F 1

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

```
DDDDDDDD    BBBBBBBB      GGGGGGGG  PPPPPPPP    RRRRRRRR      IIIIII    NN      NN  TTTTTTTTTT
DDDDDDDD    BBBBBBBB      GGGGGGGG  PPPPPPPP    RRRRRRRR      IIIIII    NN      NN  TTTTTTTTTT
DD      DD  BB      BB  GG          PP      PP  RR      RR      II      NN      NN      TT
DD      DD  BB      BB  GG          PP      PP  RR      RR      II      NN      NN      TT
DD      DD  BB      BB  GG          PP      PP  RR      RR      II      NNNN    NN      TT
DD      DD  BB      BB  GG          PP      PP  RR      RR      II      NNNN    NN      TT
DD      DD  BBBBBBBB    GG          PPPPPPPP    RRRRRRRR        II      NN  NN  NN      TT
DD      DD  BBBBBBBB    GG          PPPPPPPP    RRRRRRRR        II      NN  NN  NN      TT
DD      DD  BB      BB  GG  GGGGGG  PP          RR  RR          II      NN    NNNN      TT
DD      DD  BB      BB  GG  GGGGGG  PP          RR  RR          II      NN    NNNN      TT
DD      DD  BB      BB  GG      GG  PP          RR    RR        II      NN      NN      TT          ....
DD      DD  BB      BB  GG      GG  PP          RR    RR        II      NN      NN      TT          ....
DDDDDDDD    BBBBBBBB      GGGGGG    PP          RR      RR    IIIIII    NN      NN      TT          ....
DDDDDDDD    BBBBBBBB      GGGGGG    PP          RR      RR    IIIIII    NN      NN      TT          ....


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 DBGPRINT (IDENT = 'V04-000') =
;     2          0002  0
;     3          0003  1 BEGIN
;     4          0004  1
;     5          0005  1 !**************************************************************************
;     6          0006  1 !*                                                                        *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                  *
;    10          0010  1 !*                                                                        *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                           *
;    17          0017  1 !*                                                                        *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                           *
;    21          0021  1 !*                                                                        *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !**************************************************************************
;    27          0027  1 !
;    28          0028  1 ! WRITTEN BY
;    29          0029  1 !       Bert Beander    March, 1982
;    30          0030  1 !
;    31          0031  1 ! MODIFIED BY
;    32          0032  1 !       V. Holt         May, 1982
;    33          0033  1 !       PS              May, 1982
;    34          0034  1 !       PS              Sep, 1982
;    35          0035  1 !
;    36          0036  1 ! REVISION HISTORY
;    37          0037  1 ! 3.01  14-May-82       VJH     Added routine DBG$FLUSHBUF.  Needed to
;    38          0038  1 !                               initialize buffer after error handling.
;    39          0039  1 !
;    40          0040  1 ! 3.01  28-May-82       PS      Added DBG$PRINT_CONTROL function for DBG$PRINT.
;    41          0041  1 !
;    42          0042  1 ! 3B    07-Sep-1982     PS      Added Print Primary Symbol Name Routines.
;    43          0043  1 !
;    44          0044  1 ! 3B    01-Nov-1982     PS      Added Print language specific data routines
;    45          0045  1 !
;    46          0046  1 ! 3B    11-Nov-1982     PS      Added Symbolization routines
;    47          0047  1 !
;    48          0048  1 ! 3B-4  15-Sep-1983     WC3     Added updating of DBG$GL_CURRENT_PRIMARY for
;    49          0049  1 !                               self-referential records.
;    50          0050  1 !
;    51          0051  1 ! MODULE FUNCTION
;    52          0052  1 !       This module contains the DEBUG print formatting and output routines.
;    53          0053  1 !
;    54          0054  1
;    55          0055  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    56          0189  1
;    57          0190  1 LIBRARY 'LIB$:DBGGEN.L32';
```

```
;    58          0191  1
;    59          0192  1 FORWARD ROUTINE
;    60          0193  1     DBG$FAO_OUT: NOVALUE,                  ! Output an FAO-formatted print line
;    61          0194  1     DBG$FORMAT_FAO_OUT: NOVALUE,           ! Format an FAO-formatted print string
;    62          0195  1     DBG$FLUSHBUF:  NOVALUE,                ! Initialize a new print line (without
;    63          0196  1                                            !     output)
;    64          0197  1     DBG$LANGUAGE_FORMAT,                   ! Print value according to language
;    65          0198  1                                            !     specific format
;    66          0199  1     DBG$NEWLINE: NOVALUE,                  ! Output the current print buffer and
;    67          0200  1                                            !     initialize a new print line
;    68          0201  1     DBG$PRINT: NOVALUE,                    ! Output text to the print buffer using
;    69          0202  1                                            !     FAO formatting
;    70          0203  1     DBG$PRINT_CONTROL,                     ! Set indentations for DBG$PRINT
;    71          0204  1     DBG$PRINT_FIELD_REF: NOVALUE,          ! Print <p,s,e> information
;    72          0205  1     DBG$PRINT_IDENTIFIER,                  ! Print name from a Primary or Volatile
;    73          0206  1     DBG$PRINT_IDENTIFIER_PC,               ! Print name from address
;    74          0207  1     DBG$PRINT_OFFSET: NOVALUE,             ! Print offset value in current radix
;    75          0208  1     DBG$PRINT_SET_VALUE: NOVALUE,          ! Print Set Data
;    76          0209  1     DBG$PRINT_SET_LANGUAGE: NOVALUE,       ! Set print info for current language
;    77          0210  1     DBG$PRINT_SYMBOL_NAME: NOVALUE,        ! Print name of data item
;    78          0211  1     DBG$PRINT_SYMBOL_PATHNAME: NOVALUE,    ! Print pathname of data item
;    79          0212  1     DBG$SET_MAX_LINE_WIDTH: NOVALUE,       ! Set the maximum line width
;    80          0213  1     DBG$WRITE_LOG_FILE: NOVALUE,           ! Write a line to the DEBUG log file
;    81          0214  1     DBG$WRITE_OUTPUT: NOVALUE,             ! Write an output line to DBG$OUTPUT
;    82          0215  1                                            !     and to the log file (if any)
;    83          0216  1     GET_SUBSCRIPT_VALUE,                   ! Get array subscript value
;    84          0217  1     PRINT_PUSH;                            ! Push name string pointer on the stack
;    85          0218  1                                            !     upwards or downwards
```

```
;    87          0219  1 EXTERNAL ROUTINE
;    88          0220  1     DBG$COLLECT: NOVALUE,                ! Sanitize character vectors
;    89          0221  1     DBG$FILL_IN_VMS_DESC,                ! Fill in VMS Descriptor
;    90          0222  1     DBG$GET_DST_NAME,                    ! Get the name from DST record
;    91          0223  1     DBG$GET_MEMORY,                      ! Get a memory block
;    92          0224  1     DBG$GET_SET_TYPEID,                  ! Get the parent type of a set
;    93          0225  1     DBG$GET_TEMPMEM,                     ! Get temporary memory block
;    94          0226  1     DBG$MAKE_SKELETON_DESC,              ! Create skeleton descriptor
;    95          0227  1     DBG$NEW_SYMBOLIZE,                   ! New symbolization routine
;    96          0228  1     DBG$NGET_RADIX,                      ! Get current radix
;    97          0229  1     DBG$NPATHDESC_TO_CS: NOVALUE,        ! Print pathname from given pathname
;    98          0230  1                                          !      descriptor
;    99          0231  1     DBG$PRIM_TO_VAL: NOVALUE,            ! Turn primary into value
;   100          0232  1     DBG$PRINT_VALUE,                     ! Print value from descriptor
;   101          0233  1     DBG$PRINT_VALUE_AS_INTEGER: NOVALUE, ! Print an absolute address
;   102          0234  1     DBG$REL_MEMORY,                      ! Release a memory block
;   103          0235  1     DBG$SCR_WRITE_LINE: NOVALUE,         ! Write a line to a Screen Display
;   104          0236  1     DBG$STA_ADDRESS_TO_REGDESCR,         ! Determine if a given address is a
;   105          0237  1                                          !      register address and returns a
;   106          0238  1                                          !      register descriptor if it is
;   107          0239  1     DBG$STA_REGISTER_NAME,               ! Return a pointer to a ASCIC string
;   108          0240  1                                          !      containing the appropriate
;   109          0241  1                                          !      register name
;   110          0242  1     DBG$SYMID_TO_PRIMARY,                ! Return Primary Descriptor for the given
;   111          0243  1                                          !      SYMID
;   112          0244  1     DBG$STA_SYMNAME: NOVALUE,            ! Get name of data item
;   113          0245  1     DBG$STA_SYMPATHNAME: NOVALUE,        ! Get fully-qualified data name
;   114          0246  1     DBG$STA_TYP_ATOMIC: NOVALUE,         ! Get atomic data information
;   115          0247  1     DBG$STA_TYP_ENUM: NOVALUE,           ! Get enumeration data information
;   116          0248  1     DBG$STA_TYP_SET: NOVALUE,            ! Get set data information
;   117          0249  1     DBG$STA_TYP_SUBRNG: NOVALUE,         ! Get subrange data information
;   118          0250  1     SYS$FAO,                             ! Formatter
;   119          0251  1     SYS$FAOL;                            ! Formatted ASCII Output routine to
;   120          0252  1                                          !      do text formatting
;   121          0253  1
;   122          0254  1 EXTERNAL
;   123          0255  1     DBG$GL_CURRENT_PRIMARY,              ! Pointer to the primary being processed
;   124          0256  1     DBG$GB_VERB: BYTE,
;   125          0257  1     DBG$GL_CMND_RADIX,
;   126          0258  1     DBG$GB_RADIX: VECTOR[3, BYTE],       ! Radix settings
;   127          0259  1     DBG$GL_OUTPRAB: BLOCK[,BYTE],        ! RAB for DBG$OUTPUT
;   128          0260  1     DBG$GB_DEF_OUT: VECTOR[,BYTE],       ! Output configuration vector
;   129          0261  1     DBG$GB_LANGUAGE: BYTE,               ! Current language setting
;   130          0262  1     DBG$GB_MOD_PTR: REF VECTOR[,BYTE],   ! Current mode settings
;   131          0263  1     DBG$GL_LOG_BUF,                      ! Pointer to log file's filespec
;   132          0264  1     DBG$GL_LOGRAB: BLOCK[,BYTE],         ! RAB for LOG file
;   133          0265  1     DBG$GL_SCREEN_OUTPUT,                ! Screen output Display ID or zero
;   134          0266  1     DBG$SRC_TERM_WIDTH;                  ! Terminal width
;   135          0267  1
;   136          0268  1
;   137          0269  1 GLOBAL
;   138          0270  1     DBG$GL_ARRSUB_FLAG,                  ! Flag to indicate array subscripting
;   139          0271  1     DBG$GL_RECCMP_FLAG,                  ! Flag to indicate record component
;   140          0272  1     DBG$GL_SIGN_FLAG;                    ! Flag to indicate print '+' before
;   141          0273  1                                          !      signed variables
;   142          0274  1
;   143          0275  1 LITERAL
```

```
;  144          0276  1      PRT$K_COMMON = 0,                   ! Dummy index
;  145          0277  1      RST$K_TYPE_COMMON = 0;              ! Load in common print characters
;  146          0278  1
;  147          0279  1
;  148          0280  1  OWN
;  149          0281  1      BUFFER_DESCR: VECTOR[2,LONG]        ! Descriptor for temporary print
;  150          0282  1                    INITIAL(132, 0),     !      buffer used in DBG$PRINT
;  151          0283  1      MAX_BRK_ON_BLANKS: INITIAL(0),      ! Turn off the PRTBRK_ON_BLANKS flag,
;  152          0284  1                                          !      if the blank character position
;  153          0285  1                                          !      exceeds this size for this print
;  154          0286  1                                          !      line
;  155          0287  1      MAX_INDENT: INITIAL(0),             ! Maximum indentation length.
;  156          0288  1      MAX_PBSIZE: INITIAL(131),           ! Maximum number of characters which
;  157          0289  1                                          !      may be filled into print buffer
;  158          0290  1      PBSIZE: INITIAL(0),                 ! Current number of characters filled
;  159          0291  1                                          !      into the print buffer
;  160          0292  1      PRINT_BUFFER: VECTOR[132,BYTE],     ! The DEBUG print buffer
;  161          0293  1      PRT_CONTINUE: INITIAL(FALSE),       ! Flag set for continuation lines
;  162          0294  1      PRT_INDENT: INITIAL(0),             ! Total indentation length
;  163          0295  1      PRTSET_CONTINUE: INITIAL(0),        ! Continuation indentation length
;  164          0296  1      PRTSET_MARGIN: INITIAL(0),          ! Left margin (for DBG$PRINT buffer)
;  165          0297  1      PRTBRK_ON_BLANKS: INITIAL(0);       ! A flag to indicate break on blanks
```

```
;  167          0298  1 MACRO
;  168        M 0299  1     GET_VALUE(VALUE) =
;  169        M 0300  1
;  170        M 0301  1
;  171        M 0302  1 ! This macro is used in DBG$PRINT_IDENTIFIER.
;  172        M 0303  1 !
;  173        M 0304  1         IF NOT .SUBOVF
;  174        M 0305  1         THEN
;  175        M 0306  1             SUBOVF = GET_SUBSCRIPT_VALUE(NAMEPTR, UPLIT BYTE(%ASCIC '!SL'),
;  176        M 0307  1                 .VALUE)
;  177        M 0308  1
;  178        M 0309  1         ELSE
;  179          0310  1             GET_SUBSCRIPT_VALUE(NAMEPTR, UPLIT BYTE(%ASCIC '!SL'), .VALUE);%;
```

```
;  181          0311  1 MACRO
;  182        M 0312  1     PRINT_OFFSETS_AND_RETURN =
;  183        M 0313  1
;  184        M 0314  1
;  185        M 0315  1 ! This macro is used in DBG$PRINT_IDENTIFIER. It prints out the offset
;  186        M 0316  1 ! and/or bit-field information, as in X+offset or X<p,s,e>.
;  187        M 0317  1 ! It then must decide whether to return the original Value Descriptor or the
;  188        M 0318  1 ! constructed Primary Descriptor to DBG$EXAMINE. The general idea is
;  189        M 0319  1 ! that if we have an exact symbolization match, then return the Primary
;  190        M 0320  1 ! so that the value displayed will be of the correct type.
;  191        M 0321  1 !
;  192        M 0322  1         BEGIN
;  193        M 0323  1
;  194        M 0324  1
;  195        M 0325  1         ! Print the byte offset if it was present. If there was a byte offset,
;  196        M 0326  1         ! then we did not have an exact match so return the Value Descriptor.
;  197        M 0327  1         !
;  198        M 0328  1         IF .BYTE_OFFSET NEQ 0
;  199        M 0329  1         THEN
;  200        M 0330  1             DBG$PRINT_OFFSET(.BYTE_OFFSET);
;  201        M 0331  1
;  202        M 0332  1
;  203        M 0333  1         ! Print the <p,s,e> information if it was present.
;  204        M 0334  1         !
;  205        M 0335  1         IF .VAL_DESC NEQ 0
;  206        M 0336  1         THEN
;  207        M 0337  1             BEGIN
;  208        M 0338  1             IF .BIT_OFFSET NEQ 0
;  209        M 0339  1             THEN
;  210        M 0340  1                 DBG$PRINT_FIELD_REF(.VAL_DESC)
;  211        M 0341  1             ELSE
;  212        M 0342  1                 IF (.PRIM_DESC[DBG$B_DHDR_KIND] NEQ RST$K_DATA) AND
;  213        M 0343  1                    (.PRIM_DESC[DBG$B_DHDR_KIND] NEQ RST$K_TYPCOMP)
;  214        M 0344  1                 THEN
;  215        M 0345  1                     DBG$PRINT_FIELD_REF(.VAL_DESC)
;  216        M 0346  1                 ELSE
;  217        M 0347  1                     BEGIN
;  218        M 0348  1                     LOCAL
;  219        M 0349  1                         TMP_VALDESC: REF DBG$VALDESC;
;  220        M 0350  1                     DBG$PRIM_TO_VAL(.PRIM_DESC, DBG$K_V_VALUE_DESC, TMP_VALDESC);
;  221        M 0351  1                     IF .SAVE_VAL_LENGTH NEQ .TMP_VALDESC[DBG$W_VALUE_LENGTH]
;  222        M 0352  1                     THEN
;  223        M 0353  1                         DBG$PRINT_FIELD_REF(.VAL_DESC);
;  224        M 0354  1                     END;
;  225        M 0355  1
;  226        M 0356  1             ! Put back the original type and length.
;  227        M 0357  1             !
;  228        M 0358  1             IF .SAVE_VAL_DTYPE NEQ 0
;  229        M 0359  1             THEN
;  230        M 0360  1                 BEGIN
;  231        M 0361  1                 VAL_DESC[DBG$B_VALUE_DTYPE] = .SAVE_VAL_DTYPE;
;  232        M 0362  1                 VAL_DESC[DBG$W_VALUE_LENGTH] = .SAVE_VAL_LENGTH;
;  233        M 0363  1                 END;
;  234        M 0364  1
;  235        M 0365  1             IF .VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV
;  236        M 0366  1             OR .VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU
;  237        M 0367  1             OR .VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V
```

```
;  238          M 0368  1              OR .VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU
;  239          M 0369  1              THEN
;  240          M 0370  1                  RETURN .VAL_DESC;
;  241          M 0371  1              END;
;  242          M 0372  1
;  243          M 0373  1
;  244          M 0374  1          ! If there is no value descriptor or exact match.
;  245          M 0375  1          !
;  246          M 0376  1          IF  .VAL_DESC EQL 0
;  247          M 0377  1          OR  (.BYTE_OFFSET EQL 0 AND .BIT_OFFSET EQL 0)
;  248          M 0378  1          THEN
;  249          M 0379  1              BEGIN
;  250          M 0380  1              DBG$COLLECT(.PRIM_DESC);
;  251          M 0381  1              RETURN .PRIM_DESC;
;  252          M 0382  1              END;
;  253          M 0383  1
;  254          M 0384  1          RETURN .VAL_DESC;
;  255            0385  1          END%;
```

```
;  257          0386  1 !       M A C R O S   T O   G E N E R A T E   P R I N T   T A B L E S
;  258          0387  1 !
;  259          0388  1 !
;  260          0389  1 !
;  261          0390  1 !       These macros are used to generate the print tables used by the
;  262          0391  1 !       Print Primary Symbol Name routines to print both language-independent
;  263          0392  1 !       and language-specific constructs accepted by DEBUG.
;  264          0393  1 !
;  265          0394  1 BIND
;  266          0395  1     TABLEBASE = UPLIT BYTE(%ASCII 'BASE');
;  267          0396  1
;  268          0397  1
;  269          0398  1 ! PRINT CHARACTER TABLE
;  270          0399  1 !
;  271          0400  1 !
;  272          0401  1 ! The Print Character Table has a set of entries to define print characters
;  273          0402  1 ! to be used in printing for a given language.  For example, subscript
;  274          0403  1 ! character in PASCAL is [, in FORTRAN is (.  A Print Character Table is
;  275          0404  1 ! declared as follows:
;  276          0405  1 !
;  277          0406  1 !       PRINT_CHAR_TABLE(TBLNAME,
;  278          0407  1 !               PRINT_CHAR_ENTRY(CODE, NAMESTRING),
;  279          0408  1 !                       ...
;  280          0409  1 !               PRINT_CHAR_ENTRY(CODE, NAMESTRING));
;  281          0410  1 !
;  282          0411  1 ! Here TBLNAME is the name of the Print Character Table.  NAMESTRING is the
;  283          0412  1 ! ASCII string which constitutes the print character(s), CODE is the name
;  284          0413  1 ! of the print character.  In some languages, for the same purpose may
;  285          0414  1 ! have different codes, for example, in C, Pointer can be *P or P->X.A
;  286          0415  1 !
;  287          0416  1 MACRO
;  288        M 0417  1     PRINT_CHAR_TABLE(TBLNAME) =
;  289        M 0418  1         OWN     TBLNAME : VECTOR[PRT$K_MAX_PRTCHAR, LONG]
;  290          0419  1                 PSECT(DBG$PLIT) PRESET(%REMAINING) %;
;  291          0420  1
;  292          0421  1 MACRO
;  293        M 0422  1     PRINT_CHAR_ENTRY(CODE, NAMESTRING) =
;  294        M 0423  1         [%NAME('PRT$K_', CODE, '_CHAR')] =
;  295          0424  1                 UPLIT BYTE(%ASCIC NAMESTRING) - TABLEBASE %;
;  296          0425  1
;  297          0426  1
;  298          0427  1 ! PRINT INFORMATION TABLE
;  299          0428  1 !
;  300          0429  1 !
;  301          0430  1 ! The Print Information Table for a language is a blockvector indexed by
;  302          0431  1 ! FCODE.  For each FCODE, it gives the print routine index, and the
;  303          0432  1 ! address of the corresponding Print Character Table.  All addresses are
;  304          0433  1 ! relative to TABLEBASE.  A Print Information Table is declared as follows:
;  305          0434  1 !
;  306          0435  1 !       PRINT_INFO_TABLE(TBLNAME,
;  307          0436  1 !               PRINT_INFO_ENTRY(FCODE, ROUT_INDEX, PRINT_CHAR_TABLE),
;  308          0437  1 !                       ...
;  309          0438  1 !               PRINT_INFO_ENTRY(FCODE, ROUT_INDEX, PRINT_CHAR_TABLE);
;  310          0439  1 !
;  311          0440  1 ! Here TBLNAME is the name of the Print Information Table.  ROUT_INDEX is
;  312          0441  1 ! the Print Routine for the given FCODE.  PRINT_CHAR_TABLE points to
;  313          0442  1 ! Print Character Table which contains language-specific print characters
```

```
;  314          0443  1 ! for the given FCODE.
;  315          0444  1 !
;  316          0445  1 MACRO
;  317        M 0446  1     PRINT_INFO_TABLE(TBLNAME) =
;  318        M 0447  1         %IF %LENGTH EQL 1
;  319        M 0448  1         %THEN
;  320        M 0449  1             OWN TBLNAME: PRTINFO$TABLE PSECT(DBG$PLIT)
;  321        M 0450  1                          INITIAL(REP RST$K_TYPE_MAXIMUM OF (0))
;  322        M 0451  1         %ELSE
;  323        M 0452  1             OWN TBLNAME : PRTINFO$TABLE PSECT(DBG$PLIT) PRESET(%REMAINING)
;  324          0453  1         %FI %;
;  325          0454  1
;  326          0455  1 MACRO
;  327        M 0456  1     PRINT_INFO_ENTRY(FCODE, ROUT_INDEX, CHARTBL) =
;  328        M 0457  1         [%NAME ('RST$K_TYPE_', FCODE), PRTINFO$L_ROUT_INDEX] =
;  329        M 0458  1                 %NAME ('PRT$K_', ROUT_INDEX),
;  330        M 0459  1         [%NAME ('RST$K_TYPE_', FCODE), PRTINFO$L_CHARTBL] =
;  331          0460  1                 CHARTBL - TABLEBASE %;
;  332          0461  1
;  333          0462  1
;  334          0463  1 ! TABLE OF LANGUAGE-SPECIFIC TABLES
;  335          0464  1 !
;  336          0465  1 ! Define the macro which builds the table of pointers to the language-specific
;  337          0466  1 ! print tables.  Each pointer is relative to TABLEBASE.  The macro is used as
;  338          0467  1 ! follows:
;  339          0468  1 !
;  340          0469  1 !       LANGUAGE_PRINT_TABLES(LANGUAGE = language-name,
;  341          0470  1 !                 INFO_TABLE = Print Information Table,
;  342          0471  1 !                 INFO_FLAG  = Print Information Flag true-or-false,
;  343          0472  1 !                 REFMOD_FLAG = Reference Modifier Flag true-or-false,
;  344          0473  1 !                 ARRSUB_FLAG = Array Subscript Flag true-or-false,
;  345          0474  1 !                 RECCMP_FLAG = Record Component Flag true-or-false),
;  346          0475  1 !                 SIGN_FLAG   = Print '+' for signed variables Flag true-or-false);
;  347          0476  1 !
;  348          0477  1 ! Note: One of these days those flags should be changed to bit instead of long
;  349          0478  1 !       word.
;  350          0479  1 !
;  351          0480  1 ! Here INFO_FLAG is an  indicator to indicate how we want the name to be printed.
;  352          0481  1 ! For example, record component Y in record X, in PASCAL, we print it as X.Y
;  353          0482  1 ! (flag sets to TRUE), in COBOL, we print it as Y OF X (flag sets to FALSE).
;  354          0483  1 ! REFMOD_FLAG is an indicator to indicate how we want the reference modifier
;  355          0484  1 ! to be printed.  For example, in FORTRAN (begin:end) (flag sets to TRUE),
;  356          0485  1 ! in COBOL (begin:length) (flag sets to FALSE).  ARRSUB_FLAG is used to
;  357          0486  1 ! indicate whether the language has array concept or not.  (similar idea with
;  358          0487  1 ! RECCMP_FLAG).  These flags primarily are used for symbolization purposes.
;  359          0488  1 ! ie, using fortran program, set lang to macro, which macro does not understand
;  360          0489  1 ! the array, so we'll print address+offset instead of arr(i,j).  SIGN_FLAG
;  361          0490  1 ! is used for the language to print out '+' before the signed variables.
;  362          0491  1 !
;  363          0492  1 !
;  364          0493  1 KEYWORDMACRO
;  365          0494  1     LANGUAGE_PRINT_TABLES(LANGUAGE=UNKNOWN,
;  366          0495  1                           INFO_TABLE=TABLEBASE,
;  367          0496  1                           INFO_FLAG=TRUE,
;  368          0497  1                           REFMOD_FLAG=TRUE,
;  369          0498  1                           ARRSUB_FLAG=TRUE,
;  370          0499  1                           RECCMP_FLAG=TRUE,
```

```
;  371        M 0500  1                        SIGN_FLAG = FALSE) =
;  372        M 0501  1          BIND %NAME(LANGUAGE, '_PRINT_TABLE') =
;  373        M 0502  1                  UPLIT (INFO_TABLE - TABLEBASE,
;  374        M 0503  1                         INFO_FLAG,
;  375        M 0504  1                         REFMOD_FLAG,
;  376        M 0505  1                         ARRSUB_FLAG,
;  377        M 0506  1                         RECCMP_FLAG,
;  378          0507  1                         SIGN_FLAG): VECTOR[,LONG] %;
```

```
;  380          0508  1 !                          P R I N T   T A B L E S
;  381          0509  1 !
;  382          0510  1 !
;  383          0511  1 !
;  384          0512  1 !       This s        ncludes all the tables needed to print the ADA symbols.
;  385          0513  1 !
;  386          0514  1 !
;  387          0515  1 !
;  388          0516  1 ! Define the ADA Language Print Tables.
;  389          0517  1 !
;  390        P 0518  1 PRINT_CHAR_TABLE(ADA_COMMON_CHAR,
;  391        P 0519  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  392        P 0520  1         PRINT_CHAR_ENTRY(SEPARATOR, '..'),
;  393          0521  1         PRINT_CHAR_ENTRY(END, ')'));
;  394          0522  1
;  395        P 0523  1 PRINT_CHAR_TABLE(ADA_ARRAY_CHAR,
;  396        P 0524  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  397        P 0525  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  398          0526  1         PRINT_CHAR_ENTRY(END, ')'));
;  399          0527  1
;  400        P 0528  1 PRINT_CHAR_TABLE(ADA_POINTER_CHAR,
;  401          0529  1         PRINT_CHAR_ENTRY(SEPARATOR, '.ALL'));
;  402          0530  1
;  403        P 0531  1 PRINT_CHAR_TABLE(ADA_RECORD_CHAR,
;  404          0532  1         PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  405          0533  1
;  406        P 0534  1 PRINT_INFO_TABLE(ADA_INFO_TABLE,
;  407        P 0535  1         PRINT_INFO_ENTRY(COMMON, COMMON, ADA_COMMON_CHAR),
;  408        P 0536  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, ADA_ARRAY_CHAR),
;  409        P 0537  1         PRINT_INFO_ENTRY(TPTR, POINTER, ADA_POINTER_CHAR),
;  410        P 0538  1         PRINT_INFO_ENTRY(FILE, POINTER, ADA_POINTER_CHAR),
;  411        P 0539  1         PRINT_INFO_ENTRY(RECORD, RECORD, ADA_RECORD_CHAR),
;  412          0540  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, ADA_RECORD_CHAR));
;  413          0541  1
;  414        P 0542  1 LANGUAGE_PRINT_TABLES(LANGUAGE = ADA,
;  415        P 0543  1         INFO_TABLE = ADA_INFO_TABLE,
;  416        P 0544  1         INFO_FLAG = TRUE,
;  417          0545  1         REFMOD_FLAG = TRUE);
```

```
;  419          0546  1 !               B A S I C   P R I N T   T A B L E S
;  420          0547  1 !
;  421          0548  1 !
;  422          0549  1 !
;  423          0550  1 !       This section includes all the tables needed to print the BASIC symbols.
;  424          0551  1 !
;  425          0552  1 !
;  426          0553  1 !
;  427          0554  1 ! Define the BASIC Language Print Tables.
;  428          0555  1 !
;  429        P 0556  1 PRINT_CHAR_TABLE(BASIC_COMMON_CHAR,
;  430        P 0557  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  431        P 0558  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  432          0559  1         PRINT_CHAR_ENTRY(END, ')'));
;  433          0560  1
;  434        P 0561  1 PRINT_CHAR_TABLE(BASIC_ARRAY_CHAR,
;  435        P 0562  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  436        P 0563  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  437          0564  1         PRINT_CHAR_ENTRY(END, ')'));
;  438          0565  1
;  439        P 0566  1 PRINT_CHAR_TABLE(BASIC_RECORD_CHAR,
;  440          0567  1         PRINT_CHAR_ENTRY(SEPARATOR, '::'));
;  441          0568  1
;  442        P 0569  1 PRINT_INFO_TABLE(BASIC_INFO_TABLE,
;  443        P 0570  1         PRINT_INFO_ENTRY(COMMON, COMMON, BASIC_COMMON_CHAR),
;  444        P 0571  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, BASIC_ARRAY_CHAR),
;  445        P 0572  1         PRINT_INFO_ENTRY(RECORD, RECORD, BASIC_RECORD_CHAR),
;  446          0573  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, BASIC_RECORD_CHAR));
;  447          0574  1
;  448        P 0575  1 LANGUAGE_PRINT_TABLES(LANGUAGE = BASIC,
;  449        P 0576  1         INFO_TABLE = BASIC_INFO_TABLE,
;  450        P 0577  1         INFO_FLAG = TRUE,
;  451        P 0578  1         REFMOD_FLAG = TRUE,
;  452        P 0579  1         ARRSUB_FLAG = TRUE,
;  453        P 0580  1         RECCMP_FLAG = TRUE,
;  454          0581  1         SIGN_FLAG = FALSE);
```

```
;  456          0582  1 !               B L I S S   P R I N T   T A B L E S
;  457          0583  1 !
;  458          0584  1 !
;  459          0585  1 !
;  460          0586  1 !       This section includes all the tables needed to print the BLISS symbols.
;  461          0587  1 !
;  462          0588  1 !
;  463          0589  1 !
;  464          0590  1 ! Define the BLISS Language Print Tables.
;  465          0591  1 !
;  466        P 0592  1 PRINT_CHAR_TABLE(BLISS_ARRAY_CHAR,
;  467        P 0593  1         PRINT_CHAR_ENTRY(BEGIN, '['),
;  468        P 0594  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  469          0595  1         PRINT_CHAR_ENTRY(END, ']'));
;  470          0596  1
;  471        P 0597  1 PRINT_INFO_TABLE(BLISS_INFO_TABLE,
;  472          0598  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, BLISS_ARRAY_CHAR));
;  473          0599  1
;  474        P 0600  1 LANGUAGE_PRINT_TABLES(LANGUAGE = BLISS,
;  475        P 0601  1         INFO_TABLE = BLISS_INFO_TABLE,
;  476        P 0602  1         INFO_FLAG = TRUE,
;  477        P 0603  1         REFMOD_FLAG = TRUE,
;  478        P 0604  1         ARRSUB_FLAG = TRUE,
;  479        P 0605  1         RECCMP_FLAG = FALSE,
;  480          0606  1         SIGN_FLAG = FALSE);
```

```
;  482          0607  1 !               C   P R I N T   T A B L E S
;  483          0608  1 !
;  484          0609  1 !
;  485          0610  1 !
;  486          0611  1 !       This section includes all the tables needed to print the C symbols.
;  487          0612  1 !
;  488          0613  1 !
;  489          0614  1 !
;  490          0615  1 ! Define the C Language Print Tables.
;  491          0616  1 !
;  492        P 0617  1 PRINT_CHAR_TABLE(C_COMMON_CHAR,
;  493        P 0618  1         PRINT_CHAR_ENTRY(BEGIN, '['),
;  494        P 0619  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  495          0620  1         PRINT_CHAR_ENTRY(END, ']'));
;  496          0621  1
;  497        P 0622  1 PRINT_CHAR_TABLE(C_ARRAY_CHAR,
;  498        P 0623  1         PRINT_CHAR_ENTRY(BEGIN, '['),
;  499        P 0624  1         PRINT_CHAR_ENTRY(SEPARATOR, ']['),
;  500          0625  1         PRINT_CHAR_ENTRY(END, ']'));
;  501          0626  1
;  502        P 0627  1 PRINT_CHAR_TABLE(C_POINTER_CHAR,
;  503        P 0628  1         PRINT_CHAR_ENTRY(SEPARATOR, '*'),
;  504          0629  1         PRINT_CHAR_ENTRY(RECPTR, '->'));
;  505          0630  1                                                                          000
;  506        P 0631  1 PRINT_CHAR_TABLE(C_RECORD_CHAR,
;  507          0632  1         PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  508          0633  1
;  509        P 0634  1 PRINT_INFO_TABLE(C_INFO_TABLE,
;  510        P 0635  1         PRINT_INFO_ENTRY(COMMON, COMMON, C_COMMON_CHAR),
;  511        P 0636  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, C_ARRAY_CHAR),
;  512        P 0637  1         PRINT_INFO_ENTRY(TPTR, POINTER_C, C_POINTER_CHAR),
;  513        P 0638  1         PRINT_INFO_ENTRY(RECORD, RECORD_C_P[I, C_RECORD_CHAR),
;  514          0639  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, C_RECORD_CHAR));
;  515          0640  1
;  516        P 0641  1 LANGUAGE_PRINT_TABLES(LANGUAGE = C,
;  517        P 0642  1         INFO_TABLE = C_INFO_TABLE,
;  518        P 0643  1         INFO_FLAG = TRUE,
;  519        P 0644  1         REFMOD_FLAG = TRUE,
;  520        P 0645  1         ARRSUB_FLAG = TRUE,
;  521        P 0646  1         RECCMP_FLAG = TRUE,
;  522          0647  1         SIGN_FLAG = FALSE);
```

