  41  56  5F  53  4D  56  5F  54  4E  49  52             50  0017D                                                               ;
                                                                    45  0018B          .ASCII  \E\                                        ;
                                                                    2B  0018C P.ABJ:   .ASCII  \+\                                      ;
24  47  42  44  5C  53  45  55  4C  41  56  47  42  44             1D  0018D P.ABK:   .ASCII  <29>\DBGVALUES\<92>\DBG$PRINT_VMS_VALU\   ;
    55  4C  41  56  5F  53  4D  56  5F  54  4E  49  52             50  0019C                                                               ;
                                                                    45  001AA          .ASCII  \E\                                        ;
                                                                    2B  001AB P.ABL:   .ASCII  \+\                                      ;
24  47  42  44  5C  53  45  55  4C  41  56  47  42  44             1D  001AC P.ABM:   .ASCII  <29>\DBGVALUES\<92>\DBG$PRINT_VMS_VALU\   ;
    55  4C  41  56  5F  53  4D  56  5F  54  4E  49  52             50  001BB                                                               ;
                                                                    45  001C9          .ASCII  \E\                                        ;
                                                                    2B  001CA P.ABN:   .ASCII  \+\                                      ;
24  47  42  44  5C  53  45  55  4C  41  56  47  42  44             1D  001CB P.ABO:   .ASCII  <29>\DBGVALUES\<92>\DBG$PRINT_VMS_VALU\   ;
    55  4C  41  56  5F  53  4D  56  5F  54  4E  49  52             50  001DA                                                               ;
                                                                    45  001E8          .ASCII  \E\                                        ;
                                                                    2B  001E9 P.ABP:   .ASCII  \+\                                      ;
                                                22  46  41  21  22  05  001EA P.ABQ:   .ASCII  <5>\"!AF"\                            ;

                                                                    EXP_ZERO=           P.ABE

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

                                                                07FC 00000 DBG$PRINT_VMS_VALUE:
                                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                        ; 2277
                                            5A 00000000G    00  9E 00002        MOVAB   FOR$CVT_D_TG, R10                               ;
                                            59 00000000G    00  9E 00009        MOVAB   LIB$SIGNAL, R9                                  ;
                                            58 00000000G    00  9E 00010        MOVAB   DBG$PRINT, R8                                   ;
                                            57 00000000'    EF  9E 00017        MOVAB   FORMAT_AD, R7                                   ;
                                            5E       9C     AE  9E 0001E        MOVAB   -100(SP), SP                                    ;
                                            56       04     AC  D0 00022        MOVL    VMS_DESC, R6                                    ; 2288
58  AE                                      66               0C  28 00026        MOVC3   #12, (R6), LOCAL_DESC                           ;
                                        4C  AE  010E0040     8F  D0 0002B        MOVL    #17694784, BUFFER_DESC                          ; 2291
                                        50  AE       0C     AE  9E 00033        MOVAB   TEXT_BUFFER, BUFFER_DESC+4                      ; 2292
                                            50       02     A6  9A 00038        MOVZBL  2(R6), R0                                       ; 2294
                                            0C               50  91 0003C        CMPB    R0, #12                                         ; 2296
                                                             05  1F 0003F        BLSSU   1$                                              ;
                                            0D               50  91 00041        CMPB    R0, #13                                         ;
                                                             0A  1B 00044        BLEQU   2$                                              ;
                                            1D               50  91 00046 1$:    CMPB    R0, #29                                         ;
                                                             4E  1F 00049        BLSSU   3$                                              ;
                                            1E               50  91 0004B        CMPB    R0, #30                                         ;
                                                             49  1A 0004E        BGTRU   3$                                              ;
                                            51       58     AE  3C 00050 2$:    MOVZWL  LOCAL_DESC, R1                                  ; 2298
                                            51               02  C6 00054        DIVL2   #2, R1                                          ;
                                        58  AE               51  B0 00057        MOVW    R1, LOCAL_DESC                                  ;
                                        5A  AE               02  82 0005B        SUBB2   #2, LOCAL_DESC+2                                ; 2299
                                        5B  AE               01  90 0005F        MOVB    #1, LOCAL_DESC+3                                ; 2300
```

