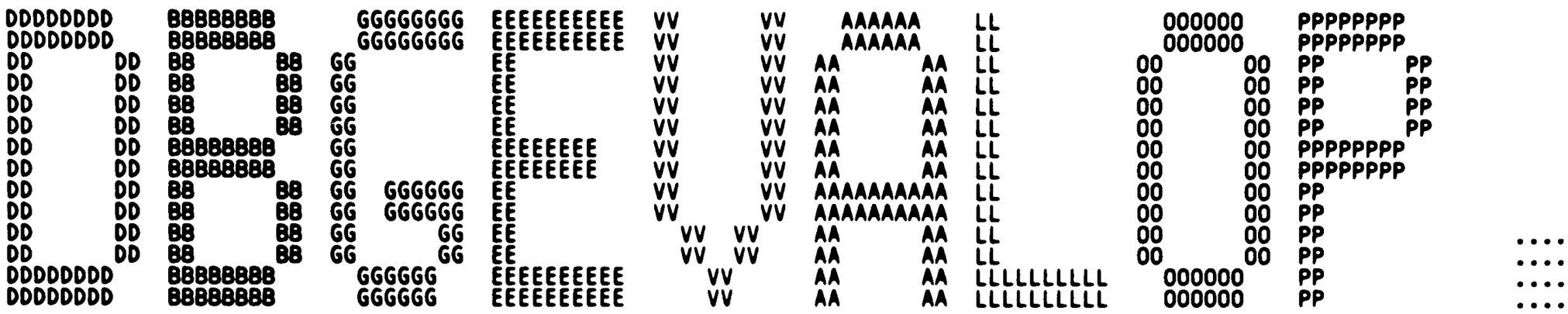
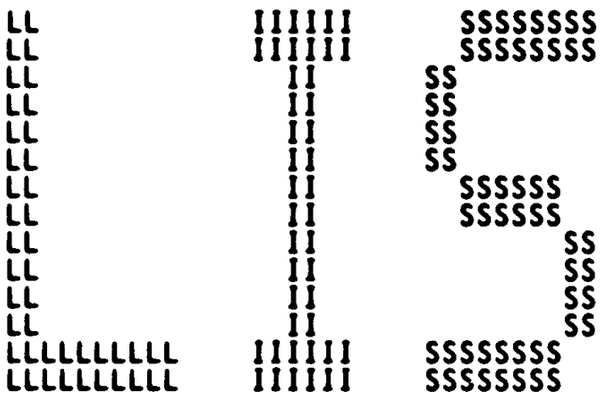
```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB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


DDDDDDDD BBBBBBBB GGGGGGGG EEEEEEEEEE VV VV AAAAAA LL 000000 PPPPPPPP
DDDDDDDD BBBBBBBB GGGGGGGG EEEEEEEEEE VV VV AAAAAA LL 000000 PPPPPPPP
DD DD BB BB GG EE VV VV AA AA LL 00 00 PP PP
DD DD BB BB GG EE VV VV AA AA LL 00 00 PP PP
DD DD BB BB GG EE VV VV AA AA LL 00 00 PP PP
DD DD BB BB GG EE VV VV AA AA LL 00 00 PP PP
DD DD BBBBBBBB GG EEEEEEEE VV VV AA AA LL 00 00 PPPPPPPP
DD DD BBBBBBBB GG EEEEEEEE VV VV AA AA LL 00 00 PPPPPPPP
DD DD BB BB GG GGGGGG EE VV VV AAAAAAAAAA LL 00 00 PP
DD DD BB BB GG GGGGGG EE VV VV AAAAAAAAAA LL 00 00 PP
DD DD BB BB GG GG EE VV VV AA AA LL 00 00 PP
DD DD BB BB GG GG EE VV VV AA AA LL 00 00 PP
DDDDDDDD BBBBBBBB GGGGGG EEEEEEEEEE VV AA AA LLLLLLLLLL 000000 PP
DDDDDDDD BBBBBBBB GGGGGG EEEEEEEEEE VV AA AA LLLLLLLLLL 000000 PP


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

```
;     1          0001  0 MODULE DBGEVALOP (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 !       R. Title        May, 1982
;    30          0030  1 !       P. Sager        Jun, 1982
;    31          0031  1 !
;    32          0032  1 ! MODULE FUNCTION
;    33          0033  1 !       The top level routine in this module, DBG$EVAL_LANG_OPERATOR,
;    34          0034  1 !       is called by the expression parser to perform an operation.
;    35          0035  1 !       This routine does type conversion, as specified by language
;    36          0036  1 !       specific type conversion tables, if needed. It then calls
;    37          0037  1 !       the routine specified by language specific operator routine
;    38          0038  1 !       tables, in order to actually perform the operation.
;    39          0039  1 !
;    40          0040  1
;    41          0041  1 REQUIRE 'SRC$:DBGPROLOG.REQ';
;    42          0175  1
;    43          0176  1 LINKAGE
;    44          0177  1     PLI_LINK = CALL(REGISTER=0, REGISTER=1, REGISTER=2, REGISTER=3);
;    45          0178  1
;    46          0179  1 FORWARD ROUTINE
;    47          0180  1     AAA_DUMMY,                          ! Dummy routine to force dump of tables
;    48          0181  1     DBG$BLISS_BITSELECT: NOVALUE,       ! Perform BLISS bit selection X<p,s,e>
;    49          0182  1     DBG$BLISS_INDIRECTION,              ! Performs BLISS indirection
;    50          0183  1     DBG$CONV_TEXT_VALUE,                ! Convert text string to value
;    51          0184  1     DBG$CONV_TQUADWORD_VALUE,           ! Convert text string to QUADWORD
;    52          0185  1     DBG$CONV_TOCTAWORD_VALUE,           ! Convert text string to OCTAWORD
;    53          0186  1     DBG$CONV_TRFA_VALUE,                ! Convert text string to RFA value
;    54          0187  1     DBG$DO_MAPPING: NOVALUE,            ! Perform type mapping
;    55          0188  1     DBG$EVALOP_SET_LANGUAGE: NOVALUE,   ! Set up pointer to Operator Information
;    56          0189  1                                         !    Tables
;    57          0190  1     DBG$EVAL_LANG_OPERATOR,             ! Evaluate a language expr. operator
```