000

000

```
;  524          0648  1 !               C O B O L   P R I N T   T A B L E S
;  525          0649  1 !
;  526          0650  1 !
;  527          0651  1 !
;  528          0652  1 !       This section includes all the tables needed to print the COBOL symbols.
;  529          0653  1 !
;  530          0654  1 !
;  531          0655  1 !
;  532          0656  1 ! Define the COBOL Language Print Tables.
;  533          0657  1 !
;  534        P 0658  1 PRINT_CHAR_TABLE(COBOL_COMMON_CHAR,
;  535        P 0659  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  536        P 0660  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  537          0661  1         PRINT_CHAR_ENTRY(END, ')'));
;  538          0662  1
;  539        P 0663  1 PRINT_CHAR_TABLE(COBOL_ARRAY_CHAR,
;  540        P 0664  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  541        P 0665  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  542          0666  1         PRINT_CHAR_ENTRY(END, ')'));
;  543          0667  1
;  544        P 0668  1 PRINT_CHAR_TABLE(COBOL_RECORD_CHAR,
;  545          0669  1         PRINT_CHAR_ENTRY(SEPARATOR, ' of '));
;  546          0670  1
;  547        P 0671  1 PRINT_INFO_TABLE(COBOL_INFO_TABLE,
;  548        P 0672  1         PRINT_INFO_ENTRY(COMMON, COMMON, COBOL_COMMON_CHAR),
;  549        P 0673  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, COBOL_ARRAY_CHAR),
;  550        P 0674  1         PRINT_INFO_ENTRY(RECORD, RECORD_COB, COBOL_RECORD_CHAR),
;  551          0675  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, COBOL_RECORD_CHAR));
;  552          0676  1
;  553        P 0677  1 LANGUAGE_PRINT_TABLES(LANGUAGE = COBOL,
;  554        P 0678  1         INFO_TABLE = COBOL_INFO_TABLE,
;  555        P 0679  1         INFO_FLAG = FALSE,
;  556        P 0680  1         REFMOD_FLAG = FALSE,                                                    000
;  557        P 0681  1         ARRSUB_FLAG = TRUE,
;  558        P 0682  1         RECCMP_FLAG = TRUE,
;  559          0683  1         SIGN_FLAG = TRUE);
```

```
;  561          0684  1 !               F O R T R A N   P R I N T   T A B L E S
;  562          0685  1 !
;  563          0686  1 !
;  564          0687  1 !
;  565          0688  1 !       This section includes all the tables needed to print the FORTRAN symbols.
;  566          0689  1 !
;  567          0690  1 !
;  568          0691  1 !
;  569          0692  1 ! Define the FORTRAN Language Print Tables.
;  570          0693  1 !
;  571        P 0694  1 PRINT_CHAR_TABLE(FORTRAN_COMMON_CHAR,
;  572        P 0695  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  573        P 0696  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  574          0697  1         PRINT_CHAR_ENTRY(END, ')'));
;  575          0698  1
;  576        P 0699  1 PRINT_CHAR_TABLE(FORTRAN_ARRAY_CHAR,
;  577        P 0700  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  578        P 0701  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  579          0702  1         PRINT_CHAR_ENTRY(END, ')'));
;  580          0703  1
;  581        P 0704  1 PRINT_CHAR_TABLE(FORTRAN_RECORD_CHAR,
;  582          0705  1         PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  583          0706  1
;  584        P 0707  1 PRINT_INFO_TABLE(FORTRAN_INFO_TABLE,
;  585        P 0708  1         PRINT_INFO_ENTRY(COMMON, COMMON, FORTRAN_COMMON_CHAR),
;  586        P 0709  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, FORTRAN_ARRAY_CHAR),
;  587          0710  1         PRINT_INFO_ENTRY(RECORD, RECORD, FORTRAN_RECORD_CHAR));
;  588          0711  1
;  589        P 0712  1 LANGUAGE_PRINT_TABLES(LANGUAGE = FORTRAN,
;  590        P 0713  1         INFO_TABLE = FORTRAN_INFO_TABLE,
;  591        P 0714  1         INFO_FLAG = TRUE,
;  592        P 0715  1         REFMOD_FLAG = TRUE,
;  593        P 0716  1         ARRSUB_FLAG = TRUE,
;  594        P 0717  1         RECCMP_FLAG = TRUE,
;  595          0718  1         SIGN_FLAG = FALSE);
```

```
;  597          0719  1 !               M A C R O   P R I N T   T A B L E S
;  598          0720  1 !
;  599          0721  1 !
;  600          0722  1 !
;  601          0723  1 !       This section includes all the tables needed to print the MACRO symbols.
;  602          0724  1 !
;  603          0725  1 !
;  604          0726  1 !
;  605          0727  1 ! Define the MACRO Language Print Tables.
;  606          0728  1 !
;  607          0729  1 PRINT_INFO_TABLE(MACRO_INFO_TABLE);
;  608          0730  1
;  609        P 0731  1 LANGUAGE_PRINT_TABLES(LANGUAGE = MACRO,
;  610        P 0732  1           INFO_TABLE = MACRO_INFO_TABLE,
;  611        P 0733  1           INFO_FLAG = TRUE,
;  612        P 0734  1           REFMOD_FLAG = TRUE,
;  613        P 0735  1           ARRSUB_FLAG = FALSE,
;  614        P 0736  1           RECCMP_FLAG = FALSE,
;  615          0737  1           SIGN_FLAG = FALSE);
```

```
;  617          0738  1 !                P A S C A L   P R I N T   T A B L E S
;  618          0739  1 !
;  619          0740  1 !
;  620          0741  1 !
;  621          0742  1 !       This section includes all the tables needed to print the PASCAL symbols.
;  622          0743  1 !
;  623          0744  1 !
;  624          0745  1 !
;  625          0746  1 ! Define the PASCAL Language Print Tables.
;  626          0747  1 !
;  627        P 0748  1 PRINT_CHAR_TABLE(PASCAL_COMMON_CHAR,
;  628        P 0749  1         PRINT_CHAR_ENTRY(BEGIN, '['),
;  629        P       1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  630          0751  1         PRINT_CHAR_ENTRY(END, ']'));
;  631          0752  1
;  632        P 0753  1 PRINT_CHAR_TABLE(PASCAL_ARRAY_CHAR,
;  633        P 0754  1         PRINT_CHAR_ENTRY(BEGIN, '['),
;  634        P 0755  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  635          0756  1         PRINT_CHAR_ENTRY(END, ']'));
;  636          0757  1
;  637        P 0758  1 PRINT_CHAR_TABLE(PASCAL_POINTER_CHAR,
;  638          0759  1         PRINT_CHAR_ENTRY(SEPARATOR, '^'));
;  639          0760  1
;  640        P 0761  1 PRINT_CHAR_TABLE(PASCAL_RECORD_CHAR,
;  641          0762  1         PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  642          0763  1
;  643        P 0764  1 PRINT_INFO_TABLE(PASCAL_INFO_TABLE,
;  644        P 0765  1         PRINT_INFO_ENTRY(COMMON, COMMON, PASCAL_COMMON_CHAR),
;  645        P 0766  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, PASCAL_ARRAY_CHAR),
;  646        P 0767  1         PRINT_INFO_ENTRY(TPTR, POINTER, PASCAL_POINTER_CHAR),
;  647        P 0768  1         PRINT_INFO_ENTRY(FILE, POINTER, PASCAL_POINTER_CHAR),
;  648        P 0769  1         PRINT_INFO_ENTRY(RECORD, RECORD, PASCAL_RECORD_CHAR),
;  649          0770  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, PASCAL_RECORD_CHAR));
;  650          0771  1
;  651        P 0772  1 LANGUAGE_PRINT_TABLES(LANGUAGE = PASCAL,
;  652        P 0773  1         INFO_TABLE = PASCAL_INFO_TABLE,
;  653        P 0774  1         INFO_FLAG = TRUE,
;  654        P 0775  1         REFMOD_FLAG = TRUE,
;  655        P 0776  1         ARRSUB_FLAG = TRUE,
;  656        P 0777  1         RECCMP_FLAG = TRUE,
;  657          0778  1         SIGN_FLAG = FALSE);
```

```
;  659          0779  1 !               P L I   P R I N T   T A B L E S
;  660          0780  1 !
;  661          0781  1 !
;  662          0782  1 !
;  663          0783  1 !       This section includes all the tables needed to print the PLI symbols.
;  664          0784  1 !
;  665          0785  1 !
;  666          0786  1 !
;  667          0787  1 ! Define the PLI Language Print Tables.
;  668          0788  1 !
;  669        P 0789  1 PRINT_CHAR_TABLE(PLI_COMMON_CHAR,
;  670        P 0790  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  671        P 0791  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),
;  672          0792  1         PRINT_CHAR_ENTRY(END, ')'));
;  673          0793  1
;  674        P 0794  1 PRINT_CHAR_TABLE(PLI_ARRAY_CHAR,
;  675        P 0795  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  676        P 0796  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  677          0797  1         PRINT_CHAR_ENTRY(END, ')'));
;  678          0798  1
;  679        P 0799  1 PRINT_CHAR_TABLE(PLI_POINTER_CHAR,
;  680          0800  1         PRINT_CHAR_ENTRY(SEPARATOR, '->'));
;  681          0801  1
;  682        P 0802  1 PRINT_CHAR_TABLE(PLI_RECORD_CHAR,
;  683          0803  1         PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  684          0804  1
;  685        P 0805  1 PRINT_INFO_TABLE(PLI_INFO_TABLE,
;  686        P 0806  1         PRINT_INFO_ENTRY(COMMON, COMMON, PLI_COMMON_CHAR),
;  687        P 0807  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, PLI_ARRAY_CHAR),
;  688        P 0808  1         PRINT_INFO_ENTRY(PTR, POINTER, PLI_POINTER_CHAR),
;  689        P 0809  1         PRINT_INFO_ENTRY(RECORD, RECORD_C_PLI, PLI_RECORD_CHAR),
;  690          0810  1         PRINT_INFO_ENTRY(VARIANT, VARIANT, PLI_RECORD_CHAR));
;  691          0811  1
;  692        P 0812  1 LANGUAGE_PRINT_TABLES(LANGUAGE = PLI,
;  693        P 0813  1         INFO_TABLE = PLI_INFO_TABLE,
;  694        P 0814  1         INFO_FLAG = TRUE,
;  695        P 0815  1         REFMOD_FLAG = TRUE,
;  696        P 0816  1         ARRSUB_FLAG = TRUE,
;  697        P 0817  1         RECCMP_FLAG = TRUE,
;  698          0818  1         SIGN_FLAG = FALSE);
```

```
;  700          0819  1 !                RPG  PRINT  TABLES
;  701          0820  1 !
;  702          0821  1 !
;  703          0822  1 !
;  704          0823  1 !       This section includes all the tables needed to print the RPG symbols.
;  705          0824  1 !
;  706          0825  1 !
;  707          0826  1 !
;  708          0827  1 ! Define the RPG Language Print Tables.
;  709          0828  1 !
;  710        P 0829  1 PRINT_CHAR_TABLE(RPG_COMMON_CHAR,
;  711        P 0830  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  712        P 0831  1         PRINT_CHAR_ENTRY(SEPARATOR, ':'),                                                    ;
;  713          0832  1         PRINT_CHAR_ENTRY(END, ')'));
;  714          0833  1
;  715        P 0834  1 PRINT_CHAR_TABLE(RPG_ARRAY_CHAR,
;  716        P 0835  1         PRINT_CHAR_ENTRY(BEGIN, '('),
;  717        P 0836  1         PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  718          0837  1         PRINT_CHAR_ENTRY(END, ')'));
;  719          0838  1
;  720          0839  1
;  721        P 0840  1 PRINT_INFO_TABLE(RPG_INFO_TABLE,
;  722        P 0841  1         PRINT_INFO_ENTRY(COMMON, COMMON, RPG_COMMON_CHAR),
;  723          0842  1         PRINT_INFO_ENTRY(ARRAY, ARRAY, RPG_ARRAY_CHAR));
;  724          0843  1
;  725        P 0844  1 LANGUAGE_PRINT_TABLES(LANGUAGE = RPG,
;  726        P 0845  1         INFO_TABLE = RPG_INFO_TABLE,
;  727        P 0846  1         INFO_FLAG = FALSE,
;  728        P 0847  1         REFMOD_FLAG = FALSE,
;  729        P 0848  1         ARRSUB_FLAG = TRUE,
;  730        P 0849  1         RECCMP_FLAG = FALSE,
;  731          0850  1         SIGN_FLAG = FALSE);
```

```
;  733          0851  1 !               U N K N O W N   P R I N T   T A B L E S
;  734          0852  1 !
;  735          0853  1 !
;  736          0854  1 !
;  737          0855  1 !       This section includes all the tables needed to print language
;  738          0856  1 !       UNKNOWN symbols.
;  739          0857  1 !
;  740          0858  1 !
;  741          0859  1 !
;  742          0860  1 ! Language UNKNOWN has records and arrays.  Define print syntax for them
;  743          0861  1 ! which corresponds to syntax given in DBGPARSER.
;  744          0862  1 !
;  745        P 0863  1 PRINT_CHAR_TABLE(UNKNOWN_ARRAY_CHAR,
;  746        P 0864  1          PRINT_CHAR_ENTRY(BEGIN, '('),
;  747        P 0865  1          PRINT_CHAR_ENTRY(SEPARATOR, ','),
;  748          0866  1          PRINT_CHAR_ENTRY(END, ')'));
;  749          0867  1
;  750        P 0868  1 PRINT_CHAR_TABLE(UNKNOWN_RECORD_CHAR,
;  751          0869  1          PRINT_CHAR_ENTRY(SEPARATOR, '.'));
;  752          0870  1
;  753        P 0871  1 PRINT_CHAR_TABLE(UNKNOWN_POINTER_CHAR,
;  754          0872  1          PRINT_CHAR_ENTRY(SEPARATOR, '^'));
;  755          0873  1
;  756        P 0874  1 PRINT_INFO_TABLE(UNKNOWN_INFO_TABLE,
;  757        P 0875  1          PRINT_INFO_ENTRY(ARRAY, ARRAY, UNKNOWN_ARRAY_CHAR),
;  758        P 0876  1          PRINT_INFO_ENTRY(TPTR, POINTER, UNKNOWN_POINTER_CHAR),
;  759        P 0877  1          PRINT_INFO_ENTRY(FILE, POINTER, UNKNOWN_POINTER_CHAR),
;  760        P 0878  1          PRINT_INFO_ENTRY(RECORD, RECORD, UNKNOWN_RECORD_CHAR),
;  761          0879  1          PRINT_INFO_ENTRY(VARIANT, VARIANT, UNKNOWN_RECORD_CHAR));
;  762          0880  1
;  763        P 0881  1 LANGUAGE_PRINT_TABLES(LANGUAGE = UNKNOWN,
;  764        P 0882  1          INFO_TABLE = UNKNOWN_INFO_TABLE,
;  765        P 0883  1          INFO_FLAG = TRUE,
;  766        P 0884  1          REFMOD_FLAG = TRUE,
;  767        P 0885  1          ARRSUB_FLAG = TRUE,
;  768        P 0886  1          RECCMP_FLAG = TRUE,
;  769          0887  1          SIGN_FLAG = FALSE);
```

```
;  771          0888  1 !       T A B L E   O F   P O I N T E R S   T O   L A N G U A G E   T A B L E S
;  772          0889  1 !
;  773          0890  1 !
;  774          0891  1 !
;  775          0892  1 !       This section contains the table of pointers to the language-specific
;  776          0893  1 !       tables of print table pointers.  In other words, this table is indexed
;  777          0894  1 !       by language code to yield a pointer (relative to TABLEBASE) to the
;  778          0895  1 !       table of table pointers build by the LANGUAGE_PRINT_TABLES macro
;  779          0896  1 !       for the individual language above.
;  780          0897  1 !
;  781          0898  1 !
;  782          0899  1 ! Define the table of pointers to language-specific tables.
;  783          0900  1 !
;  784          0901  1 OWN
;  785          0902  1     LANGUAGE_PRINT_TABLE_PTRS:
;  786          0903  1                 VECTOR[DBG$K_MAX_LANGUAGE + 1, LONG] PSECT(DBG$PLIT) PRESET(
;  787          0904  1
;  788          0905  1         [DBG$K_MACRO]   = MACRO_PRINT_TABLE   - TABLEBASE,      ! MACRO
;  789          0906  1         [DBG$K_FORTRAN] = FORTRAN_PRINT_TABLE - TABLEBASE,      ! FORTRAN
;  790          0907  1         [DBG$K_BLISS]   = BLISS_PRINT_TABLE   - TABLEBASE,      ! BLISS
;  791          0908  1         [DBG$K_COBOL]   = COBOL_PRINT_TABLE   - TABLEBASE,      ! COBOL
;  792          0909  1         [DBG$K_BASIC]   = BASIC_PRINT_TABLE   - TABLEBASE,      ! BASIC
;  793          0910  1         [DBG$K_PLI]     = PLI_PRINT_TABLE     - TABLEBASE,      ! PLI
;  794          0911  1         [DBG$K_PASCAL]  = PASCAL_PRINT_TABLE  - TABLEBASE,      ! PASCAL
;  795          0912  1         [DBG$K_C]       = C_PRINT_TABLE       - TABLEBASE,      ! C
;  796          0913  1         [DBG$K_RPG]     = RPG_PRINT_TABLE     - TABLEBASE,      ! RPG
;  797          0914  1         [DBG$K_ADA]     = ADA_PRINT_TABLE     - TABLEBASE,      ! ADA
;  798          0915  1         [DBG$K_UNKNOWN] = UNKNOWN_PRINT_TABLE - TABLEBASE);     ! UNKNOWN
```

```
;  800          0916  1 FIELD PRTID$STACK_FLD_DEF =
;  801          0917  1         SET
;  802          0918  1         PRTID$STACK_PTR = [0, L_],      ! Pointer
;  803          0919  1         PRTID$STACK_FLAG = [4, B0_]     ! Flag set to indicate PTR field is
;  804          0920  1                                         !       a pointer to value descritptor
;  805          0921  1         TES;                            !       and pointer to ASCIC string
;  806          0922  1
;  807          0923  1 LITERAL
;  808          0924  1         PRINT_STACK_SIZE = 100,         ! Stack size for the print name stack
;  809          0925  1         PRINT_STACK_START = 60;         ! Start stack pointer
;  810          0926  1
;  811          0927  1 MACRO
;  812        M 0928  1         PRTID$STACK = BLOCKVECTOR[PRINT_STACK_SIZE, 5, BYTE]
;  813          0929  1                       FIELD (PRTID$STACK_FLD_DEF) %;
;  814          0930  1
```

```
;  816          0931  1 GLOBAL ROUTINE DBG$FAO_OUT(STRING, ARGUMENTS) : NOVALUE =
;  817          0932  1 !
;  818          0933  1 ! FUNCTION
;  819          0934  1 !       This routine formats an output string using FAO and writes the
;  820          0935  1 !       formatted string to DBG$OUTPUT and the log file (if any).  FAO
;  821          0936  1 !       (Formatted ASCII Output--see the System Services Manual) does the
;  822          0937  1 !       formatting via a call on SYS$FAOL.  The formatted text string is
;  823          0938  1 !       then written out by routine DBG$WRITE_OUTPUT.
;  824          0939  1 !
;  825          0940  1 ! INPUTS
;  826          0941  1 !       STRING          - The address of a Counted ASCII FAO control string.
;  827          0942  1 !
;  828          0943  1 !       ARGUMENTS       - Zero or more arguments to be applied to the FAO
;  829          0944  1 !                         control string.
;  830          0945  1 !
;  831          0946  1 ! OUTPUTS
;  832          0947  1 !       NONE
;  833          0948  1 !
;  834          0949  1
;  835          0950  2     BEGIN
;  836          0951  2
;  837          0952  2     MAP
;  838          0953  2         STRING: REF VECTOR[,BYTE];      ! Address of FAO control string
;  839          0954  2
;  840          0955  2     LOCAL
;  841          0956  2         BUFFER: VECTOR[132,BYTE],       ! Buffer to hold output string from
;  842          0957  2                                         !       SYS$FAOL call
;  843          0958  2         INP_DESC: VECTOR[2,LONG],       ! String descriptor for input buffer
;  844          0959  2         LENGTH: WORD,                   ! Length of SYS$FAOL output string
;  845          0960  2         OUT_DESC: VECTOR[2,LONG];       ! String descriptor for SYS$FAOL
;  846          0961  2                                         !       output buffer
;  847          0962  2
;  848          0963  2
;  849          0964  2
;  850          0965  2     ! Set up the descriptors needed for the SYS$FAOL call.  Then call SYS$FAOL
;  851          0966  2     ! to format the text string and call DBG$WRITE_OUTPUT to actually output
;  852          0967  2     ! the string to DBG$OUTPUT and the log file, if any.
;  853          0968  2     !
;  854          0969  2     INP_DESC [0] = .STRING [0];
;  855          0970  2     INP_DESC [1] = STRING [1];
;  856          0971  2     OUT_DESC [0] = 131;
;  857          0972  2     OUT_DESC [1] = BUFFER;
;  858          0973  2     SYS$FAOL(INP_DESC, LENGTH, OUT_DESC, ARGUMENTS);
;  859          0974  2     DBG$WRITE_OUTPUT(.LENGTH, BUFFER);
;  860          0975  2     RETURN;
;  861          0976  2
;  862          0977  1     END;


                                                .TITLE  DBGPRINT
                                                .IDENT  \V04-000\

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

                                45  53  41  42  00000 P.AAA:  .ASCII  \BASE\                      ;
                                        28  01  00004 P.AAB:  .ASCII  <1>\(\                    ;
                                    2E  2E  02  00006 P.AAC:  .ASCII  <2>\..\                   ;
```

```
                                        29  01  00009 P.AAD:   .ASCII  <1>\)\                          ;
                                                    0000B       .BLKB   1
                    00000009  00000006  00000004   0000C ADA_COMMON_CHAR:
                                                            .LONG   4, 6, 9                         ;
                                                    00018       .BLKB   4
                                        28  01  0001C P.AAE:   .ASCII  <1>\(\                         ;
                                        2C  01  0001E P.AAF:   .ASCII  <1>\,\                         ;
                                        29  01  00020 P.AAG:   .ASCII  <1>\)\                         ;
                                                    00022       .BLKB   2
                    00000020  0000001E  0000001C   00024 ADA_ARRAY_CHAR:
                                                            .LONG   28, 30, 32                      ;
                                                    00030       .BLKB   4
                              4C  4C  41  2E  04    00034 P.AAH:   .ASCII  <4>\.ALL\                ;
                                                    00039       .BLKB   3
                                            00#     0003C ADA_POINTER_CHAR:
                                                            .BYTE   0[4]                            ;
                                        00000034    00040       .LONG   52                          ;
                                                    00044       .BLKB   8
                                        2E  01  0004C P.AAI:   .ASCII  <1>\.\                         ;
                                                    0004E       .BLKB   2
                                            00#     00050 ADA_RECORD_CHAR:
                                                            .BYTE   0[4]                            ;
                                        0000004C    00054       .LONG   76                          ;
                                                    00058       .BLKB   8
    00000024  00000001  0000000C  00000000          00060 ADA_INFO_TABLE:
                                                            .LONG   0, 12, 1, 36                    ;
                                            00#     00070       .BYTE   0[32]                       ;
    00000050  00000003  0000003C  00000002          00090       .LONG   2, 60, 3, 80                ;
                                            00#     000A0       .BYTE   0[56]                       ;
                        0000003C  00000002          000D8       .LONG   2, 60                       ;
                                            00#     000E0       .BYTE   0[24]                       ;
                        00000050  00000007          000F8       .LONG   7, 80                       ;
                                                    00100       .BLKB   24
00000000  00000001  00000001  00000001  00000001   00000060  00118 P.AAJ:   .LONG   96, 1, 1, 1, 1, 0 ;
                                                      28  01  00130 P.AAK:   .ASCII  <1>\(\           ;
                                                      3A  01  00132 P.AAL:   .ASCII  <1>\:\           ;
                                                      29  01  00134 P.AAM:   .ASCII  <1>\)\           ;
                                                              00136       .BLKB   2
                                        00000134  00000132  00000130  00138 BASIC_COMMON_CHAR:
                                                                    .LONG   304, 306, 308           ;
                                                              00144       .BLKB   4
                                                      28  01  00148 P.AAN:   .ASCII  <1>\(\           ;
                                                      2C  01  0014A P.AAO:   .ASCII  <1>\,\           ;
                                                      29  01  0014C P.AAP:   .ASCII  <1>\)\           ;
                                                              0014E       .BLKB   2
                                        0000014C  0000014A  00000148  00150 BASIC_ARRAY_CHAR:
                                                                    .LONG   328, 330, 332           ;
                                                              0015C       .BLKB   4
                                                    3A  3A  02  00160 P.AAQ:   .ASCII  <2>\::\          ;
                                                              00163       .BLKB   1
                                                      00#     00164 BASIC_RECORD_CHAR:
                                                                    .BYTE   0[4]                    ;
                                                  00000160    00168       .LONG   352                 ;
                                                              0016C       .BLKB   8
                    00000150  00000001  00000138  00000000  00174 BASIC_INFO_TABLE:
                                                                    .LONG   0, 312, 1, 336          ;
                                                      00#     00184       .BYTE   0[40]               ;
```

```
                                                00000164  00000003  001AC          .LONG   3, 356                          ;
                                                      00#  001B4          .BYTE   0[88]                           ;
                                                00000164  00000007  0020C          .LONG   7, 356                          ;
                                                          00214          .BLKB   4
00000000  00000001  00000001  00000001  00000001  00000174  0022C P.AAR:  .LONG   372, 1, 1, 1, 1, 0                 ;
                                                    5B  01  00244 P.AAS:  .ASCII  <1>\[\                          ;
                                                    2C  01  00246 P.AAT:  .ASCII  <1>\,\                          ;
                                                    5D  01  00248 P.AAU:  .ASCII  <1>\]\                          ;
                                                          0024A          .BLKB   2
                                00000248  00000246  00000244  0024C BLISS_ARRAY_CHAR:
                                                                         .LONG   580, 582, 584                   ;
                                                          00258          .BLKB   4
                                                      00#  0025C BLISS_INFO_TABLE:
                                                                         .BYTE   0[8]                            ;
                                                0000024C  00000001  00264          .LONG   1, 588                          ;
                                                          0026C          .BLKB   168
00000000  00000000  00000001  00000001  00000001  0000025C  00314 P.AAV:  .LONG   604, 1, 1, 1, 0, 0                 ;
                                                    5B  01  0032C P.AAW:  .ASCII  <1>\[\                          ;
                                                    3A  01  0032E P.AAX:  .ASCII  <1>\:\                          ;
                                                    5D  01  00330 P.AAY:  .ASCII  <1>\]\                          ;
                                                          00332          .BLKB   2
                                00000330  0000032E  0000032C  00334 C_COMMON_CHAR:
                                                                         .LONG   812, 814, 816                   ;
                                                          00340          .BLKB   4
                                                        5B  01  00344 P.AAZ:  .ASCII  <1>\[\                          ;
                                                  5B    5D  02  00346 P.ABA:  .ASCII  <2>\][\                         ;
                                                    5D  01  00349 P.ABB:  .ASCII  <1>\]\                          ;
                                                          0034B          .BLKB   1
                                00000349  00000346  00000344  0034C C_ARRAY_CHAR:
                                                                         .LONG   836, 838, 841                   ;
                                                          00358          .BLKB   4
                                                      2A  01  0035C P.ABC:  .ASCII  <1>\*\                          ;
                                                3E    2D  02  0035E P.ABD:  .ASCII  <2>\->\                         ;
                                                          00361          .BLKB   3
                                                      00#  00364 C_POINTER_CHAR:
                                                                         .BYTE   0[4]                            ;
                                                00000035C  00368          .LONG   860                             ;
                                                      00#  0036C          .BYTE   0[4]                            ;
                                                0000035E  00370          .LONG   862                             ;
                                                    2E  01  00374 P.ABE:  .ASCII  <1>\.\                          ;
                                                          00376          .BLKB   2
                                                      00#  00378 C_RECORD_CHAR:
                                                                         .BYTE   0[4]                            ;
                                                00000374  0037C          .LONG   884                             ;
                                                          00380          .BLKB   8
                  0000034C  00000001  00000334  00000000  00388 C_INFO_TABLE:
                                                                         .LONG   0, 820, 1, 844                  ;
                                                      00#  00398          .BYTE   0[32]                           ;
                  00000378  00000006  00000364  00000005  003B8          .LONG   5, 868, 6, 888                  ;
                                                      00#  003C8          .BYTE   0[88]                           ;
                                                00000378  00000007  00420          .LONG   7, 888                          ;
                                                          00428          .BLKB   24
00000000  00000001  00000001  00000001  00000001  00000388  00440 P.ABF:  .LONG   904, 1, 1, 1, 1, 0                 ;
                                                    28  01  00458 P.ABG:  .ASCII  <1>\(\                          ;
                                                    3A  01  0045A P.ABH:  .ASCII  <1>\:\                          ;
                                                    29  01  0045C P.ABI:  .ASCII  <1>\)\                          ;
                                                          0045E          .BLKB   2
```

```
                                0000045C  0000045A  00000458  00460 COBOL_COMMON_CHAR:
                                                                  .LONG   1112, 1114, 1116
                                                          0046C   .BLKB   4
                                                    28 01 00470 P.ABJ:    .ASCII  <1>\(\
                                                    2C 01 00472 P.ABK:    .ASCII  <1>\,\
                                                    29 01 00474 P.ABL:    .ASCII  <1>\)\
                                                          00476   .BLKB   2
                                00000474  00000472  00000470  00478 COBOL_ARRAY_CHAR:
                                                                  .LONG   1136, 1138, 1140
                                                          00484   .BLKB   4
                                                 20 66 6F 20 04 00488 P.ABM:    .ASCII  <4>\ of \
                                                          0048D   .BLKB   3
                                                       00# 00490 COBOL_RECORD_CHAR:
                                                                  .BYTE   0[4]
                                                00000488  00494         .LONG   1160
                                                          00498   .BLKB   8
                        00000478  00000001  00000460  00000000  004A0 COBOL_INFO_TABLE:
                                                                  .LONG   0, 1120, 1, 1144
                                                       00# 004B0         .BYTE   0[40]
                                                00000490  00000004  004D8         .LONG   4, 1168
                                                       00# 004E0         .BYTE   0[88]
                                                00000490  00000007  00538         .LONG   7, 1168
                                                          00540   .BLKB   24
00000001  00000001  00000001  00000000  00000000  000004A0  00558 P.ABN:    .LONG   1184, 0, 0, 1, 1, 1
                                                    28 01 00570 P.ABO:    .ASCII  <1>\(\
                                                    3A 01 00572 P.ABP:    .ASCII  <1>\:\
                                                    29 01 00574 P.ABQ:    .ASCII  <1>\)\
                                                          00576   .BLKB   2
                                00000574  00000572  00000570  00578 FORTRAN_COMMON_CHAR:
                                                                  .LONG   1392, 1394, 1396
                                                          00584   .BLKB   4
                                                    28 01 00588 P.ABR:    .ASCII  <1>\(\
                                                    2C 01 0058A P.ABS:    .ASCII  <1>\,\
                                                    29 01 0058C P.ABT:    .ASCII  <1>\)\
                                                          0058E   .BLKB   2
                                0000058C  0000058A  00000588  00590 FORTRAN_ARRAY_CHAR:
                                                                  .LONG   1416, 1418, 1420
                                                          0059C   .BLKB   4
                                                    2E 01 005A0 P.ABU:    .ASCII  <1>\.\
                                                          005A2   .BLKB   2
                                                       00# 005A4 FORTRAN_RECORD_CHAR:
                                                                  .BYTE   0[4]
                                                000005A0  005A8         .LONG   1440
                                                          005AC   .BLKB   8
                        00000590  00000001  00000578  00000000  005B4 FORTRAN_INFO_TABLE:
                                                                  .LONG   0, 1400, 1, 1424
                                                       00# 005C4         .BYTE   0[40]
                                                000005A4  00000003  005EC         .LONG   3, 1444
                                                          005F4   .BLKB   120
00000000  00000001  00000001  00000001  00000001  000005B4  0066C P.ABV:    .LONG   1460, 1, 1, 1, 1, 0
                                                00000000# 00684 MACRO_INFO_TABLE:
                                                                  .LONG   0[22]
                                                          006DC   .BLKB   96
00000000  00000000  00000000  00000001  00000001  00000684  0073C P.ABW:    .LONG   1668, 1, 1, 0, 0, 0
                                                    5B 01 00754 P.ABX:    .ASCII  <1>\[\
                                                    3A 01 00756 P.ABY:    .ASCII  <1>\:\
                                                    5D 01 00758 P.ABZ:    .ASCII  <1>\]\
```

```
                                                  0075A                 .BLKB   2
                                00000758  00000756  00000754  0075C PASCAL_COMMON_CHAR:
                                                                      .LONG   1876, 1878, 1880
                                                  00768               .BLKB   4
                                                    5B  01    0076C P.ACA:  .ASCII  <1>\[\
                                                    2C  01    0076E P.ACB:  .ASCII  <1>\,\
                                                    5D  01    00770 P.ACC:  .ASCII  <1>\]\
                                                  00772               .BLKB   2
                                00000770  0000076E  0000076C  00774 PASCAL_ARRAY_CHAR:
                                                                      .LONG   1900, 1902, 1904
                                                  00780               .BLKB   4
                                                    5E  01    00784 P.ACD:  .ASCII  <1>\^\
                                                  00786               .BLKB   2
                                                      00#     00788 PASCAL_POINTER_CHAR:
                                                                      .BYTE   0[4]
                                          00000784            0078C       .LONG   1924
                                                  00790               .BLKB   8
                                                    2E  01    00798 P.ACE:  .ASCII  <1>\.\
                                                  0079A               .BLKB   2
                                                      00#     0079C PASCAL_RECORD_CHAR:
                                                                      .BYTE   0[4]
                                          00000798            007A0       .LONG   1944
                                                  007A4               .BLKB   8
                    00000774  00000001  0000075C  00000000    007AC PASCAL_INFO_TABLE:
                                                                      .LONG   0, 1884, 1, 1908
                                                      00#     007BC       .BYTE   0[32]
                    0000079C  00000003  00000788  00000002    007DC       .LONG   2, 1928, 3, 1948
                                                      00#     007EC       .BYTE   0[56]
                                        00000788  00000002    00824       .LONG   2, 1928
                                                      00#     0082C       .BYTE   0[24]
                                        0000079C  00000007    00844       .LONG   7, 1948
                                                  0084C               .BLKB   24
00000000  00000001  00000001  00000001  00000001  000007AC    00864 P.ACF:  .LONG   1964, 1, 1, 1, 1, 0
                                                    28  01    0087C P.ACG:  .ASCII  <1>\(\
                                                    3A  01    0087E P.ACH:  .ASCII  <1>\:\
                                                    29  01    00880 P.ACI:  .ASCII  <1>\)\
                                                  00882               .BLKB   2
                                00000880  0000087E  0000087C  00884 PLI_COMMON_CHAR:
                                                                      .LONG   2172, 2174, 2176
                                                  00890               .BLKB   4
                                                    28  01    00894 P.ACJ:  .ASCII  <1>\(\
                                                    2C  01    00896 P.ACK:  .ASCII  <1>\,\
                                                    29  01    00898 P.ACL:  .ASCII  <1>\)\
                                                  0089A               .BLKB   2
                                00000898  00000896  00000894  0089C PLI_ARRAY_CHAR:
                                                                      .LONG   2196, 2198, 2200
                                                  008A8               .BLKB   4
                                                  3E  2D  02  008AC P.ACM:  .ASCII  <2>\->\
                                                  008AF               .BLKB   1
                                                      00#     008B0 PLI_POINTER_CHAR:
                                                                      .BYTE   0[4]
                                          000008AC            008B4       .LONG   2220
                                                  008B8               .BLKB   8
                                                    2E  01    008C0 P.ACN:  .ASCII  <1>\.\
                                                  008C2               .BLKB   2
                                                      00#     008C4 PLI_RECORD_CHAR:
                                                                      .BYTE   0[4]
```

```
                                                000008C0  008C8           .LONG   2240                                    ;
                                                          008CC           .BLKB   8
                        0000089C  00000001  00000884  00000000  008D4 PLI_INFO_TABLE:
                                                          008E4           .LONG   0, 2180, 1, 2204                        ;
                                                    00#   008E4           .BYTE   0[40]
                                  000008C4  00000006    0090C           .LONG   6, 2244
                                                    00#   00914           .BYTE   0[64]
                                  000008B0  00000002    00954           .LONG   2, 2224
                                                    00#   0095C           .BYTE   0[16]
                                  000008C4  00000007    0096C           .LONG   7, 2244
                                                          00974           .BLKB   24
00000000  00000001  00000001  00000001  00000001  000008D4  0098C P.ACO:   .LONG   2260, 1, 1, 1, 1, 0                  ;
                                                    28  01  009A4 P.ACP:   .ASCII  <1>\(\
                                                    3A  01  009A6 P.ACQ:   .ASCII  <1>\:\
                                                    29  01  009A8 P.ACR:   .ASCII  <1>\)\
                                                          009AA           .BLKB   2
                                  000009A8  000009A6  000009A4  009AC RPG_COMMON_CHAR:
                                                          009B8           .LONG   2468, 2470, 2472                        ;
                                                          009B8           .BLKB   4
                                                    28  01  009BC P.ACS:   .ASCII  <1>\(\                               ;
                                                    2C  01  009BE P.ACT:   .ASCII  <1>\,\                               ;
                                                    29  01  009C0 P.ACU:   .ASCII  <1>\)\                               ;
                                                          009C2           .BLKB   2
                                  000009C0  000009BE  000009BC  009C4 RPG_ARRAY_CHAR:
                                                          009D0           .LONG   2492, 2494, 2496                        ;
                                                          009D0           .BLKB   4
                        000009C4  00000001  000009AC  00000000  009D4 RPG_INFO_TABLE:
                                                          009E4           .LONG   0, 2476, 1, 2500                        ;
                                                          009E4           .BLKB   168
00000000  00000000  00000001  00000000  00000000  000009D4  00A8C P.ACV:   .LONG   2516, 0, 0, 1, 0, 0                  ;
                                                    28  01  00AA4 P.ACW:   .ASCII  <1>\(\                               ;
                                                    2C  01  00AA6 P.ACX:   .ASCII  <1>\,\                               ;
                                                    29  01  00AA8 P.ACY:   .ASCII  <1>\)\                               ;
                                                          00AAA           .BLKB   2
                                  00000AA8  00000AA6  00000AA4  00AAC UNKNOWN_ARRAY_CHAR:
                                                          00AB8           .LONG   2724, 2726, 2728                        ;
                                                          00AB8           .BLKB   4
                                                    2E  01  00ABC P.ACZ:   .ASCII  <1>\.\                               ;
                                                          00ABE           .BLKB   2
                                                    00#   00AC0 UNKNOWN_RECORD_CHAR:
                                                          00AC0           .BYTE   0[4]
                                                00000ABC  00AC4           .LONG   2748                                    ;
                                                          00AC8           .BLKB   8
                                                    5E  01  00AD0 P.ADA:   .ASCII  <1>\^\                               ;
                                                          00AD2           .BLKB   2
                                                    00#   00AD4 UNKNOWN_POINTER_CHAR:
                                                          00AD4           .BYTE   0[4]
                                                00000AD0  00AD8           .LONG   2768                                    ;
                                                          00ADC           .BLKB   8
                                                    00#   00AE4 UNKNOWN_INFO_TABLE:
                                                          00AE4           .BYTE   0[8]
                                  00000AAC  00000001    00AEC           .LONG   1, 2732                                 ;
                                                    00#   00AF4           .BYTE   0[32]
                        00000AC0  00000003  00000AD4  00000002    00B14           .LONG   2, 2772, 3, 2752                ;
                                                    00#   00B24           .BYTE   0[56]
                                  00000AD4  00000002    00B5C           .LONG   2, 2772                                 ;
                                                    00#   00B64           .BYTE   0[24]
```

```
                                                00000AC0  00000007  00B7C          .LONG   7, 2752
                                                          00B84      .BLKB   24
00000000  00000001  00000001  00000001  00000001  00000AE4  00B9C P.ADB:  .LONG   2788, 1, 1, 1, 1, 0
0000098C  0000022C  00000558  00000314  0000066C  0000073C  00BB4 LANGUAGE_PRINT_TABLE_PTRS:
                                                          .LONG   1852, 1644, 788, 1368, 556, 2444, 2148, -
          00000B9C  00000118  00000A8C  00000440  00000864  00BCC                 1088, 2700, 280, 2972

                                                          .PSECT  DBG$OWN,NOEXE,  PIC,2

                                                00000000  00000084  00000 BUFFER_DESCR:
                                                                    .LONG   132, 0
                                                          00000000  00008 MAX_BRK_ON_BLANKS:
                                                                    .LONG   0
                                                          00000000  0000C MAX_INDENT:
                                                                    .LONG   0
                                                          00000083  00010 MAX_PBSIZE:
                                                                    .LONG   131
                                                          00000000  00014 PBSIZE: .LONG   0
                                                                    00018 PRINT_BUFFER:
                                                                    .BLKB   132
                                                          00000000  0009C PRT_CONTINUE:
                                                                    .LONG   0
                                                          00000000  000A0 PRT_INDENT:
                                                                    .LONG   0
                                                          00000000  000A4 PRTSET_CONTINUE:
                                                                    .LONG   0
                                                          00000000  000A8 PRTSET_MARGIN:
                                                                    .LONG   0
                                                          00000000  000AC PRTBRK_ON_BLANKS:
                                                                    .LONG   0

                                                          .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

                                                00000 DBG$GL_ARRSUB_FLAG::
                                                          .BLKB   4
                                                00004 DBG$GL_RECCMP_FLAG::
                                                          .BLKB   4
                                                00008 DBG$GL_SIGN_FLAG::
                                                          .BLKB   4

                TABLEBASE=          P.AAA
                ADA_PRINT_TABLE=    P.AAJ
                BASIC_PRINT_TABLE=  P.AAR
                BLISS_PRINT_TABLE=  P.AAV
                C_PRINT_TABLE=      P.ABF
                COBOL_PRINT_TABLE=  P.ABN
                FORTRAN_PRINT_TABLE=P.ABV
                MACRO_PRINT_TABLE=  P.ABW
                PASCAL_PRINT_TABLE= P.ACF
                PLI_PRINT_TABLE=    P.ACO
                RPG_PRINT_TABLE=    P.ACV
                UNKNOWN_PRINT_TABLE=P.ADB
                        .EXTRN  DBG$COLLECT, DBG$FILL_IN_VMS_DESC
                        .EXTRN  DBG$GET_DST_NAME
                        .EXTRN  DBG$GET_MEMORY, DBG$GET_SET_TYPEID
                        .EXTRN  DBG$GET_TEMPMEM
                        .EXTRN  DBG$MAKE_SKELETON_DESC
```

```
                                        .EXTRN  DBG$NEW_SYMBOLIZE
                                        .EXTRN  DBG$NGET_RADIX, DBG$NPATHDESC_TO_CS
                                        .EXTRN  DBG$PRIM_TO_VAL
                                        .EXTRN  DBG$PRINT_VALUE
                                        .EXTRN  DBG$PRINT_VALUE_AS_INTEGER
                                        .EXTRN  DBG$REL_MEMORY, DBG$SCR_WRITE_LINE
                                        .EXTRN  DBG$STA_ADDRESS_TO_REGDESCR
                                        .EXTRN  DBG$STA_REGISTER_NAME
                                        .EXTRN  DBG$SYMID_TO_PRIMARY
                                        .EXTRN  DBG$STA_SYMNAME
                                        .EXTRN  DBG$STA_SYMPATHNAME
                                        .EXTRN  DBG$STA_TYP_ATOMIC
                                        .EXTRN  DBG$STA_TYP_ENUM
                                        .EXTRN  DBG$STA_TYP_SET
                                        .EXTRN  DBG$STA_TYP_SUBRNG
                                        .EXTRN  SYS$FAO, SYS$FAOL
                                        .EXTRN  DBG$GL_CURRENT_PRIMARY
                                        .EXTRN  DBG$GB_VERB, DBG$GL_CMND_RADIX
                                        .EXTRN  DBG$GB_RADIX, DBG$GL_OUTPRAB
                                        .EXTRN  DBG$GB_DEF_OUT, DBG$GB_LANGUAGE
                                        .EXTRN  DBG$GB_MOD_PTR, DBG$GL_LOG_BUF
                                        .EXTRN  DBG$GL_LOGRAB, DBG$GL_SCREEN_OUTPUT
                                        .EXTRN  DBG$SRC_TERM_WIDTH

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

                                0000 00000      .ENTRY  DBG$FAO_OUT, Save nothing                       ; 0931
                          5E    FF68   CE  9E 00002   MOVAB   -152(SP), SP                              ;
                    0C    AE      04   BC  9A 00007   MOVZBL  @STRING, INP_DESC                         ; 0969
              10    AE      04   AC      01  C1 0000C   ADDL3   #1, STRING, INP_DESC+4                   ; 0970
                    04    AE      83   8F  9A 00012   MOVZBL  #131, OUT_DESC                            ; 0971
                    08    AE      14   AE  9E 00017   MOVAB   BUFFER, OUT_DESC+4                        ; 0972
                                  08   AC  9F 0001C   PUSHAB  ARGUMENTS                                 ; 0973
                                  08   AE  9F 0001F   PUSHAB  OUT_DESC                                  ;
                                  08   AE  9F 00022   PUSHAB  LENGTH                                    ;
                                  18   AE  9F 00025   PUSHAB  INP_DESC                                  ;
                       00000000G  00       04  FB 00028   CALLS   #4, SYS$FAOL                        ;
                                  14   AE  9F 0002F   PUSHAB  BUFFER                                   ; 0974
                          7E      04   AE  3C 00032   MOVZWL  LENGTH, -(SP)                           ;
                         0000V    CF       02  FB 00036   CALLS   #2, DBG$WRITE_OUTPUT                ;
                                           04  00038   RET                                            ; 0977

; Routine Size:  60 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  864          0978  1 GLOBAL ROUTINE DBG$FORMAT_FAO_OUT(BUF_DESC, STRING, ARGUMENTS) : NOVALUE =
;  865          0979  1 !
;  866          0980  1 ! FUNCTION
;  867          0981  1 !       This routine formats an output string using SYS$FAOL.
;  868          0982  1 !
;  869          0983  1 ! INPUTS
;  870          0984  1 !       BUF_DESC - Output buffer String Descriptor.
;  871          0985  1 !
;  872          0986  1 !       STRING - The address of a Counted ASCII FAO control string.
;  873          0987  1 !
;  874          0988  1 !       ARGUMENTS - Zero or more arguments to be applied to the FAO control
;  875          0989  1 !                   string.
;  876          0990  1 !
;  877          0991  1 ! OUTPUTS
;  878          0992  1 !       NONE
;  879          0993  1 !
;  880          0994  1 !
;  881          0995  2     BEGIN
;  882          0996  2
;  883          0997  2     MAP
;  884          0998  2         BUF_DESC: REF VECTOR[,LONG],    ! Output buffer string descriptor
;  885          0999  2         STRING: REF VECTOR[,BYTE];      ! Address of FAO control string
;  886          1000  2
;  887          1001  2     LOCAL
;  888          1002  2         INP_DESC: VECTOR[2,LONG],       ! String descriptor for input buffer
;  889          1003  2         OUT_DESC: VECTOR[2,LONG],       ! String descriptor for output buffer
;  890          1004  2         OUTLEN: WORD,                   ! Length of SYS$FAOL output string
;  891          1005  2         STATUS;                         ! Status code returned by SYS$FAOL
;  892          1006  2
;  893          1007  2
;  894          1008  2
;  895          1009  2     ! Set up the descriptors needed for the SYS$FAOL call.
;  896          1010  2     !
;  897          1011  2     INP_DESC[0] = .STRING[0];
;  898          1012  2     INP_DESC[1] = STRING[1];
;  899          1013  2     OUT_DESC[0] = .BUF_DESC[0];
;  900          1014  2     OUT_DESC[1] = .BUF_DESC[1];
;  901          1015  2     STATUS = SYS$FAOL(INP_DESC, OUTLEN, OUT_DESC, ARGUMENTS);
;  902          1016  2
;  903          1017  2
;  904          1018  2     ! Update the output string length and pointer.  Then return.
;  905          1019  2     !
;  906          1020  2     BUF_DESC[0] = .OUT_DESC[0] - .OUTLEN;
;  907          1021  2     BUF_DESC[1] = .OUT_DESC[1] + .OUTLEN;
;  908          1022  2     RETURN;
;  909          1023  2
;  910          1024  1     END;


                                        0004 00000          .ENTRY  DBG$FORMAT_FAO_OUT, Save R2                     ; 0978
                                    5E      14   C2 00002      SUBL2   #20, SP                                         ;
                                0C  AE  08  BC   9A 00005      MOVZBL  @STRING, INP_DESC                               ; 1011
                            10  AE  08  AC      01   C1 0000A  ADDL3   #1, STRING, INP_DESC+4                          ; 1012
                                    52  04  AC  D0 00010       MOVL    BUF_DESC, R2                                    ; 1013
```

```
                    04   AE          62  7D 00014         MOVQ    (R2), OUT_DESC                  ;
                                  0C    AC  9F 00018      PUSHAB  ARGUMENTS                       ; 1015
                                  08    AE  9F 0001B      PUSHAB  OUT_DESC                        ;
                                  08    AE  9F 0001E      PUSHAB  OUTLEN                          ;
                                  18    AE  9F 00021      PUSHAB  INP_DESC                        ;
                00000000G         00      04  FB 00024   CALLS   #4, SYS$FAOL                   ;
                                  50      6E  3C 0002B    MOVZWL  OUTLEN, R0                      ; 1020
              62        04   AE          50  C3 0002E    SUBL3   R0, OUT_DESC, (R2)              ;
                                  50      6E  3C 00033    MOVZWL  OUTLEN, R0                      ; 1021
                    04   A2        08 BE40  9E 00036     MOVAB   @OUT_DESC+4[R0], 4(R2)          ;
                                           04 0003C     RET                                     ; 1024

; Routine Size:  61 bytes,    Routine Base:  DBG$CODE + 003C
```

```
;  912          1025  1 GLOBAL ROUTINE DBG$FLUSHBUF: NOVALUE =
;  913          1026  1 !
;  914          1027  1 ! FUNCTION
;  915          1028  1 !       This routine flushes the current print line as built up with calls
;  916          1029  1 !       on DBG$PRINT and initializes the print buffer to start a new line.
;  917          1030  1 !       This is accomplished by setting PBSIZE (the print buffer size) to
;  918          1031  1 !       zero so that a new empty print line is initialized.