```
                        0167   C7  9F 00063          PUSHAB  P.ABF                    ; 2301
                               01  DD 00067          PUSHL   #1
                               57  DD 00069          PUSHL   R7
                    68         03  FB 0006B          CALLS   #3, DBG$PRINT
                          58   AE  9F 0006E          PUSHAB  LOCAL_DESC              ; 2302
                8B    AF       01  FB 00071          CALLS   #1, DBG$PRINT_VMS_VALUE
                        0168   C7  9F 00075          PUSHAB  P.ABG                   ; 2303
                               01  DD 00079          PUSHL   #1
                               57  DD 0007B          PUSHL   R7
                    68         03  FB 0007D          CALLS   #3, DBG$PRINT
                    50    58   AE  3C 00080          MOVZWL  LOCAL_DESC, R0          ; 2304
                5C  AE         50  C0 00084          ADDL2   R0, LOCAL_DESC+4
                          58   AE  9F 00088          PUSHAB  LOCAL_DESC              ; 2305
              FF70  CF         01  FB 0008B          CALLS   #1, DBG$PRINT_VMS_VALUE
                        0169   C7  9F 00090          PUSHAB  P.ABH                   ; 2306
                               01  DD 00094          PUSHL   #1
                             0236  31  00096          BRW     27$
                    0A         50  91 00099 3$:      CMPB    R0, #10                 ; 2309
                               57  12 0009C          BNEQ    6$
                04  AE    5C   BE  56 0009E          CVTFD   @LOCAL_DESC+4, DVALUE   ; 2321
                               02  DD 000A3          PUSHL   #2                      ; 2322
                               01  DD 000A5          PUSHL   #1
                    7E         07  7D 000A7          MOVQ    #7, -(SP)
                          5C   AE  9F 000AA          PUSHAB  BUFFER_DESC
                          18   AE  9F 000AD          PUSHAB  DVALUE
                    6A         06  FB 000B0          CALLS   #6, FOR$CVT_D_TG
                    0F         50  E8 000B3          BLBS    R0, 4$
                        016A   C7  9F 000B6          PUSHAB  P.ABI                   ; 2331
                               01  DD 000BA          PUSHL   #1
                     00028362  8F  DD 000BC          PUSHL   #164706
                    69         03  FB 000C2          CALLS   #3, LIB$SIGNAL
            0C  AE      0040   8F      20  3B 000C5 4$:      SKPC    #32, #64, TEXT_BUFFER  ; 2337
                               53      51  D0 000CC          MOVL    R1, R3
                    63         50      20  3A 000CF          LOCC    #32, LENGTH, (DIGITS)  ; 2338
                               52      51  D0 000D3          MOVL    R1, R2
                        0161   C7  FC  A2  D1 000D6          CMPL    -4(SPACES), EXP_ZERO   ; 2339
                               03      12  000DC          BNEQ    5$
                               52      04  C2 000DE          SUBL2   #4, SPACES
                               01      6C  91 000E1 5$:      CMPB    (AP), #1               ; 2340
                                       6B  1B 000E4          BLEQU   10$
                               67  08  AC  E9 000E6          BLBC    8(AP), 10$
                               2D      63  91 000EA          CMPB    (DIGITS), #45          ; 2341
                                       62  13 000ED          BEQL    10$
                        0188   C7  9F 000EF          PUSHAB  P.ABJ                   ; 2342
                                       55  11 000F3          BRB     9$
                               0B      50  91 000F5 6$:      CMPB    R0, #11                ; 2346
                                       60  12 000F8          BNEQ    11$
                               02  DD 000FA          PUSHL   #2                      ; 2354
                               01  DD 000FC          PUSHL   #1
                               7E      10  7D 000FE          MOVQ    #16, -(SP)
                                   5C  AE  9F 00101          PUSHAB  BUFFER_DESC
                                   70  AE  DD 00104          PUSHL   LOCAL_DESC+4
                               6A      06  FB 00107          CALLS   #6, FOR$CVT_D_TG
                               0F      50  E8 0010A          BLBS    R0, 7$
                        0189   C7  9F 0010D          PUSHAB  P.ABK                   ; 2363
                               01  DD 00111          PUSHL   #1
                     00028362  8F  DD 00113          PUSHL   #164706
```