```
;    58          0191  1      DBG$GET_SET_TYPEID,                 ! Set the parent TYPEID for SET data type
;    59          0192  1      DBG$GET_DTYPE,                      ! Obtain dtype from Value Descriptor
;    60          0193  1      DBG$LANGUAGE_TYPE_CONV,             ! Language-specific type converter
;    61          0194  1      DBG$MAP_DTYPE_CLASS,                ! Given DTYPE, do a best guess at the
;    62          0195  1                                          !    CLASS
;    63          0196  1      DBG$NUM_BYTES,                      ! Returns number of bytes to hold
;    64          0197  1                                          !    a given DTYPE.
;    65          0198  1      DBG$PERFORM_TYPEID_CHECK,           ! Perform typeid check on non-atomic
;    66          0199  1                                          !    data types
;    67          0200  1      DBG$TYPE_CONV,                      ! Top level DEBUG type converter
;    68          0201  1      CONV_TEXT_PACK_VALUE,               ! A routine convert the unconverted
;    69          0202  1                                          !    string packed decimal
;    70          0203  1      GET_DATA_LENGTH,                    ! Get the length of the data
;    71          0204  1      GET_SCALE,                          ! Get scaling factor for Floating Point
;    72          0205  1                                          !    data to packed decimal conversion
;    73          0206  1      FIND_JOIN,                          ! Find join in directed acyclic graph
;    74          0207  1      FIND_PATH,                          ! Find path in directed acyclic graph
;    75          0208  1      FIND_PATH_DEPOSIT,                  ! Similar to FIND_PATH, used for DEPOSIT
;    76          0209  1      FIXUP_EMPTY_SET,                    ! Fix up typeid for empty set
;    77          0210  1      INTMED_DATA_FOR_DEP,                ! Create intermediate data for deposit
;    78          0211  1      MAKE_VAL_DESC,                      ! Build a dummy value descriptor
;    79          0212  1      MAP_NRO_DTYPE_IN_RPG,               ! Map normal and alternate least
;    80          0213  1                                          !    significant digit and sign into
;    81          0214  1                                          !    RPG standard output format
;    82          0215  1      MAP_PACKED:  NOVALUE,               ! Maps packed decimal to correct type.
;    83          0216  1      MAP_PLI_TYPE_SIZE:  NOVALUE,        ! Maps dtype to PL/I specific type; calculates size.
;    84          0217  1      MODIFY_PLI_TARGET_TYPE: NOVALUE,    ! Modify PLI target type
;    85          0218  1      PLI_TYPE_CONV,                      ! PLI Type Conv. used for PLI bit-string
;    86          0219  1                                          !    conversion
;    87          0220  1      PLI_HANDLER,                        ! Catches PL/I conversion errors.
;    88          0221  1      TYPEID_CHECK_ENUM,                  ! Perform typeid check on a pair of enum.
;    89          0222  1      TYPEID_CHECK_SET,                   ! Perform typeid check on a pair of sets
;    90          0223  1      TYPEID_RANGE_CHECK_ENUM,            ! Perform range check for enum data type
;    91          0224  1      TYPEID_RANGE_CHECK_SUBRNG;          ! Perform range check for subrange data type
;    92          0225  1
;    93          0226  1  EXTERNAL ROUTINE
;    94          0227  1      FOR$CVT_D_TE,                       ! Fortran E format routine
;    95          0228  1      FOR$CVT_G_TE,                       ! Fortran E format routine
;    96          0229  1      FOR$CVT_H_TE,                       ! Fortran E format routine
;    97          0230  1      MTH$JNOT,                           ! Bitwise complement of a Longword
;    98          0231  1      OTS$CVT_TB_L,                       ! Convert text Binary to Longword
;    99          0232  1      OTS$CVT_TI_L,                       ! Convert text (signed) to Longword
;   100          0233  1      OTS$CVT_T_F,                        ! Convert text to single Floating
;   101          0234  1      OTS$CVT_T_D,                        ! Convert text to Double Float
;   102          0235  1      OTS$CVT_T_G,                        ! Convert text to G_Float
;   103          0236  1      OTS$CVT_T_H,                        ! Convert text to H_Float
;   104          0237  1      OTS$CVT_TO_L,                       ! Convert text (octal) to Longword
;   105          0238  1      OTS$CVT_TZ_L,                       ! Convert text (hexadecimal) to Longword
;   106          0239  1      PLI$CHARABIT_R6:  PLI_LINK,         ! Convert text to aligned bit-string.
;   107          0240  1      PLI$CVRT_ANY,                       ! Used in PL/I bit-strings conversions.
;   108          0241  1      DBG$COLLECT: NOVALUE,               ! "Sanitize" Primary Descriptors
;   109          0242  1      DBG$COVER_DX_DX,                    ! Debug type converter.
;   110          0243  1      DBG$CVT_DX_DX:  NOVALUE,            ! Debug version of LIB$CVT_DX_DX
;   111          0244  1      DBG$CVT_TOQUADWORD_TO_VALUE,        ! Convert into QUADWORD value
;   112          0245  1      DBG$CVT_TOUQUADWORD_TO_VALUE,       ! Convert into unsigned QUADWORD value
;   113          0246  1      DBG$CVT_TOOCTAWORD_TO_VALUE,        ! Convert into OCTAWORD value
;   114          0247  1      DBG$CVT_TRFA_TO_VALUE,              ! Convert into RFA value
```

```
;  115     0248  1      DBG$GET_DST_NAME,                ! Obtain name of symbol
;  116     0249  1      DBG$GET_TEMPMEM,                 ! Allocate temporary memory
;  117     0250  1      DBG$MAKE_SKELETON_DESC,          ! Create skeleton Value Descriptor
;  118     0251  1      DBG$MAKE_VAL_DESC,               ! Convert VMS desc to valdesc
;  119     0252  1      DBG$MAKE_VMS_DESC,               ! Convert Primary to VMS desc
;  120     0253  1      DBG$NEWLINE: NOVALUE,            ! Close print line and start new line
;  121     0254  1      DBG$PERFORM_OPERATOR: NOVALUE,   ! Perform arithmetic operation
;  122     0255  1      DBG$PRIM_TO_ADDR,                ! Convert primary descriptor to value
;  123     0256  1                                       !    descriptor containing the address
;  124     0257  1                                       !    of the primary.
;  125     0258  1      DBG$PRIM_TO_VAL,                 ! Converts a primary descriptor
;  126     0259  1                                       !    to a value descriptor
;  127     0260  1      DBG$PRINT: NOVALUE,              ! Print FAO-formatted text
;  128     0261  1      DBG$STA_SYMVALUE: NOVALUE,       ! Get symbol's value
;  129     0262  1      DBG$STA_TYP_ATOMIC: NOVALUE,     ! Get symbol information
;  130     0263  1      DBG$STA_TYP_ENUM: NOVALUE,       ! Get symbol information
;  131     0264  1      DBG$STA_TYP_PICT: NOVALUE,       ! Get symbol information
;  132     0265  1      DBG$STA_TYP_SET: NOVALUE,        ! Get symbol information
;  133     0266  1      DBG$STA_TYP_SUBRNG: NOVALUE,     ! Get symbol information
;  134     0267  1      DBG$STA_TYP_TYPEDPTR: NOVALUE,   ! Get symbol information
;  135     0268  1      DBG$TYPEID_FOR_SET;              ! Construct a Set Constant typeid
;  136     0269  1
;  137     0270  1  EXTERNAL
;  138     0271  1      DBG$GB_LANGUAGE: BYTE,           ! Current lang setting
;  139     0272  1      DBG$GL_NEG_CONST_TOKEN,          ! Negative constant token
;  140     0273  1      DBG$GL_POS_CONST_TOKEN,          ! Positive constant token
;  141     0274  1      DBG$GL_NEG_SIGN_TOKEN,           ! Negative constant token (--> unary minus)
;  142     0275  1      DBG$GL_POS_SIGN_TOKEN,           ! Positive constant token (--> unary plus)
;  143     0276  1      DBG$GL_DEVELOPER: BITVECTOR[];   ! Set developer flag
;  144     0277  1
;  145     0278  1  BUILTIN
;  146     0279  1      ASHP,
;  147     0280  1      EDITPC;
;  148     0281  1
;  149     0282  1  LITERAL
;  150     0283  1      DBG$K_DTYPE_ARRAY  = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_ARRAY,
;  151     0284  1      DBG$K_DTYPE_ENUM   = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_ENUM,
;  152     0285  1      DBG$K_DTYPE_TPTR   = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_TPTR,
;  153     0286  1      DBG$K_DTYPE_SET    = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_SET,
;  154     0287  1      DBG$K_DTYPE_SUBRNG = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_SUBRNG,
;  155     0288  1      DBG$K_DTYPE_PTR    = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_PTR,
;  156     0289  1      DBG$K_DTYPE_PICT   = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_PICT,
;  157     0290  1      DBG$K_DTYPE_RFA    = DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_RFA;
;  158     0291  1
;  159     0292  1
;  160     0293  1  ! The following is so that the map table can do a mapping from
;  161     0294  1  ! DST$K_BOOL to DSC$K_DTYPE_TF
;  162     0295  1  !
;  163     0296  1  LITERAL
;  164     0297  1      DSC$K_DTYPE_BOOL = DST$K_BOOL;
;  165     0298  1
;  166     0299  1
;  167     0300  1  ! The following is so that the lang. cvt table can do a conversion from
;  168     0301  1  ! DST$K_DTYPE_ANY to a specify dtype or a specify dtype to DST$K_DTYPE_ANY.
;  169     0302  1  !
;  170     0303  1  LITERAL
;  171     0304  1      DSC$K_DTYPE_ANY = %X'FF';
```

```
;  172          0305  1
;  173          0306  1
;  174          0307  1 BIND
;  175          0308  1     TABLEBASE = UPLIT BYTE (%ASCII 'BASE');
;  176          0309  1
;  177          0310  1 ! OPCODE_NAME is global so that the type converter can get at it for
;  178          0311  1 ! purposes of signalling error messages.
;  179          0312  1 !
;  180          0313  1 GLOBAL
;  181          0314  1     DBG$GL_OPCODE_NAME: REF VECTOR [,BYTE];     ! Operator name in ASCIC
;  182          0315  1
;  183          0316  1 OWN
;  184          0317  1     BLISS_BITSELECTION_FLAG1:INITIAL(0),! Flag saying we've done a bit select
;  185          0318  1                                        !    operator in BLISS
;  186          0319  1     BLISS_BITSELECTION_FLAG2:INITIAL(0),! Flag saying we've done a bit select
;  187          0320  1                                        !    operator in BLISS
;  188          0321  1     BLISS_INDIRECTION_FLAG,            ! Flag saying whether we've done an
;  189          0322  1                                        !    indirection yet
;  190          0323  1     CVT_ROUND_FLAG,                    ! A flag set to TRUE indicate the
;  191          0324  1                                        !    conversion result is rounded
;  192          0325  1     CVT_TBL: REF VECTOR [,LONG],       ! Pointer to a Language Dependent Type
;  193          0326  1                                        !    Conversion Table
;  194          0327  1     CVT_TBL_SIZE,                      ! Number of entries in CVT_TBL
;  195          0328  1     CVTINFO_TABLE: REF CVTINFO$TABLE,  ! Pointer to the Type Conversion
;  196          0329  1                                        !    Table for the current language
;  197          0330  1     MAP_TBL: REF VECTOR [,WORD],       ! Pointer to a Type Mapping Table
;  198          0331  1     MAP_TBL_SIZE,                      ! Number of entries in MAP_TBL
;  199          0332  1     MAX_DEPTH,                         ! Maxinum search depth for graphs
;  200          0333  1     OPINFO_TABLE: REF OPINFO$TABLE,    ! Pointer to the Operator Information
;  201          0334  1                                        !    Table for the current language
;  202          0335  1     SAVE_RESULT_DESC: DBG$STG_DESC;    ! Saved descriptor for commnication
;  203          0336  1                                        !    between <> and . in BLISS
```