;  919          1032  1 !
;  920          1033  1 ! INPUTS
;  921          1034  1 !       NONE
;  922          1035  1 !
;  923          1036  1 ! OUTPUTS
;  924          1037  1 !       NONE
;  925          1038  1 !
;  926          1039  1
;  927          1040  2     BEGIN
;  928          1041  2
;  929          1042  2
;  930          1043  2
;  931          1044  2     ! Flush the current contents of PRINT_BUFFER by resetting PBSIZE to zero.
;  932          1045  2     !
;  933          1046  2     PBSIZE = 0;
;  934          1047  2     RETURN;
;  935          1048  2
;  936          1049  1     END;


                                0000 00000          .ENTRY  DBG$FLUSHBUF, Save nothing      ; 1025
                00000000'   EF   D4 00002          CLRL    PBSIZE                          ; 1044
                                 04 00008          RET                                     ; 1049

; Routine Size:  9 bytes,    Routine Base:  DBG$CODE + 0079
```

```
;  938          1050  1 GLOBAL ROUTINE DBG$LANGUAGE_FORMAT(VALPTR) =
;  939          1051  1 !
;  940          1052  1 ! FUNCTION
;  941          1053  1 !       This routine prints the value in a given Value Descriptor according to
;  942          1054  1 !       a language-specific formatting rule if such a rule exists.  Otherwise
;  943          1055  1 !       the value is not printed and the routine returns FALSE to indicate this.
;  944          1056  1 !
;  945          1057  1 ! INPUTS
;  946          1058  1 !       VALPTR  - Pointer to value descriptor containing the value to be
;  947          1059  1 !                 printed in a language-specific format.
;  948          1060  1 !
;  949          1061  1 ! OUTPUTS
;  950          1062  1 !       This routine returns TRUE if the value was printed according to a
;  951          1063  1 !                 language-specific formatting rule.  It returns FALSE if
;  952          1064  1 !                 there is no language-specific formatting rule to print it;
;  953          1065  1 !                 in this case the value was not printed.
;  954          1066  1 !
;  955          1067  1
;  956          1068  2     BEGIN
;  957          1069  2
;  958          1070  2     MAP
;  959          1071  2         VALPTR: REF DBG$VALDESC;        ! Pointer to value descriptor
;  960          1072  2
;  961          1073  2
;  962          1074  2
;  963          1075  2     ! If the kind is not data, then do not use language-specific rules.
;  964          1076  2     ! Note - components of records as in A.B come back with kind TYPCOMP,
;  965          1077  2     ! and this is also data.
;  966          1078  2     !
;  967          1079  2     IF (.VALPTR [DBG$B_DHDR_KIND] NEQ RST$K_DATA) AND
;  968          1080  3        (.VALPTR [DBG$B_DHDR_KIND] NEQ RST$K_TYPCOMP)
;  969          1081  2     THEN
;  970          1082  2         RETURN FALSE;
;  971          1083  2
;  972          1084  2
;  973          1085  2     ! Case on the FCODE to select the approriate language-specific formatting
;  974          1086  2     ! rule.
;  975          1087  2     !
;  976          1088  2     CASE .VALPTR[DBG$B_DHDR_FCODE] FROM RST$K_TYPE_MINIMUM
;  977          1089  2                                    TO   RST$K_TYPE_MAXIMUM OF
;  978          1090  2         SET
;  979          1091  2
;  980          1092  2         ! Atomic data items, and items described by descriptor.
;  981          1093  2         !
;  982          1094  2         [RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR]:
;  983          1095  2
;  984          1096  2                 ! Case on language.
;  985          1097  2                 !
;  986          1098  2                 CASE .DBG$GB_LANGUAGE FROM DBG$K_MIN_LANGUAGE
;  987          1099  2                                           TO DBG$K_MAX_LANGUAGE OF
;  988          1100  2                     SET
;  989          1101  2
;  990          1102  2                     ! For language PL/I we want to display bitstrings
;  991          1103  2                     ! in the format '101011100'B, for example.
;  992          1104  2                     !
;  993          1105  2                     [DBG$K_PLI]:
;  994          1106  2                         IF (.VALPTR[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU) OR
```

```
;  995          1107  2                (.VALPTR[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V) OR
;  996          1108  2                (.VALPTR[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_TF)
;  997          1109  2            THEN
;  998          1110  2                BEGIN
;  999          1111  2                LOCAL
; 1000          1112  2                    PTR: REF BITVECTOR[],
; 1001          1113  2                    START;
; 1002          1114  2                DBG$PRINT(UPLIT BYTE(%ASCIC '''));
; 1003          1115  2                PTR = .VALPTR[DBG$L_VALUE_POINTER];
; 1004          1116  2                IF .VALPTR[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS
; 1005          1117  2                THEN
; 1006          1118  2                    START = .VALPTR[DBG$L_VALUE_POS]
; 1007          1119  2                ELSE
; 1008          1120  2                    START = 0;
; 1009          1121  2                INCR I FROM .START
; 1010          1122  2                       TO .START+.VALPTR[DBG$W_VALUE_LENGTH]-1 DO
; 1011          1123  2                    IF .PTR[.I]
; 1012          1124  2                    THEN
; 1013          1125  2                        DBG$PRINT(UPLIT BYTE(%ASCIC '1'))
; 1014          1126  2                    ELSE
; 1015          1127  2                        DBG$PRINT(UPLIT BYTE(%ASCIC '0'));
; 1016          1128  2                DBG$PRINT(UPLIT BYTE(%ASCIC '''B'));
; 1017          1129  2                RETURN TRUE;
; 1018          1130  2                END;
; 1019          1131  2
; 1020          1132  2            ! Other languages - no special rules.
; 1021          1133  2            !
; 1022          1134  2            [INRANGE]: 0;
; 1023          1135  2
; 1024          1136  2            ! We do not expect any other language codes here,
; 1025          1137  2            ! so report an error if we see one.
; 1026          1138  2            !
; 1027          1139  2            [OUTRANGE]:
; 1028          1140  2                $DBG_ERROR('DBGPRINT\DBG$LANGUAGE_FORMAT');
; 1029          1141  2
; 1030          1142  2            TES;
; 1031          1143  2
; 1032          1144  2
; 1033          1145  2        ! Other FCODEs - just fall through and return FALSE.
; 1034          1146  2        !
; 1035          1147  2        [INRANGE]: 0;
; 1036          1148  2
; 1037          1149  2
; 1038          1150  2        ! If kind is data, FCODE should be in the range of the
; 1039          1151  2        ! CASE statement.
; 1040          1152  2        !
; 1041          1153  2        [OUTRANGE]:
; 1042          1154  2            $DBG_ERROR('DBGPRINT\DBG$LANGUAGE_FORMAT');
; 1043          1155  2
; 1044          1156  2        TES;
; 1045          1157  2
; 1046          1158  2
; 1047          1159  2    ! No language-specific formatting rule exists for this language and FCODE.
; 1048          1160  2    ! We therefore return FALSE to indicate the value was not formatted at all.
; 1049          1161  2    !
; 1050          1162  2    RETURN FALSE;
; 1051          1163  2
```

```
; 1052          1164  1      END;


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

                                           27  01  00BE0 P.ADC:  .ASCII  <1>\'\                                      ;
                                           31  01  00BE2 P.ADD:  .ASCII  <1>\1\                                      ;
                                           30  01  00BE4 P.ADE:  .ASCII  <1>\0\                                      ;
                                       42  27  02  00BE6 P.ADF:  .ASCII  <2>\'B\                                     ;
4C  24  47  42  44  5C  54  4E  49  52  50  47  42  44  1C  00BE9 P.ADG:  .ASCII  <28>\DBGPRINT\<92>\DBG$LANGUAGE_FORMAT\  ;
    54  41  4D  52  4F  46  5F  45  47  41  55  47  4E  41  00BF8                                                        ;
4C  24  47  42  44  5C  54  4E  49  52  50  47  42  44  1C  00C06 P.ADH:  .ASCII  <28>\DBGPRINT\<92>\DBG$LANGUAGE_FORMAT\  ;
    54  41  4D  52  4F  46  5F  45  47  41  55  47  4E  41  00C15                                                        ;


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

                                        007C  00000        .ENTRY  DBG$LANGUAGE_FORMAT, Save R2,R3,R4,R5,R6      ; 1050
                56       0000V   CF   9E  00002        MOVAB   DBG$PRINT, R6                                ;
                55   00000000'   EF   9E  00007        MOVAB   P.ADH, R5                                    ;
                52          04   AC   D0  0000E        MOVL    VALPTR, R2                                   ; 1079
                06          07   A2   91  00012        CMPB    7(R2), #6                                    ;
                                 06   13  00016        BEQL    1$                                           ;
                0A          07   A2   91  00018        CMPB    7(R2), #10                                   ; 1080
                                 77   12  0001C        BNEQ    7$                                           ;
        15      01          06   A2   8F  0001E 1$:    CASEB   6(R2), #1, #21                               ; 1088
00B7    0030    0030           00B7       00023 2$:    .WORD   15$-2$,-                                     ;
00B7    00B7    00B7           00B7       0002B                3$-2$,-                                      ;
00B7    00B7    00B7           00B7       00033                3$-2$,-                                      ;
00B7    00B7    00B7           00B7       0003B                15$-2$,-                                     ;
00B7    00B7    00B7           00B7       00043                15$-2$,-                                     ;
                00B7           00B7       0004B                15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$,-                                     ;
                                                               15$-2$                                       ;
                                 55   DD  0004F        PUSHL   R5                                           ; 1154
                                 21   11  00051        BRB     5$                                           ;
        0A      00   00000000G   00   8F  00053 3$:    CASEB   DBG$GB_LANGUAGE, #0, #10                     ; 1098
007F    007F    007F           007F       0005B 4$:    .WORD   15$-4$,-                                     ;
007F    007F    002A           007F       00063                15$-4$,-                                     ;
        007F    007F           007F       0006B                15$-4$,-                                     ;
                                                               15$-4$,-                                     ;
```

```
                                                                  15$-4$,-                                   ;
                                                                  6$-4$,-                                    ;
                                                                  15$-4$,-                                   ;
                                                                  15$-4$,-                                   ;
                                                                  15$-4$,-                                   ;
                                                                  15$-4$,-                                   ;
                                                                  15$-4$                                     ;
                              E3   A5  9F 00071          PUSHAB   P.ADG                                      ; 1140
                                   01  DD 00074  5$:     PUSHL    #1                                         ;
                      00028362     8F  DD 00076          PUSHL    #164706                                    ;
         00000000G  00             03  FB 0007C          CALLS    #3, LIB$SIGNAL                             ;
                                   55  11 00083          BRB      15$                                        ;
                    22        16   A2  91 00085  6$:     CMPB     22(R2), #34                                ; 1106
                                   0C  13 00089          BEQL     8$                                         ;
                    01        16   A2  91 0008B          CMPB     22(R2), #1                                 ; 1107
                                   06  13 0008F          BEQL     8$                                         ;
                    28        16   A2  91 00091          CMPB     22(R2), #40                                ; 1108
                                   43  12 00095  7$:     BNEQ     15$                                        ;
                              DA   A5  9F 00097  8$:     PUSHAB   P.ADC                                      ; 1114
                    66             01  FB 0009A          CALLS    #1, DBG$PRINT                              ;
                    54        18   A2  D0 0009D          MOVL     24(R2), PTR                                ; 1115
                    0D        17   A2  91 000A1          CMPB     23(R2), #13                                ; 1116
                                   06  12 000A5          BNEQ     9$                                         ;
                    50        1C   A2  D0 000A7          MOVL     28(R2), START                              ; 1118
                                   02  11 000AB          BRB      10$                                        ;
                                   50  D4 000AD  9$:     CLRL     START                                      ; 1120
                    51        14   A2  3C 000AF  10$:    MOVZWL   20(R2), R1                                 ; 1122
         53         50             51  C1 000B3          ADDL3    R1, START, R3                              ;
                    52        FF   A0  9E 000B7          MOVAB    -1(R0), I                                  ; 1123
                                   0F  11 000BB          BRB      14$                                        ;
         05         64             52  E1 000BD  11$:    BBC      I, (PTR), 12$                              ;
                              DC   A5  9F 000C1          PUSHAB   P.ADD                                      ; 1125
                                   03  11 000C4          BRB      13$                                        ;
                              DE   A5  9F 000C6  12$:    PUSHAB   P.ADE                                      ; 1127
                    66             01  FB 000C9  13$:    CALLS    #1, DBG$PRINT                              ;
         ED         52             53  F2 000CC  14$:    AOBLSS   R3, I, 11$                                 ; 1123
                              E0   A5  9F 000D0          PUSHAB   P.ADF                                      ; 1128
                    66             01  FB 000D3          CALLS    #1, DBG$PRINT                              ;
                    50             01  D0 000D6          MOVL     #1, R0                                     ; 1129
                                       04 000D9          RET                                                 ;
                                   50  D4 000DA  15$:    CLRL     R0                                         ; 1164
                                       04 000DC          RET                                                 ;

; Routine Size:  221 bytes,    Routine Base:  DBG$CODE + 0082
```

```
;  1054          1165  1 GLOBAL ROUTINE DBG$NEWLINE: NOVALUE =                                                                    ;
;  1055          1166  1 !
;  1056          1167  1 ! FUNCTION
;  1057          1168  1 !       This routine prints the current print line as built up with calls
;  1058          1169  1 !       on DBG$PRINT and initializes the print buffer to start a new line.
;  1059          1170  1 !       This is accomplished by calling DBG$WRITE_OUTPUT to output the current
;  1060          1171  1 !       contents of PRINT_BUFFER and then resetting PBSIZE (the print buffer
;  1061          1172  1 !       size) to zero so that a new empty print line is initialized.
;  1062          1173  1 !
;  1063          1174  1 ! INPUTS
;  1064          1175  1 !       NONE
;  1065          1176  1 !
;  1066          1177  1 ! OUTPUTS
;  1067          1178  1 !       NONE
;  1068          1179  1 !
;  1069          1180  1
;  1070          1181  2     BEGIN
;  1071          1182  2
;  1072          1183  2     LOCAL
;  1073          1184  2         LENGTH;                             ! Length of output buffer
;  1074          1185  2
;  1075          1186  2
;  1076          1187  2     ! Print the current contents of PRINT_BUFFER and reset its size to zero.
;  1077          1188  2     !
;  1078          1189  2     LENGTH = .PBSIZE;
;  1079          1190  2     PBSIZE = 0;
;  1080          1191  2     PRT_CONTINUE = FALSE;
;  1081          1192  2     DBG$WRITE_OUTPUT(.LENGTH, PRINT_BUFFER);
;  1082          1193  2     RETURN;
;  1083          1194  2
;  1084          1195  1     END;

                                    0004 00000          .ENTRY  DBG$NEWLINE, Save R2                                    ;  1165
                          52 00000000'  EF   9E 00002   MOVAB   PBSIZE, R2                                              ;
                          50            62   D0 00009   MOVL    PBSIZE, LENGTH                                        ;  1189
                                        62   D4 0000C   CLRL    PBSIZE                                                ;  1190
                                  0088  C2   D4 0000E   CLRL    PRT_CONTINUE                                          ;  1191
                                    04  A2   9F 00012   PUSHAB  PRINT_BUFFER                                          ;  1192
                                        50   DD 00015   PUSHL   LENGTH                                                ;
                        0000V   CF      02   FB 00017   CALLS   #2, DBG$WRITE_OUTPUT                                  ;
                                        04   04 0001C   RET                                                           ;  1195

; Routine Size:  29 bytes,    Routine Base:  DBG$CODE + 015F
```

```
;  1086          1196  1 GLOBAL ROUTINE DBG$PRINT(FAOSTRING, ARGUMENTS): NOVALUE =
;  1087          1197  1 !
;  1088          1198  1 !  FUNCTION
;  1089          1199  1 !        This routine is the DEBUG print formatting routine.  It accepts an
;  1090          1200  1 !        FAO (Formatted ASCII Output--see the System Services Manual) control
;  1091          1201  1 !        string which specifies how the output formatting is to be done and
;  1092          1202  1 !        it accepts zero or more additional arguments as called for in the
;  1093          1203  1 !        FAO control string.  It calls SYS$FAOL to format the text and then
;  1094          1204  1 !        moves the formatted text to the DEBUG print buffer (PRINT_BUFFER).
;  1095          1205  1 !
;  1096          1206  1 !        Note, however, that the formatted output is not actually printed
;  1097          1207  1 !        unless the print buffer overflows.  To force the current print buffer
;  1098          1208  1 !        to be printed, one must call the DBG$NEWLINE routine.  DBG$PRINT can
;  1099          1209  1 !        thus be called repeatedly to build up the print buffer one piece at a
;  1100          1210  1 !        time, after which DBG$NEWLINE is called to actually output the now
;  1101          1211  1 !        completed print line.
;  1102          1212  1 !
;  1103          1213  1 !        If a DBG$PRINT call overflows the print buffer, the filled buffer is
;  1104          1214  1 !        output to the current output device (or screen display) and the over-
;  1105          1215  1 !        flow text is filled into the beginning of the buffer.  The effect is
;  1106          1216  1 !        that oversized print lines are folded on the user's output device.
;  1107          1217  1 !
;  1108          1218  1 !        Before this routine is called, you can call DBG$PRINT_CONTROL to set
;  1109          1219  1 !        up the proper indentation specifications.
;  1110          1220  1 !
;  1111          1221  1 !  INPUTS
;  1112          1222  1 !        FAOSTRING - The address of an FAO control string represented as
;  1113          1223  1 !                    Counted ASCII.  This string contains text and FAO format-
;  1114          1224  1 !                    ting directives of the form "!xx" which define how the
;  1115          1225  1 !                    output text should be formatted.
;  1116          1226  1 !
;  1117          1227  1 !        ARGUMENTS - Zero or more FAO arguments.  These arguments must corre-
;  1118          1228  1 !                    spond in kind and number to the formatting directives in
;  1119          1229  1 !                    the FAO control string.
;  1120          1230  1 !
;  1121          1231  1 !  OUTPUTS
;  1122          1232  1 !        NONE
;  1123          1233  1 !
;  1124          1234  1
;  1125          1235  2     BEGIN
;  1126          1236  2
;  1127          1237  2     MAP
;  1128          1238  2         FAOSTRING: REF VECTOR[,BYTE];    ! FAO control string
;  1129          1239  2
;  1130          1240  2     LOCAL
;  1131          1241  2         BUFFER_PTR: REF VECTOR[,BYTE],   ! Pointer to FAO formatting buffer
;  1132          1242  2         FAOSTRING_DESC: VECTOR[2,LONG],  ! FAO control string descriptor
;  1133          1243  2         FOUND_BLANK_FLAG,                ! Flag set when blank is found in back-
;  1134          1244  2                                          !      ward scan for blank to break on
;  1135          1245  2         LENGTH,                          ! Length of string copied at one time
;  1136          1246  2                                          !      to PRINT_BUFFER from temp buffer
;  1137          1247  2         MAX_LINE_LENGTH,                 ! Maximum print buffer size
;  1138          1248  2         SAVEBUF: VECTOR[132, BYTE],      ! Save buffer for text after the last
;  1139          1249  2                                          !      blank if text is broken on blank
;  1140          1250  2         SAVEBUF_LEN,                     ! The current length of SAVEBUF text
;  1141          1251  2         STATUS,                          ! Status code returned by SYS$FAOL
;  1142          1252  2         STR_SIZE: WORD;                  ! String size returned by SYS$FAOL
```

```
;  1143          1253  2
;  1144          1254  2
;  1145          1255  2
;  1146          1256  2      ! Call SYS$FAOL to format the desired ASCII string.  If the output buffer
;  1147          1257  2      ! we give to SYS$FAOL is too small, we increase its size and loop until it
;  1148          1258  2      ! is large enough to contain the output string.
;  1149          1259  2      !
;  1150          1260  2      FAOSTRING_DESC[0] = .FAOSTRING[0];
;  1151          1261  2      FAOSTRING_DESC[1] = FAOSTRING[1];
;  1152          1262  2      WHILE TRUE DO
;  1153          1263  3          BEGIN
;  1154          1264  3          IF .BUFFER_DESCR[1] EQL 0
;  1155          1265  3          THEN
;  1156          1266  3              BUFFER_DESCR[1] = DBG$GET_MEMORY(.BUFFER_DESCR[0]/%UPVAL);
;  1157          1267  3
;  1158          1268  3          STATUS = SYS$FAOL(FAOSTRING_DESC, STR_SIZE, BUFFER_DESCR, ARGUMENTS);
;  1159          1269  3          IF .STATUS EQL SS$_NORMAL THEN EXITLOOP;
;  1160          1270  3          IF .STATUS NEQ SS$_BUFFEROVF
;  1161          1271  3          THEN
;  1162          1272  3              $DBG_ERROR('DBGPRINT\DBG$PRINT 10 ', .STATUS);
;  1163          1273  3
;  1164          1274  3
;  1165          1275  3          ! Make sure the buffer size won't get over 2100 bytes.  (This allows
;  1166          1276  3          ! strings up to 2000 bytes plus a symbol name.)
;  1167          1277  3          !
;  1168          1278  3          IF .BUFFER_DESCR[0] GTR 2100
;  1169          1279  3          THEN
;  1170          1280  4              BEGIN
;  1171          1281  4              BUFFER_DESCR[0] = 2100;
;  1172          1282  4              SIGNAL(DBG$_STGTRUNC);
;  1173          1283  4              EXITLOOP;
;  1174          1284  3              END;
;  1175          1285  3
;  1176          1286  3          DBG$REL_MEMORY(.BUFFER_DESCR[1]);
;  1177          1287  3          BUFFER_DESCR[1] = 0;
;  1178          1288  3          BUFFER_DESCR[0] = .BUFFER_DESCR[0] + 100;
;  1179          1289  2          END;
;  1180          1290  2
;  1181          1291  2
;  1182          1292  2      ! Move the newly formatted text to PRINT_BUFFER.  If the new text would
;  1183          1293  2      ! overflow PRINT_BUFFER, move over as much as will fit, output the buffer
;  1184          1294  2      ! by calling DBG$WRITE_OUTPUT, and loop to move the remaining text into
;  1185          1295  2      ! the print buffer.
;  1186          1296  2      !
;  1187          1297  2      MAX_LINE_LENGTH = MIN(.DBG$SRC_TERM_WIDTH, .MAX_PBSIZE);
;  1188          1298  2      BUFFER_PTR = .BUFFER_DESCR[1];
;  1189          1299  2      WHILE TRUE DO
;  1190          1300  3          BEGIN
;  1191          1301  3
;  1192          1302  3
;  1193          1303  3          ! If the print buffer is full, output its contents before we get more
;  1194          1304  3          ! text from the FAO formatting buffer.
;  1195          1305  3          !
;  1196          1306  3          IF .PBSIZE GEQ .MAX_LINE_LENGTH
;  1197          1307  3          THEN
;  1198          1308  4              BEGIN
;  1199          1309  4
```

```
;  1200     1310  4
;  1201     1311  4                ! If the Break-on-Blanks flag is set, we try to break the line on
;  1202     1312  4                ! the last blanks before the end of the line.  This prevents num-
;  1203     1313  4                ! bers and names from being broken in the middle, if possible.
;  1204     1314  4                !
;  1205     1315  4                SAVEBUF_LEN = 0;
;  1206     1316  4                IF .PRTBRK_ON_BLANKS
;  1207     1317  4                THEN
;  1208     1318  5                    BEGIN
;  1209     1319  5
;  1210     1320  5
;  1211     1321  5                    ! Loop backward over the print buffer to locate the last blank
;  1212     1322  5                    ! in the line and the last non-blank before that.
;  1213     1323  5                    !
;  1214     1324  5                    FOUND_BLANK_FLAG = FALSE;
;  1215     1325  5                    DECR I FROM .PBSIZE - 1 TO 0 DO
;  1216     1326  6                        BEGIN
;  1217     1327  6
;  1218     1328  6
;  1219     1329  6                        ! If we have already found a blank in the scan, set PBSIZE
;  1220     1330  6                        ! to include all text up to the current character.  If the
;  1221     1331  6                        ! current character is non-blank, stop the backward scan.
;  1222     1332  6                        !
;  1223     1333  6                        IF .FOUND_BLANK_FLAG
;  1224     1334  6                        THEN
;  1225     1335  7                            BEGIN
;  1226     1336  7                            PBSIZE = .I + 1;
;  1227     1337  7                            IF .PRINT_BUFFER[.I] NEQ ' ' THEN EXITLOOP;
;  1228     1338  7                            END
;  1229     1339  7
;  1230     1340  7
;  1231     1341  7                        ! We have not yet found a blank.  If this character is the
;  1232     1342  7                        ! first blank, save all text after it in SAVEBUF and set
;  1233     1343  7                        ! the found-a-blank flag.
;  1234     1344  7                        !
;  1235     1345  6                        ELSE IF .PRINT_BUFFER[.I] EQL ' '
;  1236     1346  6                        THEN
;  1237     1347  7                            BEGIN
;  1238     1348  7                            SAVEBUF_LEN = .PBSIZE - .I - 1;
;  1239     1349  7                            CH$MOVE(.SAVEBUF_LEN, PRINT_BUFFER[.I + 1], SAVEBUF[0]);
;  1240     1350  7                            FOUND_BLANK_FLAG = TRUE;
;  1241     1351  7                            END
;  1242     1352  7
;  1243     1353  7
;  1244     1354  7                        ! If we have not yet found a blank but are already below
;  1245     1355  7                        ! the maximum backward scan index, exit the loop without
;  1246     1356  7                        ! having found a place to break on.  In this case, the
;  1247     1357  7                        ! line is not broken on a blank.
;  1248     1358  7                        !
;  1249     1359  6                        ELSE IF .I LSS .MAX_BRK_ON_BLANKS
;  1250     1360  6                        THEN
;  1251     1361  6                            EXITLOOP;
;  1252     1362  6
;  1253     1363  5                    END;                ! End of loop to break line on blanks
;  1254     1364  5
;  1255     1365  4                END;                    ! End of PRTBRK_ON_BLANKS If statement
;  1256     1366  4
```

```
;  1257     1367  4
;  1258     1368  4              ! Now output the formatted print line to the output device.
;  1259     1369  4              !
;  1260     1370  4              DBG$WRITE_OUTPUT(.PBSIZE, PRINT_BUFFER);
;  1261     1371  4
;  1262     1372  4
;  1263     1373  4              ! Indent the print buffer for the next line of text.  Also copy
;  1264     1374  4              ! over any text that remained from the previous line if that line
;  1265     1375  4              ! was broken on the last blank.
;  1266     1376  4              !
;  1267     1377  4              PRT_CONTINUE = TRUE;
;  1268     1378  4              PBSIZE = .PRT_INDENT + .PRTSET_CONTINUE;
;  1269     1379  4              CH$FILL(' ', .PBSIZE, PRINT_BUFFER[0]);
;  1270     1380  4              IF .SAVEBUF_LEN GTR 0
;  1271     1381  4              THEN
;  1272     1382  5                  BEGIN
;  1273     1383  5                  CH$MOVE(.SAVEBUF_LEN, SAVEBUF[0], PRINT_BUFFER[.PBSIZE]);
;  1274     1384  5                  PBSIZE = .PBSIZE + .SAVEBUF_LEN;
;  1275     1385  4                  END;
;  1276     1386  4
;  1277     1387  3              END;                          ! End of code to output print buffer
;  1278     1388  3
;  1279     1389  3
;  1280     1390  3          ! If this is a new text line, indent the line by the current indenta-
;  1281     1391  3          ! tion amount.  Note that a continuation line can be indented more.
;  1282     1392  3          !
;  1283     1393  3          IF .PBSIZE EQL 0
;  1284     1394  3          THEN
;  1285     1395  4              BEGIN
;  1286     1396  4              PBSIZE = .PRT_INDENT;
;  1287     1397  4              IF .PRT_CONTINUE THEN PBSIZE = .PBSIZE + .PRTSET_CONTINUE;
;  1288     1398  4              CH$FILL(' ', .PBSIZE, PRINT_BUFFER[0]);
;  1289     1399  3              END;
;  1290     1400  3
;  1291     1401  3
;  1292     1402  3          ! Move the next section of text from the FAO buffer into the print
;  1293     1403  3          ! buffer and adjust the print buffer size accordingly.  As this is
;  1294     1404  3          ! done, all tabs are expanded to blanks to produce a completely
;  1295     1405  3          ! de-tabbed print buffer.
;  1296     1406  3          !
;  1297     1407  3          INCR I FROM 0 TO .STR_SIZE - 1 DO
;  1298     1408  4              BEGIN
;  1299     1409  4              STR_SIZE = .STR_SIZE - 1;
;  1300     1410  4
;  1301     1411  4
;  1302     1412  4              ! If the current character is a tab, we expand it to blanks.
;  1303     1413  4              !
;  1304     1414  4              IF .BUFFER_PTR[0] EQL DBG$K_TAB
;  1305     1415  4              THEN
;  1306     1416  5                  BEGIN
;  1307     1417  5                  LENGTH = MIN(8 - (.PBSIZE MOD 8), .MAX_LINE_LENGTH - .PBSIZE);
;  1308     1418  5                  CH$FILL(' ', .LENGTH, PRINT_BUFFER[.PBSIZE]);
;  1309     1419  5                  PBSIZE = .PBSIZE + .LENGTH;
;  1310     1420  5                  END
;  1311     1421  5
;  1312     1422  5
;  1313     1423  5              ! And if it is not a tab, we simply copy it over.
```

```
;  1314          1424  5              !
;  1315          1425  4              ELSE
;  1316          1426  5                  BEGIN
;  1317          1427  5                  PRINT_BUFFER[.PBSIZE] = .BUFFER_PTR[0];
;  1318          1428  5                  PBSIZE = .PBSIZE + 1;
;  1319          1429  4                  END;
;  1320          1430  4
;  1321          1431  4
;  1322          1432  4              ! Increment the FAO formatting buffer pointer by one.  Then, if we
;  1323          1433  4              ! are about to overflow the print buffer, we exit the copy loop.
;  1324          1434  4              ! Otherwise, we loop for the next character.
;  1325          1435  4              !
;  1326          1436  4              BUFFER_PTR = .BUFFER_PTR + 1;
;  1327          1437  4              IF .PBSIZE GEQ .MAX_LINE_LENGTH THEN EXITLOOP;
;  1328          1438  4
;  1329          1439  3              END;                            ! End of copy and de-tab loop
;  1330          1440  3
;  1331          1441  3
;  1332          1442  3          ! See how much input text we have left.  If there is none, exit this
;  1333          1443  3          ! print loop.  Otherwise, we loop to move the text that remains in the
;  1334          1444  3          ! FAO formatting buffer into the print buffer.
;  1335          1445  3          !
;  1336          1446  3          IF .STR_SIZE EQL 0 THEN EXITLOOP;
;  1337          1447  3
;  1338          1448  2          END;                            ! End of WHILE loop over print lines
;  1339          1449  2
;  1340          1450  2
;  1341          1451  2      ! The formatted text is now in PRINT_BUFFER.  Return control.
;  1342          1452  2      !
;  1343          1453  2      RETURN;
;  1344          1454  2
;  1345          1455  1      END;


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

50 24 47 42 44 5C 54 4E 49 52 50 47 42 44 16 00C23 P.ADI:  .ASCII  <22>\DBGPRINT\<92>\DBG$PRINT 10 \                ;
                     20 30 31 20 54 4E 49 52 00C32                                                               ;


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

                                                0FFC 00000      .ENTRY  DBG$PRINT, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 1196
                                                                R10,R11
                                          5E    FF68   CE  9E 00002      MOVAB   -152(SP), SP                        ;
                                    F8    AD     04   BC  9A 00007      MOVZBL  @FAOSTRING, FAOSTRING_DESC          ; 1260
                              FC    AD     04   AC      01  C1 0000C      ADDL3   #1, FAOSTRING, FAOSTRING_DESC+4     ; 1261
                                          00000000' EF  D5 00012 1$:     TSTL    BUFFER_DESCR+4                      ; 1264
                                                    16  12 00018      BNEQ    2$                                  ;
                                    7E 00000000'  EF      04  C7 0001A      DIVL3   #4, BUFFER_DESCR, -(SP)             ; 1266
                                       00000000G  00      01  FB 00022      CALLS   #1, DBG$GET_MEMORY                  ;
                                       00000000'  EF      50  D0 00029      MOVL    R0, BUFFER_DESCR+4                  ;
                                                 08   AC  9F 00030 2$:     PUSHAB  ARGUMENTS                           ; 1268
                                          00000000' EF  9F 00033      PUSHAB  BUFFER_DESCR                        ;
                                                10   AE  9F 00039      PUSHAB  STR_SIZE                            ;
```

```
                        F8    AD  9F 0003C          PUSHAB  FAOSTRING_DESC                                    ;
00000000G   00              04  FB 0003F          CALLS   #4, SYS$FAOL                                      ;
            52              50  D0 00046          MOVL    R0, STATUS                                        ;
            01              52  D1 00049          CMPL    STATUS, #1                                        ; 1269
                            66  13 0004C          BEQL    5$                                                ;
00000601    8F              52  D1 0004E          CMPL    STATUS, #1537                                     ; 1270
                            17  13 00055          BEQL    3$                                                ;
                            52  DD 00057          PUSHL   STATUS                                            ; 1272
                00000000'   EF  9F 00059          PUSHAB  P.ADI                                             ;
                            01  DD 0005F          PUSHL   #1                                                ;
                00028362    8F  DD 00061          PUSHL   #164706                                           ;
00000000G   00              04  FB 00067          CALLS   #4, LIB$SIGNAL                                    ;
00000834    8F  00000000'   EF  D1 0006E 3$:      CMPL    BUFFER_DESCR, #2100                               ; 1278
                            18  15 00079          BLEQ    4$                                                ;
00000000'   EF      0834    8F  3C 0007B          MOVZWL  #2100, BUFFER_DESCR                               ; 1281
                0002804B    8F  DD 00084          PUSHL   #163915                                           ; 1282
00000000G   00              01  FB 0008A          CALLS   #1, LIB$SIGNAL                                    ;
                            21  11 00091          BRB     5$                                                ; 1280
                00000000'   EF  DD 00093 4$:      PUSHL   BUFFER_DESCR+4                                    ; 1286
00000000G   00              01  FB 00099          CALLS   #1, DBG$REL_MEMORY                                ;
                00000000'   EF  D4 000A0          CLRL    BUFFER_DESCR+4                                    ; 1287
00000000'   EF  00000064    8F  C0 000A6          ADDL2   #100, BUFFER_DESCR                                ; 1288
                        FF5E    31 000B1          BRW     1$                                                ; 1262
            50  00000000G     00  D0 000B4 5$:      MOVL    DBG$SRC_TERM_WIDTH, R0                            ; 1297
00000000'   EF              50  D1 000BB          CMPL    R0, MAX_PBSIZE                                    ;
                            07  15 000C2          BLEQ    6$                                                ;
            50  00000000'   EF  D0 000C4          MOVL    MAX_PBSIZE, R0                                    ;
            57              50  D0 000CB 6$:      MOVL    R0, MAX_LINE_LENGTH                               ;
      04    AE  00000000'   EF  D0 000CE          MOVL    BUFFER_DESCR+4, BUFFER_PTR                        ; 1298
            5A        08    AE  3C 000D6          MOVZWL  STR_SIZE, R10                                     ; 1407
            57  00000000'   EF  D1 000DA 7$:      CMPL    PBSIZE, MAX_LINE_LENGTH                           ; 1306
                            03  18 000E1          BGEQ    8$                                                ;
                        00AE    31 000E3          BRW     14$                                               ;
                            56  D4 000E6 8$:      CLRL    SAVEBUF_LEN                                       ; 1315
                53  00000000'   EF  E9 000E8          BLBC    PRTBRK_ON_BLANKS, 13$                         ; 1316
                            6E  D4 000EF          CLRL    FOUND_BLANK_FLAG                                  ; 1324
                58  00000000'   EF  D0 000F1          MOVL    PBSIZE, I                                     ; 1325
                            45  11 000F8          BRB     12$                                               ;
                            14      6E  E9 000FA 9$:      BLBC    FOUND_BLANK_FLAG, 10$                     ; 1333
            00000000'   EF      01  A8  9E 000FD          MOVAB   1(R8), PBSIZE                             ; 1336
                20  00000000'EF48  91  00105          CMPB    PRINT_BUFFER[I], #32                          ;  337
                            30  13 0010D          BEQL    12$                                               ;
                            31  11 0010F          BRB     13$                                               ;
                20  00000000'EF48  91  00111 10$:      CMPB    PRINT_BUFFER[I], #32                          ; 1345
                            1B  12 00119          BNEQ    11$                                               ;
        50  00000000'   EF          58  C3 0011B          SUBL3   I, PBSIZE, R0                             ; 1348
                            56      FF  A0  9E 00123          MOVAB   -1(R0), SAVEBUF_LEN                       ;
  0C    AE  00000000'EF48           56  28 00127          MOVC3   SAVEBUF_LEN, PRINT_BUFFER+1[I], SAVEBUF   ; 1349
                            6E      01  D0 00131          MOVL    #1, FOUND_BLANK_FLAG                      ; 1350
                            09  11 00134          BRB     12$                                               ; 1345
            00000000'   EF          58  D1 00136 11$:      CMPL    I, MAX_BRK_ON_BLANKS                      ; 1359
                            03  19 0013D          BLSS    13$                                               ;
                B8          58  F4 0013F 12$:      SOBGEQ  I, 9$                                             ; 1325
                    00000000'   EF  9F 00142 13$:      PUSHAB  PRINT_BUFFER                              ; 1370
                    00000000'   EF  DD 00148          PUSHL   PBSIZE                                    ;
              0000V CF              02  FB 0014E          CALLS   #2, DBG$WRITE_OUTPUT                      ;
            00000000'   EF          01  D0 00153          MOVL    #1, PRT_CONTINUE                          ; 1377
```

```
                00000000'  EF  00000000'  EF  00000000'  EF  C1 0015A          ADDL3   PRTSET_CONTINUE, PRT_INDENT, PBSIZE          ; 1378
00000000'  EF              20              6E              00  2C 0016A          MOVC5   #0, (SP), #32, PBSIZE, PRINT_BUFFER         ; 1379
                                               00000000'  EF      00173
                                                          56  D5 00178          TSTL    SAVEBUF_LEN                                 ; 1380
                                                          18  15 0017A          BLEQ    14$
                                           50  00000000'  EF  9E 0017C          MOVAB   PRINT_BUFFER, R0                            ; 1383
                    00000000'FF40              0C       AE                      56  28 00183          MOVC3   SAVEBUF_LEN, SAVEBUF, @PBSIZE[R0]
                                    00000000'  EF                              56  C0 0018D          ADDL2   SAVEBUF_LEN, PBSIZE         ; 1384
                                               00000000'  EF  D5 00194  14$:   TSTL    PBSIZE                                      ; 1393
                                                          2B  12 0019A          BNEQ    16$
                                    00000000'  EF  00000000'  EF  D0 0019C          MOVL    PRT_INDENT, PBSIZE                  ; 1396
                                               0B  00000000'  EF  E9 001A7          BLBC    PRT_CONTINUE, 15$                   ; 1397
                                    00000000'  EF  00000000'  EF  C0 001AE          ADDL2   PRTSET_CONTINUE, PBSIZE
00000000'  EF              20              6E              00  2C 001B9  15$:   MOVC5   #0, (SP), #32, PBSIZE, PRINT_BUFFER         ; 1398
                                               00000000'  EF      001C2
                                           58  00000000'  EF  D0 001C7  16$:   MOVL    PBSIZE, R8                                  ; 1417
                                           59                  01  CE 001CE          MNEGL   #1, I
                                                               58  11 001D1          BRB     21$
                                                       08      AE  B7 001D3  17$:   DECW    STR_SIZE                            ; 1409
                                               09      04      BE  91 001D6          CMPB    @BUFFER_PTR, #9                     ; 1414
                                                               31  12 001DA          BNEQ    19$
                    7E              00              58              01  7A 001DC          EMUL    #1, R8, #0, -(SP)                   ; 1417
                    50              50              8E              08  7B 001E1          EDIV    #8, (SP)+, R0, R0
                                    50              08              50  C3 001E6          SUBL3   R0, #8, R0
                                    51              57              58  C3 001EA          SUBL3   R8, MAX_LINE_LENGTH, R1
                                                    51              50  D1 001EE          CMPL    R0, R1
                                                                    03  15 001F1          BLEQ    18$
                                                    50              51  D0 001F3          MOVL    R1, R0
                                                    5B              50  D0 001F6  18$:   MOVL    R0, LENGTH
                        5B          20              6E              00  2C 001F9          MOVC5   #0, (SP), #32, LENGTH, PRINT_BUFFER[R8]  ; 1418
                                                       00000000'EF48      001FE
                                    00000000'  EF                      5B  C0 00204          ADDL2   LENGTH, PBSIZE                  ; 1419
                                                                       0F  11 0020B          BRB     20$                             ; 1414
                                    00000000'EF48              04      BE  90 0020D  19$:   MOVB    @BUFFER_PTR, PRINT_BUFFER[R8]   ; 1427
                                                       00000000'  EF      D6 00216          INCL    PBSIZE                          ; 1428
                                                                  04      AE  D6 0021C  20$:   INCL    BUFFER_PTR                      ; 1436
                                                    58  00000000'  EF      D0 0021F          MOVL    PBSIZE, R8                      ; 1437
                                                    57                  58  D1 00226          CMPL    R8, MAX_LINE_LENGTH
                                                                        04  18 00229          BGEQ    22$
                                    A4              59                  5A  F2 0022B  21$:   AOBLSS  R10, I, 17$                     ; 1407
                                                    5A              08      AE  3C 0022F  22$:   MOVZWL  STR_SIZE, R10               ; 1446
                                                                            03  13 00233          BEQL    23$
                                                                        FEA2    31 00235          BRW     7$
                                                                                04 00238  23$:   RET                             ; 145C

; Routine Size:  569 bytes,    Routine Base:  DBG$CODE + 017C
```

```
;  1347          1456  1 GLOBAL ROUTINE DBG$PRINT_CONTROL =
;  1348          1457  1 !
;  1349          1458  1 ! FUNCTION
;  1350          1459  1 !       This routine sets up the indentation levels for the print buffer used
;  1351          1460  1 !       in DBG$PRINT, so that when text overflows the print buffer, the
;  1352          1461  1 !       remaining text can be indented according to the given specifications.
;  1353          1462  1 !
;  1354          1463  1 ! INPUTS
;  1355          1464  1 !       Can be any numbers of input parameters, separated by commas.  The first
;  1356          1465  1 !       parameter has to be one of the following Print Control Codes:
;  1357          1466  1 !       DBG$K_PRTSET_LMARGIN, DBG$K_PRTSET_RLMARGIN, DBG$K_PRTBRK_ON_BLANKS,
;  1358          1467  1 !       DBG$K_PRTSET_CONTINUE, and DBG$K_PRT_RESET.
;  1359          1468  1 !
;  1360          1469  1 !       Valid parameters are:
;  1361          1470  1 !          DBG$K_PRTSET_LMARGIN followed by a value (0, or +)
;  1362          1471  1 !          DBG$K_PRTSET_RLMARGIN followed by a value (0, +, or -)
;  1363          1472  1 !          DBG$K_PRTBRK_ON_BLANKS
;  1364          1473  1 !          DBG$K_PRTSET_CONTINUE followed by a value (0 or +)
;  1365          1474  1 !          DBG$K_PRT_RESET
;  1366          1475  1 !
;  1367          1476  1 !       Default for above parameters are:
;  1368          1477  1 !          DBG$K_PRESET_LMARGIN                 PRTSET_LMARGIN = 0
;  1369          1478  1 !          DBG$K_PRESET_RLMARGIN                PRTSET_RLMARGIN = 0
;  1370          1479  1 !          DBG$K_PRTSET_ON_BLANKS               PRTSET_ON_BLANKS = FALSE
;  1371          1480  1 !          DBG$K_PRTSET_CONTINUE                PRTSET_CONTINUE = 0
;  1372          1481  1 !
;  1373          1482  1 ! OUTPUTS
;  1374          1483  1 !       Return TRUE or FALSE status code to the caller.  TRUE means the
;  1375          1484  1 !                 given indentation is properly set.  FALSE means the
;  1376          1485  1 !                 previous indentation remains set.
;  1377          1486  1 !
;  1378          1487  1
;  1379          1488  2     BEGIN
;  1380          1489  2
;  1381          1490  2     BUILTIN
;  1382          1491  2         ACTUALCOUNT,                     ! The number of actual parameter
;  1383          1492  2         ACTUALPARAMETER;                 ! Select the N-th actual parameter
;  1384          1493  2
;  1385          1494  2     LOCAL
;  1386          1495  2         BRK_ON_BLANKS,                   ! Flag set to break on blanks at the end
;  1387          1496  2         CONTINUE,                        ! Indentation for continuation line
;  1388          1497  2         MARGIN,                          ! Left margin for print buffer
;  1389          1498  2         PARAMCNT;                        ! The number of input parameters
;  1390          1499  2
;  1391          1500  2
;  1392          1501  2     ! Initialization.
;  1393          1502  2     !
;  1394          1503  2     PARAMCNT = 0;
;  1395          1504  2     MARGIN = .PRTSET_MARGIN;
;  1396          1505  2     CONTINUE = .PRTSET_CONTINUE;
;  1397          1506  2     BRK_ON_BLANKS = .PRTBRK_ON_BLANKS;
;  1398          1507  2
;  1399          1508  2
;  1400          1509  2     ! Loop through the input parameters.  Perform the proper operation
;  1401          1510  2     ! for each input parameter according to the Print Control Code.
;  1402          1511  2     !
;  1403          1512  2     WHILE .PARAMCNT LSS ACTUALCOUNT() DO
```

```
;  1404          1513  3          BEGIN
;  1405          1514  3          PARAMCNT = .PARAMCNT + 1;
;  1406          1515  3
;  1407          1516  3
;  1408          1517  3          ! Get the input parameter.  Case on the Print Control Code and perform
;  1409          1518  3          ! the proper print control operation.
;  1410          1519  3          !
;  1411          1520  3          CASE ACTUALPARAMETER(.PARAMCNT) FROM DBG$K_PRTSET_LMARGIN
;  1412          1521  3                                                  TO DBG$K_PRT_RESET OF
;  1413          1522  3              SET
;  1414          1523  3
;  1415          1524  3
;  1416          1525  3              ! Set the left margin.  We need to get one more input parameter
;  1417          1526  3              ! to set the left margin value.
;  1418          1527  3              !
;  1419          1528  3              [DBG$K_PRTSET_LMARGIN]:
;  1420          1529  4                  BEGIN
;  1421          1530  4                  PARAMCNT = .PARAMCNT + 1;
;  1422          1531  4                  IF .PARAMCNT GTR ACTUALCOUNT()
;  1423          1532  4                  THEN
;  1424          1533  4                      $DBG_ERROR('DBGPRINT\DBG$PRINT_CONTROL, err. 10');
;  1425          1534  4
;  1426          1535  4                  MARGIN = ACTUALPARAMETER(.PARAMCNT);
;  1427          1536  3                  END;
;  1428          1537  3
;  1429          1538  3
;  1430          1539  3              ! Set margin relative to current setting.  We need to get one more
;  1431          1540  3              ! input parameter to set the relative margin value.
;  1432          1541  3              !
;  1433          1542  3              [DBG$K_PRTSET_RLMARGIN]:
;  1434          1543  4                  BEGIN
;  1435          1544  4                  PARAMCNT = .PARAMCNT + 1;
;  1436          1545  4                  IF .PARAMCNT GTR ACTUALCOUNT()
;  1437          1546  4                  THEN
;  1438          1547  4                      $DBG_ERROR('DBGPRINT\DBG$PRINT_CONTROL, err. 10');
;  1439          1548  4
;  1440          1549  4                  MARGIN = .MARGIN + ACTUALPARAMETER(.PARAMCNT);
;  1441          1550  3                  END;
;  1442          1551  3
;  1443          1552  3
;  1444          1553  3              ! Set Flag to indicate to break on blanks at the end of the print
;  1445          1554  3              ! line.
;  1446          1555  3              !
;  1447          1556  3              [DBG$K_PRTBRK_ON_BLANKS]:
;  1448          1557  3                  BRK_ON_BLANKS = TRUE;
;  1449          1558  3
;  1450          1559  3
;  1451          1560  3              ! Set more indentation margin for continued print line.  We need to
;  1452          1561  3              ! have one more input parameter for this Print Control Code.
;  1453          1562  3              !
;  1454          1563  3              [DBG$K_PRTSET_CONTINUE]:
;  1455          1564  4                  BEGIN
;  1456          1565  4                  PARAMCNT = .PARAMCNT + 1;
;  1457          1566  4                  IF .PARAMCNT GTR ACTUALCOUNT()
;  1458          1567  4                  THEN
;  1459          1568  4                      $DBG_ERROR('DBGPRINT\DBG$PRINT_CONTROL, err. 10');
;  1460          1569  4
```

```
;  1461     1570  4                CONTINUE = ACTUALPARAMETER(.PARAMCNT);
;  1462     1571  4                END;
;  1463     1572  4
;  1464     1573  4
;  1465     1574  4            ! Reset all the indentations.
;  1466     1575  4            !
;  1467     1576  4            [DBG$K_PRT_RESET]:
;  1468     1577  4                BEGIN
;  1469     1578  4                MARGIN = 0;
;  1470     1579  4                CONTINUE = 0;
;  1471     1580  4                BRK_ON_BLANKS = FALSE;
;  1472     1581  4                END;
;  1473     1582  4
;  1474     1583  4
;  1475     1584  4            ! We do not recognize this Print Control Code.
;  1476     1585  4            !
;  1477     1586  4            [INRANGE, OUTRANGE]:
;  1478     1587  4                $DBG_ERROR('DBGPRINT\DBG$PRINT_CONTROL, err. 20');
;  1479     1588  4
;  1480     1589  4            TES;
;  1481     1590  4
;  1482     1591  4
;  1483     1592  4        ! We have got an apparently valid parameter. Check that the
;  1484     1593  4        ! indentation limits are valid.
;  1485     1594  4        !
;  1486     1595  4        IF (.CONTINUE LSS 0) OR (.MARGIN LSS 0) OR
;  1487     1596  4          ((.MARGIN + .CONTINUE) GEQ MIN(.DBG$SRC_TERM_WIDTH,.MAX_PBSIZE))
;  1488     1597  4        THEN
;  1489     1598  4            RETURN FALSE;
;  1490     1599  3
;  1491     1600  2            END;                            ! End of WHILE loop over the parameters
;  1492     1601  2
;  1493     1602  2
;  1494     1603  2        ! All the parameters were valid.  Set the permanent print control values.
;  1495     1604  2        !
;  1496     1605  2        PRTSET_MARGIN = .MARGIN;
;  1497     1606  2        PRTSET_CONTINUE = .CONTINUE;
;  1498     1607  2        PRTBRK_ON_BLANKS = .BRK_ON_BLANKS;
;  1499     1608  2
;  1500     1609  2
;  1501     1610  2        ! Calculate the maximum indentation size.  If the new margin is greater
;  1502     1611  2        ! than this size, use this size instead.
;  1503     1612  2        !
;  1504     1613  2        MAX_INDENT = MIN(.DBG$SRC_TERM_WIDTH, .MAX_PBSIZE) / 2;
;  1505     1614  2        PRT_INDENT = MIN(.MAX_INDENT, .PRTSET_MARGIN);
;  1506     1615  2
;  1507     1616  2
;  1508     1617  2        ! Calculate the maximum break on blanks size.  We use this value to stop
;  1509     1618  2        ! the search for blanks when we later search for blanks backwards from
;  1510     1619  2        ! end of the print buffer.  Then return.
;  1511     1620  2        !
;  1512     1621  2        MAX_BRK_ON_BLANKS = .PRT_INDENT + (.DBG$SRC_TERM_WIDTH - .PRT_INDENT) / 2;
;  1513     1622  2        RETURN TRUE;
;  1514     1623  2
;  1515     1624  1        END;
```