```
                        69              03 FB 00119          CALLS   #3, LIB$SIGNAL
                0C AE       0040    8F  20 3B 0011C 7$:      SKPC    #32, #64, TEXT_BUFFER                   ;2369
                        53              51 D0 00123          MOVL    R1, R3
                    63          50      20 3A 00126          LOCC    #32, LENGTH, (DIGITS)                   ;2370
                        52              51 D0 0012A          MOVL    R1, R2
                        0161    C7  FC  A2 D1 0012D          CMPL    -4(SPACES), EXP_ZERO                    ;2371
                                        03 12 00133          BNEQ    8$
                            52          04 C2 00135          SUBL2   #4, SPACES
                            01          6C 91 00138 8$:      CMPB    (AP), #1                                ;2372
                                        14 1B 0013B          BLEQU   10$
                            10      08  AC E9 0013D          BLBC    8(AP), 10$
                            2D          63 91 00141          CMPB    (DIGITS), #45                           ;2373
                                        0B 13 00144          BEQL    10$
                                01A7    C7 9F 00146          PUSHAB  P.ABL                                   ;2374
                                        01 DD 0014A 9$:      PUSHL   #1
                                        57 DD 0014C          PUSHL   R7
                            68          03 FB 0014E          CALLS   #3, DBG$PRINT
                                        53 DD 00151 10$:     PUSHL   DIGITS                                  ;2375
                    7E          52      53 C3 00153          SUBL3   DIGITS, SPACES, -(SP)
                                    0175 31 00157            BRW     27$
                            1B          50 91 0015A 11$:     CMPB    R0, #27                                 ;2378
                                        57 12 0015D          BNEQ    14$
                                        03 DD 0015F          PUSHL   #3                                      ;2386
                                        01 DD 00161          PUSHL   #1
                            7E          0F 7D 00163          MOVQ    #15, -(SP)
                                5C      AE 9F 00166          PUSHAB  BUFFER_DESC
                                70      AE DD 00169          PUSHL   LOCAL_DESC+4
                00000000G       00      06 FB 0016C          CALLS   #6, FOR$CVT_G_TG
                                0F      50 E8 00173          BLBS    R0, 12$
                                01A8    C7 9F 00176          PUSHAB  P.ABM                                   ;2395
                                        01 DD 0017A          PUSHL   #1
                            00028362    8F DD 0017C          PUSHL   #164706
                        69              03 FB 00182          CALLS   #3, LIB$SIGNAL
                0C AE       0040    8F  20 3B 00185 12$:     SKPC    #32, #64, TEXT_BUFFER                   ;2401
                        55              51 D0 0018C          MOVL    R1, R5
                    65          50      20 3A 0018F          LOCC    #32, LENGTH, (DIGITS)                   ;2402
                        54              51 D0 00193          MOVL    R1, R4
            0161    C7      FB  A4      05 29 00196          CMPC3   #5, -5(SPACES), EXP_ZERO                ;2403
                                        03 12 0019D          BNEQ    13$
                            54          05 C2 0019F          SUBL2   #5, SPACES
                            01          6C 91 001A2 13$:     CMPB    (AP), #1                                ;2404
                                        70 1B 001A5          BLEQU   18$
                            6C      08  AC E9 001A7          BLBC    8(AP), 18$
                            2D          65 91 001AB          CMPB    (DIGITS), #45                           ;2405
                                        67 13 001AE          BEQL    18$
                                01C6    C7 9F 001B0          PUSHAB  P.ABN                                   ;2406
                                        5A 11 001B4          BRB     17$
                            1C          50 91 001B6 14$:     CMPB    R0, #28                                 ;2410
                                        65 12 001B9          BNEQ    19$
                                        04 DD 001BB          PUSHL   #4                                      ;2418
                                        01 DD 001BD          PUSHL   #1
                            7E          21 7D 001BF          MOVQ    #33, -(SP)
                                5C      AE 9F 001C2          PUSHAB  BUFFER_DESC
                                70      AE DD 001C5          PUSHL   LOCAL_DESC+4
                00000000G       00      06 FB 001C8          CALLS   #6, FOR$CVT_H_TG
                                0F      50 E8 001CF          BLBS    R0, 15$
                                01C7    C7 9F 001D2          PUSHAB  P.ABO                                   ;2427
```