```
;  205          0337  1 !       M A C R O S   F O R   O P E R A T O R   R O U T I N E   T A B L E S
;  206          0338  1 !
;  207          0339  1 !
;  208          0340  1 !
;  209          0341  1 !       These macros are used to generate the tables which are used to select
;  210          0342  1 !       operator routines based on the operand data types and which define any
;  211          0343  1 !       type conversion behavior appropriate to the operator.  For example,
;  212          0344  1 !       INT*REAL has to be converted to REAL*REAL in most languages, after
;  213          0345  1 !       which the REAL*REAL multiply routine is invoked; this kind of behavior
;  214          0346  1 !       is specified in the tables whose structure is defined here.
;  215          0347  1 !
;  216          0348  1 !
;  217          0349  1 !
;  218          0350  1 !
```

```
;  220          0351  1 ! OPERATOR ROUTINE TABLE
;  221          0352  1 !
;  222          0353  1 !
;  223          0354  1 ! The Operator Routine Table for a given operator contains one entry for each
;  224          0355  1 ! routine which can be associated with that operator.  Each such entry contains
;  225          0356  1 ! the operand types accepted by that routine (e.g., REAL,REAL for F_Floating
;  226          0357  1 ! multiply) and a routine index which identifies the routine to be invoked.
;  227          0358  1 ! And also there is an optional routine index for typeid check.
;  228          0359  1 ! The types are represented as standard Dtypes and
;  229          0360  1 ! the routine index is simply a CASE index used to select the routine body.
;  230          0361  1 !
;  231          0362  1 ! An Operator Routine Table for an operator is declared as follows:
;  232          0363  1 !
;  233          0364  1 !       OPERATOR_ROUTINE_TABLE(TBLNAME,
;  234          0365  1 !           OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
;  235          0366  1 !                            TYPEID_CHECK_ROUT_NAME),
;  236          0367  1 !
;  237          0368  1 !           OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
;  238          0369  1 !                            TYPEID_CHECK_ROUT_NAME));
;  239          0370  1 !
;  240          0371  1 ! Here TBLNAME is the name of the table.  LEFT_TYPE is the Type Index Value of
;  241          0372  1 ! the left argument (or the only argument if the operator is unary), RIGHT_TYPE
;  242          0373  1 ! is the Type Index Value of the right argument (or 0 if the operator is unary),
;  243          0374  1 ! and ROUT_NAME is the name of the corresponding semantic routine.  ROUT_NAME
;  244          0375  1 ! is automatically prefixed by "ORT$K_" to yield the routine index value.
;  245          0376  1 ! TYPEID_CHECK_ROUT_NAME is the name of the corresponding typeid check semantic
;  246          0377  1 ! routine.  TYPEID_CHECK_ROUT_NAME is automatically prefixed by "ORT$K_TYPEID"
;  247          0378  1 ! to yield the routine index value.
;  248          0379  1 !
;  249          0380  1 ! Define the macros that generate the Operator Routine Table.
;  250          0381  1 !
;  251          0382  1 MACRO
;  252        M 0383  1     OPERATOR_ROUTINE_TABLE(TBLNAME) =
;  253          0384  1         BIND TBLNAME = PLIT BYTE(%REMAINING): VECTOR[,BYTE] %;
;  254          0385  1
;  255          0386  1 MACRO
;  256          0387  1     OPERATOR_ROUTINE(LEFT_TYPE, RIGHT_TYPE, RESULT_TYPE, ROUT_NAME,
;  257        M 0388  1                      TYPEID_CHECK_ROUT_NAME) =
;  258        M 0389  1         %IF %DECLARED (%NAME ('RST$K_TYPE_', RIGHT_TYPE))
;  259        M 0390  1         %THEN
;  260        M 0391  1             DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', RIGHT_TYPE)
;  261        M 0392  1         %ELSE
;  262        M 0393  1             %NAME ('DSC$K_DTYPE_', RIGHT_TYPE)
;  263        M 0394  1         %FI
;  264        M 0395  1         %IF %DECLARED (%NAME ('RST$K_TYPE_', LEFT_TYPE))
;  265        M 0396  1         %THEN
;  266        M 0397  1             DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', LEFT_TYPE)
;  267        M 0398  1         %ELSE
;  268        M 0399  1             %NAME ('DSC$K_DTYPE_', LEFT_TYPE)
;  269        M 0400  1         %FI
;  270        M 0401  1         %NAME ('ORT$K_', ROUT_NAME) AND %X'00FF',
;  271        M 0402  1         %NAME ('ORT$K_', ROUT_NAME)^-8,
;  272        M 0403  1         %IF %LENGTH LEQ 4
;  273        M 0404  1         %THEN
;  274        M 0405  1             0
;  275        M 0406  1         %ELSE
;  276        M 0407  1             %NAME ('ORT$K_TYPEID_', TYPEID_CHECK_ROUT_NAME) AND %X'00FF'
```

```
;  277          M 0408  1          %FI,
;  278          M 0409  1          %IF %LENGTH LEQ 4
;  279          M 0410  1          %THEN
;  280          M 0411  1              0
;  281          M 0412  1          %ELSE
;  282          M 0413  1              %NAME ('ORT$K_TYPEID_', TYPEID_CHECK_ROUT_NAME)^-8
;  283          M 0414  1          %FI,
;  284          M 0415  1          %IF %DECLARED (%NAME ('ORT$K_RESULT_', RESULT_TYPE))
;  285          M 0416  1          %THEN
;  286          M 0417  1              %NAME ('ORT$K_RESULT_', RESULT_TYPE)
;  287          M 0418  1          %ELSE
;  288          M 0419  1              (%IF %DECLARED (%NAME ('RST$K_TYPE_', RESULT_TYPE))
;  289          M 0420  1               %THEN
;  290          M 0421  1                   DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', RESULT_TYPE)
;  291          M 0422  1               %ELSE
;  292          M 0423  1                   %NAME ('DSC$K_DTYPE_', RESULT_TYPE)
;  293          M 0424  1               %FI)
;  294          M 0425  1          %FI,
;  295            0426  1          0%;
;  296            0427  1
;  297            0428  1
;  298            0429  1  ! The TYPE_GRAPH_EDGE macro is used by the tables below.
;  299            0430  1  !
;  300            0431  1  MACRO
;  301          M 0432  1      TYPE_GRAPH_EDGE (LOWER_TYPE, HIGHER_TYPE) =
;  302          M 0433  1          (%IF %DECLARED (%NAME ('RST$K_TYPE_', HIGHER_TYPE))
;  303          M 0434  1          %THEN
;  304          M 0435  1              DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', HIGHER_TYPE)
;  305          M 0436  1          %ELSE
;  306          M 0437  1              %NAME ('DSC$K_DTYPE_', HIGHER_TYPE)
;  307          M 0438  1          %FI)^8
;  308          M 0439  1          OR
;  309          M 0440  1          (%IF %DECLARED (%NAME ('RST$K_TYPE_', LOWER_TYPE))
;  310          M 0441  1          %THEN
;  311          M 0442  1              DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', LOWER_TYPE)
;  312          M 0443  1          %ELSE
;  313          M 0444  1              %NAME ('DSC$K_DTYPE_', LOWER_TYPE)
;  314            0445  1          %FI) %;
```