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

50 24 47 42 44 5C 54 4E 49 52 50 47 42 44 23 00C3A P.ADJ:  .ASCII  \#DBGPRINT\<92>\DBG$PRINT_CONTROL, err. \
65 20 2C 4C 4F 52 54 4E 4F 43 5F 54 4E 49 52 00C49
                                              20 2E 72 72 00C58
                                                 30 31 00C5C        .ASCII  \10\
50 24 47 42 44 5C 54 4E 49 52 50 47 42 44 23 00C5E P.ADK:  .ASCII  \#DBGPRINT\<92>\DBG$PRINT_CONTROL, err. \
65 20 2C 4C 4F 52 54 4E 4F 43 5F 54 4E 49 52 00C6D
                                              20 2E 72 72 00C7C
                                                 30 31 00C80        .ASCII  \10\
50 24 47 42 44 5C 54 4E 49 52 50 47 42 44 23 00C82 P.ADL:  .ASCII  \#DBGPRINT\<92>\DBG$PRINT_CONTROL, err. \
65 20 2C 4C 4F 52 54 4E 4F 43 5F 54 4E 49 52 00C91
                                              20 2E 72 72 00CA0
                                                 30 31 00CA4        .ASCII  \10\
50 24 47 42 44 5C 54 4E 49 52 50 47 42 44 23 00CA6 P.ADM:  .ASCII  \#DBGPRINT\<92>\DBG$PRINT_CONTROL, err. \
65 20 2C 4C 4F 52 54 4E 4F 43 5F 54 4E 49 52 00CB5
                                              20 2E 72 72 00CC4
                                                 30 32 00CC8        .ASCII  \20\


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

                                                 03FC 00000         .ENTRY  DBG$PRINT_CONTROL, Save R2,R3,R4,R5,R6,R7,- ; 1456
                                                                                R8,R9
                                59 00000000G   00  9E 00002        MOVAB   DBG$SRC_TERM_WIDTH, R9
                                58 00000000G   00  9E 00009        MOVAB   LIB$SIGNAL, R8
                                57 00000000'   EF  9E 00010        MOVAB   P.ADM, R7
                                56 00000000'   EF  9E 00017        MOVAB   PRTSET_MARGIN, R6
                                                 53  D4 0001E        CLRL    PARAMCNT                                    ; 1503
                                                 52  66 00020        MOVL    PRTSET_MARGIN, MARGIN                       ; 1504
                                                 54  FC A6 00023    MOVL    PRTSET_CONTINUE, CONTINUE                   ; 1505
                                                 55  04 A6 00027    MOVL    PRTBRK_ON_BLANKS, BRK_ON_BLANKS             ; 1506
                                53              6C  08  00 ED 0002B 1$:     CMPZV   #0, #8, (AP), PARAMCNT              ; 1512
                                                        03 14 00030 2$:     BGTR    2$
                                                     00A3 31 00032          BRW     14$
                                                        53 D6 00035         INCL    PARAMCNT                            ; 1514
                                                04      01  6C43 CF 00037  CASEL   (AP)[PARAMCNT], #1, #4              ; 1520
                                0058            0053    0036    0019 0003C 3$:     .WORD   4$-3$,-
                                                                0075 00044         6$-3$,-
                                                                                   8$-3$,-
                                                                                   9$-3$,-
                                                                                   11$-3$
                                                        57 DD 00046         PUSHL   R7                                  ; 1587
                                                        01 DD 00048         PUSHL   #1
                                                00028362   8F DD 0004A      PUSHL   #164706
                                                    68     03 FB 00050      CALLS   #3, LIB$SIGNAL
                                                        60 11 00053         BRB     12$
                                                        53 D6 00055 4$:     INCL    PARAMCNT                            ; 1530
                                53              6C  08  00 ED 00057        CMPZV   #0, #8, (AP), PARAMCNT              ; 1531
                                                        0E 18 0005C        BGEQ    5$
                                                    94     A7 9F 0005E     PUSHAB  P.ADJ                               ; 1533
                                                        01 DD 00061        PUSHL   #1
                                                00028362   8F DD 00063     PUSHL   #164706
                                                    68     03 FB 00069     CALLS   #3, LIB$SIGNAL
                                                    52      6C43 D0 0006C 5$:  MOVL    (AP)[PARAMCNT], MARGIN          ; 1535
```

```
                                    43  11 00070          BRB     12$                                     ; 1520
                                    53  D6 00072 6$:      INCL    PARAMCNT                                ; 1544
        53      6C          08          00  ED 00074      CMPZV   #0, #8, (AP), PARAMCNT                  ; 1545
                                        0E  18 00079      BGEQ    7$                                      ;
                                B8      A7  9F 0007B      PUSHAB  P.ADK                                   ; 1547
                                        01  DD 0007E      PUSHL   #1                                      ;
                         00028362       8F  DD 00080      PUSHL   #164706                                 ;
                                68      03  FB 00086      CALLS   #3, LIB$SIGNAL                          ;
                                52    6C43  C0 00089 7$:  ADDL2   (AP)[PARAMCNT], MARGIN                 ; 1549
                                        26  11 0008D      BRB     12$                                     ; 1520
                                55      01  D0 0008F 8$:  MOVL    #1, BRK_ON_BLANKS                       ; 1557
                                        21  11 00092      BRB     12$                                     ;
                                        53  D6 00094 9$:  INCL    PARAMCNT                                ; 1565
        53      6C          08          00  ED 00096      CMPZV   #0, #8, (AP), PARAMCNT                  ; 1566
                                        0E  18 0009B      BGEQ    10$                                     ;
                                DC      A7  9F 0009D      PUSHAB  P.ADL                                   ; 1568
                                        01  DD 000A0      PUSHL   #1                                      ;
                         00028362       8F  DD 000A2      PUSHL   #164706                                 ;
                                68      03  FB 000A8      CALLS   #3, LIB$SIGNAL                          ;
                                54    6C43  D0 000AB 10$: MOVL    (AP)[PARAMCNT], CONTINUE                ; 1570
                                        04  11 000AF      BRB     12$                                     ; 1520
                                        52  D4 000B1 11$: CLRL    MARGIN                                  ; 1578
                                        54  7C 000B3      CLRQ    CONTINUE                                ; 1579
                                        54  D5 000B5 12$: TSTL    CONTINUE                                ; 1595
                                        66  19 000B7      BLSS    17$                                     ;
                                        52  D5 000B9      TSTL    MARGIN                                  ;
                                        62  19 000BB      BLSS    17$                                     ;
        51                      52      54  C1 000BD      ADDL3   CONTINUE, MARGIN, R1                    ; 1596
                                50      69  D0 000C1      MOVL    DBG$SRC_TERM_WIDTH, R0                  ;
                      FF68      C6      50  D1 000C4      CMPL    R0, MAX_PBSIZE                          ;
                                        05  15 000C9      BLEQ    13$                                     ;
                                50  FF68  C6  D0 000CB      MOVL    MAX_PBSIZE, R0                          ;
                                50      51  D1 000D0 13$: CMPL    R1, R0                                  ;
                                        4A  18 000D3      BGEQ    17$                                     ;
                                      FF53  31 000D5      BRW     1$                                      ;
                                66      52  D0 000D8 14$: MOVL    MARGIN, PRTSET_MARGIN                   ; 1605
                                FC      A6  54  D0 000DB      MOVL    CONTINUE, PRTSET_CONTINUE               ; 1606
                                04      A6  55  D0 000DF      MOVL    BRK_ON_BLANKS, PRTBRK_ON_BLANKS         ; 1607
                                51      69  D0 000E3      MOVL    DBG$SRC_TERM_WIDTH, R1                  ; 1613
                                50      51  D0 000E6      MOVL    R1, R0                                  ;
                      FF68      C6      50  D1 000E9      CMPL    R0, MAX_PBSIZE                          ;
                                        05  15 000EE      BLEQ    15$                                     ;
                                50  FF68  C6  D0 000F0      MOVL    MAX_PBSIZE, R0                          ;
        FF64    C6              50      02  C7 000F5 15$: DIVL3   #2, R0, MAX_INDENT                     ;
                                50  FF64  C6  D0 000FB      MOVL    MAX_INDENT, R0                          ; 1614
                                66      50  D1 00100      CMPL    R0, PRTSET_MARGIN                       ;
                                        03  15 00103      BLEQ    16$                                     ;
                                50      66  D0 00105      MOVL    PRTSET_MARGIN, R0                       ;
                                F8      A6  50  D0 00108 16$: MOVL    R0, PRT_INDENT                          ;
                50                      51  F8  A6  C3 0010C      SUBL3   PRT_INDENT, R1, R0                      ; 1621
                                50      02  C6 00111      DIVL2   #2, R0                                  ;
                      FF60      C6      F8 B640 9E 00114      MOVAB   @PRT_INDENT[R0], MAX_BRK_ON_BLANKS      ;
                                50      01  D0 0011B      MOVL    #1, R0                                  ; 1622
                                        04  0011E          RET                                             ;
                                        50  D4 0011F 17$: CLRL    R0                                      ; 1624
                                            04 00121      RET                                             ;
```

; Routine Size: 290 bytes,   Routine Base: DBG$CODE + 03B5

```
;  1517          1625  1 GLOBAL ROUTINE DBG$PRINT_FIELD_REF(VAL_DESC, TF_FLAG) : NOVALUE =
;  1518          1626  1 !
;  1519          1627  1 ! FUNCTION
;  1520          1628  1 !       This routine prints the field reference information as in X<p,s,e>.
;  1521          1629  1 !       It is called from DBG$PRINT_IDENTIFIER and from DBG$EVALUATE (in
;  1522          1630  1 !       the EVAL/ADDR code).  If the address passed as a Value Descriptor
;  1523          1631  1 !       to this routine represents a bit-field, then the "<p,s,e>" informa-
;  1524          1632  1 !       tion is filled into the output buffer via DBG$PRINT.
;  1525          1633  1 !
;  1526          1634  1 ! INPUTS
;  1527          1635  1 !       VAL_DESC - A pointer to a volatile value descriptor representing
;  1528          1636  1 !                  the address to be printed.
;  1529          1637  1 !
;  1530          1638  1 !       TF_FLAG - An optional flag, which if present and TRUE, indicates that
;  1531          1639  1 !                 we want to see field info for type TF.
;  1532          1640  1 !
;  1533          1641  1 ! OUTPUTS
;  1534          1642  1 !       NONE
;  1535          1643  1 !
;  1536          1644  1
;  1537          1645  2     BEGIN
;  1538          1646  2
;  1539          1647  2     BUILTIN
;  1540          1648  2         ACTUALCOUNT;                    ! The number of actual parameters
;  1541          1649  2
;  1542          1650  2     MAP
;  1543          1651  2         VAL_DESC: REF DBG$VALDESC;      ! Pointer to input Value Descriptor
;  1544          1652  2
;  1545          1653  2     LOCAL
;  1546          1654  2         FLAG;
;  1547          1655  2
;  1548          1656  2     IF ACTUALCOUNT() LSS 2
;  1549          1657  2     THEN
;  1550          1658  2         FLAG = FALSE
;  1551          1659  2
;  1552          1660  2     ELSE
;  1553          1661  2         FLAG = .TF_FLAG;
;  1554          1662  2
;  1555          1663  2
;  1556          1664  2     ! Check whether the Value Descriptor represents a bit-field.
;  1557          1665  2     ! If so, print the <p,s,e> information.
;  1558          1666  2     !
;  1559          1667  2     IF (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV)  OR
;  1560          1668  2        (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU) OR
;  1561          1669  2        (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V)   OR
;  1562          1670  2        (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU)  OR
;  1563          1671  2        (.VAL_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS) OR
;  1564          1672  3        (.FLAG AND (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_TF))
;  1565          1673  2     THEN
;  1566          1674  3         BEGIN
;  1567          1675  3
;  1568          1676  3
;  1569          1677  3         ! We have a bit-field.  We now print the "<p,s,e>" information unless
;  1570          1678  3         ! the bitfield just happens to cover a byte-aligned longword.
;  1571          1679  3         !
;  1572          1680  4         IF NOT (.VAL_DESC[DBG$L_VALUE_POS] EQL 0 AND
;  1573          1681  4                 .VAL_DESC[DBG$W_VALUE_LENGTH] EQL 32)
```

```
;  1574          1682  3          THEN
;  1575          1683  4              BEGIN
;  1576          1684  4              DBG$PRINT(UPLIT BYTE (%ASCIC '<'));
;  1577          1685  4              DBG$PRINT(UPLIT BYTE (%ASCIC '!UL'), .VAL_DESC[DBG$L_VALUE_POS]);
;  1578          1686  4              DBG$PRINT(UPLIT BYTE (%ASCIC ','));
;  1579          1687  4              DBG$PRINT(UPLIT BYTE (%ASCIC '!UW'), .VAL_DESC[DBG$W_VALUE_LENGTH]);
;  1580          1688  4              DBG$PRINT(UPLIT BYTE (%ASCIC ','));
;  1581          1689  4              IF (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV) OR
;  1582          1690  5                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU)
;  1583          1691  4              THEN
;  1584          1692  4                  DBG$PRINT(UPLIT BYTE (%ASCIC '1>'))
;  1585          1693  4
;  1586          1694  4              ELSE
;  1587          1695  4                  DBG$PRINT(UPLIT BYTE (%ASCIC '0>'));
;  1588          1696  4
;  1589          1697  3              END;
;  1590          1698  3
;  1591          1699  2          END;
;  1592          1700  2
;  1593          1701  2
;  1594          1702  2      ! We are now done.  Return to the caller.
;  1595          1703  2      !
;  1596          1704  2      RETURN;
;  1597          1705  2
;  1598          1706  1      END;


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

                                3C  01   00CCA P.ADN:   .ASCII  <1>\<\
                          4C    55  21   03   00CCC P.ADO:   .ASCII  <3>\!UL\                                                  ;
                                2C  01   00CD0 P.ADP:   .ASCII  <1>\,\                                                       ;
                          57    55  21   03   00CD2 P.ADQ:   .ASCII  <3>\!UW\                                                  ;
                                2C  01   00CD6 P.ADR:   .ASCII  <1>\,\                                                       ;
                                3E  31   02   00CD8 P.ADS:   .ASCII  <2>\1>\                                                 ;
                                3E  30   02   00CDB P.ADT:   .ASCII  <2>\0>\                                                 ;


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

                                   007C 00000           .ENTRY  DBG$PRINT_FIELD_REF, Save R2,R3,R4,R5,R6                     ; 1625
                          56     FC9F  CF   9E 00002           MOVAB   DBG$PRINT, R6                                          ;
                          55 00000000'  EF   9E 00007           MOVAB   P.ADN, R5                                              ;
                          02            6C   91 0000E           CMPB    (AP), #2                                             ; 1656
                                        04   1E 00011           BGEQU   1$                                                   ;
                                        50   D4 00013           CLRL    FLAG                                                 ; 1658
                                        04   11 00015           BRB     2$                                                   ;
                          50      08    AC   D0 00017 1$:       MOVL    TF_FLAG, FLAG                                        ; 1661
                          52      04    AC   D0 0001B 2$:       MOVL    VAL_DESC, R2                                         ; 1667
                          53      16    A2   9A 0001F           MOVZBL  22(R2), R3                                           ;
                                        54   D4 00023           CLRL    R4                                                   ;
                          29            53   91 00025           CMPB    R3, #41                                              ;
                                        04   12 00028           BNEQ    3$                                                   ;
                                        54   D6 0002A           INCL    R4                                                   ;
                                        1D   11 0002C           BRB     4$                                                   ;
```

```
                53  91  0002E 3$:      CMPB    R3, #42                         ; 1668
                18  13  00031          BEQL    4$
                53  91  00033          CMPB    R3, #1                          ; 1669
                13  13  00036          BEQL    4$
                53  91  00038          CMPB    R3, #34                         ; 1670
                0E  13  0003B          BEQL    4$
            17  A2  91  0003D          CMPB    23(R2), #13                     ; 1671
                08  13  00041          BEQL    4$
                50  E9  00043          BLBC    FLAG, 9$                        ; 1672
                53  91  00046          CMPB    R3, #40
                42  12  00049          BNEQ    9$
            1C  A2  D5  0004B 4$:       TSTL    28(R2)                          ; 1680
                06  12  0004E          BNEQ    5$
            14  A2  B1  00050          CMPW    20(R2), #32                     ; 1681
                37  13  00054          BEQL    9$
                55  DD  00056 5$:       PUSHL   R5                              ; 1684
                01  FB  00058          CALLS   #1, DBG$PRINT
            1C  A2  DD  0005B          PUSHL   28(R2)                          ; 1685
            02  A5  9F  0005E          PUSHAB  P.ADO
                02  FB  00061          CALLS   #2, DBG$PRINT
            06  A5  9F  00064          PUSHAB  P.ADP                           ; 1686
                01  FB  00067          CALLS   #1, DBG$PRINT
            14  A2  3C  0006A          MOVZWL  20(R2), -(SP)                   ; 1687
            08  A5  9F  0006E          PUSHAB  P.ADQ
                02  FB  00071          CALLS   #2, DBG$PRINT
            0C  A5  9F  00074          PUSHAB  P.ADR                           ; 1688
                01  FB  00077          CALLS   #1, DBG$PRINT
                54  E8  0007A          BLBS    R4, 6$                          ; 1689
                53  91  0007D          CMPB    R3, #42                         ; 1690
                05  12  00080          BNEQ    7$
            0E  A5  9F  00082 6$:       PUSHAB  P.ADS                           ; 1692
                03  11  00085          BRB     8$
            11  A5  9F  00087 7$:       PUSHAB  P.ADT                           ; 1695
        66          01  FB  0008A 8$:       CALLS   #1, DBG$PRINT
                    04  0008D 9$:       RET                                     ; 1706

; Routine Size:  142 bytes,    Routine Base:  DBG$CODE + 04D7
```

```
;  1600          1707  1 GLOBAL ROUTINE DBG$PRINT_IDENTIFIER(PRIM_DESC) =
;  1601          1708  1 !
;  1602          1709  1 ! FUNCTION
;  1603          1710  1 !       This routine takes a primary descriptor or turns volatile value descriptor
;  1604          1711  1 !       into primary descriptor + offset, then based on the language code
;  1605          1712  1 !       in the primary descriptor and the print format code, prints out the
;  1606          1713  1 !       primary symbol name + offset.  If there is no symbolization can be
;  1607          1714  1 !       done, (address --X--> primary + offset) then absolute address will
;  1608          1715  1 !       be printed.
;  1609          1716  1 !
;  1610          1717  1 ! INPUTS
;  1611          1718  1 !       PRIM_DESC       - Pointer to Primary Descriptor or Value Descriptor.
;  1612          1719  1 !
;  1613          1720  1 !       argument2       - Print Format Code.
;  1614          1721  1 !                         Format Code 0: data component name, single data name
;  1615          1722  1 !                         Format Code 1: data name
;  1616          1723  1 !                         Format code 2: M\R\data name
;  1617          1724  1 !
;  1618          1725  1 ! OUTPUTS
;  1619          1726  1 !       None.
;  1620          1727  1 !
;  1621          1728  1
;  1622          1729  2     BEGIN
;  1623          1730  2
;  1624          1731  2     MAP
;  1625          1732  2         PRIM_DESC: REF DBG$PRIMARY;     ! Pointer to Primary Root Node
;  1626          1733  2
;  1627          1734  2     BUILTIN
;  1628          1735  2         ACTUALCOUNT,                    ! The number of actual parameters
;  1629          1736  2         ACTUALPARAMETER;                ! Select the N-th actual parameter
;  1630          1737  2
;  1631          1738  2     LITERAL
;  1632          1739  2         DESCR_SIZE = 20;                ! Buffer size in bytes
;  1633          1740  2
;  1634          1741  2     BIND
;  1635          1742  2         BACK_SLASH = UPLIT BYTE(%ASCIC '\'): VECTOR[,BYTE];
;  1636          1743  2
;  1637          1744  2     LOCAL
;  1638          1745  2         ADDRESS: VECTOR[2, LONG],       ! Address descriptor byte and offset
;  1639          1746  2         ARRAY_FLAG,                     ! Flag set to indicate array subscript
;  1640          1747  2                                         !      begin is already printed
;  1641          1748  2         BITLEN,                         ! Bit length of array element
;  1642          1749  2         BIT_OFFSET,                     ! Bit offset from SYMID
;  1643          1750  2         BYTE_OFFSET,                    ! Byte offset from the Primary Descriptor
;  1644          1751  2         BOTTOM,                         ! Bottom of the stack
;  1645          1752  2         CHAR_TABLE: REF VECTOR[,LONG],  ! Pointer to Print Character Table
;  1646          1753  2         FCODE,                          ! Format Code
;  1647          1754  2         FORMAT,                         ! Print Format for Primary Symbols
;  1648          1755  2         HERE,                           ! Temporary stack pointer
;  1649          1756  2         INFO_FLAG,                      ! Print Information Flag
;  1650          1757  2         INFO_TABLE: REF PRTINFO$TABLE,  ! Pointer to Print Information Table
;  1651          1758  2         KIND,                           ! Symid's kind
;  1652          1759  2         NAMEPTR: REF VECTOR[,BYTE],     ! Pointer to a Counted string
;  1653          1760  2         NEXT_NODE: REF DBG$PRIM_NODE,   ! Next Sub-Node in Primary Descriptor
;  1654          1761  2         PATHNAME: REF PTH$PATHNAME,     ! Pointer to Pathname Descriptor
;  1655          1762  2         PATHNAME_FLAG,                  ! Flag set to true to indicate we
;  1656          1763  2                                         !      have a pathname
```

```
;  1657          1764  2          REFMOD_FLAG,                        ! Reference Modifier Flag
;  1658          1765  2          PRIM_HEAD,                          ! Address of Sub-Node list head in Root
;  1659          1766  2          PRIM_NODE: REF DBG$PRIM_NODE,       ! Pointer to Primary Sub-Node
;  1660          1767  2          PTR: REF VECTOR[,LONG],             ! Pointer to print language tables
;  1661          1768  2          PTR_FLAG,                           ! Flag set to indicate the pointer is
;  1662          1769  2                                              !       encountered
;  1663          1770  2          REGDESCR: REF DBG$REGDESCR,         ! Pointer to register descriptor
;  1664          1771  2          SAVE_VAL_DTYPE,                     ! Saved Dtype from Value Descriptor
;  1665          1772  2          SAVE_VAL_LENGTH,                    ! Saved Length from Value Descriptor
;  1666          1773  2          SYMID: REF RST$ENTRY,               ! Pointer to SYMID
;  1667          1774  2          STACK: PRTID$STACK,                 ! A blockvector of pointers
;  1668          1775  2          STKOVF,                             ! Flag set to indicate stack overflow
;  1669          1776  2          SUBOVF,                             ! Flag set to indicate overflow
;  1670          1777  2          SUBVECTOR:                          ! Pointer to subscript block-vector
;  1671          1778  2                  REF DBG$PRIM_NODE_SUBS,     !       in Primary Descr Array Sub-Node
;  1672          1779  2          TOP,                                ! Top of the stack
;  1673          1780  2          TYPEID: REF RST$ENTRY,              ! Pointer to TYPEID
;  1674          1781  2          VALPTR: REF DBG$VALDESC,            ! Pointer to value descriptor
;  1675          1782  2          VALUE,                              ! Temporary variable
;  1676          1783  2          VAL_DESC: REF DBG$VALDESC,          ! Value descriptor
;  1677          1784  2          VMS_DESC: BLOCK [8,BYTE];           ! A VMS descriptor
;  1678          1785  2
;  1679          1786  2
;  1680          1787  2      ! First we set up some print flags by the current language setting.
;  1681          1788  2      !
;  1682          1789  2      DBG$PRINT_SET_LANGUAGE(.DBG$GB_LANGUAGE);
;  1683          1790  2
;  1684          1791  2
;  1685          1792  2      ! Volatile Value Descriptors.  We do our best to symbolize this absolute
;  1686          1793  2      ! address back to primary, or primary+offset.  If we cannot symbolize it
;  1687          1794  2      ! at all then we print it as absolute address.
;  1688          1795  2      !
;  1689          1796  2      VAL_DESC = 0;
;  1690          1797  2      BYTE_OFFSET = 0;
;  1691          1798  2      BIT_OFFSET = 0;
;  1692          1799  2      IF .PRIM_DESC[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC
;  1693          1800  2      THEN
;  1694          1801  3          BEGIN
;  1695          1802  3          VAL_DESC = .PRIM_DESC;
;  1696          1803  3          SAVE_VAL_DTYPE = 0;
;  1697          1804  3          SAVE_VAL_LENGTH = .VAL_DESC[DBG$W_VALUE_LENGTH];
;  1698          1805  3
;  1699          1806  3          ! First we try to see if the given address is a register address.
;  1700          1807  3          ! If it is, a register descriptor is returned.  Then DBG$STA_REGISTER_NAME
;  1701          1808  3          ! is called to converts the register descriptor into a counted ASCII
;  1702          1809  3          ! name.  Otherwise, DBG$STA_GETSYMOFF is called.
;  1703          1810  3          !
;  1704          1811  3          ! Note: The correct logic should call DBG$STA_GETSYMOFF first, if every-
;  1705          1812  3          ! thing failed then try to see if the given address is a register
;  1706          1813  3          ! address.  If it fails, then print it as an absolute address.
;  1707          1814  3          ! Currently, DBG$STA_GETSYMOFF tries to symbolize a given address in
;  1708          1815  3          ! DBG$REG_VALUES and returns the first SYMID found bound to that
;  1709          1816  3          ! register.  Symbolization needs to associate the scope with the register.
;  1710          1817  3          ! Till this piece of code is in place, we'll not symbolize the register
;  1711          1818  3          ! address.  We'll just print it and immediately return.
;  1712          1819  3          !
;  1713          1820  3          REGDESCR = DBG$STA_ADDRESS_TO_REGDESCR(.VAL_DESC[DBG$L_VALUE_POINTER]);
```

```
;  1714          1821  3          IF .REGDESCR NEQ 0
;  1715          1822  3          THEN
;  1716          1823  4              BEGIN
;  1717          1824  4              NAMEPTR = DBG$STA_REGISTER_NAME(.REGDESCR);
;  1718          1825  4              DBG$PRINT(UPLIT BYTE(%ASCIC '!AC'), .NAMEPTR);
;  1719          1826  4              RETURN .VAL_DESC;
;  1720          1827  3              END;
;  1721          1828  3
;  1722          1829  3
;  1723          1830  3          ! The address is not a register address.  We thus try to symbolize it,
;  1724          1831  3          ! but only if we are in symbolic mode.
;  1725          1832  3          !
;  1726          1833  3          IF .DBG$GB_MOD_PTR[MODE_SYMBOLS]
;  1727          1834  3          THEN
;  1728          1835  4              BEGIN
;  1729          1836  4              ADDRESS[0] = .VAL_DESC[DBG$L_VALUE_POINTER];
;  1730          1837  4              IF .VAL_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS
;  1731          1838  4              THEN
;  1732          1839  4                  ADDRESS[1] = .VAL_DESC[DBG$L_VALUE_POS]
;  1733          1840  4
;  1734          1841  4              ELSE
;  1735          1842  4                  ADDRESS[1] = 0;
;  1736          1843  4
;  1737          1844  4              DBG$NEW_SYMBOLIZE (ADDRESS, SYMID, BIT_OFFSET);
;  1738          1845  4              END
;  1739          1846  4
;  1740          1847  4
;  1741          1848  4          ! We are in non-symbolic mode, so we set SYMID to zero.
;  1742          1849  4          !
;  1743          1850  3          ELSE
;  1744          1851  3              SYMID = 0;
;  1745          1852  3
;  1746          1853  3
;  1747          1854  3          ! If we cannot symbolize it, we print it as an absolute address and
;  1748          1855  3          ! then return to the caller.
;  1749          1856  3          !
;  1750          1857  3          IF .SYMID EQL 0
;  1751          1858  3          THEN
;  1752          1859  4              BEGIN
;  1753          1860  4
;  1754          1861  4
;  1755          1862  4              ! Build a VMS Descriptor of type longword unsigned, which
;  1756          1863  4              ! is the type for absolute addresses.
;  1757          1864  4              !
;  1758          1865  4              VMS_DESC[DSC$B_CLASS] = DSC$K_CLASS_Z;
;  1759          1866  4              VMS_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_LU;
;  1760          1867  4              VMS_DESC[DSC$W_LENGTH] = 4;
;  1761          1868  4              VMS_DESC[DSC$A_POINTER] = VAL_DESC[DBG$L_VALUE_POINTER];
;  1762          1869  4              DBG$PRINT_VALUE_AS_INTEGER(VMS_DESC);
;  1763          1870  4
;  1764          1871  4
;  1765          1872  4              ! If this is a bit substring, print the angle brackets with the
;  1766          1873  4              ! offset, length, and sign extension information.  Then return.
;  1767          1874  4              !
;  1768          1875  4              DBG$PRINT_FIELD_REF(.VAL_DESC);
;  1769          1876  4              RETURN .VAL_DESC;
;  1770          1877  3              END;
```

```
;  1771          1878  3
;  1772          1879  3
;  1773          1880  3          ! We can symbolize it, so we build a Primary Descriptor from the
;  1774          1881  3          ! SYMID and offset.
;  1775          1882  3          !
;  1776          1883  3          ! Note. there is one case here we need to pay special attention to.
;  1777          1884  3          ! I'll use an example to explain.  If you say,
;  1778          1885  3          !
;  1779          1886  3          ! E FAT_RECORD<6,10,0> what we got is a volatile value descriptor
;  1780          1887  3          ! comes into this routine, with the Class UBS, one of the Dtype
;  1781          1888  3          ! SV, SVU, V, VU, and Byte address for FAT_RECORD, Bit offset and
;  1782          1889  3          ! length.  When we symbolize the address to symid to primary, and
;  1783          1890  3          ! some bit offsets, we know the fact that we need to print bit offsets
;  1784          1891  3          ! to the primary from the given CLASS UBS, and DTYPE SV, SVU, V, VU.
;  1785          1892  3          ! The following code gets these information based on these facts.
;  1786          1893  3          ! So, we'll print it M\R\FAT_RECORD.BG2<1,0,0>: 40, say the address
;  1787          1894  3          ! of FAT_RECORD.BG2 is 7FFAF4D7<5,11,0>.
;  1788          1895  3          !
;  1789          1896  3          ! Now, consider another case,  If you say,
;  1790          1897  3          ! E 7FFAF4D8 (this is FAT_RECORD.BG2 + some offsets).  Since the
;  1791          1898  3          ! volatile value descriptor for this address has Class S, Dtype
;  1792          1899  3          ! L, length 4.  So after we turn the symid to primary, we forgot
;  1793          1900  3          ! the fact that we really have bit offsets to the primary for there
;  1794          1901  3          ! are no CLASS and DTYPE to drive this information out.
;  1795          1902  3          !
;  1796          1903  3          ! SAVE_VAL_DTYPE is the hack to do the trick.  In order words,
;  1797          1904  3          ! now that we know we have bit offsets to primary, we plung in
;  1798          1905  3          ! Dtype V to the value descriptor to do the bit offset symbolization.
;  1799          1906  3          ! After we are done we put the orignal back.  The reason we do not
;  1800          1907  3          ! put in the primary is see the first example, since we do not
;  1801          1908  3          ! build primary with the offsets for FAT_RECORD<6,10,0> case, there
;  1802          1909  3          ! is not a consistent way for this routine to build that way.
;  1803          1910  3          ! We'll change it later if there is a need.
;  1804          1911  3          !
;  1805          1912  3          ! If the above case showed up, we always print the length zero.
;  1806          1913  3          ! SAVE_VAL_LENGTH is used to saved the original length.  So the
;  1807          1914  3          ! original value descriptor can be returned back.
;  1808          1915  3          !
;  1809          1916  3          PRIM_DESC = DBG$SYMID_TO_PRIMARY(.SYMID, BIT_OFFSET);
;  1810          1917  3          IF .BIT_OFFSET NEQ 0
;  1811          1918  3          THEN
;  1812          1919  4              BEGIN
;  1813          1920  4              IF (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV)  OR
;  1814          1921  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU) OR
;  1815          1922  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V)   OR
;  1816          1923  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU)  OR
;  1817          1924  5                 (.VAL_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS)
;  1818          1925  4              THEN
;  1819          1926  4                  SAVE_VAL_DTYPE = 0
;  1820          1927  4
;  1821          1928  4              ELSE
;  1822          1929  4
;  1823          1930  4
;  1824          1931  4                  ! Note: This trick only applies to the item that does
;  1825          1932  4                  ! not have a byte offset.
;  1826          1933  4                  !
;  1827          1934  5                  BEGIN
```

```
;1828     1935  5              IF (.BIT_OFFSET - ((.BIT_OFFSET / 8) * 8)) NEQ 0
;1829     1936  5              THEN
;1830     1937  6                  BEGIN
;1831     1938  6                  SAVE_VAL_DTYPE = .VAL_DESC[DBG$B_VALUE_DTYPE];
;1832     1939  6                  VAL_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_V;
;1833     1940  6                  VAL_DESC[DBG$W_VALUE_LENGTH] = 8 * .VAL_DESC[DBG$W_VALUE_LENGTH];
;1834     1941  6                  END
;1835     1942  6
;1836     1943  5              ELSE
;1837     1944  5                  SAVE_VAL_DTYPE = 0;
;1838     1945  4              END;
;1839     1946  4
;1840     1947  4          BYTE_OFFSET = .BIT_OFFSET / 8;
;1841     1948  4          BIT_OFFSET = .BIT_OFFSET - .BYTE_OFFSET * 8;
;1842     1949  4          IF (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV)  OR
;1843     1950  4             (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU) OR
;1844     1951  4             (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V)   OR
;1845     1952  4             (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU)  OR
;1846     1953  5             (.VAL_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS)
;1847     1954  4          THEN
;1848     1955  5              BEGIN
;1849     1956  5              IF .BIT_OFFSET EQL 0
;1850     1957  5              THEN
;1851     1958  6                  BEGIN
;1852     1959  6                  IF .VAL_DESC[DBG$W_VALUE_LENGTH] LSS 32
;1853     1960  6                  THEN
;1854     1961  7                      BEGIN
;1855     1962  7                      BIT_OFFSET = .BYTE_OFFSET * 8;
;1856     1963  7                      BYTE_OFFSET = 0;
;1857     1964  6                      END;
;1858     1965  6
;1859     1966  5                  END;
;1860     1967  5
;1861     1968  4              END;
;1862     1969  4
;1863     1970  3          END;
;1864     1971  3
;1865     1972  3
;1866     1973  3          ! Adjust the offset for printing.
;1867     1974  3          !
;1868     1975  3          IF .BIT_OFFSET NEQ 0
;1869     1976  3          THEN
;1870     1977  4              BEGIN
;1871     1978  4              IF (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SV)  OR
;1872     1979  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_SVU) OR
;1873     1980  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_V)  OR
;1874     1981  4                 (.VAL_DESC[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU) OR
;1875     1982  5                 (.VAL_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_UBS)
;1876     1983  4              THEN
;1877     1984  4                  VAL_DESC[DBG$L_VALUE_POS] = .BIT_OFFSET;
;1878     1985  4
;1879     1986  3             END;
;1880     1987  3
;1881     1988  2          END;                            ! End of code for Volatile Value Descr.
;1882     1989  2
;1883     1990  2
;1884     1991  2      ! Set up the Primary Symbol Print Format.  For example:
```

```
;  1885          1992  2      !       Format 0 - X or (2)
;  1886          1993  2      !       Format 1 - A.X or A(2)
;  1887          1994  2      !       Format 2 - M\R\A or M\R\A.X or M\R\A(2)
;  1888          1995  2      !
;  1889          1996  2      FORMAT = 2;
;  1890          1997  2      IF ACTUALCOUNT() GTR 1 THEN FORMAT = MINU(ACTUALPARAMETER(2), 2);
;  1891          1998  2      IF .FORMAT EQL 2
;  1892          1999  2      THEN
;  1893          2000  2          DBG$STA_SYMPATHNAME(.PRIM_DESC[DBG$L_DHDR_SYMIDO], PATHNAME);
;  1894          2001  2
;  1895          2002  2
;  1896          2003  2      ! Output the none-data (code) item.
;  1897          2004  2      !
;  1898          2005  2      IF .PRIM_DESC[DBG$B_DHDR_FCODE] EQL 0
;  1899          2006  2      THEN
;  1900          2007  3          BEGIN
;  1901          2008  3          IF .FORMAT LSS 2
;  1902          2009  3          THEN
;  1903          2010  3              DBG$STA_SYMNAME(.PRIM_DESC[DBG$L_DHDR_SYMIDO], NAMEPTR)
;  1904          2011  3
;  1905          2012  3          ELSE
;  1906          2013  3              DBG$NPATHDESC_TO_CS(.PATHNAME, NAMEPTR);
;  1907          2014  3
;  1908          2015  3          DBG$PRINT(UPLIT BYTE(%ASCIC '!AC'), .NAMEPTR);
;  1909          2016  3          PRINT_OFFSETS_AND_RETURN;
;  1910          2017  2          END;
;  1911          2018  2
;  1912          2019  2
;  1913          2020  2      ! Data items are language-dependent.  Load in Language Print Table
;  1914          2021  2      ! according to the language code in the Root Node.  Set up the stack
;  1915          2022  2      ! pointers.  This stack grows two ways, so we set up TOP and BOTTOM
;  1916          2023  2      ! pointers relative to STACK_START.
;  1917          2024  2      !
;  1918          2025  2      PTR = .LANGUAGE_PRINT_TABLE_PTRS[.PRIM_DESC[DBG$B_DHDR_LANG]] + TABLEBASE;
;  1919          2026  2      INFO_TABLE = .PTR[0] + TABLEBASE;
;  1920          2027  2      INFO_FLAG = .PTR[1];
;  1921          2028  2      REFMOD_FLAG = .PTR[2];
;  1922          2029  2      TOP = BOTTOM = PRINT_STACK_START;
;  1923          2030  2      ARRAY_FLAG = FALSE;
;  1924          2031  2      PTR_FLAG = FALSE;
;  1925          2032  2      STKOVF = FALSE;
;  1926          2033  2      SUBOVF = FALSE;
;  1927          2034  2      PATHNAME_FLAG = 0;
;  1928          2035  2
;  1929          2036  2
;  1930          2037  2      ! Start from the Primary Root Node, following the forward linked list,
;  1931          2038  2      ! get to Sub-Nodes which describe the individual components of the symbol
;  1932          2039  2      ! described by the Primary Descriptor.  Save the Address of Sub-Node
;  1933          2040  2      ! list head in root, and some other information.
;  1934          2041  2      !
;  1935          2042  2      PRIM_HEAD = PRIM_DESC[DBG$A_PRIM_FLINK];
;  1936          2043  2      PRIM_NODE = .PRIM_HEAD;
;  1937          2044  2      WHILE TRUE DO
;  1938          2045  3          BEGIN
;  1939          2046  3
;  1940          2047  3
;  1941          2048  3          ! Link to the next Primary Descriptor Sub-Node in the list.  If that
```