```
                                    01 DD 001D6          PUSHL   #1                                          ;
                          00028362  8F DD 001D8          PUSHL   #164706                                     ;
                         69         03 FB 001DE          CALLS   #3, LIB$SIGNAL                              ;
                   0C AE     0040  8F 20 3B 001E1 15$:   SKPC    #32, #64, TEXT_BUFFER                       ; 2433
                         55        51 D0 001E8          MOVL    R1, R5                                      ;
                      65           50 20 3A 001EB       LOCC    #32, LENGTH, (DIGITS)                       ; 2434
                         54        51 D0 001EF          MOVL    R1, R4                                      ;
                 0161 C7      FA  A4 06 29 001F2        CMPC3   #6, -6(SPACES), EXP_ZERO                    ; 2435
                                    03 12 001F9          BNEQ    16$                                         ;
                         54        06 C2 001FB          SUBL2   #6, SPACES                                  ;
                         01        6C 91 001FE 16$:     CMPB    (AP), #1                                    ; 2436
                                   14 1B 00201          BLEQU   18$                                         ;
                         10     08 AC E9 00203          BLBC    8(AP), 18$                                  ;
                         2D        65 91 00207          CMPB    (DIGITS), #45                               ; 2437
                                   0B 13 0020A          BEQL    18$                                         ;
                              01E5 C7 9F 0020C          PUSHAB  P.ABP                                       ; 2438
                                   01 DD 00210 17$:     PUSHL   #1                                          ;
                                   57 DD 00212          PUSHL   R7                                          ;
                         68        03 FB 00214          CALLS   #3, DBG$PRINT                               ;
                                   55 DD 00217 18$:     PUSHL   DIGITS                                      ; 2439
                      7E           54 C3 00219          SUBL3   DIGITS, SPACES, -(SP)                       ;
                                00AF 31 0021D           BRW     27$                                         ;
                         0E        50 91 00220 19$:     CMPB    R0, #14                                     ; 2442
                                   3C 12 00223          BNEQ    22$                                         ;
                         50     58 AE 3C 00225          MOVZWL  LOCAL_DESC, R0                              ; 2450
                   0800  8F        50 B1 00229          CMPW    R0, #2048                                   ;
                                   05 1B 0022E          BLEQU   20$                                         ;
                         50  0800 8F 3C 00230          MOVZWL  #2048, R0                                   ;
                      58 AE        50 B0 00235 20$:     MOVW    R0, LOCAL_DESC                              ;
                         51     5C AE D0 00239          MOVL    LOCAL_DESC+4, ADDR                          ; 2462
                         50     58 AE 3C 0023D          MOVZWL  LOCAL_DESC, BYTES                           ; 2463
                      61           50 00 0C 00241       PROBER  #0, BYTES, (ADDR)                           ; 2464
                                   0D 12 00245          BNEQ    21$                                         ;
                                   51 DD 00247          PUSHL   ADDR                                        ; 2466
                                   01 DD 00249          PUSHL   #1                                          ;
                          00028228 8F DD 0024B          PUSHL   #164392                                     ;
                         69        03 FB 00251          CALLS   #3, LIB$SIGNAL                              ;
                                5C AE DD 00254 21$:     PUSHL   LOCAL_DESC+4                                ; 2469
                            7E  5C AE 3C 00257          MOVZWL  LOCAL_DESC, -(SP)                           ; 2468
                              01E6 C7 9F 0025B          PUSHAB  P.ABQ                                       ;
                                   70 11 0025F          BRB     28$                                         ;
                         01 00000000G 00 91 00261 22$:  CMPB    DBG$GB_LANGUAGE, #1                         ; 2481
                                   22 12 00268          BNEQ    25$                                         ;
                         02        5A AE 91 0026A       CMPB    LOCAL_DESC+2, #2                            ; 2483
                                   06 12 0026E          BNEQ    23$                                         ;
                   5A AE           06 90 00270          MOVB    #6, LOCAL_DESC+2                            ; 2484
                                   16 11 00274          BRB     25$                                         ;
                         03        5A AE 91 00276 23$:  CMPB    LOCAL_DESC+2, #3                            ; 2485
                                   06 12 0027A          BNEQ    24$                                         ;
                   5A AE           07 90 0027C          MOVB    #7, LOCAL_DESC+2                            ; 2486
                                   0A 11 00280          BRB     25$                                         ;
                         04        5A AE 91 00282 24$:  CMPB    LOCAL_DESC+2, #4                            ; 2487
                                   04 12 00286          BNEQ    25$                                         ;
                   5A AE           08 90 00288          MOVB    #8, LOCAL_DESC+2                            ; 2488
                                   5E DD 0028C 25$:     PUSHL   SP                                          ; 2490
                                50 AE 9F 0028E          PUSHAB  BUFFER_DESC                                 ;
                                60 AE 9F 00291          PUSHAB  LOCAL_DESC                                  ;
```