```
;  316          0446  1 ! TYPE HIERARCHY TABLE
;  317          0447  1 !
;  318          0448  1 ! The Type Hierarchy Table specifies the implicit type conversions to
;  319          0449  1 ! be done within an expression. A type hierarchy table may be specific
;  320          0450  1 ! to a given operator in a given language, or may be shared across
;  321          0451  1 ! operators and languages.
;  322          0452  1 !
;  323          0453  1 ! The Type Hierarchy Table is a directed acyclic graph. For example,
;  324          0454  1 ! for the addition operator in Fortran, a part of the Type Hierarchy
;  325          0455  1 ! Table might look like this:
;  326          0456  1 !
;  327          0457  1 !                              ,F_COMPLEX  -> D_COMPLEX
;  328          0458  1 !                             /                      /
;  329          0459  1 !                            /                      /
;  330          0460  1 !       LONG -> F_FLOAT     /                      /
;  331          0461  1 !                           \                      /
;  332          0462  1 !                            \                    /
;  333          0463  1 !                             v                  /
;  334          0464  1 !                              D_FLOAT -> H_FLOAT
;  335          0465  1 !
;  336          0466  1 ! A Type Hierarchy Table is specified by giving its edges.
;  337          0467  1 !
;  338          0468  1 ! An example of a Type Hierarchy Table definition is:
;  339          0469  1 !
;  340          0470  1 ! TYPE_HIERARCHY_TABLE ( FORADD_TYPE_HIER_TABLE,
;  341          0471  1 !       TYPE_GRAPH_EDGE (L, F),
;  342          0472  1 !       TYPE_GRAPH_EDGE (F, FC),
;  343          0473  1 !       TYPE_GRAPH_EDGE (F, D),
;  344          0474  1 !       ...,0 );
;  345          0475  1 !
;  346          0476  1 ! Define the macro which declares a Type Hierarchy Table.
;  347          0477  1 !
;  348          0478  1 MACRO
;  349        M 0479  1     TYPE_HIERARCHY_TABLE (TBLNAME) =
;  350          0480  1         BIND TBLNAME = PLIT WORD(%REMAINING) : VECTOR [,WORD] %;
;  351          0481  1
```

```
;  353          0482  1 ! TYPE INCOMPATIBILITY TABLE
;  354          0483  1 !
;  355          0484  1 ! The Type Incompatibility Table specifies which pairs of types are
;  356          0485  1 ! incompatible (i.e., it is illegal for both to be in the same
;  357          0486  1 ! operator expression). For example, D_FLOAT and G_FLOAT are
;  358          0487  1 ! incompatible in FORTRAN. A Type Incompatibility Table may be specific
;  359          0488  1 ! to a given operator in a given language, or may be shared across
;  360          0489  1 ! operators and languages.
;  361          0490  1 !
;  362          0491  1 ! A Type Incompatibility Table is specified by giving a list of
;  363          0492  1 ! incompatible type pairs.
;  364          0493  1 !
;  365          0494  1 ! An example of a Type Incompatibility Table definition is:
;  366          0495  1 !
;  367          0496  1 ! TYPE_INCOMP_TABLE ( FORADD_TYPE_INCOMP_TABLE,
;  368          0497  1 !       TYPE_GRAPH_EDGE (D, G),0);
;  369          0498  1 !
;  370          0499  1
;  371          0500  1 ! Define the macro which declares a Type Hierarchy Table.
;  372          0501  1 !
;  373          0502  1 MACRO
;  374          0503  1     TYPE_INCOMP_TABLE (TBLNAME)
;  375          0504  1         = BIND TBLNAME = PLIT WORD(%REMAINING) : VECTOR [,WORD] %;
;  376          0505  1
```

```
;  378          0506  1 !
;  379          0507  1 ! OPERATOR INFORMATION TABLE
;  380          0508  1 !
;  381          0509  1 !
;  382          0510  1 ! The Operator Information Table for a language is a blockvector indexed by
;  383          0511  1 ! operator code.  For each operator, it gives the address of the corresponding
;  384          0512  1 ! Operator Routine Table, Type Hierarchy Table, Type Incompatibility Table,
;  385          0513  1 ! and a flag.  All addresses are relative to TABLEBASE.  An Operator
;  386          0514  1 ! Information Table is declared as follows:
;  387          0515  1 !
;  388          0516  1 !       OPERATOR_INFO_TABLE(TBLNAME,
;  389          0517  1 !               OPERATOR_INFO_ENTRY(OPCODE, ROUTTBL, HIERTBL, INCOMPTBL,
;  390          0518  1 !                       FETCH_FLAG),
;  391          0519  1 !                       .
;  392          0520  1 !               OPERATOR_INFO_ENTRY(OPCODE, ROUTTBL, HIERTBL, INCOMPTBL,
;  393          0521  1 !                       FETCH_FLAG));
;  394          0522  1 !
;  395          0523  1 ! Here TBLNAME is the name of the whole Operator Information Table.  For each
;  396          0524  1 ! operator accepted by the language, OPCODE is the operator code name for the
;  397          0525  1 ! operator (this is automatically prefixed by "TOKEN$K_" by the macro).
;  398          0526  1 !
;  399          0527  1 ! Define the macros which declare Operator Information Tables.
;  400          0528  1 !
;  401          0529  1 MACRO
;  402        M 0530  1     OPERATOR_INFO_TABLE(TBLNAME) =
;  403          0531  1         OWN TBLNAME : OPINFO$TABLE PSECT(DBG$PLIT) PRESET(%REMAINING) %;
;  404          0532  1
;  405          0533  1 MACRO
;  406        M 0534  1     OPERATOR_INFO_ENTRY (OPERATOR, ROUTTBL, HIERTBL, INCOMPTBL, FALSE) =
;  407        M 0535  1         [ %NAME ('TOKEN$K_', OPERATOR), OPINFO$L_ROUTTBL] =
;  408        M 0536  1             ROUTTBL - TABLEBASE,
;  409        M 0537  1         [ %NAME ('TOKEN$K_', OPERATOR), OPINFO$L_HIERTBL] =
;  410        M 0538  1             HIERTBL - TABLEBASE,
;  411        M 0539  1         [ %NAME ('TOKEN$K_', OPERATOR), OPINFO$L_INCOMPTBL] =
;  412        M 0540  1             INCOMPTBL - TABLEBASE,
;  413        M 0541  1         %IF %LENGTH LEQ 4
;  414        M 0542  1         %THEN
;  415        M 0543  1             [ %NAME ('TOKEN$K_', OPERATOR), OPINFO$V_FETCH] = TRUE
;  416        M 0544  1         %ELSE
;  417        M 0545  1             [ %NAME ('TOKEN$K_', OPERATOR), OPINFO$V_FETCH] = FALSE
;  418          0546  1         %FI %;
```

```
;  420          0547  1 ! TYPE MAPPING TABLE
;  421          0548  1 !
;  422          0549  1 ! The Type Mapping Table specifies the mapping of data types to be done
;  423          0550  1 ! prior to any expression evaluation. For example, in FORTRAN data type
;  424          0551  1 ! WU is treated the same as type W in expressions, so the mapping from
;  425          0552  1 ! WU to W is done here.
;  426          0553  1 !
;  427          0554  1 ! A Type Mapping Table is represented as a sequence of pairs of data types.
;  428          0555  1 !
;  429          0556  1 ! An example of a Type Mapping Table definition is:
;  430          0557  1 !
;  431          0558  1 ! TYPE_MAPPING_TABLE ( FORTRAN_TYPE_MAPPING_TABLE,
;  432          0559  1 !       TYPE_GRAPH_EDGE (BU, B),
;  433          0560  1 !       TYPE_GRAPH_EDGE (WU, W),
;  434          0561  1 !       TYPE_GRAPH_EDGE (LU, L),
;  435          0562  1 !       0);
;  436          0563  1 !
;  437          0564  1
;  438          0565  1 ! Define the macro which declares a Type Mapping Table.
;  439          0566  1 !
;  440          0567  1 MACRO
;  441          0568  1     TYPE_MAPPING_TABLE (TBLNAME)
;  442          0569  1         = BIND TBLNAME = PLIT WORD (%REMAINING) : VECTOR [,WORD] %;
;  443          0570  1
```