```
;  1942          2049  3          ! takes us back to the list head, exit the loop.
;  1943          2050  3          !
;  1944          2051  3          PRIM_NODE = .PRIM_NODE[DBG$L_PNODE_FLINK];
;  1945          2052  3          IF .PRIM_NODE EQLA .PRIM_HEAD THEN EXITLOOP;
;  1946          2053  3
;  1947          2054  3
;  1948          2055  3          ! Go ahead and get the Primary Symbol's name.  Check to see if this
;  1949          2056  3          ! symbol is the first symbol in the node; if it is, then check to see
;  1950          2057  3          ! if this symbol needs a pathname (as determined by the FORMAT param-
;  1951          2058  3          ! eter).  If so, get the full pathname in Counted ASCII, and if not,
;  1952          2059  3          ! just get the individual symbol name.
;  1953          2060  3          !
;  1954          2061  3          IF .PRIM_NODE[DBG$L_PNODE_SYMID] EQL .PRIM_DESC[DBG$L_DHDR_SYMIDO]
;  1955          2062  3          THEN
;  1956          2063  4              BEGIN
;  1957          2064  4              DBG$STA_SYMNAME(.PRIM_NODE[DBG$L_PNODE_SYMID], NAMEPTR);
;  1958          2065  4
;  1959          2066  4
;  1960          2067  4              ! Make a note that this symbol has M\R pathname only.
;    61          2068  4              !
;  1962          2069  4              IF .NAMEPTR[0] EQL 0 THEN PATHNAME_FLAG = 1;
;  1963          2070  4              IF .FORMAT GEQ 2
;  1964          2071  4              THEN
;  1965          2072  5                  BEGIN
;  1966          2073  5                  DBG$NPATHDESC_TO_CS(.PATHNAME, NAMEPTR);
;  1967          2074  5
;  1968          2075  5
;  1969          2076  5                      ! So far, this is only affect cobol.
;  1970          2077  5                      !
;  1971          2078  5                      ! Push Pathname (M\R) and \ (In this case, there is no data name,
;  1972          2079  5                      ! indicated by PATHNAME_FLAG set to 1).  After we have done
;  1973          2080  5                      ! this, we set the PATHNAME_FLAG to 2 to indicate the next
;  1974          2081  5                      ! data name needs to go after \ (need to put it at the bottom
;  1975          2082  5                      ! of the stack).
;  1976          2083  5                      !
;  1977          2084  5                      IF .PATHNAME_FLAG EQL 1
;  1978          2085  5                      THEN
;  1979          2086  6                          BEGIN
;  1980          2087  6                          STKOVF = PRINT_PUSH(TRUE, STACK, TOP, BACK_SLASH);
;  1981          2088  6                          STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  1982          2089  6                          PATHNAME_FLAG = 2;
;  1983          2090  6
;  1984          2091  6
;  1985          2092  6                          ! Since we have done a push, go get one more data node.
;  1986          2093  6                          !
;  1987          2094  6                          PRIM_NODE = .PRIM_NODE[DBG$L_PNODE_FLINK];
;  1988          2095  6                          IF .PRIM_NODE[DBG$L_PNODE_SYMID] NEQ 0
;  1989          2096  6                          THEN
;  1990          2097  6                              DBG$STA_SYMNAME(.PRIM_NODE[DBG$L_PNODE_SYMID], NAMEPTR)
;  1991          2098  6
;  1992          2099  6                          ELSE
;  1993          2100  6                              NAMEPTR = 0;
;  1994          2101  5                          END;
;  1995          2102  4                  END;
;  1996          2103  4
;  1997          2104  4          END
;  1998          2105  4
```

```
;1999        2106  4
;2000        2107  4          ! This is not the first symbol in the full name, so we do not need the
;2001        2108  4          ! full pathname.  Just get the individual symbol name if one exists.
;2002        2109  4          !
;2003        2110  3          ELSE
;2004        2111  4              BEGIN
;2005        2112  4              IF .PRIM_NODE[DBG$L_PNODE_SYMID] NEQ 0
;2006        2113  4              THEN
;2007        2114  4                  DBG$STA_SYMNAME(.PRIM_NODE[DBG$L_PNODE_SYMID], NAMEPTR)
;2008        2115  4
;2009        2116  4              ELSE
;2010        2117  4                  NAMEPTR = 0;
;2011        2118  4
;2012        2119  3              END;
;2013        2120  3
;2014        2121  3
;2015        2122  3          ! Get each node's FCODE.  Based on the FCODE, load in proper print
;2016        2123  3          ! characters and print routine index.
;2017        2124  3          !
;2018        2125  3          FORMAT = (IF .FORMAT EQL 0 THEN FALSE ELSE TRUE);
;2019        2126  3          FCODE = .PRIM_NODE[DBG$B_PNODE_FCODE];
;2020        2127  3          CHAR_TABLE = .INFO_TABLE[.FCODE, PRTINFO$L_CHARTBL] + TABLEBASE;
;2021        2128  3          CASE .INFO_TABLE[.FCODE, PRTINFO$L_ROUT_INDEX] FROM PRT$K_MIN_ROUT
;2022        2129  3                                                          TO   PRT$K_MAX_ROUT OF
;2023        2130  3              SET
;2024        2131  3
;2025        2132  3
;2026        2133  3              ! General data item name, it fcode is not one of the following
;2027        2134  3              ! for example, ARRAY, or RECORD, or Pointer.
;2028        2135  3              !
;2029        2136  3              [PRT$K_OTHER]:
;2030        2137  4                  BEGIN
;2031        2138  4                  IF .PATHNAME_FLAG EQL 2
;2032        2139  4                  THEN
;2033        2140  5                      BEGIN
;2034        2141  5                      STKOVF = PRINT_PUSH(FALSE, STACK, BOTTOM, .NAMEPTR);
;2035        2142  5                      PATHNAME_FLAG = 3;
;2036        2143  5                      END
;2037        2144  5
;2038        2145  4                  ELSE
;2039        2146  4                      STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;2040        2147  3                  END;
;2041        2148  3
;2042        2149  3
;2043        2150  3              ! Print Array.
;2044        2151  3              !
;2045        2152  3              [PRT$K_ARRAY]:
;2046        2153  4                  BEGIN
;2047        2154  4
;2048        2155  4
;2049        2156  4                  !  If the pathname_flag set to 2, we always push the name
;2050        2157  4                  !  at the bottom of the stack after the \.  After the operation
;2051        2158  4                  !  set the flag to 3 to indiate the operation is back to normal.
;2052        2159  4                  !
;2053        2160  4                  IF .PATHNAME_FLAG EQL 2
;2054        2161  4                  THEN
;2055        2162  5                      BEGIN
```

```
;2056     2163  5                STKOVF = PRINT_PUSH(FALSE, STACK, BOTTOM, .NAMEPTR);
;2057     2164  5                PATHNAME_FLAG = 3;
;2058     2165  5                END
;2059     2166  5
;2060     2167  4            ELSE
;2061     2168  5                BEGIN
;2062     2169  5
;2063     2170  5
;2064     2171  5                ! Get the name of the array if the Print Format Code is not 0.
;2065     2172  5                !
;2066     2173  5                IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;2067     2174  5                THEN
;2068     2175  6                    BEGIN
;2069     2176  6                    IF .FORMAT
;2070     2177  6                    THEN
;2071     2178  6                        STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;2072     2179  6
;2073     2180  6                    END
;2074     2181  6
;2075     2182  5                ELSE
;2076     2183  5                    STKOVF = PRINT_PUSH(TRUE, STACK. TOP, .NAMEPTR);
;2077     2184  4                END;
;2078     2185  4
;2079     2186  4
;2080     2187  4            ! Get the array subscripts.  There are two way of getting it:
;2081     2188  4            ! In COBOL, array subscripts are printed at the end of the
;2082     2189  4            ! name string, where in other languages, array subscripts are
;2083     2190  4            ! printed along with the array name.  This is indicated by
;2084     2191  4            ! INFO_FLAG.  For example, flag sets to FALSE for COBOL.
;2085     2192  4            !
;2086     2193  5            IF (NOT .FORMAT AND
;2087     2194  5                .PRIM_NODE[DBG$V_PNODE_EVAL] AND
;2088     2195  4                ..PRIM_NODE[0,0,32,0] EQL .PRIM_HEAD) OR
;2089     2196  5               (.FORMAT AND
;2090     2197  5                .PRIM_NODE[DBG$V_PNODE_EVAL])
;2091     2198  4            THEN
;2092     2199  5                BEGIN
;2093     2200  5
;2094     2201  5
;2095     2202  5                ! Set Pathname_flag back to 0 to indicate we should be
;2096     2203  5                ! all done with our funny business of pushing the
;2097     2204  5                ! the data name after pathname\.
;2098     2205  5                !
;2099     2206  5                PATHNAME_FLAG = 0;
;2100     2207  5                IF .INFO_FLAG
;2101     2208  5                THEN
;2102     2209  5                    HERE = .TOP
;2103     2210  5
;2104     2211  5                ELSE
;2105     2212  5                    HERE = .BOTTOM;
;2106     2213  5
;2107     2214  5
;2108     2215  5                ! Place in the begin array subscript.
;2109     2216  5                !
;2110     2217  6                IF NOT (NOT .INFO_FLAG AND .ARRAY_FLAG)
;2111     2218  5                THEN
;2112     2219  6                    BEGIN
```

```
;  2113          2220  6                 STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2114          2221  6                                     .CHAR_TABLE[PRT$K_BEGIN_CHAR] + TABLEBASE);
;  2115          2222  6                 ARRAY_FLAG = TRUE;
;  2116          2223  5                 END;
;  2117          2224  5
;  2118          2225  5
;  2119          2226  5             ! Get the subscript value.
;  2120          2227  5             !
;  2121          2228  5             SUBVECTOR = PRIM_NODE[DBG$A_PNARR_SVECTOR];
;  2122          2229  5             INCR I FROM 0 TO .PRIM_NODE[DBG$B_PNARR_DIMCNT] - 1 DO
;  2123          2230  6                 BEGIN
;  2124          2231  6                 IF .SUBVECTOR[.I, DBG$L_PNSUB_TYPEID] EQL 0
;  2125          2232  6                 THEN
;  2126          2233  7                     BEGIN
;  2127          2234  7                     GET_VALUE(SUBVECTOR[.I, DBG$L_PNSUB_SVALUE]);
;  2128          2235  7                     STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2129          2236  7                     END
;  2130          2237  7
;  2131          2238  7
;  2132          2239  7                 ! This type needs some more work.  Build a value
;  2133          2240  7                 ! descriptor for this TYPEID.
;  2134          2241  7                 !
;  2135          2242  6                 ELSE
;  2136          2243  7                     BEGIN
;  2137          2244  7                     VALPTR = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
;  2138          2245  7                     TYPEID = .SUBVECTOR[.I, DBG$L_PNSUB_TYPEID];
;  2139          2246  7                     VALPTR[DBG$B_DHDR_LANG] = .PRIM_DESC[DBG$B_DHDR_LANG];
;  2140          2247  7                     VALPTR[DBG$B_DHDR_KIND] = .TYPEID[RST$B_KIND];
;  2141          2248  7                     VALPTR[DBG$B_DHDR_FCODE] = .TYPEID[RST$B_FCODE];
;  2142          2249  7                     VALPTR[DBG$L_DHDR_TYPEID] = .TYPEID;
;  2143          2250  7                     VALPTR[DBG$L_DHDR_SYMIDO] = .PRIM_DESC[DBG$L_DHDR_SYMIDO];
;  2144          2251  7                     DBG$GL_CURRENT_PRIMARY = .PRIM_DESC;
;  2145          2252  7                     DBG$FILL_IN_VMS_DESC(.TYPEID[RST$B_FCODE], .TYPEID,
;  2146          2253  7                         .PRIM_DESC[DBG$L_DHDR_SYMIDO],
;  2147          2254  7                         VALPTR[DBG$A_VALUE_VMSDESC], BITLEN, BIT_OFFSET);
;  2148          2255  7                     VALPTR[DBG$L_VALUE_POINTER] = VALPTR[DBG$A_VALUE_ADDRESS];
;  2149          2256  7                     VALPTR[DBG$L_VALUE_VALUE0] = .SUBVECTOR[.I, DBG$L_PNSUB_SVALUE];
;  2150          2257  7                     STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2151          2258  7                                         .VALPTR, TRUE);
;  2152          2259  6                     END;
;  2153          2260  6
;  2154          2261  6
;  2155          2262  6                 ! Place in Array separator.
;  2156          2263  6                 !
;  2157          2264  6                 STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2158          2265  6                                     .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2159          2266  6
;  2160          2267  5                 END;            ! End of INCR getting the subscripts.
;  2161          2268  5
;  2162          2269  5
;  2163          2270  5             ! Get offset, length, and sign.
;  2164          2271  5             !
;  2165          2272  5             IF .PRIM_DESC[DBG$V_DHDR_BLIBLK]
;  2166          2273  5             THEN
;  2167          2274  6                 BEGIN
;  2168          2275  6                 IF .PRIM_DESC[DBG$V_DHDR_SUBREF]
;  2169          2276  6                 THEN
```

```
;  2170          2277  7                    BEGIN
;  2171          2278  7
;  2172          2279  7                    ! Get offset.
;  2173          2280  7                    !
;  2174          2281  7                    GET_VALUE(PRIM_DESC[DBG$W_PRIM_OFFSET]);
;  2175          2282  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2176          2283  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2177          2284  7                                .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2178          2285  7
;  2179          2286  7                    ! Get length.
;  2180          2287  7                    !
;  2181          2288  7                    GET_VALUE(PRIM_DESC[DBG$W_PRIM_LENGTH]);
;  2182          2289  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2183          2290  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2184          2291  7                                .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2185          2292  7
;  2186          2293  7                    ! Get sign.
;  2187          2294  7                    !
;  2188          2295  7                    GET_VALUE(PRIM_DESC[DBG$V_DHDR_SGNEXT]);
;  2189          2296  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2190          2297  7                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2191          2298  7                                .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2192          2299  7                    END;
;  2193          2300  7
;  2194          2301  7                END;
;  2195          2302  6
;  2196          2303  6
;  2197          2304  5            IF .INFO_FLAG
;  2198          2305  5            THEN
;  2199          2306  5                TOP = .HERE
;  2200          2307  5
;  2201          2308  5            ELSE
;  2202          2309  5                BOTTOM = .HERE;
;  2203          2310  5
;  2204          2311  5                ! Put in close subscript, but not in COBOL case yet. We
;  2205          2312  5                ! have one separator too many, so we overlay the last one.
;  2206          2313  5                !
;  2207          2314  5                IF .INFO_FLAG
;  2208          2315  5                THEN
;  2209          2316  5                    BEGIN
;  2210          2317  5                    TOP = .TOP + 1;
;  2211          2318  5                    STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, TOP,
;  2212          2319  6                                .CHAR_TABLE[PRT$K_END_CHAR] + TABLEBASE);
;  2213          2320  6                    END;
;  2214          2321  6
;  2215          2322  6                END;                ! End of getting subscripts.
;  2216          2323  5
;  2217          2324  5
;  2218          2325  4            END;
;  2219          2326  4
;  2220          2327  3
;  2221          2328  3
;  2222          2329  3        ! Print general record/pointer format.
;  2223          2330  3        !
;  2224          2331  3        [PRT$K_RECORD,
;  2225          2332  3         PRT$K_POINTER]:
;  2226          2333  3
```

```
;  2227     2334  4              BEGIN
;  2228     2335  4              IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;  2229     2336  4              THEN
;  2230     2337  5                  BEGIN
;  2231     2338  5                  IF .FORMAT
;  2232     2339  5                  THEN
;  2233     2340  6                      BEGIN
;  2234     2341  6                      STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2235     2342  6                      STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;  2236     2343  6                                  .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2237     2344  5                      END;
;  2238     2345  5
;  2239     2346  5                  END
;  2240     2347  5
;  2241     2348  4              ELSE
;  2242     2349  4                  STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2243     2350  4
;  2244     2351  4              PTR_FLAG = FALSE;
;  2245     2352  4              IF .INFO_TABLE[.FCODE, PRTINFO$L_ROUT_INDEX] EQL PRT$K_POINTER
;  2246     2353  4              THEN
;  2247     2354  4                  PTR_FLAG = TRUE;
;  2248     2355  4
;  2249     2356  3              END;
;  2250     2357  3
;  2251     2358  3
;  2252     2359  3          ! In cobol there are record structure started with no name cases:
;  2253     2360  3          ! For example, in program FOO,
;  2254     2361  3          !     01.
;  2255     2362  3          !         02 A    PIC X.
;  2256     2363  3          !
;  2257     2364  3          ! We want to print it as "FOO\A", instead of "A of FOO".
;  2258     2365  3          ! So we use PATHNAME_FLAG to fudge this case.
;  2259     2366  3          !
;  2260     2367  3          ! Note: Since PLI and C go through different path of code,
;  2261     2368  3          ! so this fudge does not affect PLI and C.
;  2262     2369  3          !
;  2263     2370  3              [PRT$K_RECORD_COB]:
;  2264     2371  4                  BEGIN
;  2265     2372  4                  IF .PATHNAME_FLAG EQL 2
;  2266     2373  4                  THEN
;  2267     2374  5                      BEGIN
;  2268     2375  5                      IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;  2269     2376  5                      THEN
;  2270     2377  6                          BEGIN
;  2271     2378  6                          IF .FORMAT
;  2272     2379  6                          THEN
;  2273     2380  7                              BEGIN
;  2274     2381  7                              IF .NAMEPTR NEQ 0
;  2275     2382  7                              THEN
;  2276     2383  8                                  BEGIN
;  2277     2384  8                                  IF .NAMEPTR[0] NEQ 0
;  2278     2385  8                                  THEN
;  2279     2386  9                                      BEGIN
;  2280     2387  9                                      STKOVF = PRINT_PUSH(FALSE, STACK, BOTTOM, .NAMEPTR);
;  2281     2388  9                                      STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;  2282     2389  9                                              .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2283     2390  9                                      PATHNAME_FLAG = 3;
```

```
;  2284          2391  8                        END;
;  2285          2392  7                    END;
;  2286          2393  6                END;
;  2287          2394  6            END
;  2288          2395  5        ELSE
;  2289          2396  6            BEGIN
;  2290          2397  6            IF .NAMEPTR NEQ 0
;  2291          2398  6            THEN
;  2292          2399  7                BEGIN
;  2293          2400  7                IF .NAMEPTR[0] NEQ 0
;  2294          2401  7                THEN
;  2295          2402  8                    BEGIN
;  2296          2403  8                    STKOVF = PRINT_PUSH(FALSE, STACK, BOTTOM, .NAMEPTR);
;  2297          2404  8                    PATHNAME_FLAG = 3;
;  2298          2405  7                    END;
;  2299          2406  6                END;
;  2300          2407  5            END;
;  2301          2408  5
;  2302          2409  5            END
;  2303          2410  5
;  2304          2411  5        ! Back to the normal case.
;  2305          2412  5        !
;  2306          2413  4        ELSE
;  2307          2414  5            BEGIN
;  2308          2415  5            IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;  2309          2416  5            THEN
;  2310          2417  6                BEGIN
;  2311          2418  6                IF .FORMAT
;  2312          2419  6                THEN
;  2313          2420  7                    BEGIN
;  2314          2421  7                    IF .NAMEPTR NEQ 0
;  2315          2422  7                    THEN
;  2316          2423  8                        BEGIN
;  2317          2424  8                        IF .NAMEPTR[0] NEQ 0
;  2318          2425  8                        THEN
;  2319          2426  9                            BEGIN
;  2320          2427  9                            STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2321          2428  9                            STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;  2322          2429  9                                .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2323          2430  8                            END;
;  2324          2431  8                        END
;  2325          2432  7                    ELSE
;  2326          2433  7                        STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;  2327          2434  7                            .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2328          2435  6                    END;
;  2329          2436  6                END
;  2330          2437  5            ELSE
;  2331          2438  6                BEGIN
;  2332          2439  6                IF .NAMEPTR NEQ 0
;  2333          2440  6                THEN
;  2334          2441  7                    BEGIN
;  2335          2442  7                    IF .NAMEPTR[0] NEQ 0
;  2336          2443  7                    THEN
;  2337          2444  7                        STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2338          2445  6                    END;
;  2339          2446  5                END;
;  2340          2447  4            END;
```

```
;  2341          2448  4
;  2342          2449  4                    PTR_FLAG = FALSE;
;  2343          2450  4                    IF .INFO_TABLE[.FCODE, PRTINFO$L_ROUT_INDEX] EQL PRT$K_POINTER
;  2344          2451  4                    THEN
;  2345          2452  4                        PTR_FLAG = TRUE;
;  2346          2453  4
;  2347          2454  3                    END;
;  2348          2455  3
;  2349          2456  3
;  2350          2457  3                ! Print C pointer.
;  2351          2458  3                !
;  2352          2459  3                [PRT$K_POINTER_C]:
;  2353          2460  4                    BEGIN
;  2354          2461  4                    NEXT_NODE = .PRIM_NODE[DBG$L_PNODE_FLINK];
;  2355          2462  4                    IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;  2356          2463  4                    THEN
;  2357          2464  5                        BEGIN
;  2358          2465  5                        IF .FORMAT
;  2359          2466  5                        THEN
;  2360          2467  6                            BEGIN
;  2361          2468  6                            IF ..PRIM_NODE[0,0,32,0] EQL .PRIM_HEAD OR
;  2362          2469  6                               .NEXT_NODE[DBG$B_PNODE_FCODE] EQL RST$K_TYPE_TPTR OR
;  2363          2470  6                               .NEXT_NODE[DBG$B_PNODE_FCODE] EQL RST$K_TYPE_PTR
;  2364          2471  6                            THEN
;  2365          2472  7                                BEGIN
;  2366          2473  7                                STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2367          2474  7                                STKOVF = PRINT_PUSH(FALSE, STACK, BOTTOM,
;  2368          2475  7                                        .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2369          2476  7                                END
;  2370          2477  7
;  2371          2478  6                            ELSE
;  2372          2479  7                                BEGIN
;  2373          2480  7                                STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2374          2481  7                                STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;  2375          2482  7                                        .CHAR_TABLE[PRT$K_RECPTR_CHAR] + TABLEBASE);
;  2376          2483  7                                PTR_FLAG = TRUE;
;  2377          2484  6                                END;
;  2378          2485  6
;  2379          2486  5                            END;
;  2380          2487  5
;  2381          2488  5                        END
;  2382          2489  5
;  2383          2490  4                    ELSE
;  2384          2491  4                        STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;  2385          2492  4
;  2386          2493  3                    END;
;  2387          2494  3
;  2388          2495  3
;  2389          2496  3                ! Print C and PLI record format.
;  2390          2497  3                !
;  2391          2498  3                [PRT$K_RECORD_C_PLI]:
;  2392          2499  4                    BEGIN
;  2393          2500  4                    IF .PRIM_NODE[DBG$V_PNODE_EVAL]
;  2394          2501  4                    THEN
;  2395          2502  5                        BEGIN
;  2396          2503  5                        IF .FORMAT
;  2397          2504  5                        THEN
```

```
;2398     2505  6                   BEGIN
;2399     2506  6                   IF .PTR_FLAG
;2400     2507  6                   THEN
;2401     2508  6                       STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR)
;2402     2509  6
;2403     2510  6                   ELSE
;2404     2511  7                       BEGIN
;2405     2512  7                       STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;2406     2513  7                       STKOVF = PRINT_PUSH(TRUE, STACK, TOP,
;2407     2514  7                                .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;2408     2515  7
;2409     2516  6                       END;
;2410     2517  6
;2411     2518  5                   END;
;2412     2519  5
;2413     2520  5               END
;2414     2521  5
;2415     2522  4           ELSE
;2416     2523  4               STKOVF = PRINT_PUSH(TRUE, STACK, TOP, .NAMEPTR);
;2417     2524  4
;2418     2525  4           PTR_FLAG = FALSE;
;2419     2526  3           END;
;2420     2527  3
;2421     2528  3       ! If this is a Variant Sub-Node, do nothing.  Record Variants have
;2422     2529  3       ! no effect on the printed variable name, so we simply skip this
;2423     2530  3       ! Sub-Node and go on to the next one.
;2424     2531  3       !
;2425     2532  3       [PRT$K_VARIANT]:
;2426     2533  3           0;
;2427     2534  3
;2428     2535  3
;2429     2536  3           ! If we get here, there is an invalid print routine index in the
;2430     2537  3           ! Print Information Table.  Signal an internal error.
;2431     2538  3           !
;2432     2539  3           [INRANGE, OUTRANGE]:
;2433     2540  3               $DBG_ERROR('DBGPRINT\DBG$PRINT_IDENTIFIER unknown print index');
;2434     2541  3
;2435     2542  3           TES;                        ! End of CASE on Print Routine Index.
;2436     2543  3
;2437     2544  2       END;                            ! End of WHILE Primary Descriptor Loop.
;2438     2545  2
;2439     2546  2   ! Fix up array close subscript for COBOL.
;2440     2547  2   !
;2441     2548  2   IF NOT .INFO_FLAG
;2442     2549  2   THEN
;2443     2550  2       BEGIN
;2444     2551  3       IF .BOTTOM GTR PRINT_STACK_START AND
;2445     2552  3          .PATHNAME_FLAG EQL 0
;2446     2553  3       THEN
;2447     2554  4           BEGIN
;2448     2555  4           BOTTOM = .BOTTOM - 1;
;2449     2556  4           CHAR_TABLE = .INFO_TABLE[RST$K_TYPE_ARRAY, PRTINFO$L_CHARTBL] + TABLEBASE;
;2450     2557  4           STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, BOTTOM,
;2451     2558  4                               .CHAR_TABLE[PRT$K_END_CHAR] + TABLEBASE);
;2452     2559  4           END;
;2453     2560  4
;2454     2561  3
```

```
;  2455          2562  3
;  2456          2563  2          END;
;  2457          2564  2
;  2458          2565  2
;  2459          2566  2      ! Get the reference modifier.  For example, in COBOL, FORTRAN, one can say
;  2460          2567  2      ! CHAR(1,10)(2:3). Or if this is a bit-string sub-reference, get that.
;  2461          2568  2      !
;  2462          2569  2      IF .PRIM_DESC[DBG$V_DHDR_SUBREF]
;  2463          2570  2      THEN
;  2464          2571  3          BEGIN
;  2465          2572  3          IF NOT .PRIM_DESC[DBG$V_DHDR_BLIBLK]
;  2466          2573  3          THEN
;  2467          2574  4              BEGIN
;  2468          2575  4              IF NOT .PRIM_DESC[DBG$V_DHDR_AGGR]
;  2469          2576  4              THEN
;  2470          2577  5                  BEGIN
;  2471          2578  5                  CHAR_TABLE = .INFO_TABLE[RST$K_TYPE_COMMON,
;  2472          2579  5                                           PRTINFO$L_CHARTBL] + TABLEBASE;
;  2473          2580  5                  HERE = (IF .INFO_FLAG THEN .TOP ELSE .BOTTOM);
;  2474          2581  5
;  2475          2582  5
;  2476          2583  5                  ! Place in the begin reference modifier subscript.
;  2477          2584  5                  !
;  2478          2585  5                  STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2479          2586  5                              .CHAR_TABLE[PRT$K_BEGIN_CHAR] + TABLEBASE);
;  2480          2587  5
;  2481          2588  5
;  2482          2589  5                  ! Get the value.
;  2483          2590  5                  !
;  2484          2591  5                  GET_VALUE(PRIM_DESC[DBG$W_PRIM_OFFSET]);
;  2485          2592  5                  STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2486          2593  5                  STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2487          2594  5                              .CHAR_TABLE[PRT$K_SEPARATOR_CHAR] + TABLEBASE);
;  2488          2595  5
;  2489          2596  5
;  2490          2597  5                  ! Get the value.
;  2491          2598  5                  !
;  2492          2599  5                  IF .REFMOD_FLAG
;  2493          2600  5                  THEN
;  2494          2601  5                      VALUE = .PRIM_DESC[DBG$W_PRIM_OFFSET] +
;  2495          2602  5                              .PRIM_DESC[DBG$W_PRIM_LENGTH] - 1
;  2496          2603  5
;  2497          2604  5                  ELSE
;  2498          2605  5                      VALUE = .PRIM_DESC[DBG$W_PRIM_LENGTH];
;  2499          2606  5
;  2500          2607  5                  GET_VALUE(VALUE);
;  2501          2608  5                  STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE, .NAMEPTR);
;  2502          2609  5                  STKOVF = PRINT_PUSH(.INFO_FLAG, STACK, HERE,
;  2503          2610  5                                      .CHAR_TABLE[PRT$K_END_CHAR] + TABLEBASE);
;  2504          2611  5
;  2505          2612  5                  IF .INFO_FLAG
;  2506          2613  5                  THEN
;  2507          2614  5                      TOP = .HERE
;  2508          2615  5
;  2509          2616  5                  ELSE
;  2510          2617  5                      BOTTOM = .HERE;
;  2511          2618  5
```