```
                    00000000G   00          03 FB 00294          CALLS   #3, DBG$CVT_DX_DX                                   ;
                                50       5A AE 9A 0029B          MOVZBL  LOCAL_DESC+2, R0                                    ; 2492
                 21 00000000'   EF          50 E1 0029F          BBC     R0, SIGNED_DTYPE, 26$                               ;
                                01          6C 91 002A7          CMPB    (AP), #1                                            ; 2493
                                            1C 1B 002AA          BLEQU   26$                                                 ;
                                18       08 AC E9 002AC          BLBC    8(AP), 26$                                          ;
                                2D       0C AE 91 002B0          CMPB    TEXT_BUFFER, #45                                    ; 2494
                                            12 13 002B4          BEQL    26$                                                 ;
                                2B       0C AE 91 002B6          CMPB    TEXT_BUFFER, #43                                    ;
                                            0C 13 002BA          BEQL    26$                                                 ;
                 0D   AE        0C    AE     6E 28 002BC          MOVC3   TEXT_LENGTH, TEXT_BUFFER, TEXT_BUFFER+1             ; 2496
                              0C    AE        2B 90 002C2          MOVB    #43, TEXT_BUFFER                                    ; 2497
                                            6E B6 002C6          INCW    TEXT_LENGTH                                         ; 2498
                                0C       0C AE 9F 002C8 26$:     PUSHAB  TEXT_BUFFER                                         ; 2501
                                7E       04 AE 3C 002CB          MOVZWL  TEXT_LENGTH, -(SP)                                  ;
                                            57 DD 002CF 27$:     PUSHL   R7                                                  ;
                                68          03 FB 002D1 28$:     CALLS   #3, DBG$PRINT                                       ;
                                            04 002D4             RET                                                         ; 2504

; Routine Size:  725 bytes,    Routine Base:  DBG$CODE + 1623
```

```
; 2400          2505  1 END
; 2401          2506  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$OWN                         6  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                      496  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     6392  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        55         0        1000       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         0         0           7       00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545       191        12          97       00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418        15         3          31       00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386        10         2          22       00:00.3


;                                   COMMAND QUALIFIERS

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

; Size:          6392 code + 502 data bytes
; Run Time:        01:43.8
; Elapsed Time:    01:53.6
; Lines/CPU Min:    1448
; Lexemes/CPU-Min: 18095
; Memory Used:  478 pages
; Compilation Complete
```

DBGTASK
LIS

DBGVALUES
LIS

DBGTBK
LIS

GETMEMORY
LIS

DSTRECORDS
LIS

ISSH
LIS

RESETSSI
LIS