```
;  445          0571  1 !
;  446          0572  1 ! TYPE CONVERSION INFORMATION TABLE
;  447          0573  1 !
;  448          0574  1 !
;  449          0575  1 ! The Type Conversion Table for a language is a vector of longwords.  For
;  450          0576  1 ! each longword, it gives the address of the corresponding Type Mapping Table,
;  451          0577  1 ! Type Conversion Table needed for the language.  All addresses are relative
;  452          0578  1 ! to TABLEBASE.  An Operator Information Table is declared as follows:
;  453          0579  1 !
;  454          0580  1 !       CONVERSION_INFO_TABLE(TBLNAME,
;  455          0581  1 !               CONVERSION_INFO_ENTRY(MAPTBL, CONVTBL, CVT_ROUNDING_FLAG));
;  456          0582  1 !
;  457          0583  1 ! Here TBLNAME is the name of the whole Type Conversion Information Table.
;  458          0584  1 !
;  459          0585  1 ! Define the macros which declare Type Conversion Information Tables.
;  460          0586  1 !
;  461          0587  1 MACRO
;  462        M 0588  1     CONVERSION_INFO_TABLE(TBLNAME) =
;  463          0589  1         OWN TB[NAME : CVTINFO$TABLE PSECT(DBG$PLIT) PRESET(%REMAINING) %;
;  464          0590  1
;  465          0591  1 MACRO
;  466        M 0592  1     CONVERSION_INFO_ENTRY (MAPTBL, CVTTBL, TRUE) =
;  467        M 0593  1         [CVTINFO$L_MAPTBL] = MAPTBL - TABLEBASE,
;  468        M 0594  1         [CVTINFO$L_CVTTBL] = CVTTBL - TABLEBASE,
;  469        M 0595  1         %IF %LENGTH LEQ 2
;  470        M 0596  1         %THEN
;  471        M 0597  1             [CVTINFO$V_ROUND] = FALSE
;  472        M 0598  1         %ELSE
;  473        M 0599  1             [CVTINFO$V_ROUND] = TRUE
;  474          0600  1         %FI %;
```

```
;  476          0601  1 ! LANGUAGE SPECIFIC TYPE CONVERSION TABLE
;  477          0602  1 !
;  478          0603  1 ! The Language Specific Type Conversion Table specifies which type conversions
;  479          0604  1 ! are to be done in a language specific manner.  For example, although PL/I
;  480          0605  1 ! bit-strings have a dtype of V or VU, they are handled very differently from
;  481          0606  1 ! most other data having those data types (as in BLISS or Pascal).
;  482          0607  1 !
;  483          0608  1 ! A language specific conversion table is specified by giving a case index,
;  484          0609  1 ! and a higher and lower type.
;  485          0610  1 !
;  486          0611  1 ! An example of a language specific conversion table entry is:
;  487          0612  1 !       LANG_CVT_TABLE (PLI_CVT_TABLE,
;  488          0613  1 !           LANG_CVT_INDEX (CVT$PLI_TF_L, L, TF),
;  489          0614  1 !           0);
;  490          0615  1 !
;  491          0616  1
;  492          0617  1
;  493          0618  1 ! Define the macro which declares the entries for Language Specific Type
;  494          0619  1 ! Conversion Table.
;  495          0620  1 !
;  496          0621  1 MACRO
;  497        M 0622  1     LANG_CVT_ENTRY (INDEX, LOWER_TYPE, HIGHER_TYPE) =
;  498        M 0623  1         ((((%NAME ('CVT$K_', INDEX))^8)
;  499        M 0624  1         OR
;  500        M 0625  1         (%IF %DECLARED (%NAME ('RST$K_TYPE_', HIGHER_TYPE))
;  501        M 0626  1         %THEN
;  502        M 0627  1             DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', HIGHER_TYPE)
;  503        M 0628  1         %ELSE
;  504        M 0629  1             %NAME ('DSC$K_DTYPE_', HIGHER_TYPE)
;  505        M 0630  1         %FI))^8)
;  506        M 0631  1         OR
;  507        M 0632  1         (%IF %DECLARED (%NAME ('RST$K_TYPE_', LOWER_TYPE))
;  508        M 0633  1         %THEN
;  509        M 0634  1             DBG$K_MAXIMUM_DTYPE + %NAME ('RST$K_TYPE_', LOWER_TYPE)
;  510        M 0635  1         %ELSE
;  511        M 0636  1             %NAME ('DSC$K_DTYPE_', LOWER_TYPE)
;  512          0637  1         %FI) %;
;  513          0638  1
;  514          0639  1
;  515          0640  1 ! Define the macro which declares a Language Specific Type Conversion Table.
;  516          0641  1 !
;  517          0642  1 MACRO
;  518          0643  1     LANG_CVT_TABLE (TBLNAME)
;  519          0644  1         = BIND TBLNAME = PLIT LONG (%REMAINING):  VECTOR[, LONG] %;
;  520          0645  1
```

```
;  522          0646  1 !       A D A   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  523          0647  1 !
;  524          0648  1 ! This section contains the Operator Routine and Type tables needed to
;  525          0649  1 ! evaluate expressions in the ADA language.
;  526          0650  1 !
;  527          0651  1 !
;  528          0652  1 ! Define the Type Conversion Information Table for ADA.
;  529          0653  1 ! For now, there are no exceptions to the standard DBG$CVT_DX_DX conversion
;  530          0654  1 ! rules so we do not specify any tables here.
;  531          0655  1 !
;  532        P 0656  1 CONVERSION_INFO_TABLE (ADA_CVTINFO_TABLE,
;  533          0657  1     CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
;  534          0658  1
;  535          0659  1
;  536          0660  1 ! Define the Type Hierarchy Table for ADA. This table specifies the implicit
;  537          0661  1 ! type conversions to be done on arithmetic operations.
;  538          0662  1 !
;  539          0663  1 ! For debugging purposes, we have decided to adopting looser rules than
;  540          0664  1 ! the compiler. For example, to add an integer X to a float Y in ADA,
;  541          0665  1 ! you have to say something like "FLOAT(X) + Y". But in the debugger, we
;  542          0666  1 ! will do the implicit conversion and allow just "X + Y".
;  543          0667  1 !
;  544          0668  1 ! We thus allow integer to be converted to either FIXED or FLOAT.
;  545          0669  1 ! We allow the conversion of FIXED to H float (but not to F, D, or G
;  546          0670  1 ! since that could lose precision).
;  547          0671  1 !
;  548          0672  1 !       BU,WU,LU              G
;  549          0673  1 !                \           /   \
;  550          0674  1 !         B,W ----> L -> F -> D -> H
;  551          0675  1 !                    \             /
;  552          0676  1 !                      - FIXED --
;  553          0677  1 !
;  554        P 0678  1 TYPE_HIERARCHY_TABLE (ADA_HIER_TABLE,
;  555        P 0679  1     TYPE_GRAPH_EDGE (B,  L),
;  556        P 0680  1     TYPE_GRAPH_EDGE (BU, L),
;  557        P 0681  1     TYPE_GRAPH_EDGE (W,  L),
;  558        P 0682  1     TYPE_GRAPH_EDGE (WU, L),
;  559        P 0683  1     TYPE_GRAPH_EDGE (LU, L),
;  560        P 0684  1     TYPE_GRAPH_EDGE (L,  F),
;  561        P 0685  1     TYPE_GRAPH_EDGE (L,  FIXED),
;  562        P 0686  1     TYPE_GRAPH_EDGE (F,  D),
;  563        P 0687  1     TYPE_GRAPH_EDGE (F,  G),
;  564        P 0688  1     TYPE_GRAPH_EDGE (FIXED, H),
;  565        P 0689  1     TYPE_GRAPH_EDGE (D,  H),
;  566        P 0690  1     TYPE_GRAPH_EDGE (G,  H),
;  567          0691  1     0);
;  568          0692  1
;  569          0693  1
;  570          0694  1 ! Define the Type Hierarchy Table for ADA deposit.
;  571          0695  1 !
;  572        P 0696  1 TYPE_HIERARCHY_TABLE (ADA_DEPOSIT_TABLE,
;  573        P 0697  1
;  574        P 0698  1     ! Allow any numeric type to any other numeric type.
;  575        P 0699  1     !
;  576        P 0700  1     TYPE_GRAPH_EDGE (B, BU),
;  577        P 0701  1     TYPE_GRAPH_EDGE (BU, W),
;  578        P 0702  1     TYPE_GRAPH_EDGE (W, WU),
```