```
;  2512          2619  4                 END;
;  2513          2620  4
;  2514          2621  3             END;                    ! End of getting reference modifier.
;  2515          2622  3
;  2516          2623  2         END;                        ! End of getting sub-reference.
;  2517          2624  2
;  2518          2625  2
;  2519          2626  2         ! If there is any truncation during the process, give an informational
;  2520          2627  2         ! signal.
;  2521          2628  2         !
;  2522          2629  2         IF .STKOVF OR .SUBOVF THEN SIGNAL(DBG$_STGTRUNC);
;  2523          2630  2
;  2524          2631  2
;  2525          2632  2         TOP = MAX(.TOP + 1, 0);
;  2526          2633  2         BOTTOM = MIN(.BOTTOM, PRINT_STACK_SIZE - 1);
;  2527          2634  2
;  2528          2635  2
;  2529          2636  2         ! Go through the pointers to ASCIC strings, accumulated the strings.
;  2530          2637  2         ! Print the stack from bottom to top direction.
;  2531          2638  2         !
;  2532          2639  2         IF .INFO_FLAG
;  2533          2640  2         THEN
;  2534          2641  2             DECR I FROM .BOTTOM TO .TOP DO
;  2535          2642  3                 BEGIN
;  2536          2643  3                 IF .STACK[.I, PRTID$STACK_FLAG]
;  2537          2644  3                 THEN
;  2538          2645  3                     DBG$PRINT_VALUE(.STACK[.I, PRTID$STACK_PTR], DBG$K_DEFAULT, FALSE, FALSE)
;  2539          2646  3
;  2540          2647  3                 ELSE
;  2541          2648  3                     DBG$PRINT(UPLIT BYTE(%ASCIC '!AC'), .STACK[.I, PRTID$STACK_PTR]);
;  2542          2649  3
;  2543          2650  3                 END
;  2544          2651  3
;  2545          2652  3
;  2546          2653  3         ! Print the stack from top to bottom direction.
;  2547          2654  3         !
;  2548          2655  2         ELSE
;  2549          2656  3             BEGIN
;  2550          2657  3             INCR I FROM .TOP TO .BOTTOM DO
;  2551          2658  4                 BEGIN
;  2552          2659  4                 IF .STACK[.I, PRTID$STACK_FLAG]
;  2553          2660  4                 THEN
;  2554          2661  4                     DBG$PRINT_VALUE(.STACK[.I, PRTID$STACK_PTR], DBG$K_DEFAULT, FALSE, FALSE)
;  2555          2662  4                 ELSE
;  2556          2663  4                     DBG$PRINT(UPLIT BYTE(%ASCIC '!AC'), .STACK[.I, PRTID$STACK_PTR]);
;  2557          2664  4
;  2558          2665  3                 END;
;  2559          2666  3
;  2560          2667  2             END;
;  2561          2668  2
;  2562          2669  2
;  2563          2670  2         ! We are all done with the printing, print the offsets if there is any.
;  2564          2671  2         ! Do some clean up and return.
;  2565          2672  2         !
;  2566          2673  2         PRINT_OFFSETS_AND_RETURN;
;  2567          2674  2
;  2568          2675  1         END;
```

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

                                                    5C  01  00CDE P.ADU:  .ASCII  <1><92>                                                ;
                                                43  41  21  03  00CE0 P.ADV:  .ASCII  <3>\.AC\                                           ;
                                                43  41  21  03  00CE4 P.ADW:  .ASCII  <3>\!AC\                                           ;
                                                4C  53  21  03  00CE8 P.ADX:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00CEC P.ADY:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00CF0 P.ADZ:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00CF4 P.AEA:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00CF8 P.AEB:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00CFC P.AEC:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00D00 P.AED:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00D04 P.AEE:  .ASCII  <3>\!SL\                                           ;
50  24  47  42  44  5C  54  4E  49  52  50  47  42  44  31  00D08 P.AEF:  .ASCII  \1DBGPRINT\<92>\DBG$PRINT_IDENTIFIER unk\                 ;
52  45  49  46  49  54  4E  45  44  49  5F  54  4E  49  52  00D17                                                                        ;
                                            6B  6E  75  20  00D26                                                                        ;
65  64  6E  69  20  74  6E  69  72  70  20  6E  77  6F  6E  00D2A          .ASCII  \nown print index\                                       ;
                                                        78  00D39                                                                        ;
                                                4C  53  21  03  00D3A P.AEG:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00D3E P.AEH:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00D42 P.AEI:  .ASCII  <3>\!SL\                                           ;
                                                4C  53  21  03  00D46 P.AEJ:  .ASCII  <3>\!SL\                                           ;
                                                43  41  21  03  00D4A P.AEK:  .ASCII  <3>\!AC\                                           ;
                                                43  41  21  03  00D4E P.AEL:  .ASCII  <3>\.AC\                                           ;

                                                                BACK_SLASH=         P.ADU

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

                                                        0FFC 00000          .ENTRY  DBG$PRINT_IDENTIFIER, Save R2,R3,R4,R5,R6,-          ; 1707
                                                                                    R7,R8,R9,R10,R11                                     ;
                                            5E      FD8C    CE  9E 00002            MOVAB   -628(SP), SP                                 ;
                                            7E 00000000G    00  9A 00007            MOVZBL  DBG$GB_LANGUAGE, -(SP)                       ; 1789
                                    0000V   CF              01  FB 0000E            CALLS   #1, DBG$PRINT_SET_LANGUAGE                   ;
                                                            54  D4 00013            CLRL    VAL_DESC                                     ; 1796
                                                        08  AE  D4 00015            CLRL    BYTE_OFFSET                                  ; 1797
                                                        54  AE  D4 00018            CLRL    BIT_OFFSET                                   ; 1798
00000083    8F      04  BC              08                  10  ED 0001B            CMPZV   #16, #8, @PRIM_DESC, #131                    ; 1799
                                                            03  13 00025            BEQL    1$                                           ;
                                                        014D    31 00027            BRW     14$                                          ;
                                            54          04  AC  D0 0002A 1$:       MOVL    PRIM_DESC, VAL_DESC                          ; 1802
                                                        04  AE  D4 0002E            CLRL    SAVE_VAL_DTYPE                               ; 1803
                                            52          14  A4  9E 00031            MOVAB   20(VAL_DESC), R2                             ; 1804
                                            6E              62  3C 00035            MOVZWL  (R2), SAVE_VAL_LENGTH                        ;
                                                        18  A4  DD 00038            PUSHL   24(VAL_DESC)                                 ; 1820
                                    00000000G   00              01  FB 0003B            CALLS   #1, DBG$STA_ADDRESS_TO_REGDESCR              ;
                                                            50  D5 00042            TSTL    REGDESCR                                     ; 1821
                                                            1D  13 00044            BEQL    2$                                           ;
                                                            50  DD 00046            PUSHL   REGDESCR                                     ; 1824
                                    00000000G   00              01  FB 00048            CALLS   #1, DBG$STA_REGISTER_NAME                    ;
                                          64    AE              50  D0 0004F            MOVL    R0, NAMEPTR                                  ;
                                                        64  AE  DD 00053            PUSHL   NAMEPTR                                      ; 1825
                                                00000000'   EF  9F 00056            PUSHAB  P.ADV                                        ;
```

```
                FBB6   CF                02   FB 0005C          CALLS   #2, DBG$PRINT                                ;
                                         58   11 00061          BRB     7$                                          ; 1826
                       50 00000000G      00   D0 00063 2$:      MOVL    DBG$GB_MOD_PTR, R0                          ; 1833
                       27          02    A0   E9 00066          BLBC    2(R0), 5$                                   ;
                   F8  AD          18    A4   D0 0006A          MOVL    24(VAL_DESC), ADDRESS                       ; 1836
                       0D          03    A2   91 0006E          CMPB    3(R2), #13                                  ; 1837
                                         07   12 00073          BNEQ    3$                                          ;
                   FC  AD          1C    A4   D0 00075          MOVL    28(VAL_DESC), ADDRESS+4                     ; 1839
                                         03   11 0007A          BRB     4$                                          ;
                                   FC    AD   D4 0007C 3$:      CLRL    ADDRESS+4                                   ; 1842
                                   54    AE   9F 0007F 4$:      PUSHAB  BIT_OFFSET                                  ; 1844
                                   4C    AE   9F 00082          PUSHAB  SYMID                                       ;
                                   F8    AD   9F 00085          PUSHAB  ADDRESS                                     ;
                00000000G          00        FB 00088          CALLS   #3, DBG$NEW_SYMBOLIZE                       ;
                                         03   11 0008F          BRB     6$                                          ; 1833
                                   48    AE   D4 00091 5$:      CLRL    SYMID                                       ; 1851
                                   48    AE   D5 00094 6$:      TSTL    SYMID                                       ; 1857
                                         21   12 00097          BNEQ    8$                                          ;
                   70  AE 00040004          8F   D0 00099          MOVL    #262148, VMS_DESC                           ; 1867
                   74  AE          18    A4   9E 000A1          MOVAB   24(VAL_DESC), VMS_DESC+4                    ; 1868
                                   70    AE   9F 000A6          PUSHAB  VMS_DESC                                    ; 1869
                00000000G          00        FB 000A9          CALLS   #1, DBG$PRINT_VALUE_AS_INTEGER              ;
                                         54   DD 000B0          PUSHL   VAL_DESC                                    ; 1875
                    FEB7   CF            01   FB 000B2          CALLS   #1, DBG$PRINT_FIELD_REF                     ;
                                      0A26    31 000B7 7$:      BRW     133$                                        ; 1876
                                   54    AE   9F 000BA 8$:      PUSHAB  BIT_OFFSET                                  ; 1916
                                   4C    AE   DD 000BD          PUSHL   SYMID                                       ;
                00000000G          00        FB 000C0          CALLS   #2, DBG$SYMID_TO_PRIMARY                    ;
                       04  AC              50   D0 000C7          MOVL    R0, PRIM_DESC                               ;
                           50          54    AE   D0 000CB          MOVL    BIT_OFFSET, R0                              ; 1917
                                         7A   13 000CF          BEQL    12$                                         ;
                           29          02    A2   91 000D1          CMPB    2(R2), #41                                  ; 1920
                                         32   13 000D5          BEQL    9$                                          ;
                           2A          02    A2   91 000D7          CMPB    2(R2), #42                                  ; 1921
                                         2C   13 000DB          BEQL    9$                                          ;
                           01          02    A2   91 000DD          CMPB    2(R2), #1                                   ; 1922
                                         26   13 000E1          BEQL    9$                                          ;
                           22          02    A2   91 000E3          CMPB    2(R2), #34                                  ; 1923
                                         20   13 000E7          BEQL    9$                                          ;
                           0D          03    A2   91 000E9          CMPB    3(R2), #13                                  ; 1924
                                         1A   13 000ED          BEQL    9$                                          ;
                       51                50   08   C7 000EF          DIVL3   #8, R0, R1                                  ; 1935
                                         51   08   C4 000F3          MULL2   #8, R1                                      ;
                                         51   50   D1 000F6          CMPL    R0, R1                                      ;
                                         0E   13 000F9          BEQL    9$                                          ;
                                   04    AE   02   A2 000FB          MOVZBL  2(R2), SAVE_VAL_DTYPE                       ; 1938
                                   02    A2   01   90 00100          MOVB    #1, 2(R2)                                   ; 1939
                                         62   08   A4 00104          MULW2   #8, (R2)                                    ; 1940
                                         03   11 00107          BRB     10$                                         ; 1935
                                   04    AE   D4 00109 9$:      CLRL    SAVE_VAL_DTYPE                              ; 1944
                08  AE                50   08   C7 0010C 10$:      DIVL3   #8, R0, BYTE_OFFSET                         ; 1947
                    50          08    AE   03   78 00111          ASHL    #3, BYTE_OFFSET, R0                         ; 1948
                                   54    AE   50   C2 00116          SUBL2   R0, BIT_OFFSET                              ;
                                         29   02   A2 91 0011A          CMPB    2(R2), #41                                  ; 1949
                                         18   13 00120          BEQL    11$                                         ;
                                         2A   02   A2 91 00122          CMPB    2(R2), #42                                  ; 1950
                                         12   13 00126          BEQL    11$                                         ;
```

```
                                01      02   A2  91 0012A           CMPB    2(R2), #1                       ;1951
                                                0C  13 0012E           BEQL    11$
                                22      02   A2  91 00130           CMPB    2(R2), #34                      ;1952
                                                06  13 00134           BEQL    11$
                                0D      03   A2  91 00136           CMPB    3(R2), #13                      ;1953
                                                13  12 0013A           BNEQ    12$
                                        54   AE  D5 0013C 11$:       TSTL    BIT_OFFSET                      ;1956
                                                0E  12 0013F           BNEQ    12$
                                20           62  B1 00141           CMPW    (R2), #32                       ;1959
                                                09  1E 00144           BGEQU   12$
                54   AE          08      AE      03  78 00146           ASHL    #3, BYTE_OFFSET, BIT_OFFSET     ;1962
                                        08   AE  D4 0014C           CLRL    BYTE_OFFSET                     ;1963
                                        54   AE  D5 0014F 12$:       TSTL    BIT_OFFSET                      ;1975
                                                23  13 00152           BEQL    14$
                                29      02   A2  91 00154           CMPB    2(R2), #41                      ;1978
                                                18  13 00158           BEQL    13$
                                2A      02   A2  91 0015A           CMPB    2(R2), #42                      ;1979
                                                12  13 0015E           BEQL    13$
                                01      02   A2  91 00160           CMPB    2(R2), #1                       ;1980
                                                0C  13 00164           BEQL    13$
                                22      02   A2  91 00166           CMPB    2(R2), #34                      ;1981
                                                06  13 0016A           BEQL    13$
                                0D      03   A2  91 0016C           CMPB    3(R2), #13                      ;1982
                                                05  12 00170           BNEQ    14$
                        1C      A4      54   AE  D0 00172 13$:       MOVL    BIT_OFFSET, 28(VAL_DESC)        ;1984
                        10      AE          02  D0 00177 14$:       MOVL    #2, FORMAT                      ;1996
                                01          6C  91 0017B           CMPB    (AP), #1                        ;1997
                                                10  1B 0017E           BLEQU   16$
                                50      08   AC  D0 00180           MOVL    8(AP), R0
                                02          50  D1 00184           CMPL    R0, #2
                                                03  1B 00187           BLEQU   15$
                                50          02  D0 00189           MOVL    #2, R0
                            10  AE          50  D0 0018C 15$:       MOVL    R0, FORMAT
                                02      10   AE  D1 00190 16$:       CMPL    FORMAT, #2                      ;1998
                                                11  12 00194           BNEQ    17$
                                        4C   AE  9F 00196           PUSHAB  PATHNAME                        ;2000
                                50      04   AC  D0 00199           MOVL    PRIM_DESC, R0
                                        0C   A0  DD 0019D           PUSHL   12(R0)
                    00000000G   00          02  FB 001A0           CALLS   #2, DBG$STA_SYMPATHNAME
                                59      04   AC  D0 001A7 17$:       MOVL    PRIM_DESC, R9                   ;2005
                            30  AE          04  A9 9E 001AB           MOVAB   4(R9), 48(SP)
00          30  BE              08          10  ED 001B0           CMPZV   #16, #8, @48(SP), #0
                                                03  13 001B6           BEQL    18$
                                          0087  31 001B8           BRW     27$
                                02      10   AE  D1 001BB 18$:       CMPL    FORMAT, #2                      ;2008
                                                0F  18 001BF           BGEQ    19$
                                        64   AE  9F 001C1           PUSHAB  NAMEPTR                         ;2010
                                        0C   A9  DD 001C4           PUSHL   12(R9)
                    00000000G   00          02  FB 001C7           CALLS   #2, DBG$STA_SYMNAME
                                                0D  11 001CE           BRB     20$
                                        64   AE  9F 001D0 19$:       PUSHAB  NAMEPTR                         ;2013
                                        50   AE  DD 001D3           PUSHL   PATHNAME
                    00000000G   00          02  FB 001D6           CALLS   #2, DBG$NPATHDESC_TO_CS
                                        64   AE  DD 001DD 20$:       PUSHL   NAMEPTR                         ;2015
                                00000000'   EF  9F 001E0           PUSHAB  P.ADW
                            FA2C    CF          02  FB 001E6           CALLS   #2, DBG$PRINT
                                        08   AE  D5 001EB           TSTL    BYTE_OFFSET
```

```
                                      08   13 001EE        BEQL    21$                                      ;
                                 08   AE   DD 001F0        PUSHL   BYTE_OFFSET                              ;
                     0000V  CF        01   FB 001F3        CALLS   #1, DBG$PRINT_OFFSET                     ;
                                      54   D5 001F8 21$:   TSTL    VAL_DESC                                 ;
                                      03   12 001FA        BNEQ    22$                                      ;
                                    08CA   31 001FC        BRW     131$                                     ;
                                 54   AE   D5 001FF 22$:   TSTL    BIT_OFFSET                               ;
                                      2C   12 00202        BNEQ    24$                                      ;
  06      30   BE           08        18   ED 00204        CMPZV   #24, #8, @48(SP), #6                     ;
                                      08   13 0020A        BEQL    23$                                      ;
  0A      30   BE           08        18   ED 0020C        CMPZV   #24, #8, @48(SP), #10                    ;
                                      1C   12 00212        BNEQ    24$                                      ;
                                 50   AE   9F 00214 23$:   PUSHAB  TMP_VALDESC                              ;
                            7E   83   8F   9A 00217        MOVZBL  #131, -(SP)                              ;
                                      59   DD 0021B        PUSHL   R9                                       ;
                 00000000G  00        03   FB 0021D        CALLS   #3, DBG$PRIM_TO_VAL                      ;
                            50   50   AE   D0 00224        MOVL    TMP_VALDESC, R0                          ;
  6E      14   A0           10        00   ED 00228        CMPZV   #0, #16, 20(R0), SAVE_VAL_LENGTH         ;
                                      07   13 0022E        BEQL    25$                                      ;
                                      54   DD 00230 24$:   PUSHL   VAL_DESC                                 ;
                     FD3B   CF        01   FB 00232        CALLS   #1, DBG$PRINT_FIELD_REF                  ;
                                 04   AE   D5 00237 25$:   TSTL    SAVE_VAL_DTYPE                           ;
                                      03   12 0023A        BNEQ    26$                                      ;
                                    0872   31 0023C        BRW     130$                                     ;
                                    0866   31 0023F 26$:   BRW     129$                                     ;
                       1C   AE      03    A9   9A 00242 27$: MOVZBL 3(R9), 28(SP)                           ; 2025
                            50 00000000'  EF   9E 00247    MOVAB   TABLEBASE, R0                            ;
                            51      1C    AE   D0 0024E    MOVL    28(SP), R1                               ;
                            50 00000000'EF41   C0 00252    ADDL2   LANGUAGE_PRINT_TABLE_PTRS[R1], PTR       ;
                            51 00000000'  EF   9E 0025A    MOVAB   TABLEBASE, R1                            ; 2026
                   5B       51            60   C1 00261    ADDL3   (PTR), R1, INFO_TABLE                    ;
                            5A      04    A0   D0 00265    MOVL    4(PTR), INFO_FLAG                        ; 2027
                       44   AE      08    A0   D0 00269    MOVL    8(PTR), REFMOD_FLAG                      ; 2028
                       60   AE            3C   D0 0026E    MOVL    #60, BOTTOM                              ; 2029
                       5C   AE            3C   D0 00272    MOVL    #60, TOP                                 ;
                                 28   AE   7C 00276        CLRQ    SUBOVF                                   ; 2033
                                 3C   AE   7C 00279        CLRQ    ARRAY_FLAG                               ; 2030
                                 14   AE   D4 0027C        CLRL    PATHNAME_FLAG                            ; 2034
                       34   AE   14   A9   9E 0027F        MOVAB   20(R9), PRIM_HEAD                        ; 2042
                            53   34   AE   D0 00284        MOVL    PRIM_HEAD, PRIM_NODE                     ; 2043
                            53        63   D0 00288 28$:   MOVL    (PRIM_NODE), PRIM_NODE                   ; 2051
                       34   AE        53   D1 0028B        CMPL    PRIM_NODE, PRIM_HEAD                     ; 2052
                                      03   12 0028F        BNEQ    29$                                      ;
                                    05A9   31 00291        BRW     98$                                      ;
                       0C   A9   10   A3   D1 00294 29$:   CMPL    16(PRIM_NODE), 12(R9)                    ; 2061
                                      63   12 00299        BNEQ    31$                                      ;
                                 64   AE   9F 0029B        PUSHAB  NAMEPTR                                  ; 2064
                                 10   A3   DD 0029E        PUSHL   16(PRIM_NODE)                            ;
                 00000000G  00        02   FB 002A1        CALLS   #2, DBG$STA_SYMNAME                      ;
                                 64   BE   95 002A8        TSTB    @NAMEPTR                                 ; 2069
                                      04   12 002AB        BNEQ    30$                                      ;
                       14   AE        01   D0 002AD        MOVL    #1, PATHNAME_FLAG                        ;
                            02   10   AE   D1 002B1 30$:   CMPL    FORMAT, #2                               ; 2070
                                      5E   19 002B5        BLSS    33$                                      ;
                                 64   AE   9F 002B7        PUSHAB  NAMEPTR                                  ; 2073
                                 50   AE   DD 002BA        PUSHL   PATHNAME                                 ;
                 00000000G  00        02   FB 002BD        CALLS   #2, DBG$NPATHDESC_TO_CS                  ;
```

```
                    01        14    AE  D1 002C4          CMPL    PATHNAME_FLAG, #1                       ; 2084
                                    4B  12 002C8          BNEQ    33$                                     ;
                         00000000'  EF  9F 002CA          PUSHAB  BACK_SLASH                              ; 2087
                                60  AE  9F 002D0          PUSHAB  TOP                                     ;
                              0080  CE  9F 002D3          PUSHAB  STACK                                   ;
                                    01  DD 002D7          PUSHL   #1                                      ;
                0000V       CF          04  FB 002D9          CALLS   #4, PRINT_PUSH                      ;
                  40        AE          50  D0 002DE          MOVL    R0, STKOVF                          ;
                                64  AE  DD 002E2          PUSHL   NAMEPTR                                 ; 2088
                                60  AE  9F 002E5          PUSHAB  TOP                                     ;
                              0080  CE  9F 002E8          PUSHAB  STACK                                   ;
                                    01  DD 002EC          PUSHL   #1                                      ;
                0000V       CF          04  FB 002EE          CALLS   #4, PRINT_PUSH                      ;
                  40        AE          50  D0 002F3          MOVL    R0, STKOVF                          ;
                  14        AE          02  D0 002F7          MOVL    #2, PATHNAME_FLAG                   ; 2089
                            53          63  D0 002FB          MOVL    (PRIM_NODE), PRIM_NODE              ; 2094
                                10  A3  D5 002FE 31$:        TSTL    16(PRIM_NODE)                       ; 2112
                                    0F  13 00301          BEQL    32$                                     ;
                                64  AE  9F 00303          PUSHAB  NAMEPTR                                 ; 2114
                                10  A3  DD 00306          PUSHL   16(PRIM_NODE)                           ;
                00000000G   00          02  FB 00309          CALLS   #2, DBG$STA_SYMNAME                 ;
                                    03  11 00310          BRB     33$                                     ;
                                64  AE  D4 00312 32$:      CLRL    NAMEPTR                                 ; 2117
                                10  AE  D5 00315 33$:      TSTL    FORMAT                                  ; 2125
                                    05  12 00318          BNEQ    34$                                     ;
                                10  AE  D4 0031A          CLRL    FORMAT                                  ;
                                    04  11 0031D          BRB     35$                                     ;
                  10        AE          01  D0 0031F 34$:      MOVL    #1, FORMAT                          ;
                            58        08  A3  9E 00323 35$:      MOVAB   8(PRIM_NODE), R8                    ; 2126
                            57        01  A8  9A 00327          MOVZBL  1(R8), FCODE                        ;
                            50 00000000'  EF  9E 0032B          MOVAB   TABLEBASE, R0                       ; 2127
                                    04 AB47  7F 00332          PUSHAQ  4(INFO_TABLE)[FCODE]                ;
                        52              50      9E  C1 00336          ADDL3   @(SP)+, R0, CHAR_TABLE          ;
                                      6B47  7F 0033A          PUSHAQ  (INFO_TABLE)[FCODE]                 ; 2128
                        07              00      9E  CF 0033D          CASEL   @(SP)+, #0, #7                  ;
0333                    0333          004C      0027      00341 36$:      .WORD   37$-36$,-                   ;
FF47                    04B3          040F      0347      00349           40$-36$,-                           ;
                                                                          72$-36$,-                           ;
                                                                          72$-36$,-                           ;
                                                                          74$-36$,-                           ;
                                                                          86$-36$,-                           ;
                                                                          93$-36$,-                           ;
                                                                          28$-36$                             ;
                                00000000'  EF  9F 00351          PUSHAB  P.AEF                               ; 2541
                                    01  DD 00357          PUSHL   #1                                      ;
                                00028362  8F  DD 00359          PUSHL   #164706                             ;
                00000000G   00          03  FB 0035F          CALLS   #3, LIB$SIGNAL                      ;
                                    22  11 00366          BRB     39$                                     ;
                    02        14    AE  D1 00368 37$:      CMPL    PATHNAME_FLAG, #2                       ; 2138
                                    03  13 0036C          BEQL    38$                                     ;
                                  046C  31 0036E          BRW     91$                                     ;
                                64  AE  DD 00371 38$:      PUSHL   NAMEPTR                                 ; 2141
                                64  AE  9F 00374          PUSHAB  BOTTOM                                  ;
                              0080  CE  9F 00377          PUSHAB  STACK                                   ;
                                    7E  D4 0037B          CLRL    -(SP)                                   ;
                0000V       CF          04  FB 0037D          CALLS   #4, PRINT_PUSH                      ;
                  40        AE          50  D0 00382          MOVL    R0, STKOVF                          ;
```

```
                14   AE          03   D0 00386          MOVL    #3, PATHNAME_FLAG                       ; 2142
                          FEFB   31   00038A 39$:       BRW     28$                                     ; 2138
                     02   14   AE   D1 00038D 40$:      CMPL    PATHNAME_FLAG, #2                       ; 2160
                          1B   12   00391               BNEQ    41$                                     ;
                     64   AE   DD   00393               PUSHL   NAMEPTR                                 ; 2163
                     64   AE   9F   00396               PUSHAB  BOTTOM                                  ;
                   0080   CE   9F   00399               PUSHAB  STACK                                   ;
                          7E   D4   0039D               CLRL    -(SP)                                   ;
              0000V   CF          04   FB 0039F         CALLS   #4, PRINT_PUSH                          ;
                40   AE          50   D0 003A4         MOVL    R0, STKOVF                              ;
                14   AE          03   D0 003A8         MOVL    #3, PATHNAME_FLAG                       ; 2164
                          1D   11   003AC               BRB     43$                                     ; 2160
                     04   02   A8   E9 003AE 41$:      BLBC    2(R8), 42$                              ; 2173
                     19   10   AE   E9 003B2           BLBC    FORMAT, 44$                             ; 2176
                     64   AE   DD   003B6 42$:         PUSHL   NAMEPTR                                 ; 2183
                     60   AE   9F   003B9               PUSHAB  TOP                                     ;
                   0080   CE   9F   003BC               PUSHAB  STACK                                   ;
                          01   DD   003C0               PUSHL   #1                                      ;
              0000V   CF          04   FB 003C2         CALLS   #4, PRINT_PUSH                          ;
                40   AE          50   D0 003C7         MOVL    R0, STKOVF                              ;
                     0F   10   AE   E8 003CB 43$:      BLBS    FORMAT, 46$                             ; 2193
                     07   02   A8   E9 003CF 44$:      BLBC    2(R8), 45$                              ; 2194
                34   AE   00   B3   D1 003D3           CMPL    @0(PRIM_NODE), PRIM_HEAD                ; 2195
                          08   13   003D8               BEQL    47$                                     ;
                     AC   10   AE   E9 003DA 45$:      BLBC    FORMAT, 39$                             ; 2196
                     A8   02   A8   E9 003DE 46$:      BLBC    2(R8), 39$                              ; 2197
                     14   AE   D4   003E2 47$:         CLRL    PATHNAME_FLAG                           ; 2206
                     07          5A   E9 003E5         BLBC    INFO_FLAG, 48$                          ; 2207
                68   AE   5C   AE   D0 003E8           MOVL    TOP, HERE                               ; 2209
                          05   11   003ED               BRB     49$                                     ;
                68   AE   60   AE   D0 003EF 48$:      MOVL    BOTTOM, HERE                            ; 2212
                     04          5A   E8 003F4 49$:      BLBS    INFO_FLAG, 50$                          ; 2217
                     21   3C   AE   E8 003F7           BLBS    ARRAY_FLAG, 51$                         ;
                     50 00000000'  EF   9E 003FB 50$:      MOVAB   TABLEBASE, R0                           ; 2221
                              00 B240   9F   00402       PUSHAB  @0(CHAR_TABLE)[R0]                      ;
                              6C   AE   9F   00406       PUSHAB  HERE                                    ; 2220
                            0080   CE   9F   00409       PUSHAB  STACK                                   ;
                                   5A   DD   0040D       PUSHL   INFO_FLAG                               ;
                     0000V   CF          04   FB 0040F   CALLS   #4, PRINT_PUSH                          ;
                       40   AE          50   D0 00414   MOVL    R0, STKOVF                              ;
                       3C   AE          01   D0 00418   MOVL    #1, ARRAY_FLAG                          ; 2222
                            56   28   A3   9E 0041C 51$:      MOVAB   40(R3), SUBVECTOR                       ; 2228
                       38   AE   1B   A3   9A 00420       MOVZBL  27(PRIM_NODE), 56(SP)                   ; 2229
                       24   AE          01   CE 00425   MNEGL   #1, I                                   ; 2265
                                    00EC   31   00429     BRW     57$                                     ;
          58           24   AE          14   C5 0042C 52$:      MULL3   #20, I, R8                      ; 2231
                              10 A846   9F   00431       PUSHAB  16(R8)[SUBVECTOR]                       ;
                       18   AE          9E   D0 00435   MOVL    @(SP)+, 24(SP)                          ;
                                   43   12   00439       BNEQ    55$                                     ;
                            19   28   AE   E8 0043B       BLBS    SUBOVF, 53$                             ; 2234
                                 6846   9F   0043F       PUSHAB  (R8)[SUBVECTOR]                         ;
                                   9E   DD   00442       PUSHL   @(SP)+                                  ;
                              00000000'  EF   9F 00444     PUSHAB  P.ADX                                   ;
                                   6C   AE   9F 0044A     PUSHAB  NAMEPTR                                 ;
                     0000V   CF          03   FB 0044D   CALLS   #3, GET_SUBSCRIPT_VALUE                 ;
                       28   AE          50   D0 00452   MOVL    R0, SUBOVF                              ;
                                        13   11 00456     BRB     54$                                     ;
```

```
                                6846  9F 00458 53$:    PUSHAB  (R8)[SUBVECTOR]                         ;
                                  9E  DD 0045B        PUSHL   @(SP)+                                  ;
                        00000000'  EF  9F 0045D       PUSHAB  P.ADY                                   ;
                                6C  AE  9F 00463      PUSHAB  NAMEPTR                                 ;
                          0000V CF      03  FB 00466  CALLS   #3, GET_SUBSCRIPT_VALUE                 ;
                                64  AE  DD 0046B 54$:  PUSHL   NAMEPTR                               ; 2235
                                6C  AE  9F 0046E      PUSHAB  HERE                                    ;
                              0080  CE  9F 00471      PUSHAB  STACK                                   ;
                                5A  DD 00475          PUSHL   INFO_FLAG                               ;
                          0000V CF      04  FB 00477  CALLS   #4, PRINT_PUSH                          ;
                                79  11 0047C         BRB     56$                                     ;
                                04  DD 0047E 55$:    PUSHL   #4                                      ; 2244
                             7E   7A  8F  9A 00480    MOVZBL  #122, -(SP)                             ;
                    00000000G 00      02  FB 00484    CALLS   #2, DBG$MAKE_SKELETON_DESC              ;
                             55       50  D0 0048B    MOVL    R0, VALPTR                              ;
                           0C AE  18  AE  D0 0048E    MOVL    24(SP), TYPEID                          ; 2245
                           03 A5  1C  AE  90 00493    MOVB    28(SP), 3(VALPTR)                       ; 2246
                        50 0C AE      14  C1 00498    ADDL3   #20, TYPEID, R0                         ; 2247
                           07 A5      60  90 0049D    MOVB    (R0), 7(VALPTR)                         ;
                        50 0C AE      18  C1 004A1    ADDL3   #24, TYPEID, R0                         ; 2248
                           06 A5      60  90 004A6    MOVB    (R0), 6(VALPTR)                         ;
                           08 A5  0C  AE  D0 004AA    MOVL    TYPEID, 8(VALPTR)                       ; 2249
                           0C A5  0C  A9  D0 004AF    MOVL    12(R9), 12(VALPTR)                      ; 2250
                    00000000G 00      59  D0 004B4    MOVL    R9, DBG$GL_CURRENT_PRIMARY              ; 2251
                                54  AE  9F 004BB      PUSHAB  BIT_OFFSET                              ; 2254
                                5C  AE  9F 004BE      PUSHAB  BITLEN                                  ;
                                14  A5  9F 004C1      PUSHAB  20(VALPTR)                              ;
                                0C  A9  DD 004C4      PUSHL   12(R9)                                  ;
                                1C  AE  DD 004C7      PUSHL   TYPEID                                  ;
                        7E      20  AE      18  C1 004CA    ADDL3   #24, TYPEID, -(SP)                  ;
                                   7E       9E  9A 004CF    MOVZBL  @(SP)+, -(SP)                       ;
                    00000000G 00      06  FB 004D2    CALLS   #6, DBG$FILL_IN_VMS_DESC                ;
                           18 A5  20  A5  9E 004D9    MOVAB   32(VALPTR), 24(VALPTR)                  ; 2255
                                6846  9F 004DE        PUSHAB  (R8)[SUBVECTOR]                         ; 2256
                           20 A5      9E  D0 004E1    MOVL    @(SP)+, 32(VALPTR)                      ;
                                01  DD 004E5          PUSHL   #1                                      ; 2257
                                55  DD 004E7          PUSHL   VALPTR                                  ; 2258
                                70  AE  9F 004E9      PUSHAB  HERE                                    ; 2257
                              0084  CE  9F 004EC      PUSHAB  STACK                                   ;
                                5A  DD 004F0          PUSHL   INFO_FLAG                               ;
                          0000V CF      05  FB 004F2  CALLS   #5, PRINT_PUSH                          ;
                           40 AE        50  D0 004F7 56$:    MOVL    R0, STKOVF                      ;
                                50 00000000'  EF  9E 004FB    MOVAB   TABLEBASE, R0                   ; 2265
                                   04 B240  9F 00502      PUSHAB  @4(CHAR_TABLE)[R0]                  ;
                                6C  AE  9F 00506      PUSHAB  HERE                                    ; 2264
                              0080  CE  9F 00509      PUSHAB  STACK                                   ;
                                5A  DD 0050D          PUSHL   INFO_FLAG                               ;
                          0000V CF      04  FB 0050F  CALLS   #4, PRINT_PUSH                          ;
                           40 AE        50  D0 00514    MOVL    R0, STKOVF                              ;
                        02 24 AE      38  AE  F2 00518 57$:    AOBLSS  56(SP), I, 58$                  ; 2229
                                03  11 0051E          BRB     59$                                     ;
                                  FF09  31 00520 58$:  BRW     52$                                     ;
                        03 30 BE        04  E0 00523 59$:    BBS     #4, @48(SP), 61$                ; 2272
                                011A  31 00528 60$:  BRW     68$                                     ;
                        F8 30 BE        01  E1 0052B 61$:    BBC     #1, @48(SP), 60$                ; 2275
                                18  28  AE  E8 00530        BLBS    SUBOVF, 62$                     ; 2282
                                7E  10  A9  32 00534        CVTWL   16(R9), -(SP)                   ;
```

```
                    00000000'  EF  9F 00538          PUSHAB  P.ADZ                                   ;
                           6C  AE  9F 0053E          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 00541          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
                28         AE      50 D0 00546          MOVL    R0, SUBOVF                           ;
                                   12 11 0054A          BRB     63$                                  ;
                        7E      10 A9 32 0054C 62$:     CVTWL   16(R9), -(SP)                        ;
                    00000000'  EF  9F 00550          PUSHAB  P.AEA                                   ;
                           6C  AE  9F 00556          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 00559          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
                           64  AE DD 0055E 63$:     PUSHL   NAMEPTR                                 ; 2283
                           6C  AE  9F 00561          PUSHAB  HERE                                    ;
                         0080  CE  9F 00564          PUSHAB  STACK                                   ;
                               5A DD 00568          PUSHL   INFO_FLAG                               ;
              0000V        CF      04 FB 0056A          CALLS   #4, PRINT_PUSH                       ;
                40         AE      50 D0 0056F          MOVL    R0, STKOVF                           ;
                        50  00000000'  EF  9E 00573          MOVAB   TABLEBASE, R0                   ; 2285
58                      50      04 A2 C1 0057A          ADDL3   4(CHAR_TABLE), R0, R8                ;
                               58 DD 0057F          PUSHL   R8                                      ;
                           6C  AE  9F 00581          PUSHAB  HERE                                    ; 2284
                         0080  CE  9F 00584          PUSHAB  STACK                                   ;
                               5A DD 00588          PUSHL   INFO_FLAG                               ;
              0000V        CF      04 FB 0058A          CALLS   #4, PRINT_PUSH                       ;
                40         AE      50 D0 0058F          MOVL    R0, STKOVF                           ;
                18         28  AE  E8 00593          BLBS    SUBOVF, 64$                          ; 2290
                        7E      12 A9 3C 00597          MOVZWL  18(R9), -(SP)                        ;
                    00000000'  EF  9F 0059B          PUSHAB  P.AEB                                   ;
                           6C  AE  9F 005A1          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 005A4          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
                28         AE      50 D0 005A9          MOVL    R0, SUBOVF                           ;
                                   12 11 005AD          BRB     65$                                  ;
                        7E      12 A9 3C 005AF 64$:     MOVZWL  18(R9), -(SP)                        ;
                    00000000'  EF  9F 005B3          PUSHAB  P.AEC                                   ;
                           6C  AE  9F 005B9          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 005BC          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
                           64  AE DD 005C1 65$:     PUSHL   NAMEPTR                                 ; 2291
                           6C  AE  9F 005C4          PUSHAB  HERE                                    ;
                         0080  CE  9F 005C7          PUSHAB  STACK                                   ;
                               5A DD 005CB          PUSHL   INFO_FLAG                               ;
              0000V        CF      04 FB 005CD          CALLS   #4, PRINT_PUSH                       ;
                40         AE      50 D0 005D2          MOVL    R0, STKOVF                           ;
                                   58 DD 005D6          PUSHL   R8                                   ; 2293
                           6C  AE  9F 005D8          PUSHAB  HERE                                    ; 2292
                         0080  CE  9F 005DB          PUSHAB  STACK                                   ;
                               5A DD 005DF          PUSHL   INFO_FLAG                               ;
              0000V        CF      04 FB 005E1          CALLS   #4, PRINT_PUSH                       ;
                40         AE      50 D0 005E6          MOVL    R0, STKOVF                           ;
                        1A      28 AE E8 005EA          BLBS    SUBOVF, 66$                          ; 2298
7E      30  BE              01      03 EF 005EE          EXTZV   #3, #1, @48(SP), -(SP)               ;
                    00000000'  EF  9F 005F4          PUSHAB  P.AED                                   ;
                           6C  AE  9F 005FA          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 005FD          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
                28         AE      50 D0 00602          MOVL    R0, SUBOVF                           ;
                                   14 11 00606          BRB     67$                                  ;
7E      30  BE              01      03 EF 00608 66$:     EXTZV   #3, #1, @48(SP), -(SP)               ;
                    00000000'  EF  9F 0060E          PUSHAB  P.AEE                                   ;
                           6C  AE  9F 00614          PUSHAB  NAMEPTR                                 ;
              0000V        CF      03 FB 00617          CALLS   #3, GET_SUBSCRIPT_VALUE              ;
```

```
                64    AE   DD 0061C 67$:    PUSHL   NAMEPTR                         ; 2299
                6C    AE   9F 0061F         PUSHAB  HERE
              0080    CE   9F 00622         PUSHAB  STACK
                      5A   DD 00626         PUSHL   INFO_FLAG
0000V   CF            04   FB 00628         CALLS   #4, PRINT_PUSH
  40    AE            50   D0 0062D         MOVL    R0, STKOVF
                      58   DD 00631         PUSHL   R8                              ; 2301
                6C    AE   9F 00633         PUSHAB  HERE                            ; 2300
              0080    CE   9F 00636         PUSHAB  STACK
                      5A   DD 0063A         PUSHL   INFO_FLAG
0000V   CF            04   FB 0063C         CALLS   #4, PRINT_PUSH
  40    AE            50   D0 00641         MOVL    R0, STKOVF
        07            5A   E9 00645 68$:    BLBC    INFO_FLAG, 69$                  ; 2308
  5C    AE      68    AE   D0 00648         MOVL    HERE, TOP
                      05   11 0064D         BRB     70$
  60    AE      68    AE   D0 0064F 69$:    MOVL    HERE, BOTTOM                    ; 2311
        03            5A   E8 00654 70$:    BLBS    INFO_FLAG, 71$                  ; 2317
                FC2E        31 00657         BRW     28$
              5C    AE   D6 0065A 71$:    INCL    TOP                             ; 2320
        50 00000000'  EF   9E 0065D         MOVAB   TABLEBASE, R0                   ; 2322
              08 B240   9F 00664         PUSHAB  @8(CHAR_TABLE)[R0]
              60    AE   9F 00668         PUSHAB  TOP                             ; 2321
              0080  CE   9F 0066B         PUSHAB  STACK
                    5A   DD 0066F         PUSHL   INFO_FLAG
                0175      31 00671         BRW     92$
        0A      02    A8   E9 00674 72$:    BLBC    2(R8), 73$                      ; 2335
        76      10    AE   E9 00678         BLBC    FORMAT, 78$                     ; 2338
                64    AE   DD 0067C         PUSHL   NAMEPTR                         ; 2341
                0080      31 0067F         BRW     80$
                64    AE   DD 00682 73$:    PUSHL   NAMEPTR                         ; 2349
                00A5      31 00685         BRW     83$
        02      14    AE   D1 00688 74$:    CMPL    PATHNAME_FLAG, #2               ; 2372
                60        12 0068C         BNEQ    77$
        38      02    A8   E9 0068E         BLBC    2(R8), 75$                      ; 2375
        5C      10    AE   E9 00692         BLBC    FORMAT, 78$                     ; 2378
        50      64    AE   D0 00696         MOVL    NAMEPTR, R0                     ; 2381
                      62   13 0069A         BEQL    79$
                      60   95 0069C         TSTB    (R0)                            ; 2384
                      5E   13 0069E         BEQL    79$
                      50   DD 006A0         PUSHL   R0                              ; 2387
                64    AE   9F 006A2         PUSHAB  BOTTOM
              0080    CE   9F 006A5         PUSHAB  STACK
                      7E   D4 006A9         CLRL    -(SP)
0000V   CF            04   FB 006AB         CALLS   #4, PRINT_PUSH
  40    AE            50   D0 006B0         MOVL    R0, STKOVF
        50 00000000'  EF   9E 006B4         MOVAB   TABLEBASE, R0                   ; 2389
              04 B240   9F 006BB         PUSHAB  @4(CHAR_TABLE)[R0]
              60    AE   9F 006BF         PUSHAB  TOP                             ; 2388
              0080  CE   9F 006C2         PUSHAB  STACK
                    01   DD 006C6         PUSHL   #1
                    15   11 006C8         BRB     76$
        50      64    AE   D0 006CA 75$:    MOVL    NAMEPTR, R0                     ; 2397
                      6F   13 006CE         BEQL    84$
                      60   95 006D0         TSTB    (R0)                            ; 2400
                      6B   13 006D2         BEQL    84$
                      50   DD 006D4         PUSHL   R0                              ; 2403
                64    AE   9F 006D6         PUSHAB  BOTTOM
```

```
                        0080   CE  9F 006D9          PUSHAB  STACK                                   ;
                               7E  D4 006DD          CLRL    -(SP)                                   ;
0000V CF                       04  FB 006DF 76$:     CALLS   #4, PRINT_PUSH                          ;
  40  AE                       50  D0 006E4          MOVL    R0, STKOVF                              ;
  14  AE                       03  D0 006E8          MOVL    #3, PATHNAME_FLAG                       ; 2404
                               51  11 006EC          BRB     84$                                     ; 2372
                 2F      02    A8  E9 006EE 77$:     BLBC    2(R8), 82$                              ; 2415
                 49      10    AE  E9 006F2 78$:     BLBC    FORMAT, 84$                             ; 2418
                 50      64    AE  D0 006F6          MOVL    NAMEPTR, R0                             ; 2421
                               18  13 006FA          BEQL    81$                                     ;
                               60  95 006FC          TSTB    (R0)                                    ; 2424
                               3F  13 006FE 79$:     BEQL    84$                                     ;
                               50  DD 00700          PUSHL   R0                                      ; 2427
                         60    AE  9F 00702 80$:     PUSHAB  TOP                                     ;
                        0080   CE  9F 00705          PUSHAB  STACK                                   ;
                               01  DD 00709          PUSHL   #1                                      ;
0000V CF                       04  FB 0070B          CALLS   #4, PRINT_PUSH                          ;
  40  AE                       50  D0 00710          MOVL    R0, STKOVF                              ;
                 50 00000000'  EF  9E 00714 81$:     MOVAB   TABLEBASE, R0                           ; 2434
                          04 B240  9F 0071B          PUSHAB  @4(CHAR_TABLE)[R0]                      ;
                               0C  11 0071F          BRB     83$                                     ; 2433
                 50      64    AE  D0 00721 82$:     MOVL    NAMEPTR, R0                             ; 2439
                               18  13 00725          BEQL    84$                                     ;
                               60  95 00727          TSTB    (R0)                                    ; 2442
                               14  13 00729          BEQL    84$                                     ;
                               50  DD 0072B          PUSHL   R0                                      ; 2444
                         60    AE  9F 0072D 83$:     PUSHAB  TOP                                     ;
                        0080   CE  9F 00730          PUSHAB  STACK                                   ;
                               01  DD 00734          PUSHL   #1                                      ;
              0000V CF                 04  FB 00736          CALLS   #4, PRINT_PUSH                  ;
                40  AE                 50  D0 0073B          MOVL    R0, STKOVF                      ;
                          2C    AE     D4  0073F 84$:     CLRL    PTR_FLAG                            ; 2449
                              6B47     7F  00742          PUSHAQ  (INFO_TABLE)[FCODE]                 ; 2450
                 02                    9E  D1 00745          CMPL    @(SP)+, #2                      ;
                                       03  12 00748          BNEQ    85$                             ;
                                     008A  31 0074A          BRW     90$                             ;
                                     FB38  31 0074D 85$:     BRW     28$                             ;
                          20    AE       63  D0 00750 86$:     MOVL    (PRIM_NODE), NEXT_NODE          ; 2461
                          03    02    A8  E8 00754          BLBS    2(R8), 87$                      ; 2462
                                     0082  31 00758          BRW     91$                             ;
                          EE    10    AE  E9 0075B 87$:     BLBC    FORMAT, 85$                     ; 2465
                          34    AE    00  B3  D1 0075F          CMPL    @0(PRIM_NODE), PRIM_HEAD        ; 2468
                                       14  13 00764          BEQL    88$                             ;
              50          20    AE       09  C1 00766          ADDL3   #9, NEXT_NODE, R0               ; 2469
                                06       60  91 0076B          CMPB    (R0), #6                        ;
                                       0A  13 0076E          BEQL    88$                             ;
              50          20    AE       09  C1 00770          ADDL3   #9, NEXT_NODE, R0               ; 2470
                                10       60  91 00775          CMPB    (R0), #16                       ;
                                       2B  12 00778          BNEQ    89$                             ;
                                 64    AE  DD 0077A 88$:     PUSHL   NAMEPTR                         ; 2473
                                 60    AE  9F 0077D          PUSHAB  TOP                             ;
                                0080   CE  9F 00780          PUSHAB  STACK                           ;
                                       01  DD 00784          PUSHL   #1                              ;
                      0000V CF                 04  FB 00786          CALLS   #4, PRINT_PUSH          ;
                        40  AE                 50  D0 0078B          MOVL    R0, STKOVF              ;
                                  50 00000000'  EF  9E 0078F          MOVAB   TABLEBASE, R0           ; 2475
                                           04 B240  9F 00796          PUSHAB  @4(CHAR_TABLE)[R0]      ;
```

```
                    64    AE  9F 0079A          PUSHAB  BOTTOM                                    ; 2474
                  0080    CE  9F 0079D          PUSHAB  STACK                                     ;
                          7E  D4 007A1          CLRL    -(SP)                                     ;
                          44  11 007A3          BRB     92$                                       ;
                    64    AE  DD 007A5 89$:     PUSHL   NAMEPTR                                   ; 2480
                    60    AE  9F 007A8          PUSHAB  TOP                                       ;
                  0080    CE  9F 007AB          PUSHAB  STACK                                     ;
                          01  DD 007AF          PUSHL   #1                                        ;
            0000V CF            04  FB 007B1          CALLS   #4, PRINT_PUSH                    ;
              40  AE            50  D0 007B6          MOVL    R0, STKOVF                        ;
                50 00000000'    EF  9E 007BA          MOVAB   TABLEBASE, R0                     ; 2482
                        0C B240 9F  9E 007C1          PUSHAB  @12(CHAR_TABLE)[R0]               ;
                    60    AE  9F 007C5          PUSHAB  TOP                                       ; 2481
                  0080    CE  9F 007C8          PUSHAB  STACK                                     ;
                          01  DD 007CC          PUSHL   #1                                        ;
            0000V CF            04  FB 007CE          CALLS   #4, PRINT_PUSH                    ;
              40  AE            50  D0 007D3          MOVL    R0, STKOVF                        ;
              2C  AE            01  D0 007D7 90$:     MOVL    #1, PTR_FLAG                      ; 2483
                                5D  11 007DB          BRB     97$                               ; 2465
                    64    AE  DD 007DD 91$:     PUSHL   NAMEPTR                                   ; 2491
                    60    AE  9F 007E0          PUSHAB  TOP                                       ;
                  0080    CE  9F 007E3          PUSHAB  STACK                                     ;
                          01  DD 007E7          PUSHL   #1                                        ;
            0000V CF            04  FB 007E9 92$:     CALLS   #4, PRINT_PUSH                    ;
              40  AE            50  D0 007EE          MOVL    R0, STKOVF                        ;
                                46  11 007F2          BRB     97$                               ; 2128
              2A        02    A8  E9 007F4 93$:     BLBC    2(R8), 94$                        ; 2500
              3B        10    AE  E9 007F8          BLBC    FORMAT, 96$                       ; 2503
              22        2C    AE  E8 007FC          BLBS    PTR_FLAG, 94$                     ; 2506
                    64    AE  DD 00800          PUSHL   NAMEPTR                                   ; 2512
                    60    AE  9F 00803          PUSHAB  TOP                                       ;
                  0080    CE  9F 00806          PUSHAB  STACK                                     ;
                          01  DD 0080A          PUSHL   #1                                        ;
            0000V CF            04  FB 0080C          CALLS   #4, PRINT_PUSH                    ;
              40  AE            50  D0 00811          MOVL    R0, STKOVF                        ;
                50 00000000'    EF  9E 00815          MOVAB   TABLEBASE, R0                     ; 2514
                        04 B240 9F  9E 0081C          PUSHAB  @4(CHAR_TABLE)[R0]                ;
                                03  11 00820          BRB     95$                               ; 2513
                    64    AE  DD 00822 94$:     PUSHL   NAMEPTR                                   ; 2523
                    60    AE  9F 00825 95$:     PUSHAB  TOP                                       ;
                  0080    CE  9F 00828          PUSHAB  STACK                                     ;
                          01  DD 0082C          PUSHL   #1                                        ;
            00C0V CF            04  FB 0082E          CALLS   #4, PRINT_PUSH                    ;
              40  AE            50  D0 00833          MOVL    R0, STKOVF                        ;
                    2C    AE  D4 00837 96$:     CLRL    PTR_FLAG                          ; 2525
                        FA4B    31  00839 97$:     BRW     28$                               ; 2128
              37                5A  E8 0083D 98$:     BLBS    INFO_FLAG, 99$                    ; 2550
              3C        60    AE  D1 00840          CMPL    BOTTOM, #60                       ; 2553
                                31  15 00844          BLEQ    99$                               ;
                    14    AE  D5 00846          TSTL    PATHNAME_FLAG                     ; 2554
                                2C  12 00849          BNEQ    99$                               ;
                    60    AE  D7 0084B          DECL    BOTTOM                            ; 2557
                50 00000000'    EF  9E 0084E          MOVAB   TABLEBASE, R0                     ; 2558
52              50        0C    AB  C1 00855          ADDL3   12(INFO_TABLE), R0, CHAR_TABLE    ;
                50 00000000'    EF  9E 0085A          MOVAB   TABLEBASE, R0                     ; 2560
                        08 B240 9F  00861          PUSHAB  @8(CHAR_TABLE)[R0]                ;
                    64    AE  9F 00865          PUSHAB  BOTTOM                            ; 2559
```

```
                 0080   CE  9F 00868          PUSHAB  STACK
                        5A  DD 0086C          PUSHL   INFO_FLAG
          0000V  CF             04  FB 0086E          CALLS   #4, PRINT_PUSH
            40   AE             50  D0 00873          MOVL    R0, STKOVF
03          30   BE             01  E0 00877 99$:     BBS     #1, @48(SP), 101$                       ;2569
                          0127   31  0087C 100$:     BRW     111$
F8          30   BE             04  E0 0087F 101$:    BBS     #4, @48(SP), 100$                       ;2572
                 F4       30    BE  E8 00884          BLBS    @48(SP), 100$                           ;2575
                 50 00000000'   EF  9E 00888          MOVAB   TABLEBASE, R0                           ;2579
52               50       04    AB  C1 0088F          ADDL3   4(INFO_TABLE), R0, CHAR_TABLE
                 06             5A  E9 00894          BLBC    INFO_FLAG, 102$                         ;2580
                 50       5C    AE  D0 00897          MOVL    TOP, R0
                                04  11 0089B          BRB     103$
                 50       60    AE  D0 0089D 102$:    MOVL    BOTTOM, R0
              68 AE             50  D0 008A1 103$:    MOVL    R0, HERE
                 50 00000000'   EF  9E 008A5          MOVAB   TABLEBASE, R0                           ;2586
                       00 B240  9F  008AC          PUSHAB  @0(CHAR_TABLE)[R0]
                          6C    AE  9F 008B0          PUSHAB  HERE                                    ;2585
                       0080     CE  9F 008B3          PUSHAB  STACK
                                5A  DD 008B7          PUSHL   INFO_FLAG
          0000V  CF             04  FB 008B9          CALLS   #4, PRINT_PUSH
            40   AE             50  D0 008BE          MOVL    R0, STKOVF
                 57       10    A9  9E 008C2          MOVAB   16(R9), R7                              ;2591
                 19       28    AE  E8 008C6          BLBS    SUBOVF, 104$
                 53             67  32 008CA          CVTWL   (R7), R3
                                53  DD 008CD          PUSHL   R3
                    00000000'   EF  9F 008CF          PUSHAB  P.AEG
                          6C    AE  9F 008D5          PUSHAB  NAMEPTR
          0000V  CF             03  FB 008D8          CALLS   #3, GET_SUBSCRIPT_VALUE
            28   AE             50  D0 008DD          MOVL    R0, SUBOVF
                                13  11 008E1          BRB     105$
                 53             67  32 008E3 104$:    CVTWL   (R7), R3
                                53  DD 008E6          PUSHL   R3
                    00000000'   EF  9F 008E8          PUSHAB  P.AEH
                          6C    AE  9F 008EE          PUSHAB  NAMEPTR
          0000V  CF             03  FB 008F1          CALLS   #3, GET_SUBSCRIPT_VALUE
                       64       AE  DD 008F6 105$:    PUSHL   NAMEPTR                                 ;2592
                          6C    AE  9F 008F9          PUSHAB  HERE
                       0080     CE  9F 008FC          PUSHAB  STACK
                                5A  DD 00900          PUSHL   INFO_FLAG
          0000V  CF             04  FB 00902          CALLS   #4, PRINT_PUSH
            40   AE             50  D0 00907          MOVL    R0, STKOVF
                 50 00000000'   EF  9E 0090B          MOVAB   TABLEBASE, R0                           ;2594
                       04 B240  9F  00912          PUSHAB  @4(CHAR_TABLE)[R0]
                          6C    AE  9F 00916          PUSHAB  HERE                                    ;2593
                       0080     CE  9F 00919          PUSHAB  STACK
                                5A  DD 0091D          PUSHL   INFO_FLAG
          0000V  CF             04  FB 0091F          CALLS   #4, PRINT_PUSH
            40   AE             50  D0 00924          MOVL    R0, STKOVF
                 0B       44    AE  E9 00928          BLBC    REFMOD_FLAG, 106$                       ;2599
                 50       02    A7  3C 0092C          MOVZWL  2(R7), R0                               ;2602
                 50       FF A043   9E 00930          MOVAB   -1(R0)[R3], VALUE
                                04  11 00935          BRB     107$                                    ;2601
                 50       02    A7  3C 00937 106$:    MOVZWL  2(R7), VALUE                            ;2605
                 16       28    AE  E8 0093B 107$:    BLBS    SUBOVF, 108$                            ;2607
                                50  DD 0093F          PUSHL   VALUE
                    00000000'   EF  9F 00941          PUSHAB  P.AEI
```

```
                        6C    AE  9F 00947          PUSHAB  NAMEPTR                          ;
        0000V   CF            03  FB 0094A          CALLS   #3, GET_SUBSCRIPT_VALUE          ;
           28   AE            50  D0 0094F          MOVL    R0, SUBOVF                       ;
                              10  11 00953          BRB     109$                             ;
                              50  DD 00955 108$:    PUSHL   VALUE                            ;
                   00000000'  EF  9F 00957          PUSHAB  P.AEJ                            ;
                        6C    AE  9F 0095D          PUSHAB  NAMEPTR                          ;
        0000V   CF            03  FB 00960          CALLS   #3, GET_SUBSCRIPT_VALUE          ;
                        64    AE  DD 00965 109$:    PUSHL   NAMEPTR                          ; 2608
                        6C    AE  9F 00968          PUSHAB  HERE                             ;
                      0080    CE  9F 0096B          PUSHAB  STACK                            ;
                              5A  DD 0096F          PUSHL   INFO_FLAG                        ;
        0000V   CF            04  FB 00971          CALLS   #4, PRINT_PUSH                   ;
           40   AE            50  D0 00976          MOVL    R0, STKOVF                       ;
                 50 00000000' EF  9E 0097A          MOVAB   TABLEBASE, R0                    ; 2610
                      08 B240     9F 00981          PUSHAB  @8(CHAR_TABLE)[R0]               ;
                        6C    AE  9F 00985          PUSHAB  HERE                             ; 2609
                      0080    CE  9F 00988          PUSHAB  STACK                            ;
                              5A  DD 0098C          PUSHL   INFO_FLAG                        ;
        0000V   CF            04  FB 0098E          CALLS   #4, PRINT_PUSH                   ;
           40   AE            50  D0 00993          MOVL    R0, STKOVF                       ;
                07            5A  E9 00997          BLBC    INFO_FLAG, 110$                  ; 2614
           5C   AE      68    AE  D0 0099A          MOVL    HERE, TOP                        ;
                              05  11 0099F          BRB     111$                             ;
           60   AE      68    AE  D0 009A1 110$:    MOVL    HERE, BOTTOM                     ; 2617
                04      40    AE  E8 009A6 111$:    BLBS    STKOVF, 112$                     ; 2629
                0D      28    AE  E9 009AA          BLBC    SUBOVF, 113$                     ;
                   0002804B   8F  DD 009AE 112$:    PUSHL   #163915                          ;
    00000000G   00            01  FB 009B4          CALLS   #1, LIB$SIGNAL                   ;
50         5C   AE            01  C1 009BB 113$:    ADDL3   #1, TOP, R0                      ; 2632
                              02  18 009C0          BGEQ    114$                             ;
                              50  D4 009C2          CLRL    R0                               ;
           5C   AE            50  D0 009C4 114$:    MOVL    R0, TOP                          ;
                50      60    AE  D0 009C8          MOVL    BOTTOM, R0                       ; 2633
     00000063   8F            50  D1 009CC          CMPL    R0, #99                          ;
                              04  15 009D3          BLEQ    115$                             ;
                50      63    8F  9A 009D5          MOVZBL  #99, R0                          ;
           60   AE            50  D0 009D9 115$:    MOVL    R0, BOTTOM                       ;
                3F            5A  E9 009DD          BLBC    INFO_FLAG, 120$                  ; 2641
                53      60    AE  D0 009E0          MOVL    BOTTOM, I                        ;
                              31  11 009E4          BRB     119$                             ;
50              53            05  C5 009E6 116$:    MULL3   #5, I, R0                        ; 2643
14         7C AE40            00  E1 009EA          BBC     #0, STACK+4[R0], 117$            ;
                              7E  7C 009F0          CLRQ    -(SP)                            ; 2645
                              01  DD 009F2          PUSHL   #1                               ;
                      0084 CE40   9F 009F4          PUSHAB  STACK[R0]                        ;
                              9E  DD 009F9          PUSHL   @(SP)+                           ;
    00000000G   00            04  FB 009FB          CALLS   #4, DBG$PRINT_VALUE              ;
                              11  11 00A02          BRB     118$                             ;
                        78 AE40   9F 00A04 117$:    PUSHAB  STACK[R0]                        ; 2648
                              9E  DD 00A08          PUSHL   @(SP)+                           ;
                   00000000'  EF  9F 00A0A          PUSHAB  P.AEK                            ;
         F202   CF            02  FB 00A10          CALLS   #2, DBG$PRINT                    ;
                              53  D7 00A15 118$:    DECL    I                                ; 2641
           5C   AE            53  D1 00A17 119$:    CMPL    I, TOP                           ;
                              C9  18 00A1B          BGEQ    116$                             ;
                              3B  11 00A1D          BRB     124$                             ;
```

```
                53        5C    AE              01  C3 00A1F 120$:  SUBL3   #1, TOP, I                          ; 2657
                                                2F  11 00A24        BRB     123$
                50              53              05  C5 00A26 121$:  MULL3   #5, I, R0                           ; 2659
                14        7C AE40               00  E1 00A2A        BBC     #0, STACK+4[R0], 122$
                                                7E  7C 00A30        CLRQ    -(SP)                               ; 2661
                                                01  DD 00A32        PUSHL   #1
                                          0084 CE40             9F  00A34        PUSHAB  STACK[R0]
                                                9E  DD 00A39        PUSHL   @(SP)+
                   00000000G    00              04  FB 00A3B        CALLS   #4, DBG$PRINT_VALUE
                                                11  11 00A42        BRB     123$
                                            78 AE40             9F  00A44 122$:  PUSHAB  STACK[R0]               ; 2663
                                                9E  DD 00A48        PUSHL   @(SP)+
                                     00000000'  EF  9F 00A4A        PUSHAB  P.AEL
                         F1C2       CF              02  FB 00A50        CALLS   #2, DBG$PRINT
                CC              53          60      AE  F3 00A55 123$:  AOBLEQ  BOTTOM, I, 121$                    ; 2657
                                            08      AE  D5 00A5A 124$:  TSTL    BYTE_OFFSET                     ; 2667
                                                08  13 00A5D        BEQL    125$
                                            08      AE  DD 00A5F        PUSHL   BYTE_OFFSET
                         0000V      CF              01  FB 00A62        CALLS   #1, DBG$PRINT_OFFSET
                                                54  D5 00A67 125$:  TSTL    VAL_DESC
                                                5E  13 00A69        BEQL    131$
                                            54      AE  D5 00A6B        TSTL    BIT_OFFSET
                                                2C  12 00A6E        BNEQ    127$
        06      30    BE              08              18  ED 00A70        CMPZV   #24, #8, @48(SP), #6
                                                08  13 00A76        BEQL    126$
        0A      30    BE              08              18  ED 00A78        CMPZV   #24, #8, @48(SP), #10
                                                1C  12 00A7E        BNEQ    127$
                                            6C      AE  9F  00A80 126$:  PUSHAB  TMP_VALDESC
                                7E          83      8F  9A 00A83        MOVZBL  #131, -(SP)
                                                59  DD 00A87        PUSHL   R9
                   00000000G    00              03  FB 00A89        CALLS   #3, DBG$PRIM_TO_VAL
                                50          6C      AE  D0 00A90        MOVL    TMP_VALDESC, R0
        6E      14    A0              10              00  ED 00A94        CMPZV   #0, #16, 20(R0), SAVE_VAL_LENGTH
                                                07  13 00A9A        BEQL    128$
                                                54  DD 00A9C 127$:  PUSHL   VAL_DESC
                         F4CF       CF              01  FB 00A9E        CALLS   #1, DBG$PRINT_FIELD_REF
                                            04      AE  D5 00AA3 128$:  TSTL    SAVE_VAL_DTYPE
                                                09  13 00AA6        BEQL    130$
                           16       A4      04      AE  90 00AA8 129$:  MOVB    SAVE_VAL_DTYPE, 22(VAL_DESC)
                           14       A4              6E  B0 00AAD        MOVW    SAVE_VAL_LENGTH, 20(VAL_DESC)
                                    29      16      A4  91 00AB1 130$:  CMPB    22(VAL_DESC), #41
                                                2D  13 00AB5        BEQL    133$
                                    2A      16      A4  91 00AB7        CMPB    22(VAL_DESC), #42
                                                27  13 00ABB        BEQL    133$
                                    01      16      A4  91 00ABD        CMPB    22(VAL_DESC), #1
                                                21  13 00AC1        BEQL    133$
                                    22      16      A4  91 00AC3        CMPB    22(VAL_DESC), #34
                                                1B  13 00AC7        BEQL    133$
                                                54  D5 00AC9 131$:  TSTL    VAL_DESC
                                                0A  13 00ACB        BEQL    132$
                                            08      AE  D5 00ACD        TSTL    BYTE_OFFSET
                                                12  12 00AD0        BNEQ    133$
                                            54      AE  D5 00AD2        TSTL    BIT_OFFSET
                                                0D  12 00AD5        BNEQ    133$
                                                59  DD 00AD7 132$:  PUSHL   R9
                   00000000G    00              01  FB 00AD9        CALLS   #1, DBG$COLLECT
                                50              59  D0 00AE0        MOVL    R9, R0
```

```
                              04 00AE3         RET                                   ;
             50            54 D0 00AE4 133$:   MOVL    VAL_DESC, R0                  ;
                              04 00AE7         RET                                   ; 2675

; Routine Size:  2792 bytes,    Routine Base:  DBG$CODE + 0565
```

```
;  2570          2676  1 GLOBAL ROUTINE DBG$PRINT_IDENTIFIER_PC(PC) =
;  2571          2677  1 !
;  2572          2678  1 ! FUNCTION
;  2573          2679  1 !       This routine takes in a 32 bit address, build a Volatile Value
;  2574          2680  1 !       Descriptor, and then call DBG$PRINT_IDENTIFIER to print out that
;  2575          2681  1 !       address as an absolute address.
;  2576          2682  1 !
;  2577          2683  1 ! INPUTS
;  2578          2684  1 !       PC      - 32 bits virtual address.
;  2579          2685  1 !
;  2580          2686  1 ! OUTPUTS
;  2581          2687  1 !       A Primary descriptor for the exact match or a Volatile Value
;  2582          2688  1 !                 Descriptor will be returned.
;  2583          2689  1 !
;  2584          2690  1
;  2585          2691  2     BEGIN
;  2586          2692  2
;  2587          2693  2     LOCAL
;  2588          2694  2         STATUS,                             ! Return value
;  2589          2695  2         VAL_DESC: REF DBG$VALDESC;          ! Pointer to value descriptor
;  2590          2696  2
;  2591          2697  2
;  2592          2698  2     VAL_DESC = DBG$MAKE_SKELETON_DESC (DBG$K_V_VALUE_DESC);
;  2593          2699  2     VAL_DESC[DBG$B_DHDR_LANG] = DBG$GB_LANGUAGE;
;  2594          2700  2     VAL_DESC[DBG$B_DHDR_KIND] = RST$K_DATA;
;  2595          2701  2     VAL_DESC[DBG$B_DHDR_FCODE] = 0;
;  2596          2702  2     VAL_DESC[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  2597          2703  2     VAL_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_LU;
;  2598          2704  2     VAL_DESC[DBG$W_VALUE_LENGTH] = 4;
;  2599          2705  2     VAL_DESC[DBG$L_VALUE_POINTER] = .PC;
;  2600          2706  2     STATUS = DBG$PRINT_IDENTIFIER(.VAL_DESC);
;  2601          2707  2     RETURN .STATUS;
;  2602          2708  2
;  2603          2709  1     END;


                                                0000 00000      .ENTRY  DBG$PRINT_IDENTIFIER_PC, Save nothing           ; 2676
                        7E          83      8F  9A 00002      MOVZBL  #131, -(SP)                                     ; 2698
                00000000G   00              01  FB 00006      CALLS   #1, DBG$MAKE_SKELETON_DESC                      ;
                            51  00000000G    00  9E 0000D      MOVAB   DBG$GB_LANGUAGE, R1                             ; 2699
                      03    A0              51  90 00014      MOVB    R1, 3(VAL_DESC)                                 ;
                      06    A0        0600  8F  B0 00018      MOVW    #1536, 6(VAL_DESC)                              ; 2701
                      14    A0  01040004  8F  D0 0001E      MOVL    #17039364, 20(VAL_DESC)                         ; 2704
                      18    A0          04  AC  D0 00026      MOVL    PC, 24(VAL_DESC)                                ; 2705
                                            50  DD 0002B      PUSHL   VAL_DESC                                        ; 2706
                    F4E6    CF              01  FB 0002D      CALLS   #1, DBG$PRINT_IDENTIFIER                        ;
                                            04  00032      RET                                                     ; 2709

; Routine Size:  51 bytes,    Routine Base:  DBG$CODE + 104D
```

```
;2605    2710  1 GLOBAL ROUTINE DBG$PRINT_OFFSET(VALUE): NOVALUE =
;2606    2711  1 !
;2607    2712  1 ! FUNCTION
;2608    2713  1 !       This function prints a value in the current radix.  It suppresses
;2609    2714  1 !       leading zeroes, except before hex A-F.  It is designed to be used in
;2610    2715  1 !       printing positive offsets; it will not print "0", and will not print
;2611    2716  1 !       negative numbers.
;2612    2717  1 !
;2613    2718  1 ! INPUTS
;2614    2719  1 !       VALUE - The value to be printed.  Should be a number greater than 0.
;2615    2720  1 !
;2616    2721  1 ! OUTPUTS
;2617    2722  1 !       The routine either prints the value in the correct radix, or signals
;2618    2723  1 !       an error.
;2619    2724  1 !
;2620    2725  1
;2621    2726  2     BEGIN
;2622    2727  2
;2623    2728  2     LOCAL
;2624    2729  2         I,                                  ! Index.
;2625    2730  2         RADIX,                              ! Current radix.
;2626    2731  2         FAO_LENGTH: WORD,                   ! Length returned by SYS$FAO.
;2627    2732  2         CONTROL_DESC: BLOCK[8, BYTE],       ! Control desc. for SYS$FAO.
;2628    2733  2         OUT_DESC: BLOCK[8, BYTE],           ! Output desc. for SYS$FAO.
;2629    2734  2         OUTBUF: VECTOR[20, BYTE];           ! Output buffer.
;2630    2735  2
;2631    2736  2
;2632    2737  2
;2633    2738  2     ! Set up output descriptor.
;2634    2739  2     !
;2635    2740  2     OUT_DESC[DSC$W_LENGTH] = 19;
;2636    2741  2     OUT_DESC[DSC$A_POINTER] = OUTBUF[1];
;2637    2742  2
;2638    2743  2
;2639    2744  2     ! Get current radix.
;2640    2745  2     !
;2641    2746  2     IF .DBG$GB_VERB EQL DBG$K_EXAMINE_VERB
;2642    2747  2     THEN
;2643    2748  3         BEGIN
;2644    2749  3         RADIX = .DBG$GL_CMND_RADIX;
;2645    2750  3         IF .RADIX EQL DBG$K_DEFAULT
;2646    2751  3         THEN
;2647    2752  3             RADIX = DBG$NGET_RADIX();
;2648    2753  3         END
;2649    2754  2     ELSE
;2650    2755  2         RADIX = DBG$NGET_RADIX();
;2651    2756  2
;2652    2757  2
;2653    2758  2     ! Decide on form of output based on radix mode.
;2654    2759  2     !
;2655    2760  2     CASE .RADIX FROM DBG$K_DEFAULT TO DBG$K_HEX OF
;2656    2761  2         SET
;2657    2762  2
;2658    2763  2
;2659    2764  2         ! Output the offset in octal.
;2660    2765  2         !
;2661    2766  2         [DBG$K_OCTAL]:
```

```
;  2662     2767  3          BEGIN
;  2663     2768  3
;  2664     2769  3
;  2665     2770  3          ! Set up control descriptor for octal output; call SYS$FAO.
;  2666     2771  3          !
;  2667     2772  3          CONTROL_DESC[DSC$W_LENGTH] = %CHARCOUNT ('!OL');
;  2668     2773  3          CONTROL_DESC[DSC$A_POINTER] = UPLIT BYTE ('!OL');
;  2669     2774  3          IF NOT SYS$FAO (CONTROL_DESC, FAO_LENGTH, OUT_DESC, .VALUE)
;  2670     2775  3          THEN
;  2671     2776  3              $DBG_ERROR('DBGPRINT\DBG$PRINT_OFFSET, SYS$FAO failed');
;  2672     2777  3
;  2673     2778  3
;  2674     2779  3          ! Suppress leading zeroes.
;  2675     2780  3          !
;  2676     2781  3          I = 1;
;  2677     2782  3          WHILE .OUTBUF[.I] EQL %C'0' DO
;  2678     2783  4              BEGIN
;  2679     2784  4              I = .I + 1;
;  2680     2785  4              IF .I GTR .FAO_LENGTH THEN EXITLOOP;
;  2681     2786  3              END;
;  2682     2787  3
;  2683     2788  3          OUT_DESC[DSC$W_LENGTH] = .FAO_LENGTH - .I + 1;
;  2684     2789  3          OUT_DESC[DSC$A_POINTER] = OUTBUF[.I];
;  2685     2790  3
;  2686     2791  3
;  2687     2792  3          ! Output the value.
;  2688     2793  3          !
;  2689     2794  3          IF .OUT_DESC[DSC$W_LENGTH] GTR 0
;  2690     2795  3          THEN
;  2691     2796  3              DBG$PRINT (UPLIT BYTE (%ASCIC '+!AS'), OUT_DESC);
;  2692     2797  3
;  2693     2798  2          END;
;  2694     2799  2
;  2695     2800  2
;  2696     2801  2          ! Output the offset in hexadecimal.
;  2697     2802  2          !
;  2698     2803  2          [DBG$K_HEX]:
;  2699     2804  3              BEGIN
;  2700     2805  3
;  2701     2806  3
;  2702     2807  3              ! Set up control descriptor for hex output; call SYS$FAO.
;  2703     2808  3              !
;  2704     2809  3              CONTROL_DESC[DSC$W_LENGTH] = %CHARCOUNT ('!XL');
;  2705     2810  3              CONTROL_DESC[DSC$A_POINTER] = UPLIT BYTE ('!XL');
;  2706     2811  3              IF NOT SYS$FAO (CONTROL_DESC, FAO_LENGTH, OUT_DESC, .VALUE)
;  2707     2812  3              THEN
;  2708     2813  3                  $DBG_ERROR('DBGPRINT\DBG$PRINT_OFFSET, SYS$FAO failed');
;  2709     2814  3
;  2710     2815  3
;  2711     2816  3              ! Suppress leading zeroes.
;  2712     2817  3              !
;  2713     2818  3              I = 1;
;  2714     2819  3              WHILE .OUTBUF[.I] EQL %C'0' DO
;  2715     2820  4                  BEGIN
;  2716     2821  4                  I = .I + 1;
;  2717     2822  4                  IF .I GTR .FAO_LENGTH THEN EXITLOOP;
;  2718     2823  3                  END;
```

```
;  2719          2824  3
;  2720          2825  3
;  2721          2826  3              ! If first non-zero character is A-F, then print one leading zero.
;  2722          2827  3              !
;  2723          2828  3              IF .OUTBUF[.I] GTR '9'
;  2724          2829  3              THEN
;  2725          2830  4                  BEGIN
;  2726          2831  4                  I = .I - 1;
;  2727          2832  4                  OUTBUF[.I] = %C'0';
;  2728          2833  3                  END;
;  2729          2834  3
;  2730          2835  3              OUT_DESC[DSC$W_LENGTH] = .FAO_LENGTH - .I + 1;
;  2731          2836  3              OUT_DESC[DSC$A_POINTER] = OUTBUF[.I];
;  2732          2837  3
;  2733          2838  3
;  2734          2839  3              ! Output the value.
;  2735          2840  3              !
;  2736          2841  3              IF .OUT_DESC[DSC$W_LENGTH] GTR 0
;  2737          2842  3              THEN
;  2738          2843  3                  DBG$PRINT (UPLIT BYTE (%ASCIC '+!AS'), OUT_DESC);
;  2739          2844  3
;  2740          2845  2              END;
;  2741          2846  2
;  2742          2847  2
;  2743          2848  2          ! In all other cases, just format the offset in decimal.  Here there
;  2744          2849  2          ! are no leading zeroes to worry about.
;  2745          2850  2          !
;  2746          2851  2          [INRANGE, OUTRANGE]:
;  2747          2852  2              DBG$PRINT(UPLIT BYTE (%ASCIC '+!UL'), .VALUE);
;  2748          2853  2
;  2749          2854  2          TES;
;  2750          2855  2
;  2751          2856  2
;  2752          2857  2          ! The offset has been format (or omitted) as appropriate.  Now return.
;  2753          2858  2          !
;  2754          2859  2          RETURN;
;  2755          2860  2
;  2756          2861  1          END;


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

                                                                  4C  4F  21  00D52 P.AEM:  .ASCII  \!OL\
50  24  47  42  44  5C  54  4E  49  52  50  47  42  44  29  00D55 P.AEN:  .ASCII  \)DBGPRINT\<92>\DBG$PRINT_OFFSET, SYS$FA\
59  53  20  2C  54  45  53  46  46  4F  5F  54  4E  49  52  00D64
                                          41  46  24  53  00D73
                        64  65  6C  69  61  66  20  4F  00D77       .ASCII  \0 failed\
                                    53  41  21  2B  04  00D7F P.AEO:  .ASCII  <4>\+!AS\
                                                4C  58  21  00D84 P.AEP:  .ASCII  \!XL\
50  24  47  42  44  5C  54  4E  49  52  50  47  42  44  29  00D87 P.AEQ:  .ASCII  \)DBGPRINT\<92>\DBG$PRINT_OFFSET, SYS$FA\
59  53  20  2C  54  45  53  46  46  4F  5F  54  4E  49  52  00D96
                                          41  46  24  53  00DA5
                        64  65  6C  69  61  66  20  4F  00DA9       .ASCII  \0 failed\
                                    53  41  21  2B  04  00DB1 P.AER:  .ASCII  <4>\+!AS\
                                    4C  55  21  2B  04  00DB6 P.AES:  .ASCII  <4>\+!UL\
```

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

                                          003C 00000          .ENTRY  DBG$PRINT_OFFSET, Save R2,R3,R4,R5          ; 2710
                    55 00000000G    00     9E 00002          MOVAB   LIB$SIGNAL, R5                               ;
                    54 00000000G    00     9E 00009          MOVAB   SYS$FAO, R4                                  ;
                    53 00000000'    EF     9E 00010          MOVAB   P.AES, R3                                    ;
                    5E              28     C2 00017          SUBL2   #40, SP                                      ;
           18       AE              13     B0 0001A          MOVW    #19, OUT_DESC                                ; 2740
           1C       AE        05    AE     9E 0001E          MOVAB   OUTBUF+1, OUT_DESC+4                         ; 2741
                    07 00000000G    00     91 00023          CMPB    DBG$GB_VERB, #7                              ; 2746
                                    0C     12 0002A          BNEQ    1$                                           ;
                    50 00000000G    00     D0 0002C          MOVL    DBG$GL_CMND_RADIX, RADIX                     ; 2749
                    01              50     D1 00033          CMPL    RADIX, #1                                    ; 2750
                                    07     12 00036          BNEQ    2$                                           ;
           00000000G  00            00     FB 00038 1$:      CALLS   #0, DBG$NGET_RADIX                           ; 2755
             0F       01            50     CF 0003F 2$:      CASEL   RADIX, #1, #15                               ; 2760
  0020      0020      0020        0020        00043 3$:      .WORD   4$-3$,-                                      ;
  0027      0020      0020        0020        0004B                  4$-3$,-                                      ;
  0020      0020      0020        0020        00053                  4$-3$,-                                      ;
  0081      0020      0020        0020        0005B                  4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     5$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     4$-3$,-                                      ;
                                                                     10$-3$                                       ;
                              04    AC     DD 00063 4$:      PUSHL   VALUE                                        ; 2852
                                    53     DD 00066          PUSHL   R3                                           ;
                                    58     11 00068          BRB     9$                                           ;
           20       AE              03     B0 0006A 5$:      MOVW    #3, CONTROL_DESC                             ; 2772
           24       AE        9C    A3     9E 0006E          MOVAB   P.AEM, CONTROL_DESC+4                        ; 2773
                              04    AC     DD 00073          PUSHL   VALUE                                        ; 2774
                              1C    AE     9F 00076          PUSHAB  OUT_DESC                                     ;
                              08    AE     9F 00079          PUSHAB  FAO_LENGTH                                   ;
                              2C    AE     9F 0007C          PUSHAB  CONTROL_DESC                                 ;
                    64              04     FB 0007F          CALLS   #4, SYS$FAO                                  ;
                    0E              50     E8 00082          BLBS    R0, 6$                                       ;
                              9F    A3     9F 00085          PUSHAB  P.AEN                                        ; 2776
                                    01     DD 00088          PUSHL   #1                                           ;
                       00028362     8F     DD 0008A          PUSHL   #164706                                      ;
                    65              03     FB 00090          CALLS   #3, LIB$SIGNAL                               ;
                    52              01     D0 00093 6$:      MOVL    #1, I                                        ; 2781
                    30        04 AE42      91 00096 7$:      CMPB    OUTBUF[I], #48                               ; 2782
                                    09     12 0009B          BNEQ    8$                                           ;
                                    52     D6 0009D          INCL    I                                            ; 2784
  52        6E              10      00     ED 0009F          CMPZV   #0, #16, FAO_LENGTH, I                       ; 2785
                                    F0     18 000A4          BGEQ    7$                                           ;
                    50              6E     3C 000A6 8$:      MOVZWL  FAO_LENGTH, R0                               ; 2788
                    50              52     C2 000A9          SUBL2   I, R0                                        ;
```

```
                18   AE          50            01  A1 000AC          ADDW3   #1, R0, OUT_DESC                                    ;
                        1C   AE        04 AE42  9E 000B1          MOVAB   OUTBUF[I], OUT_DESC+4                               ; 2789
                             18   AE   B5 000B7          TSTW    OUT_DESC                                                    ; 2794
                                  73   13 000BA          BEQL    16$                                                         ;
                             18   AE   9F 000BC          PUSHAB  OUT_DESC                                                    ; 2796
                             C9   A3   9F 000BF          PUSHAB  P.AEO                                                       ;
                                  66   11 000C2 9$:       BRB     15$                                                         ;
                        20   AE        03   B0 000C4 10$:      MOVW    #3, CONTROL_DESC                                    ; 2809
                        24   AE   CE   A3   9E 000C8          MOVAB   P.AEP, CONTROL_DESC+4                               ; 2810
                             04   AC   DD 000CD          PUSHL   VALUE                                                       ; 2811
                             1C   AE   9F 000D0          PUSHAB  OUT_DESC                                                    ;
                             08   AE   9F 000D3          PUSHAB  FAO_LENGTH                                                  ;                 :
                             2C   AE   9F 000D6          PUSHAB  CONTROL_DESC                                                ;
                        64             04   FB 000D9          CALLS   #4, SYS$FAO                                         ;
                        0E             50   E8 000DC          BLBS    R0, 11$                                             ;
                             D1   A3   9F 000DF          PUSHAB  P.AEQ                                                       ; 2813
                                  01   DD 000E2          PUSHL   #1                                                          ;
                           00028362   8F   DD 000E4          PUSHL   #164706                                             ;
                        65             03   FB 000EA          CALLS   #3, LIB$SIGNAL                                      ;
                        52             01   D0 000ED 11$:      MOVL    #1, I                                               ; 2818
                        30        04 AE42   91 000F0 12$:      CMPB    OUTBUF[I], #48                                      ; 2819
                                  09   12 000F5          BNEQ    13$                                                         ;
                                  52   D6 000F7          INCL    I                                                           ; 2821
           52                6E   10   00   ED 000F9          CMPZV   #0, #16, FAO_LENGTH, I                              ; 2822
                                  F0   18 000FE          BGEQ    12$                                                         ;
                        39        04 AE42   91 00100 13$:      CMPB    OUTBUF[I], #57                                      ; 2828
                                  07   1B 00105          BLEQU   14$                                                         ;
                                  52   D7 00107          DECL    I                                                           ; 2831
                             04 AE42   30   90 00109          MOVB    #48, OUTBUF[I]                                      ; 2832
                                  50   6E   3C 0010E 14$:      MOVZWL  FAO_LENGTH, R0                                      ; 2835
                                  50   52   C2 00111          SUBL2   I, R0                                               ;
                18   AE             50   01   A1 00114          ADDW3   #1, R0, OUT_DESC                                    ;
                             1C   AE   04 AE42  9E 00119          MOVAB   OUTBUF[I], OUT_DESC+4                               ; 2836
                             18   AE   B5 0011F          TSTW    OUT_DESC                                                    ; 2841
                                  0B   13 00122          BEQL    16$                                                         ;
                             18   AE   9F 00124          PUSHAB  OUT_DESC                                                    ; 2843
                             FB   A3   9F 00127          PUSHAB  P.AER                                                       ;
                        EFCD   CF        02   FB 0012A 15$:      CALLS   #2, DBG$PRINT                                       ;
                                       04 0012F 16$:      RET                                                         ; 2861

; Routine Size:  304 bytes,    Routine Base:  DBG$CODE + 1080
```

```
;  2758          2862  1 GLOBAL ROUTINE DBG$PRINT_SET_VALUE(VALPTR): NOVALUE =
;  2759          2863  1 !
;  2760          2864  1 ! FUNCTION
;  2761          2865  1 !       Given value descriptor for Set data, print its value.
;  2762          2866  1 !
;  2763          2867  1 ! INPUTS
;  2764          2868  1 !       VALPTR  - Pointer to value descriptor.
;  2765          2869  1 !
;  2766          2870  1 ! OUTPUTS
;  2767          2871  1 !       None.
;  2768          2872  1 !
;  2769          2873  1
;  2770          2874  2     BEGIN
;  2771          2875  2
;  2772          2876  2     MAP
;  2773          2877  2         VALPTR: REF DBG$VALDESC;        ! Pointer to value descriptor
;  2774          2878  2
;  2775          2879  2     LOCAL
;  2776          2880  2         COMMA_FLAG,                     ! Flag to indicate it is time to print ,
;  2777          2881  2         DUMMY,                          ! Subrange typeid in Set
;  2778          2882  2         PARENT_TYPE: REF RST$ENTRY,     ! Set parent typeid
;  2779          2883  2         PRINT_FLAG,                     ! Flag to indicate it is time to print
;  2780          2884  2                                         !       the set value
;  2781          2885  2         PRINT_VALPTR: REF DBG$VALDESC,  ! Pointer to Print value descriptor
;  2782          2886  2         SETVALUE: REF BITVECTOR[],      ! A bit vector of set value
;  2783          2887  2         SIZE;                           ! The length of the set in bits
;  2784          2888  2
;  2785          2889  2
;  2786          2890  2
;  2787          2891  2     PARENT_TYPE = DBG$GET_SET_TYPEID(.VALPTR[DBG$L_DHDR_TYPEID], DUMMY);
;  2788          2892  2     PRINT_VALPTR = DBG$MAKE_SKELETON_DESC(DBG$K_VALUE_DESC, 4);
;  2789          2893  2     PRINT_VALPTR[DBG$B_DHDR_LANG] = .VALPTR[DBG$B_DHDR_LANG];
;  2790          2894  2     PRINT_VALPTR[DBG$B_DHDR_KIND] = .PARENT_TYPE[RST$B_KIND];
;  2791          2895  2     PRINT_VALPTR[DBG$B_DHDR_FCODE] = .PARENT_TYPE[RST$B_FCODE];
;  2792          2896  2     PRINT_VALPTR[DBG$L_DHDR_TYPEID] = .PARENT_TYPE;
;  2793          2897  2     IF .PRINT_VALPTR[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC
;  2794          2898  2     THEN
;  2795          2899  3         BEGIN
;  2796          2900  3         DBG$STA_TYP_ATOMIC(.PARENT_TYPE, PRINT_VALPTR[DBG$B_VALUE_DTYPE], SIZE);
;  2797          2901  3         IF .PRINT_VALPTR[DBG$B_VALUE_DTYPE] EQL DST$K_BOOL
;  2798          2902  3         THEN
;  2799          2903  3             PRINT_VALPTR[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_TF;
;  2800          2904  3
;  2801          2905  3         PRINT_VALPTR[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  2802          2906  3         END
;  2803          2907  3
;  2804          2908  2     ELSE
;  2805          2909  3         BEGIN
;  2806          2910  3         PRINT_VALPTR[DBG$B_VALUE_CLASS] = 0;
;  2807          2911  3         PRINT_VALPTR[DBG$B_VALUE_DTYPE] = 0;
;  2808          2912  3         SIZE = 32;
;  2809          2913  2         END;
;  2810          2914  2
;  2811          2915  2     PRINT_VALPTR[DBG$W_VALUE_LENGTH] = .SIZE / 8;
;  2812          2916  2     PRINT_VALPTR[DBG$L_VALUE_POINTER] = PRINT_VALPTR[DBG$A_VALUE_ADDRESS];
;  2813          2917  2     DBG$PRINT(UPLIT BYTE(%ASCIC '['));
;  2814          2918  2     SETVALUE = .VALPTR[DBG$L_VALUE_POINTER];
```

```
;  2815          2919  2      PRINT_FLAG = TRUE;
;  2816          2920  2      COMMA_FLAG = FALSE;
;  2817          2921  2      INCR I FROM 0 TO (.VALPTR[DBG$W_VALUE_LENGTH] * 8 - 1) DO
;  2818          2922  3          BEGIN
;  2819          2923  3          IF .SETVALUE[.I]
;  2820          2924  3          THEN
;  2821          2925  4              BEGIN
;  2822          2926  4              IF .PRINT_FLAG
;  2823          2927  4              THEN
;  2824          2928  5                  BEGIN
;  2825          2929  5                  IF .COMMA_FLAG
;  2826          2930  5                  THEN
;  2827          2931  5                      DBG$PRINT(UPLIT BYTE(%ASCIC ', '));
;  2828          2932  5
;  2829          2933  5                  COMMA_FLAG = TRUE;
;  2830          2934  5                  PRINT_VALPTR[DBG$L_VALUE_VALUE0] = .I;
;  2831          2935  5                  DBG$PRINT_VALUE(.PRINT_VALPTR, DBG$K_DEFAULT, FALSE, FALSE);
;  2832          2936  5                  IF .I EQL (.VALPTR[DBG$W_VALUE_LENGTH] * 8 - 1) THEN EXITLOOP;
;  2833          2937  5                  IF .SETVALUE[.I+1]
;  2834          2938  5                  THEN
;  2835          2939  6                      BEGIN
;  2836          2940  6                      IF (.I + 1) EQL (.VALPTR[DBG$W_VALUE_LENGTH] * 8 - 1) OR
;  2837          2941  7                         (NOT .SETVALUE[.I + 2])
;  2838          2942  6                      THEN
;  2839          2943  6                          DBG$PRINT(UPLIT BYTE(%ASCIC ', '))
;  2840          2944  6
;  2841          2945  6                      ELSE
;  2842          2946  6                          DBG$PRINT(UPLIT BYTE(%ASCIC '..'));
;  2843          2947  6
;  2844          2948  6                      PRINT_FLAG = FALSE;
;  2845          2949  5                      END;
;  2846          2950  5
;  2847          2951  5                  END
;  2848          2952  5
;  2849          2953  4              ELSE
;  2850          2954  5                  BEGIN
;  2851          2955  6                  IF .I EQL (.VALPTR[DBG$W_VALUE_LENGTH] * 8 - 1)
;  2852          2956  5                  THEN
;  2853          2957  6                      BEGIN
;  2854          2958  6                      PRINT_VALPTR[DBG$L_VALUE_VALUE0] = .I;
;  2855          2959  6                      DBG$PRINT_VALUE(.PRINT_VALPTR, DBG$K_DEFAULT, FALSE, FALSE);
;  2856          2960  6                      EXITLOOP;
;  2857          2961  5                      END;
;  2858          2962  5
;  2859          2963  5                  IF NOT .SETVALUE[.I + 1]
;  2860          2964  5                  THEN
;  2861          2965  6                      BEGIN
;  2862          2966  6                      PRINT_VALPTR[DBG$L_VALUE_VALUE0] = .I;
;  2863          2967  6                      DBG$PRINT_VALUE(.PRINT_VALPTR, DBG$K_DEFAULT, FALSE, FALSE);
;  2864          2968  6                      PRINT_FLAG = TRUE;
;  2865          2969  5                      END;
;  2866          2970  5
;  2867          2971  4                  END;
;  2868          2972  4
;  2869          2973  3              END;
;  2870          2974  3
;  2871          2975  2      END;
```

```
;  2872          2976  2
;  2873          2977  2     DBG$PRINT(UPLIT BYTE(%ASCIC ']'));
;  2874          2978  2     RETURN 0;
;  2875          2979  1     END;


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

                                                    5B  01  00DBB P.AET:  .ASCII  <1>\[\                                    ;
                                                20  2C  02  00DBD P.AEU:  .ASCII  <2>\, \                                    ;
                                                20  2C  02  00DC0 P.AEV:  .ASCII  <2>\, \                                    ;
                                                2E  2E  02  00DC3 P.AEW:  .ASCII  <2>\..\                                    ;
                                                    5D  01  00DC6 P.AEX:  .ASCII  <1>\]\                                    ;


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

                                                          0FFC 00000        .ENTRY  DBG$PRINT_SET_VALUE, Save R2,R3,R4,R5,R6,-      ; 2862
                                                                                    R7,R8,R9,R10,R11                                ;
                                                5B 00000000G   00  9E 00002         MOVAB   DBG$PRINT_VALUE, R11                    ;
                                                5A 00000000'   EF  9E 00009         MOVAB   P.AET, R10                              ;
                                                5E             08  C2 00010         SUBL2   #8, SP                                  ;
                                                               5E  DD 00013         PUSHL   SP                                      ; 2891
                                                54        04   AC  D0 00015         MOVL    VALPTR, R4                              ;
                                                           08  A4  DD 00019         PUSHL   8(R4)                                   ;
                                            00000000G   00         02  FB 0001C    CALLS   #2, DBG$GET_SET_TYPEID                  ;
                                                55             50  D0 00023         MOVL    R0, PARENT_TYPE                         ;
                                                               04  DD 00026         PUSHL   #4                                      ; 2892
                                                7E        7A   8F  9A 00028         MOVZBL  #122, -(SP)                             ;
                                            00000000G   00         02  FB 0002C    CALLS   #2, DBG$MAKE_SKELETON_DESC              ;
                                                53             50  D0 00033         MOVL    R0, PRINT_VALPTR                        ;
                                                03   A3   03   A4  90 00036         MOVB    3(R4), 3(PRINT_VALPTR)                  ; 2893
                                                07   A3   14   A5  90 0003B         MOVB    20(PARENT_TYPE), 7(PRINT_VALPTR)        ; 2894
                                                06   A3   18   A5  90 00040         MOVB    24(PARENT_TYPE), 6(PRINT_VALPTR)        ; 2895
                                                08   A3             55  D0 00045    MOVL    PARENT_TYPE, 8(PRINT_VALPTR)            ; 2896
                                                     52   14   A3  9E 00049         MOVAB   20(PRINT_VALPTR), R2                    ; 2901
                                                     02   06   A3  91 0004D         CMPB    6(PRINT_VALPTR), #2                     ; 2897
                                                               20  12 00051         BNEQ    2$                                      ;
                                                          04   AE  9F 00053         PUSHAB  SIZE                                    ; 2900
                                                          16   A3  9F 00056         PUSHAB  22(PRINT_VALPTR)                        ;
                                                               55  DD 00059         PUSHL   PARENT_TYPE                             ;
                                            00000000G   00         03  FB 0005B    CALLS   #3, DBG$STA_TYP_ATOMIC                  ;
                                                9E   8F   02   A2  91 00062         CMPB    2(R2), #158                             ; 2901
                                                               04  12 00067         BNEQ    1$                                      ;
                                                02   A2             28  90 00069    MOVB    #40, 2(R2)                              ; 2903
                                                03   A2             01  90 0006D 1$: MOVB    #1, 3(R2)                               ; 2905
                                                               07  11 00071         BRB     3$                                      ; 2897
                                                          02   A2  B4 00073 2$:     CLRW    2(R2)                                   ; 2911
                                                04   AE             20  D0 00076    MOVL    #32, SIZE                               ; 2912
                                           50   04   AE             08  C7 00079 3$: DIVL3   #8, SIZE, R0                            ; 2915
                                                62                  50  B0 0007F    MOVW    R0, (R2)                                ;
                                                18   A3   20   A3  9E 00082         MOVAB   32(PRINT_VALPTR), 24(PRINT_VALPTR)      ; 2916
                                                               5A  DD 00087         PUSHL   R10                                     ; 2917
                                                EF3E   CF          01  FB 00089    CALLS   #1, DBG$PRINT                           ;
                                                     56   18   A4  D0 0008E         MOVL    24(R4), SETVALUE                        ; 2918
```

```
                57          01  7D 00092          MOVQ    #1, PRINT_FLAG                          ; 2919
                50      14  A4  3C 00095          MOVZWL  20(R4), R0                              ; 2921
          59    50          03  78 00099          ASHL    #3, R0, R9                              ;
                52          01  CE 0009D          MNEGL   #1, I                                   ; 2923
                            54  11 000A0          BRB     9$                                      ;
          50    66          52  E1 000A2  4$:     BBC     I, (SETVALUE), 9$                       ;
                55      01  A2  9E 000A6          MOVAB   1(R2), R5                               ; 2937
                4B          57  E9 000AA          BLBC    PRINT_FLAG, 10$                         ; 2926
                08          58  E9 000AD          BLBC    COMMA_FLAG, 5$                          ; 2929
                        02  AA  9F 000B0          PUSHAB  P.AEU                                   ; 2931
            EF14    CF          01  FB 000B3          CALLS   #1, DBG$PRINT                       ;
                58              01  D0 000B8  5$:     MOVL    #1, COMMA_FLAG                      ; 2933
              20    A3          52  D0 000BB          MOVL    I, 32(PRINT_VALPTR)                 ; 2934
                                7E  7C 000BF          CLRQ    -(SP)                               ; 2935
                                01  DD 000C1          PUSHL   #1                                  ;
                                53  DD 000C3          PUSHL   PRINT_VALPTR                        ;
                6B              04  FB 000C5          CALLS   #4, DBG$PRINT_VALUE                 ;
                50      14  A4  3C 000C8          MOVZWL  20(R4), R0                              ; 2936
                50              08  C4 000CC          MULL2   #8, R0                              ;
                                50  D7 000CF          DECL    R0                                  ;
                50              52  D1 000D1          CMPL    I, R0                               ;
                                5C  13 000D4          BEQL    14$                                 ;
          4F    66              55  E1 000D6          BBC     R5, (SETVALUE), 12$                 ; 2937
                50              55  D1 000DA          CMPL    R5, R0                              ; 2940
                                08  13 000DD          BEQL    6$                                  ;
                50      02  A2  9E 000DF          MOVAB   2(R2), R0                               ; 2941
          05    66              50  E0 000E3          BBS     R0, (SETVALUE), 7$                  ;
                        05  AA  9F 000E7  6$:     PUSHAB  P.AEV                                   ; 2943
                            03  11 000EA          BRB     8$                                      ;
                        08  AA  9F 000EC  7$:     PUSHAB  P.AEW                                   ; 2946
            EED8    CF          01  FB 000EF  8$:     CALLS   #1, DBG$PRINT                       ;
                                57  D4 000F4          CLRL    PRINT_FLAG                          ; 2948
                            31  11 000F6  9$:     BRB     12$                                     ; 2926
                50      14  A4  3C 000F8  10$:    MOVZWL  20(R4), R0                              ; 2955
                50              08  C4 000FC          MULL2   #8, R0                              ;
                                50  D7 000FF          DECL    R0                                  ;
                50              52  D1 00101          CMPL    I, R0                               ;
                                0F  12 00104          BNEQ    11$                                 ;
              20    A3          52  D0 00106          MOVL    I, 32(PRINT_VALPTR)                 ; 2958
                                7E  7C 0010A          CLRQ    -(SP)                               ; 2959
                                01  DD 0010C          PUSHL   #1                                  ;
                                53  DD 0010E          PUSHL   PRINT_VALPTR                        ;
                6B              04  FB 00110          CALLS   #4, DBG$PRINT_VALUE                 ;
                            1D  11 00113          BRB     14$                                     ; 2957
          10    66          55  E0 00115  11$:     BBS     R5, (SETVALUE), 12$                     ; 2963
              20    A3          52  D0 00119          MOVL    I, 32(PRINT_VALPTR)                 ; 2966
                                7E  7C 0011D          CLRQ    -(SP)                               ; 2967
                                01  DD 0011F          PUSHL   #1                                  ;
                                53  DD 00121          PUSHL   PRINT_VALPTR                        ;
                6B              04  FB 00123          CALLS   #4, DBG$PRINT_VALUE                 ;
                57              01  D0 00126          MOVL    #1, PRINT_FLAG                      ; 2968
          02    52          59  F2 00129  12$:     AOBLSS  R9, I, 13$                              ; 2921
                            03  11 0012D          BRB     14$                                     ;
                        FF70    31 0012F  13$:     BRW     4$                                      ;
                        0B  AA  9F 00132  14$:     PUSHAB  P.AEX                                   ; 2977
            EE92    CF          01  FB 00135          CALLS   #1, DBG$PRINT                       ;
                                04  13A          RET                                             ; 2979
```

; Routine Size: 315 bytes,   Routine Base: DBG$CODE + 11B0