```
;  579        P 0703  1     TYPE_GRAPH_EDGE (WU, L),
;  580        P 0704  1     TYPE_GRAPH_EDGE (L, LU),
;  581        P 0705  1     TYPE_GRAPH_EDGE (LU, FIXED),
;  582        P 0706  1     TYPE_GRAPH_EDGE (FIXED, F),
;  583        P 0707  1     TYPE_GRAPH_EDGE (F ,D),
;  584        P 0708  1     TYPE_GRAPH_EDGE (D, G),
;  585        P 0709  1     TYPE_GRAPH_EDGE (G, H),
;  586        P 0710  1     TYPE_GRAPH_EDGE (H, B),
;  587        P 0711  1
;  588        P 0712  1     ! Allow number -> SUBRNG. The constraint check should be done
;  589        P 0713  1     ! by the DBG$PERFORM_TYPEID_CHECK routine. Also allow ENUM->SUBRNG
;  590        P 0714  1     ! and character -> SUBRNG.
;  591        P 0715  1     !
;  592        P 0716  1     TYPE_GRAPH_EDGE (ENUM, SUBRNG),
;  593        P 0717  1     TYPE_GRAPH_EDGE (T, SUBRNG),
;  594        P 0718  1     TYPE_GRAPH_EDGE (L, SUBRNG),
;  595        P 0719  1     TYPE_GRAPH_EDGE (FIXED, SUBRNG),
;  596        P 0720  1     TYPE_GRAPH_EDGE (F, SUBRNG),
;  597        P 0721  1     TYPE_GRAPH_EDGE (D, SUBRNG),
;  598        P 0722  1     TYPE_GRAPH_EDGE (G, SUBRNG),
;  599        P 0723  1     TYPE_GRAPH_EDGE (H, SUBRNG),
;  600          0724  1     0);
;  601          0725  1
;  602          0726  1
;  603          0727  1 ! Define the Operator Routine Table for ADA unary plus.
;  604          0728  1 !
;  605        P 0729  1 OPERATOR_ROUTINE_TABLE (ADA_UNARY_PLUS_TABLE,
;  606        P 0730  1
;  607        P 0731  1 ! The following are not language dependent types.  This is needed for DEBUG
;  608        P 0732  1 ! types.  For example, DEP/QUAD L= +1.
;  609        P 0733  1 !
;  610        P 0734  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  611        P 0735  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  612        P 0736  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  613        P 0737  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  614        P 0738  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  615        P 0739  1
;  616        P 0740  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  617        P 0741  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_PLUS_FIXED),
;  618        P 0742  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  619        P 0743  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  620        P 0744  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  621          0745  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H));
;  622          0746  1
;  623          0747  1
;  624          0748  1 ! Define the Operator Routine Table for ADA unary minus.
;  625          0749  1 !
;  626        P 0750  1 OPERATOR_ROUTINE_TABLE (ADA_UNARY_MINUS_TABLE,
;  627        P 0751  1
;  628        P 0752  1 ! The following are not language dependent types.  This is needed for DEBUG
;  629        P 0753  1 ! types.  For example, DEP/QUAD L= -1.
;  630        P 0754  1 !
;  631        P 0755  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  632        P 0756  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  633        P 0757  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  634        P 0758  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  635        P 0759  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
```

```
;  636        P 0760  1
;  637        P 0761  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  638        P 0762  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_MINUS_FIXED),
;  639        P 0763  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  640        P 0764  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  641        P 0765  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  642           0766  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H);
;  643           0767  1
;  644           0768  1
;  645           0769  1 ! Define the Operator Routine Table for ADA absolute value.
;  646           0770  1 !
;  647        P 0771  1 OPERATOR_ROUTINE_TABLE (ADA_ABSOLUTE_TABLE,
;  648        P 0772  1     OPERATOR_ROUTINE (L, L, L, ABS_L),
;  649        P 0773  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ABS_FIXED),
;  650        P 0774  1     OPERATOR_ROUTINE (F, F, F, ABS_F),
;  651        P 0775  1     OPERATOR_ROUTINE (D, D, D, ABS_D),
;  652        P 0776  1     OPERATOR_ROUTINE (G, G, G, ABS_G),
;  653           0777  1     OPERATOR_ROUTINE (H, H, H, ABS_H);
;  654           0778  1
;  655           0779  1
;  656           0780  1 ! Define the Operator Routine Table for ADA addition.
;  657           0781  1 !
;  658        P 0782  1 OPERATOR_ROUTINE_TABLE (ADA_ADD_TABLE,
;  659        P 0783  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;  660        P 0784  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ADD_FIXED_FIXED),
;  661        P 0785  1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  662        P 0786  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;  663        P 0787  1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
;  664           0788  1     OPERATOR_ROUTINE (H, H, H, ADD_H_H);
;  665           0789  1
;  666           0790  1
;  667           0791  1 ! Define the Operator Routine Table for ADA subtraction.
;  668           0792  1 !
;  669        P 0793  1 OPERATOR_ROUTINE_TABLE (ADA_SUBTRACT_TABLE,
;  670        P 0794  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  671        P 0795  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, SUB_FIXED_FIXED),
;  672        P 0796  1     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  673        P 0797  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  674        P 0798  1     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  675           0799  1     OPERATOR_ROUTINE (H, H, H, SUB_H_H);
;  676           0800  1
;  677           0801  1
;  678           0802  1 ! Define the Operator Routine Table for ADA multiplication.
;  679           0803  1 !
;  680        P 0804  1 OPERATOR_ROUTINE_TABLE (ADA_MULTIPLY_TABLE,
;  681        P 0805  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;  682        P 0806  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, MUL_FIXED_FIXED),
;  683        P 0807  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  684        P 0808  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;  685        P 0809  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;  686           0810  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H);
;  687           0811  1
;  688           0812  1
;  689           0813  1 ! Define the Operator Routine Table for ADA division.
;  690           0814  1 !
;  691        P 0815  1 OPERATOR_ROUTINE_TABLE (ADA_DIVIDE_TABLE,
;  692        P 0816  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
```

```
;  693        P 0817  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, DIV_FIXED_FIXED),
;  694        P 0818  1     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
;  695        P 0819  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
;  696        P 0820  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;  697           0821  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H);
;  698           0822  1
;  699           0823  1
;  700           0824  1 ! Define the Operator Routine Table for ADA Modulus.
;  701           0825  1 !
;  702        P 0826  1 OPERATOR_ROUTINE_TABLE (ADA_MODULUS_TABLE,
;  703           0827  1     OPERATOR_ROUTINE (L, L, L, MOD_L_L));
;  704           0828  1
;  705           0829  1
;  706           0830  1 ! Define the Operator Routine Table for ADA Remainder.
;  707           0831  1 !
;  708        P 0832  1 OPERATOR_ROUTINE_TABLE (ADA_REMAINDER_TABLE,
;  709           0833  1     OPERATOR_ROUTINE (L, L, L, REM_L_L));
;  710           0834  1
;  711           0835  1
;  712           0836  1 ! Define the Operator Routine Table for ADA Exponentiation.
;  713           0837  1 !
;  714        P 0838  1 OPERATOR_ROUTINE_TABLE (ADA_POWER_OF_TABLE,
;  715        P 0839  1     OPERATOR_ROUTINE (L, L, L, POWER_L_L),
;  716        P 0840  1     OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;  717        P 0841  1     OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;  718        P 0842  1     OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;  719           0843  1     OPERATOR_ROUTINE (H, H, H, POWER_H_H));
;  720           0844  1
;  721           0845  1
;  722           0846  1 ! Define the Operator Routine Table for ADA Logical Not.
;  723           0847  1 !
;  724        P 0848  1 OPERATOR_ROUTINE_TABLE (ADA_NOT_TABLE,
;  725           0849  1     OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
;  726           0850  1
;  727           0851  1
;  728           0852  1 ! Define the Operator Routine Table for ADA Logical And.
;  729           0853  1 !
;  730        P 0854  1 OPERATOR_ROUTINE_TABLE (ADA_AND_TABLE,
;  731           0855  1     OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
;  732           0856  1
;  733           0857  1
;  734           0858  1 ! Define the Operator Routine Table for ADA Logical Or.
;  735           0859  1 !
;  736        P 0860  1 OPERATOR_ROUTINE_TABLE (ADA_OR_TABLE,
;  737           0861  1     OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
;  738           0862  1
;  739           0863  1
;  740           0864  1 ! Define the Operator Routine Table for ADA Logical Exclusive Or.
;  741           0865  1 !
;  742        P 0866  1 OPERATOR_ROUTINE_TABLE (ADA_XOR_TABLE,
;  743           0867  1     OPERATOR_ROUTINE (TF, TF, TF, XOR_L_L));
;  744           0868  1
;  745           0869  1
;  746           0870  1 ! Define the Operator Routine Table for ADA Equal.
;  747           0871  1 !
;  748        P 0872  1 OPERATOR_ROUTINE_TABLE (ADA_EQUAL_TABLE,
;  749        P 0873  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, EQL_L_L, ENUM_ENUM),
```