```
;  2877          2980  1 GLOBAL ROUTINE DBG$PRINT_SET_LANGUAGE(LANGUAGE) : NOVALUE =
;  2878          2981  1 !
;  2879          2982  1 ! FUNCTION:
;  2880          2983  1 !       This routine is called from the set language command.  It sets two
;  2881          2984  1 !       global flags to indicate whether the language has the array or record
;  2882          2985  1 !       type of data.  When we do the symbolization, we symbolize to array
;  2883          2986  1 !       subscript or record component by the current language setting.  If the
;  2884          2987  1 !       current language does not have array or record then we'll symbolize
;  2885          2988  1 !       to data+offset.  For example, if we debug a variable declared in
;  2886          2989  1 !       fortran, but the language set to macro, then we'll symbolize the
;  2887          2990  1 !       variable to data+offset instead to data(i,j).
;  2888          2991  1 !
;  2889          2992  1 ! INPUTS
;  2890          2993  1 !       LANGUAGE - The current language setting.
;  2891          2994  1 !
;  2892          2995  1 ! OUTPUTS
;  2893          2996  1 !       None.
;  2894          2997  1 !
;  2895          2998  1
;  2896          2999  2     BEGIN
;  2897          3000  2
;  2898          3001  2     LOCAL
;  2899          3002  2         PTR: REF VECTOR[,LONG];
;  2900          3003  2
;  2901          3004  2
;  2902          3005  2     ! Load in language independent print information table.  The rest of
;  2903          3006  2     ! the information is set by the language code in the primary/value
;  2904          3007  2     ! descritpor.  But, for the subscripting or record component is printed
;  2905          3008  2     ! by the current language setting.
;  2906          3009  2     !
;  2907          3010  2     PTR = .LANGUAGE_PRINT_TABLE_PTRS[.LANGUAGE] + TABLEBASE;
;  2908          3011  2     DBG$GL_ARRSUB_FLAG = .PTR[3];
;  2909          3012  2     DBG$GL_RECCMP_FLAG = .PTR[4];
;  2910          3013  2     DBG$GL_SIGN_FLAG = .PTR[5];
;  2911          3014  2     RETURN 0;
;  2912          3015  1     END;


                                                0000 00000          .ENTRY  DBG$PRINT_SET_LANGUAGE, Save nothing          ; 2980
                                    50      04  AC   D0 00002       MOVL    LANGUAGE, R0                                  ; 3010
                                    51 00000000'EF   9E 00006       MOVAB   TABLEBASE, R1                                 ;
                                50  51 00000000'EF40   C1 0000D       ADDL3   LANGUAGE_PRINT_TABLE_PTRS[R0], R1, PTR        ;
                   00000000'    EF      0C  A0   7D 00016       MOVQ    12(PTR), DBG$GL_ARRSUB_FLAG                   ; 3011
                   00000000'    EF      14  A0   D0 0001E       MOVL    20(PTR), DBG$GL_SIGN_FLAG                     ; 3013
                                                04 00026       RET                                                   ; 3015

; Routine Size:  39 bytes,    Routine Base:  DBG$CODE + 12EB

;  2913          3016  1
```

```
;  2915          3017  1 GLOBAL ROUTINE DBG$PRINT_SYMBOL_NAME(SYMID) : NOVALUE =
;  2916          3018  1 !
;  2917          3019  1 ! FUNCTION
;  2918          3020  1 !       This routine prints the symbol name without pathname qualification
;  2919          3021  1 !       of a symbol specified by a SYMID.  If that SYMID is zero, nothing
;  2920          3022  1 !       is printed.  The printing is achieved by calling DBG$PRINT.
;  2921          3023  1 !
;  2922          3024  1 ! INPUTS
;  2923          3025  1 !       SYMID   - The SYMID of the symbol whose name is to be printed.
;  2924          3026  1 !
;  2925          3027  1 ! OUTPUTS
;  2926          3028  1 !       NONE
;  2927          3029  1 !
;  2928          3030  1
;  2929          3031  2     BEGIN
;  2930          3032  2
;  2931          3033  2     LOCAL
;  2932          3034  2         NAMEPTR;                        ! Pointer to ASCIC symbol name
;  2933          3035  2
;  2934          3036  2
;  2935          3037  2
;  2936          3038  2     ! If the SYMID is non-zero, print the corresponding symbol name.
;  2937          3039  2     !
;  2938          3040  2     IF .SYMID EQL 0 THEN RETURN;
;  2939          3041  2     DBG$STA_SYMNAME(.SYMID,NAMEPTR);
;  2940          3042  2     DBG$PRINT(UPLIT BYTE(%ASCIC '!AC'), .NAMEPTR);
;  2941          3043  2     RETURN;
;  2942          3044  2
;  2943          3045  1     END;


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

                        43  41  21  03  00DC8 P.AEY:  .ASCII  <3>\!AC\                          ;


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

                                    0000 00000          .ENTRY  DBG$PRINT_SYMBOL_NAME, Save nothing     ; 3017
                    5E                  04  C2 00002     SUBL2   #4, SP
                                04      AC  D5 00005     TSTL    SYMID                                  ; 3040
                                        19  13 00008     BEQL    1$
                                        5E  DD 0000A     PUSHL   SP                                     ; 3041
                                04      AC  DD 0000C     PUSHL   SYMID
          00000000G   00                02  FB 0000F     CALLS   #2, DBG$STA_SYMNAME
                                        6E  DD 00016     PUSHL   NAMEPTR                                ; 3042
                            00000000'   EF  9F 00018     PUSHAB  P.AEY
              EE47      CF                02  FB 0001E     CALLS   #2, DBG$PRINT
                                            04 00023 1$:  RET                                            ; 3045

; Routine Size:  36 bytes,    Routine Base:  DBG$CODE + 1312
```

```
;  2945          3046  1 GLOBAL ROUTINE DBG$PRINT_SYMBOL_PATHNAME(SYMID) : NOVALUE =
;  2946          3047  1 !
;  2947          3048  1 ! FUNCTION
;  2948          3049  1 !       This routine prints the symbol name including pathname qualification
;  2949          3050  1 !       of a symbol specified by a SYMID.  If that SYMID is zero, nothing
;  2950          3051  1 !       is printed.  The printing is achieved by calling DBG$PRINT.
;  2951          3052  1 !
;  2952          3053  1 ! INPUTS
;  2953          3054  1 !       SYMID   - The SYMID of the symbol whose name and pathname qualfication
;  2954          3055  1 !                 is to be printed.
;  2955          3056  1 !
;  2956          3057  1 ! OUTPUTS
;  2957          3058  1 !       NONE
;  2958          3059  1 !
;  2959          3060  1
;  2960          3061  2     BEGIN
;  2961          3062  2
;  2962          3063  2     LOCAL
;  2963          3064  2         PATHNAME_DESC: REF PTH$PATHNAME,! Pointer to Pathname Descr.
;  2964          3065  2         SYMBOL_NAME: REF VECTOR[,BYTE]; ! Pointer to counted ASCII string
;  2965          3066  2
;  2966          3067  2
;  2967          3068  2     ! Get a Pathname Descriptor for the SYMID symbol from SYMPATHNAME, convert
;  2968          3069  2     ! that to a Counted ASCII string through NPATHDESC_TO_CS, and then print
;  2969          3070  2     ! that string and return.
;  2970          3071  2     !
;  2971          3072  2     DBG$STA_SYMPATHNAME (.SYMID, PATHNAME_DESC);
;  2972          3073  2     DBG$NPATHDESC_TO_CS (.PATHNAME_DESC, SYMBOL_NAME);
;  2973          3074  2     DBG$PRINT (UPLIT BYTE (%ASCIC '!AC'), .SYMBOL_NAME);
;  2974          3075  2     RETURN;
;  2975          3076  2
;  2976          3077  1     END;

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

                                43  41  21  03  00DCC P.AEZ:  .ASCII  <3>\!AC\                                  ;

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

                                      0000 00000        .ENTRY  DBG$PRINT_SYMBOL_PATHNAME, Save nothing       ; 3046
                            5E          08  C2 00002     SUBL2   #8, SP                                        ;
                                        5E  DD 00005     PUSHL   SP                                            ; 3072
                                    04  AC  DD 00007     PUSHL   SYMID                                         ;
                00000000G   00          02  FB 0000A     CALLS   #2, DBG$STA_SYMPATHNAME                       ;
                                    04  AE  9F 00011     PUSHAB  SYMBOL_NAME                                   ; 3073
                                    04  AE  DD 00014     PUSHL   PATHNAME_DESC                                 ;
                00000000G   00          02  FB 00017     CALLS   #2, DBG$NPATHDESC_TO_CS                       ;
                                    04  AE  DD 0001E     PUSHL   SYMBOL_NAME                                   ; 3074
                                00000000' EF  9F 00021     PUSHAB  P.AEZ                                         ;
                    EE1A    CF          02  FB 00027     CALLS   #2, DBG$PRINT                                 ;
                                            04 0002C     RET                                                   ; 3077

; Routine Size:  45 bytes,    Routine Base:  DBG$CODE + 1336
```

```
;  2978          3078  1 GLOBAL ROUTINE DBG$SET_MAX_LINE_WIDTH(LINE_WIDTH): NOVALUE =
;  2979          3079  1 !
;  2980          3080  1 ! FUNCTION
;  2981          3081  1 !       This routine sets the maximum width of DEBUG output lines so that
;  2982          3082  1 !       output lines that exceed this width get folded at the maximum width.
;  2983          3083  1 !       The maximum width cannot be set to a value less than 40 or greater
;  2984          3084  1 !       than 131 characters.  The width setting is accomplished by setting
;  2985          3085  1 !       the new width into the OWN variable MAX_PBSIZE.
;  2986          3086  1 !
;  2987          3087  1 ! INPUTS
;  2988          3088  1 !       LINE_WIDTH - The new maximum print line width to be set.
;  2989          3089  1 !
;  2990          3090  1 ! OUTPUTS
;  2991          3091  1 !       NONE
;  2992          3092  1 !
;  2993          3093  1
;  2994          3094  2     BEGIN
;  2995          3095  2
;  2996          3096  2
;  2997          3097  2
;  2998          3098  2     ! Set the new maximum line width and make sure it is within range.
;  2999          3099  2     !
;  3000          3100  2     MAX_PBSIZE = .LINE_WIDTH;
;  3001          3101  2     IF .MAX_PBSIZE LSS 40 THEN MAX_PBSIZE = 40;
;  3002          3102  2     IF .MAX_PBSIZE GTR 131 THEN MAX_PBSIZE = 131;
;  3003          3103  2     RETURN;
;  3004          3104  2
;  3005          3105  1     END;

                                    0004 00000          .ENTRY  DBG$SET_MAX_LINE_WIDTH, Save R2          ;  3078
                          52 00000000'  EF  9E 00002    MOVAB   MAX_PBSIZE, R2                           ;
                          62        04  AC  D0 00009    MOVL    LINE_WIDTH, MAX_PBSIZE                   ;  3100
                          28            62  D1 0000D    CMPL    MAX_PBSIZE, #40                          ;  3101
                                        03  18 00010    BGEQ    1$                                       ;
                          62            28  D0 00012    MOVL    #40, MAX_PBSIZE                          ;
                00000083  8F            62  D1 00015 1$: CMPL    MAX_PBSIZE, #131                         ;  3102
                                        04  15 0001C    BLEQ    2$                                       ;
                          62        83  8F  9A 0001E    MOVZBL  #131, MAX_PBSIZE                         ;
                                        04  00022 2$:   RET                                              ;  3105

; Routine Size:  35 bytes,    Routine Base:  DBG$CODE + 1363
```

```
;  3007          3106  1 GLOBAL ROUTINE DBG$WRITE_LOG_FILE(LENGTH, BUFFER): NOVALUE =
;  3008          3107  1 !
;  3009          3108  1 ! FUNCTION
;  3010          3109  1 !       This routine writes an already formatted line of output to the DEBUG
;  3011          3110  1 !       log file.  It accepts the length and address of the output line as
;  3012          3111  1 !       input and outputs that buffer to the log file.  It then returns.  It
;  3013          3112  1 !       is assumed that the caller has inserted the leading "!" in the out-
;  3014          3113  1 !       put line (if appropriate) before calling this routine.
;  3015          3114  1 !
;  3016          3115  1 ! INPUTS
;  3017          3116  1 !       LENGTH  - The number of bytes of text to be output to the log file
;  3018          3117  1 !                 from the BUFFER location
;  3019          3118  1 !
;  3020          3119  1 !       BUFFER  - The address of the buffer to be output to the log file.
;  3021          3120  1 !
;  3022          3121  1 ! OUTPUTS
;  3023          3122  1 !       NONE
;  3024          3123  1 !
;  3025          3124  1
;  3026          3125  2     BEGIN
;  3027          3126  2
;  3028          3127  2     LOCAL
;  3029          3128  2         FAB_PTR: REF $FAB_DECL,          ! Pointer to the log file FAB
;  3030          3129  2         MSG_DESC: BLOCK[8,BYTE],         ! String descriptor pointing to file
;  3031          3130  2                                          !       name--used in error messages
;  3032          3131  2         STATUS;                          ! Status code returned by RMS
;  3033          3132  2
;  3034          3133  2
;  3035          3134  2
;  3036          3135  2     ! If output logging is not active, return immediately without writing
;  3037          3136  2     ! anything to the log file.
;  3038          3137  2     !
;  3039          3138  2     IF NOT .DBG$GB_DEF_OUT[OUT_LOG] THEN RETURN;
;  3040          3139  2
;  3041          3140  2
;  3042          3141  2     ! Output the buffer to the log file and check the status.  Then return.
;  3043          3142  2     !
;  3044          3143  2     DBG$GL_LOGRAB[RAB$L_RBF] = .BUFFER;
;  3045          3144  2     DBG$GL_LOGRAB[RAB$W_RSZ] = .LENGTH;
;  3046          3145  2     STATUS = $PUT(RAB = DBG$GL_LOGRAB);
;  3047          3146  2
;  3048          3147  2
;  3049          3148  2     ! If we got a record stream active error, wait for the current I/O on
;  3050          3149  2     ! the file to finish and then try the I/O again.
;  3051          3150  2     !
;  3052          3151  2     IF .STATUS EQL RMS$_RSA
;  3053          3152  2     THEN
;  3054          3153  3         BEGIN
;  3055          3154  3         $WAIT(RAB = DBG$GL_LOGRAB);
;  3056          3155  3         STATUS = $PUT(RAB = DBG$GL_LOGRAB);
;  3057          3156  2         END;
;  3058          3157  2
;  3059          3158  2
;  3060          3159  2     ! If the I/O operation failed, signal the error.  The error message
;  3061          3160  2     ! will include the file name and the STS and STV status values.
;  3062          3161  2     !
;  3063          3162  2     IF NOT .STATUS
```

```
;  3064          3163  2     THEN
;  3065          3164  3         BEGIN
;  3066          3165  3         MSG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  3067          3166  3         MSG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  3068          3167  3         FAB_PTR = .DBG$GL_LOGRAB[RAB$L_FAB];
;  3069          3168  3         IF .DBG$GL_LOG_BUF NEQ 0
;  3070          3169  3         THEN
;  3071          3170  4             BEGIN
;  3072          3171  4             MSG_DESC[DSC$W_LENGTH] = .FAB_PTR[FAB$B_FNS];
;  3073          3172  4             MSG_DESC[DSC$A_POINTER] = .FAB_PTR[FAB$L_FNA];
;  3074          3173  4             END
;  3075          3174  4
;  3076          3175  3         ELSE
;  3077          3176  4             BEGIN
;  3078          3177  4             MSG_DESC[DSC$W_LENGTH] = .FAB_PTR[FAB$B_DNS];
;  3079          3178  4             MSG_DESC[DSC$A_POINTER] = .FAB_PTR[FAB$L_DNA];
;  3080          3179  3             END;
;  3081          3180  3
;  3082          3181  3         SIGNAL (SHR$_WRITEERR + DBG_FAC_CODE, 1, MSG_DESC,
;  3083          3182  3                 .DBG$GL_LOGRAB[RAB$L_STS], .DBG$GL_LOGRAB[RAB$L_STV]);
;  3084          3183  3
;  3085          3184  2         END;
;  3086          3185  2
;  3087          3186  2
;  3088          3187  2     ! Everything worked fine--return to the caller.
;  3089          3188  2     !
;  3090          3189  2     RETURN;
;  3091          3190  2
;  3092          3191  1     END;


                                                .EXTRN  SYS$PUT, SYS$WAIT

                                001C 00000      .ENTRY  DBG$WRITE_LOG_FILE, Save R2,R3,R4                   ; 3106
                54 00000000G    00   9E 00002  MOVAB   SYS$PUT, R4
                53 00000000G    00   9E 00009  MOVAB   DBG$GL_LOGRAB, R3
                5E                   08 C2 00010  SUBL2   #8, SP
                6B 00000000G    00   E9 00013  BLBC    DBG$GB_DEF_OUT, 4$                                  ; 3138
          28    A3        08    AC   D0 0001A  MOVL    BUFFER, DBG$GL_LOGRAB+40                            ; 3143
          22    A3        04    AC   B0 0001F  MOVW    LENGTH, DBG$GL_LOGRAB+34                            ; 3144
                                  53   DD 00024  PUSHL   R3                                                 ; 3145
                64                01   FB 00026  CALLS   #1, SYS$PUT
                52                50   D0 00029  MOVL    R0, STATUS
000182DA        8F                52   D1 0002C  CMPL    STATUS, #99034                                    ; 3151
                                  11   12 00033  BNEQ    1$
                                  53   DD 00035  PUSHL   R3                                                 ; 3154
00000000G       00                01   FB 00037  CALLS   #1, SYS$WAIT
                                  53   DD 0003E  PUSHL   R3                                                 ; 3155
                64                01   FB 00040  CALLS   #1, SYS$PUT
                52                50   D0 00043  MOVL    R0, STATUS
                3C                52   E8 00046  1$:     BLBS    STATUS, 4$                                  ; 3162
          02    AE      010E      8F   B0 00049  MOVW    #270, MSG_DESC+2                                    ; 3166
                50        3C      A3   D0 0004F  MOVL    DBG$GL_LOGRAB+60, FAB_PTR                           ; 3167
                   00000000G      00   D5 00053  TSTL    DBG$GL_LOG_BUF                                      ; 3168
                                  0B   13 00059  BEQL    2$
                6E        34      A0   9B 0005B  MOVZBW  52(FAB_PTR), MSG_DESC                              ; 3171
```

```
                04   AE      2C   A0  DO 0005F             MOVL    44(FAB_PTR), MSG_DESC+4                 ; 3172
                                  09  11 00064             BRB     3$                                      ; 3168
                     6E      35   A0  9B 00066 2$:         MOVZBW  53(FAB_PTR), MSG_DESC                   ; 3177
                04   AE      30   A0  DO 0006A             MOVL    48(FAB_PTR), MSG_DESC+4                 ; 3178
                     7E      08   A3  7D 0006F 3$:         MOVQ    DBG$GL_LOGRAB+8, -(SP)                  ; 3182
                             08   AE  9F 00073             PUSHAB  MSG_DESC                                ; 3181
                                  01  DD 00076             PUSHL   #1                                      ;
                        000210D0  8F  DD 00078             PUSHL   #135376                                 ;
                00000000G   00        05  FB 0007E             CALLS   #5, LIB$SIGNAL                          ;
                                      04 00085 4$:         RET                                             ; 3191

; Routine Size:  134 bytes,    Routine Base:  DBG$CODE + 1386
```

```
;  3094          3192  1 ROUTINE DBG$WRITE_OUTPUT(LENGTH, BUFFER): NOVALUE =
;  3095          3193  1 !
;  3096          3194  1 ! FUNCTION
;  3097          3195  1 !       This routine writes a text buffer containing an ASCII character
;  3098          3196  1 !       string to file DBG$OUTPUT (normally the user's terminal) and to
;  3099          3197  1 !       the current log file (if one exists).  The write to DBG$OUTPUT
;  3100          3198  1 !       is done using RMS, but is skipped entirely if the user has sup-
;  3101          3199  1 !       pressed it with a SET OUTPUT NOTERMINAL command.
;  3102          3200  1 !
;  3103          3201  1 !       If there is a log file, the comment character "!" is inserted in
;  3104          3202  1 !       front of the character string before the string is written out.
;  3105          3203  1 !       This is done by copying the string to a temporary log file buffer.
;  3106          3204  1 !       The output is then done by calling DBG$WRITE_LOG_FILE.
;  3107          3205  1 !
;  3108          3206  1 ! INPUTS
;  3109          3207  1 !       LENGTH  - The length in bytes of the ASCII character string to be
;  3110          3208  1 !                 output to DBG$OUTPUT and the log file.
;  3111          3209  1 !
;  3112          3210  1 !       BUFFER  - The address of the buffer containing the ASCII character
;  3113          3211  1 !                 string to be output to DBG$OUTPUT and the log file.
;  3114          3212  1 !
;  3115          3213  1 ! OUTPUTS
;  3116          3214  1 !       NONE
;  3117          3215  1 !
;  3118          3216  1
;  3119          3217  2     BEGIN
;  3120          3218  2
;  3121          3219  2     LOCAL
;  3122          3220  2         LOG_BUFFER: VECTOR[132,BYTE],   ! Temporary buffer for log file output
;  3123          3221  2         LOG_LENGTH,                     ! Length of string in log file buffer
;  3124          3222  2         MSG_DESC: BLOCK[8,BYTE],        ! String descriptor pointing to file
;  3125          3223  2                                         !     name--used in error messages
;  3126          3224  2         STATUS;                         ! RMS status code
;  3127          3225  2
;  3128          3226  2
;  3129          3227  2
;  3130          3228  2     ! If we are in Screen Mode, we direct the print output to the currently
;  3131          3229  2     ! active Screen Display instead of printing it through RMS in the normal
;  3132          3230  2     ! manner.
;  3133          3231  2     !
;  3134          3232  2     IF .DBG$GL_SCREEN_OUTPUT NEQ 0
;  3135          3233  2     THEN
;  3136          3234  2         DBG$SCR_WRITE_LINE(.LENGTH, .BUFFER, .DBG$GL_SCREEN_OUTPUT)
;  3137          3235  2
;  3138          3236  2
;  3139          3237  2     ! If the user wants output on DBG$OUTPUT (the normal situation unless
;  3140          3238  2     ! SET OUTPUT NOTERMINAL has been specified), set up and do the write.
;  3141          3239  2     ! DBG$OUTPUT is normally defined to be the user's terminal.
;  3142          3240  2     !
;  3143          3241  2     ELSE IF .DBG$GB_DEF_OUT[OUT_TERM]
;  3144          3242  2     THEN
;  3145          3243  3         BEGIN
;  3146          3244  3
;  3147          3245  3
;  3148          3246  3         ! Write the output buffer to DBG$OUTPUT.
;  3149          3247  3         !
;  3150          3248  3         DBG$GL_OUTPRAB[RAB$L_RBF] = .BUFFER;
```

```
;  3151          3249  3          DBG$GL_OUTPRAB[RAB$W_RSZ] = .LENGTH;
;  3152          3250  3          STATUS = $PUT(RAB = DBG$GL_OUTPRAB);
;  3153          3251  3
;  3154          3252  3
;  3155          3253  3          ! If we got a record stream active error, wait for the current I/O
;  3156          3254  3          ! operation to finish and then retry the write.
;  3157          3255  3          !
;  3158          3256  3          IF .STATUS EQL RMS$_RSA
;  3159          3257  3          THEN
;  3160          3258  4              BEGIN
;  3161          3259  4              $WAIT(RAB = DBG$GL_OUTPRAB);
;  3162          3260  4              STATUS = $PUT(RAB = DBG$GL_OUTPRAB);
;  3163          3261  3              END;
;  3164          3262  3
;  3165          3263  3
;  3166          3264  3          ! If we got any other I/O error, signal an error message which includes
;  3167          3265  3          ! the DBG$OUTPUT file name and the STS and STV status values.
;  3168          3266  3          !
;  3169          3267  3          IF NOT .STATUS
;  3170          3268  3          THEN
;  3171          3269  4              BEGIN
;  3172          3270  4              MSG_DESC[DSC$W_LENGTH]  = 10;
;  3173          3271  4              MSG_DESC[DSC$A_POINTER] = UPLIT BYTE(%ASCII 'DBG$OUTPUT');
;  3174          3272  4              SIGNAL(SHR$_WRITEERR + DBG_FAC_CODE, 1, MSG_DESC,
;  3175          3273  4                  .DBG$GL_OUTPRAB[RAB$L_STS], .DBG$GL_OUTPRAB[RAB$L_STV]);
;  3176          3274  3              END;
;  3177          3275  3
;  3178          3276  2          END;                                ! End of DBG$OUTPUT code
;  3179          3277  2
;  3180          3278  2
;  3181          3279  2      ! If a LOG file is being written, write the output buffer to it, but
;  3182          3280  2      ! with the comment character appended in front of it.
;  3183          3281  2      !
;  3184          3282  2      IF .DBG$GB_DEF_OUT[OUT_LOG]
;  3185          3283  2      THEN
;  3186          3284  3          BEGIN
;  3187          3285  3
;  3188          3286  3
;  3189          3287  3          ! Put the comment character "!" at the beginning of a temporary log
;  3190          3288  3          ! file buffer.  Then copy the input string into the log file buffer
;  3191          3289  3          ! after the comment character.  This causes all DEBUG output to look
;  3192          3290  3          ! like comments in the log file.
;  3193          3291  3          !
;  3194          3292  3          LOG_BUFFER[0] = '!';
;  3195          3293  3          LOG_LENGTH = MIN(.LENGTH + 1, 132);
;  3196          3294  3          CH$MOVE(.LOG_LENGTH - 1, .BUFFER, LOG_BUFFER[1]);
;  3197          3295  3
;  3198          3296  3
;  3199          3297  3          ! Write the output buffer to the log file.
;  3200          3298  3          !
;  3201          3299  3          DBG$WRITE_LOG_FILE(.LOG_LENGTH, LOG_BUFFER[0]);
;  3202          3300  2          END;
;  3203          3301  2
;  3204          3302  2
;  3205          3303  2      ! Everything worked fine--return to the caller.
;  3206          3304  2      !
;  3207          3305  2      RETURN;
```

```
;  3208          3306  2
;  3209          3307  1     END;


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

    54  55  50  54  55  4F  24  47  42  44  00DD0 P.AFA:  .ASCII  \DBG$OUTPUT\                          ;


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

                                            01FC 00000 DBG$WRITE_OUTPUT:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8               ; 3192
                                    58 00000000G   00    9E 00002          MOVAB   SYS$PUT, R8         ;
                                    57 00000000G   00    9E 00009          MOVAB   DBG$GL_OUTPRAB, R7  ;
                                    5E      FF74   CE    9E 00010          MOVAB   -140(SP), SP        ;
                                    50 00000000G   00    D0 00015          MOVL    DBG$GL_SCREEN_OUTPUT, R0    ; 3232
                                                   0F    13 0001C          BEQL    1$                  ;
                                                   50    DD 0001E          PUSHL   R0                  ; 3234
                                            7E          04  AC  7D 00020    MOVQ    LENGTH, -(SP)       ;
                                00000000G   00              03  FB 00024    CALLS   #3, DBG$SCR_WRITE_LINE  ;
                                                   57    11 0002B          BRB     3$                  ;
                                    50 00000000G   00    E9 0002D 1$:      BLBC    DBG$GB_DEF_OUT+1, 3$        ; 3241
                                28  A7          08  AC  D0 00034          MOVL    BUFFER, DBG$GL_OUTPRAB+40   ; 3248
                                22  A7          04  AC  B0 00039          MOVW    LENGTH, DBG$GL_OUTPRAB+34   ; 3249
                                                   57    DD 0003E          PUSHL   R7                  ; 3250
                                    68                  01  FB 00040      CALLS   #1, SYS$PUT         ;
                                    52                  50  D0 00043      MOVL    R0, STATUS          ;
                                000182DA    8F          52  D1 00046      CMPL    STATUS, #99034      ; 3256
                                                   11    12 0004D          BNEQ    2$                  ;
                                                   57    DD 0004F          PUSHL   R7                  ; 3259
                                00000000G   00          01  FB 00051      CALLS   #1, SYS$WAIT        ;
                                                   57    DD 00058          PUSHL   R7                  ; 3260
                                    68                  01  FB 0005A      CALLS   #1, SYS$PUT         ;
                                    52                  50  D0 0005D      MOVL    R0, STATUS          ;
                                    21                  52  E8 00060 2$:  BLBS    STATUS, 3$          ; 3267
                                    6E                  0A  B0 00063      MOVW    #10, MSG_DESC       ; 3270
                                        04      AE 00000000'    EF  9E 00066  MOVAB   P.AFA, MSG_DESC+4   ; 3271
                                    7E              08  A7  7D 0006E      MOVQ    DBG$GL_OUTPRAB+8, -(SP)     ; 3273
                                                    08  AE  9F 00072      PUSHAB  MSG_DESC            ; 3272
                                                   01    DD 00075          PUSHL   #1                  ;
                                        000210D0    8F  DD 00077          PUSHL   #135376             ;
                                00000000G   00          05  FB 0007D      CALLS   #5, LIB$SIGNAL      ;
                                    2D 00000000G   00    E9 00084 3$:      BLBC    DBG$GB_DEF_OUT, 5$          ; 3282
                                        08      AE              21  90 0008B  MOVB    #33, LOG_BUFFER     ; 3292
                                50      04      AC              01  C1 0008F  ADDL3   #1, LENGTH, R0      ; 3293
                                00000084    8F                  50  D1 00094  CMPL    R0, #132            ;
                                                   04    15 0009B          BLEQ    4$                  ;
                                    50          84  8F  9A 0009D          MOVZBL  #132, R0            ;
                                    56                  50  D0 000A1 4$:  MOVL    R0, LOG_LENGTH      ;
                                    50          FF  A6  9E 000A4          MOVAB   -1(R6), R0          ; 3294
09  AE          08  BC                              50  28 000A8          MOVC3   R0, @BUFFER, LOG_BUFFER+1   ;
                                                    08  AE  9F 000AE      PUSHAB  LOG_BUFFER          ; 3299
                                                   56    DD 000B1          PUSHL   LOG_LENGTH          ;
                                FEC2    CF                  02  FB 000B3  CALLS   #2, DBG$WRITE_LOG_FILE  ;
```

```
                04 000B8 5$:    RET                                     ; 3307

; Routine Size:  185 bytes,    Routine Base:  DBG$CODE + 140C
```

```
;  3211     3308  1 ROUTINE GET_SUBSCRIPT_VALUE(NAME, STRING, ARGUMENTS) =
;  3212     3309  1 !
;  3213     3310  1 ! FUNCTION
;  3214     3311  1 !       This routine takes an array subscript value, formats that value into
;  3215     3312  1 !       an output string using FAO, and returns a pointer to the formatted
;  3216     3313  1 !       string to the NAMEPTR parameter.
;  3217     3314  1 !
;  3218     3315  1 ! INPUTS
;  3219     3316  1 !       NAMEPTR - The address of a longword location to receive a pointer
;  3220     3317  1 !                 to the formatted array subscript value.
;  3221     3318  1 !
;  3222     3319  1 !       STRING  - The address of an ASCIC FAO control string.
;  3223     3320  1 !
;  3224     3321  1 !       ARGUMENTS - Array subscript value.
;  3225     3322  1 !
;  3226     3323  1 ! OUTPUTS
;  3227     3324  1 !       NAMEPTR - A pointer to the formatted array subscript value is returned
;  3228     3325  1 !                 to the NAMEPTR parameter.
;  3229     3326  1 !
;  3230     3327  1 !       The routine returns the value TRUE if the formatting was successful and
;  3231     3328  1 !                 it returns FALSE if the value was truncated when formatted.
;  3232     3329  1 !
;  3233     3330  1
;  3234     3331  2     BEGIN
;  3235     3332  2
;  3236     3333  2     MAP
;  3237     3334  2         NAME: REF VECTOR[,LONG],          ! Pointer to ASCIC string
;  3238     3335  2         STRING: REF VECTOR[,BYTE];        ! Address of FAO control string
;  3239     3336  2
;  3240     3337  2     LITERAL
;  3241     3338  2         SUBBUF = 20;                      ! Buffer length in bytes
;  3242     3339  2
;  3243     3340  2     LOCAL
;  3244     3341  2         INP_DESC: VECTOR[2,LONG],         ! String descriptor for input buffer
;  3245     3342  2         LENGTH: WORD,                     ! Length of SYS$FAOL output string
;  3246     3343  2         NAMEPTR: REF VECTOR[,BYTE],       ! Pointer to ASCIC string
;  3247     3344  2         OUT_DESC: VECTOR[2,LONG],         ! String descriptor for output buffer
;  3248     3345  2         STATUS;                           ! Return status from FAO
;  3249     3346  2
;  3250     3347  2
;  3251     3348  2
;  3252     3349  2     ! ----------------------------------------------------------------------
;  3253     3350  2     !
;  3254     3351  2     INP_DESC[0] = .STRING[0];
;  3255     3352  2     INP_DESC[1] = STRING[1];
;  3256     3353  2     OUT_DESC[0] = SUBBUF;
;  3257     3354  2     OUT_DESC[1] = DBG$GET_MEMORY(SUBBUF / 4);
;  3258     3355  2     STATUS = SYS$FAOL(INP_DESC, LENGTH, OUT_DESC, ARGUMENTS);
;  3259     3356  2     IF .STATUS EQL SS$_ACCVIO OR .STATUS EQL SS$_BADPARAM
;  3260     3357  2     THEN
;  3261     3358  3         BEGIN
;  3262     3359  3         DBG$REL_MEMORY(.OUT_DESC[1]);
;  3263     3360  3         SIGNAL(.STATUS);
;  3264     3361  2         END;
;  3265     3362  2
;  3266     3363  2     NAMEPTR = DBG$GET_TEMPMEM((.LENGTH + 5)/4);
;  3267     3364  2     CH$MOVE(.LENGTH, .OUT_DESC[1], NAMEPTR[1]);
```

```
;  3268          3365  2     NAMEPTR[0] = .LENGTH;
;  3269          3366  2     DBG$REL_MEMORY(.OUT_DESC[1]);
;  3270          3367  2     .NAME = .NAMEPTR;
;  3271          3368  2     IF .STATUS EQL SS$_NORMAL THEN RETURN FALSE;
;  3272          3369  2     RETURN TRUE;
;  3273          3370  2
;  3274          3371  1     END;


                                                01FC 00000 GET_SUBSCRIPT_VALUE:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 3308
                                    58 00000000G   00  9E 00002    MOVAB   DBG$REL_MEMORY, R8
                                    5E             14  C2 00009    SUBL2   #20, SP
                                0C  AE         08  BC  9A 0000C    MOVZBL  @STRING, INP_DESC                    ; 3351
                            10  AE     08      AC      01  C1 00011    ADDL3   #1, STRING, INP_DESC+4           ; 3352
                                04      AE              14  D0 00017    MOVL    #20, OUT_DESC                    ; 3353
                                                        05  DD 0001B    PUSHL   #5                               ; 3354
                                00000000G   00          01  FB 0001D    CALLS   #1, DBG$GET_MEMORY
                                    08      AE          50  D0 00024    MOVL    R0, OUT_DESC+4
                                                0C      AC  9F 00028    PUSHAB  ARGUMENTS                        ; 3355
                                                08      AE  9F 0002B    PUSHAB  OUT_DESC
                                                08      AE  9F 0002E    PUSHAB  LENGTH
                                                18      AE  9F 00031    PUSHAB  INP_DESC
                                00000000G   00          04  FB 00034    CALLS   #4, SYS$FAOL
                                    57              50  D0 0003B    MOVL    R0, STATUS
                                    0C              57  D1 0003E    CMPL    STATUS, #12                      ; 3356
                                                    05  13 00041    BEQL    1$
                                    14              57  D1 00043    CMPL    STATUS, #20
                                                    0F  12 00046    BNEQ    2$
                                                08  AE  DD 00048 1$:  PUSHL   OUT_DESC+4                      ; 3359
                                    68              01  FB 0004B    CALLS   #1, DBG$REL_MEMORY
                                                    57  DD 0004E    PUSHL   STATUS                           ; 3360
                                00000000G   00      01  FB 00050    CALLS   #1, LIB$SIGNAL
                                    50              6E  3C 00057 2$:  MOVZWL  LENGTH, R0                       ; 3363
                                    50              05  C0 0005A    ADDL2   #5, R0
                                7E  50              04  C7 0005D    DIVL3   #4, R0, -(SP)
                                00000000G   00      01  FB 00061    CALLS   #1, DBG$GET_TEMPMEM
                                    56              50  D0 00068    MOVL    R0, NAMEPTR
                        01  A6      08      BE      6E  28 0006B    MOVC3   LENGTH, @OUT_DESC+4, 1(NAMEPTR)  ; 3364
                                    66              6E  90 00071    MOVB    LENGTH, (NAMEPTR)                ; 3365
                                                08  AE  DD 00074    PUSHL   OUT_DESC+4                       ; 3366
                                    68              01  FB 00077    CALLS   #1, DBG$REL_MEMORY
                                04      BC          56  D0 0007A    MOVL    NAMEPTR, @NAME                   ; 3367
                                    01              57  D1 0007E    CMPL    STATUS, #1                       ; 3368
                                                    04  13 00081    BEQL    3$
                                    50              01  D0 00083    MOVL    #1, R0                           ; 3369
                                                    04  00086    RET
                                                    50  D4 00087 3$:  CLRL    R0                               ; 3371
                                                    04  00089    RET

; Routine Size:  138 bytes,    Routine Base:  DBG$CODE + 14C5
```

```
;  3276     3372  1 ROUTINE PRINT_PUSH(FLAG, STACK, I, NAMEPTR) =
;  3277     3373  1 !
;  3278     3374  1 ! FUNCTION
;  3279     3375  1 !       Push the Pointer to ASCIC name string on the top of the stack upwards
;  3280     3376  1 !       or downwards.
;  3281     3377  1 !
;  3282     3378  1 ! INPUTS
;  3283     3379  1 !       FLAG    - Flag to indicate the stack grows upwards or downwards.
;  3284     3380  1 !
;  3285     3381  1 !       STACK   - Pointer to the beginning of the stack.
;  3286     3382  1 !
;  3287     3383  1 !       I       - Current stack pointer.
;  3288     3384  1 !
;  3289     3385  1 !       NAMEPTR - Pointer to ASCIC name string.
;  3290     3386  1 !
;  3291     3387  1 !       arg. 5  - Flag set to indicate NAMEPTR contains a value descritpor.
;  3292     3388  1 !
;  3293     3389  1 ! OUTPUTS
;  3294     3390  1 !       TRUE or FALSE.  TRUE to indicate stack underflow or overflow.
;  3295     3391  1 !
;  3296     3392  1
;  3297     3393  2     BEGIN
;  3298     3394  2
;  3299     3395  2     MAP
;  3300     3396  2         I: REF VECTOR[,LONG],
;  3301     3397  2         NAMEPTR: REF VECTOR[,BYTE],
;  3302     3398  2         STACK: REF PRTID$STACK;
;  3303     3399  2                                                                    63
;  3304     3400  2     BUILTIN
;  3305     3401  2         ACTUALCOUNT,
;  3306     3402  2         ACTUALPARAMETER;                                           6C
;  3307     3403  2
;  3308     3404  2     LOCAL
;  3309     3405  2         VALFLAG;                                                   75
;  3310     3406  2
;  3311     3407  2
;  3312     3408  2                                                                    6E
;  3313     3409  2     ! ------------------------------------------------------------------
;  3314     3410  2     !                                                              65
;  3315     3411  2     IF (..I LSS 0) OR (..I GEQ PRINT_STACK_SIZE) THEN RETURN TRUE;
;  3316     3412  2     IF .NAMEPTR NEQ 0
;  3317     3413  2     THEN                                                           74
;  3318     3414  3         BEGIN
;  3319     3415  3         VALFLAG = (IF ACTUALCOUNT() GTR 4 THEN 1 ELSE 0);
;  3320     3416  3
;  3321     3417  3
;  3322     3418  3         ! Push upwards.
;  3323     3419  3         !
;  3324     3420  3         IF .FLAG
;  3325     3421  3         THEN
;  3326     3422  4             BEGIN                                                  69
;  3327     3423  4             STACK[..I, PRTID$STACK_PTR] = .NAMEPTR;
;  3328     3424  4             STACK[..I, PRTID$STACK_FLAG] = .VALFLAG;
;  3329     3425  4             .I = ..I - 1;                                          69
;  3330     3426  4             END                                                    63
;  3331     3427  4
;  3332     3428  4
```

```
;  3333          3429  4          ! Push downwards.
;  3334          3430  4          !
;  3335          3431  3          ELSE
;  3336          3432  4              BEGIN
;  3337          3433  4              .I = ..I + 1;
;  3338          3434  4              STACK[..I, PRTID$STACK_PTR] = .NAMEPTR;
;  3339          3435  4              STACK[..I, PRTID$STACK_FLAG] = .VALFLAG;
;  3340          3436  3              END;
;  3341          3437  3
;  3342          3438  2          END;
;  3343          3439  2
;  3344          3440  2      RETURN FALSE;
;  3345          3441  2
;  3346          3442  1      END;


                                        001C 00000 PRINT_PUSH:
                                                   .WORD   Save R2,R3,R4                ; 3372
                51          0C      AC  D0 00002           MOVL    I, R1                ; 3411
                                    61  D5 00006           TSTL    (R1)                 ;
                                    09  19 00008           BLSS    1$                   ;
00000064        8F                  61  D1 0000A           CMPL    (R1), #100           ;
                                    04  19 00011           BLSS    2$                   ;
                50                  01  D0 00013 1$:       MOVL    #1, R0               ;
                                        04 00016           RET                          ;
                53          10      AC  D0 00017 2$:       MOVL    NAMEPTR, R3          ; 3412
                                    3E  13 0001B           BEQL    6$                   ;
                04                  6C  91 0001D           CMPB    (AP), #4             ; 3415
                                    05  1B 00020           BLEQU   3$                   ;
                54                  01  D0 00022           MOVL    #1, VALFLAG          ;
                                    02  11 00025           BRB     4$                   ;
                                    54  D4 00027 3$:       CLRL    VALFLAG              ;
                18          04      AC  E9 00029 4$:       BLBC    FLAG, 5$             ; 3420
     50         61                  05  C5 0002D           MULL3   #5, (R1), R0         ; 3423
     52         50          08      AC  C1 00031           ADDL3   STACK, R0, R2        ;
                62                  53  D0 00036           MOVL    R3, (R2)             ;
                50          08      AC  C0 00039           ADDL2   STACK, R0            ; 3424
                    04      A0          54  90 0003D       MOVB    VALFLAG, 4(R0)       ;
                                    61  D7 00041           DECL    (R1)                 ; 3425
                                    16  11 00043           BRB     6$                   ; 3420
                                    61  D6 00045 5$:       INCL    (R1)                 ; 3433
     50         61                  05  C5 00047           MULL3   #5, (R1), R0         ; 3434
     52         50          08      AC  C1 0004B           ADDL3   STACK, R0, R2        ;
                62                  53  D0 00050           MOVL    R3, (R2)             ;
                50          08      AC  C0 00053           ADDL2   STACK, R0            ; 3435
                    04      A0          54  90 00057       MOVB    VALFLAG, 4(R0)       ;
                                    50  D4 0005B 6$:       CLRL    R0                   ; 3440
                                        04 0005D           RET                          ; 3442

; Routine Size:  94 bytes,    Routine Base:  DBG$CODE + 154F

;  3347          3443  1
;  3348          3444  0 END ELUDOM
```

```
                                .EXTRN  LIB$SIGNAL

;                       PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  DBG$GLOBAL                     12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                       176  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$PLIT                     3546  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$CODE                     5549  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        33         0        1000        00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1
;                                      32         2         6           7        00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1
;                                    1545       195        12          97        00:02.0
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                     418        17         4          31        00:00.4
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1
;                                     386         2         0          22        00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGGEN.L32;1
;                                     150         4         2          12        00:00.3


;                               COMMAND QUALIFIERS

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

; Size:         5549 code + 3734 data bytes
; Run Time:       02:00.3
; Elapsed Time:   02:26.6
; Lines/CPU Min:   1717
; Lexemes/CPU-Min: 18293
; Memory Used:  856 pages
; Compilation Complete
```

DBGSCREEN
LIS

DBGPRINT
LIS