```
;  750        P 0874  1     OPERATOR_ROUTINE (L,  L,  TF, EQL_L_L),
;  751        P 0875  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, EQL_FIXED_FIXED),
;  752        P 0876  1     OPERATOR_ROUTINE (F,  F,  TF, EQL_F_F),
;  753        P 0877  1     OPERATOR_ROUTINE (D,  D,  TF, EQL_D_D),
;  754        P 0878  1     OPERATOR_ROUTINE (G,  G,  TF, EQL_G_G),
;  755        P 0879  1     OPERATOR_ROUTINE (H,  H,  TF, EQL_H_H),
;  756          0880  1     OPERATOR_ROUTINE (TF, TF, TF, EQL_TF_TF));
;  757          0881  1
;  758          0882  1
;  759          0883  1 ! Define the Operator Routine Table for ADA Not Equal.
;  760          0884  1 !
;  761        P 0885  1 OPERATOR_ROUTINE_TABLE (ADA_NOT_EQUAL_TABLE,
;  762        P 0886  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, NEQ_L_L, ENUM_ENUM),
;  763        P 0887  1     OPERATOR_ROUTINE (L,  L,  TF, NEQ_L_L),
;  764        P 0888  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, NEQ_FIXED_FIXED),
;  765        P 0889  1     OPERATOR_ROUTINE (F,  F,  TF, NEQ_F_F),
;  766        P 0890  1     OPERATOR_ROUTINE (D,  D,  TF, NEQ_D_D),
;  767        P 0891  1     OPERATOR_ROUTINE (G,  G,  TF, NEQ_G_G),
;  768        P 0892  1     OPERATOR_ROUTINE (H,  H,  TF, NEQ_H_H),
;  769          0893  1     OPERATOR_ROUTINE (TF, TF, TF, NEQ_TF_TF));
;  770          0894  1
;  771          0895  1
;  772          0896  1 ! Define the Operator Routine Table for ADA Less Than.
;  773          0897  1 !
;  774        P 0898  1 OPERATOR_ROUTINE_TABLE (ADA_LSS_THAN_TABLE,
;  775        P 0899  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, LSS_L_L, ENUM_ENUM),
;  776        P 0900  1     OPERATOR_ROUTINE (L,  L,  TF, LSS_L_L),
;  777        P 0901  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, LSS_FIXED_FIXED),
;  778        P 0902  1     OPERATOR_ROUTINE (F,  F,  TF, LSS_F_F),
;  779        P 0903  1     OPERATOR_ROUTINE (D,  D,  TF, LSS_D_D),
;  780        P 0904  1     OPERATOR_ROUTINE (G,  G,  TF, LSS_G_G),
;  781        P 0905  1     OPERATOR_ROUTINE (H,  H,  TF, LSS_H_H),
;  782          0906  1     OPERATOR_ROUTINE (TF, TF, TF, LSS_TF_TF));
;  783          0907  1
;  784          0908  1
;  785          0909  1 ! Define the Operator Routine Table for ADA Greater Than.
;  786          0910  1 !
;  787        P 0911  1 OPERATOR_ROUTINE_TABLE (ADA_GTR_THAN_TABLE,
;  788        P 0912  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, GTR_L_L, ENUM_ENUM),
;  789        P 0913  1     OPERATOR_ROUTINE (L,  L,  TF, GTR_L_L),
;  790        P 0914  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, GTR_FIXED_FIXED),
;  791        P 0915  1     OPERATOR_ROUTINE (F,  F,  TF, GTR_F_F),
;  792        P 0916  1     OPERATOR_ROUTINE (D,  D,  TF, GTR_D_D),
;  793        P 0917  1     OPERATOR_ROUTINE (G,  G,  TF, GTR_G_G),
;  794        P 0918  1     OPERATOR_ROUTINE (H,  H,  TF, GTR_H_H),
;  795          0919  1     OPERATOR_ROUTINE (TF, TF, TF, GTR_TF_TF));
;  796          0920  1
;  797          0921  1
;  798          0922  1 ! Define the Operator Routine Table for ADA Less Than or Equal to.
;  799          0923  1 !
;  800        P 0924  1 OPERATOR_ROUTINE_TABLE (ADA_LSS_EQUAL_TABLE,
;  801        P 0925  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, LEQ_L_L, ENUM_ENUM),
;  802        P 0926  1     OPERATOR_ROUTINE (L,  L,  TF, LEQ_L_L),
;  803        P 0927  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, LEQ_FIXED_FIXED),
;  804        P 0928  1     OPERATOR_ROUTINE (F,  F,  TF, LEQ_F_F),
;  805        P 0929  1     OPERATOR_ROUTINE (D,  D,  TF, LEQ_D_D),
;  806        P 0930  1     OPERATOR_ROUTINE (G,  G,  TF, LEQ_G_G),
```

```
;  807        P 0931  1     OPERATOR_ROUTINE (H,  H,  TF, LEQ_H_H),
;  808          0932  1     OPERATOR_ROUTINE (TF, TF, TF, LEQ_TF_TF));
;  809          0933  1
;  810          0934  1
;  811          0935  1 ! Define the Operator Routine Table for ADA Greater Than or Equal to.
;  812          0936  1 !
;  813        P 0937  1 OPERATOR_ROUTINE_TABLE (ADA_GTR_EQUAL_TABLE,
;  814        P 0938  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, GEQ_L_L, ENUM_ENUM),
;  815        P 0939  1     OPERATOR_ROUTINE (L,  L,  TF, GEQ_L_L),
;  816        P 0940  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, GEQ_FIXED_FIXED),
;  817        P 0941  1     OPERATOR_ROUTINE (F,  F,  TF, GEQ_F_F),
;  818        P 0942  1     OPERATOR_ROUTINE (D,  D,  TF, GEQ_D_D),
;  819        P 0943  1     OPERATOR_ROUTINE (G,  G,  TF, GEQ_G_G),
;  820        P 0944  1     OPERATOR_ROUTINE (H,  H,  TF, GEQ_H_H),
;  821          0945  1     OPERATOR_ROUTINE (TF, TF, TF, GEQ_TF_TF));
;  822          0946  1
;  823          0947  1
;  824          0948  1 ! Define the Operator Routine Table for ADA Concatenate.
;  825          0949  1 !
;  826        P 0950  1 OPERATOR_ROUTINE_TABLE (ADA_CONCATENATE_TABLE,
;  827          0951  1     OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
;  828          0952  1
;  829          0953  1
;  830          0954  1 ! Define the Operator Information Table for ADA.
;  831          0955  1 !
;  832        P 0956  1 OPERATOR_INFO_TABLE (ADA_OPINFO_TABLE,
;  833        P 0957  1
;  834        P 0958  1     ! Unary arithmetic.
;  835        P 0959  1     !
;  836        P 0960  1     OPERATOR_INFO_ENTRY
;  837        P 0961  1         (UNARY_PLUS,     ADA_UNARY_PLUS_TABLE,   ADA_HIER_TABLE, TABLEBASE),
;  838        P 0962  1     OPERATOR_INFO_ENTRY
;  839        P 0963  1         (UNARY_MINUS,    ADA_UNARY_MINUS_TABLE,  ADA_HIER_TABLE, TABLEBASE),
;  840        P 0964  1     OPERATOR_INFO_ENTRY
;  841        P 0965  1         (ABSOLUTE,       ADA_ABSOLUTE_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  842        P 0966  1
;  843        P 0967  1     ! Binary arithmetic.
;  844        P 0968  1     !
;  845        P 0969  1     OPERATOR_INFO_ENTRY
;  846        P 0970  1         (ADD,            ADA_ADD_TABLE,          ADA_HIER_TABLE, TABLEBASE),
;  847        P 0971  1     OPERATOR_INFO_ENTRY
;  848        P 0972  1         (SUBTRACT,       ADA_SUBTRACT_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  849        P 0973  1     OPERATOR_INFO_ENTRY
;  850        P 0974  1         (MULTIPLY,       ADA_MULTIPLY_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  851        P 0975  1     OPERATOR_INFO_ENTRY
;  852        P 0976  1         (DIVIDE,         ADA_DIVIDE_TABLE,       ADA_HIER_TABLE, TABLEBASE),
;  853        P 0977  1     OPERATOR_INFO_ENTRY
;  854        P 0978  1         (MODULUS,        ADA_MODULUS_TABLE,      ADA_HIER_TABLE, TABLEBASE),
;  855        P 0979  1     OPERATOR_INFO_ENTRY
;  856        P 0980  1         (REMAINDER,      ADA_REMAINDER_TABLE,    ADA_HIER_TABLE, TABLEBASE),
;  857        P 0981  1     OPERATOR_INFO_ENTRY
;  858        P 0982  1         (POWER_OF,       ADA_POWER_OF_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  859        P 0983  1
;  860        P 0984  1     ! Logical operations.
;  861        P 0985  1     !
;  862        P 0986  1     OPERATOR_INFO_ENTRY
;  863        P 0987  1         (NOT,            ADA_NOT_TABLE,          ADA_HIER_TABLE, TABLEBASE),
```

```
;  864          P 0988  1     OPERATOR_INFO_ENTRY
;  865          P 0989  1         (AND,           ADA_AND_TABLE,          ADA_HIER_TABLE, TABLEBASE),
;  866          P 0990  1     OPERATOR_INFO_ENTRY
;  867          P 0991  1         (OR,            ADA_OR_TABLE,           ADA_HIER_TABLE, TABLEBASE),
;  868          P 0992  1     OPERATOR_INFO_ENTRY
;  869          P 0993  1         (XOR,           ADA_XOR_TABLE,          ADA_HIER_TABLE, TABLEBASE),
;  870          P 0994  1
;  871          P 0995  1     ! Relationals.
;  872          P 0996  1     !
;  873          P 0997  1     OPERATOR_INFO_ENTRY
;  874          P 0998  1         (EQUAL,         ADA_EQUAL_TABLE,        ADA_HIER_TABLE, TABLEBASE),
;  875          P 0999  1     OPERATOR_INFO_ENTRY
;  876          P 1000  1         (NOT_EQUAL,     ADA_NOT_EQUAL_TABLE,    ADA_HIER_TABLE, TABLEBASE),
;  877          P 1001  1     OPERATOR_INFO_ENTRY
;  878          P 1002  1         (LSS_THAN,      ADA_LSS_THAN_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  879          P 1003  1     OPERATOR_INFO_ENTRY
;  880          P 1004  1         (GTR_THAN,      ADA_GTR_THAN_TABLE,     ADA_HIER_TABLE, TABLEBASE),
;  881          P 1005  1     OPERATOR_INFO_ENTRY
;  882          P 1006  1         (LSS_EQUAL,     ADA_LSS_EQUAL_TABLE,    ADA_HIER_TABLE, TABLEBASE),
;  883          P 1007  1     OPERATOR_INFO_ENTRY
;  884          P 1008  1         (GTR_EQUAL,     ADA_GTR_EQUAL_TABLE,    ADA_HIER_TABLE, TABLEBASE),
;  885          P 1009  1
;  886          P 1010  1     ! String operations.
;  887          P 1011  1     !
;  888          P 1012  1     OPERATOR_INFO_ENTRY
;  889          P 1013  1         (CONCATENATE,   ADA_CONCATENATE_TABLE,  ADA_HIER_TABLE, TABLEBASE),
;  890          P 1014  1
;  891          P 1015  1     ! Convert, Deposit, and Identity.
;  892          P 1016  1     !
;  893          P 1017  1     OPERATOR_INFO_ENTRY
;  894          P 1018  1         (CONVERT,       TABLEBASE,              ADA_DEPOSIT_TABLE, TABLEBASE),
;  895          P 1019  1     OPERATOR_INFO_ENTRY
;  896          P 1020  1         (DEPOSIT,       TABLEBASE,              ADA_DEPOSIT_TABLE, TABLEBASE),
;  897          P 1021  1     OPERATOR_INFO_ENTRY
;  898            1022  1         (IDENTITY,      TABLEBASE,              TABLEBASE, TABLEBASE));
;  899            1023  1
```

```
;  901          1024  1 !
;  902          1025  1 !  B A S I C   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  903          1026  1 !
;  904          1027  1 ! This section contains the Operator Routine and Type tables needed to
;  905          1028  1 ! evaluate expressions in the BASIC language.
;  906          1029  1 !
;  907          1030  1 !
;  908          1031  1 ! The following summarizes the information in the BASIC manual about
;  909          1032  1 ! data types, type conversions, and operators. There is further documentation
;  910          1033  1 ! within the tables below, describing exactly how we translate this into
;  911          1034  1 ! the DEBUG tables.
;  912          1035  1 !
;  913          1036  1 ! BASIC Data Types:
;  914          1037  1 !       . Integer (signed only)
;  915          1038  1 !         byte (8-bit), word (16-bit), long (32-bit)
;  916          1039  1 !       . Real
;  917          1040  1 !         single (f-float), double (d-float), gfloat, hfloat
;  918          1041  1 !       . Packed Decimal
;  919          1042  1 !         0-16 bytes; specifies number of digits and location of decimal point
;  920          1043  1 !       . String
;  921          1044  1 !         one character per byte
;  922          1045  1 !       . RFA
;  923          1046  1 !         6 bytes; specifies record file address - block number and offset
;  924          1047  1 !
;  925          1048  1 ! BASIC Constants:
;  926          1049  1 !       . Any of BASIC's data types
;  927          1050  1 !
;  928          1051  1 ! BASIC Aggregates:
;  929          1052  1 !       . Array
;  930          1053  1 !       . Record
;  931          1054  1 !
;  932          1055  1 ! Expressions:
;  933          1056  1 !       . Numeric
;  934          1057  1 !         Floating-point or integer operands separated by arithmetic operators
;  935          1058  1 !         (+, -, *, /, ^, **) and optionally grouped by parentheses.
;  936          1059  1 !         Result:  numeric (see type conversion).
;  937          1060  1 !       . String
;  938          1061  1 !         Strings separated by "+" (concatenation) or by combinations of
;  939          1062  1 !         string functions.
;  940          1063  1 !         Result:  string.
;  941          1064  1 !       . Relational
;  942          1065  1 !         Operands may be either numeric or string (not mixed)
;  943          1066  1 !         Operators:  =, <, >, <= (or =<), >= (or =>), <> (or ><), ==
;  944          1067  1 !               (note:  "==" is different for string and numeric)
;  945          1068  1 !         Result:  true (-1) or false (0)
;  946          1069  1 !       . Logical
;  947          1070  1 !         Operands:  integer only.
;  948          1071  1 !         Operators:  NOT, AND, OR, XOR, EQV, IMP
;  949          1072  1 !         Result:  true (-1) or false (0)
;  950          1073  1 !       . Assignment, conditional
;  951          1074  1 !
;  952          1075  1 ! Type Conversion:
;  953          1076  1 !       . Arithmetic
;  954          1077  1 !         Note that, with one exception, the resulting data type is the same
;  955          1078  1 !         as that of the operand with the higher data type.  The exception is
;  956          1079  1 !         when the operands are DOUBLE and GFLOAT:  BASIC promotes both values
;  957          1080  1 !         to HFLOAT, and the result is HFLOAT.  This preserves both precision
```

```
;  958          1081  1 !       and magnitude.
;  959          1082  1 !
;  960          1083  1 !                  !  BYTE    WORD    LONG    SINGLE  DOUBLE  GFLOAT  HFLOAT
;  961          1084  1 !       ------------------------------------------------------------------
;  962          1085  1 !           BYTE   !  BYTE    WORD    LONG    SINGLE  DOUBLE  GFLOAT  HFLOAT
;  963          1086  1 !       ------------------------------------------------------------------
;  964          1087  1 !           WORD   !  WORD    WORD    LONG    SINGLE  DOUBLE  GFLOAT  HFLOAT
;  965          1088  1 !       ------------------------------------------------------------------
;  966          1089  1 !           LONG   !  LONG    LONG    LONG    SINGLE  DOUBLE  GFLOAT  HFLOAT
;  967          1090  1 !       ------------------------------------------------------------------
;  968          1091  1 !          SINGLE  !  SINGLE  SINGLE  SINGLE  SINGLE  DOUBLE  GFLOAT  HFLOAT
;  969          1092  1 !       ------------------------------------------------------------------
;  970          1093  1 !          DOUBLE  !  DOUBLE  DOUBLE  DOUBLE  DOUBLE  DOUBLE  HFLOAT  HFLOAT
;  971          1094  1 !       ------------------------------------------------------------------
;  972          1095  1 !          GFLOAT  !  GFLOAT  GFLOAT  GFLOAT  GFLOAT  HFLOAT  GFLOAT  HFLOAT
;  973          1096  1 !       ------------------------------------------------------------------
;  974          1097  1 !          HFLOAT  !  HFLOAT  HFLOAT  HFLOAT  HFLOAT  HFLOAT  HFLOAT  HFLOAT
;  975          1098  1 !
;  976          1099  1 !
;  977          1100  1 !     . Packed Decimal Conversion
;  978          1101  1 !       1.  If both operands are Decimal with the same digit and scale
;  979          1102  1 !           values, no conversion is performed.  If the operands have
;  980          1103  1 !           different digit and scale values, BASIC always used the larger
;  981          1104  1 !           number of specified digits for the result.  The debugger uses
;  982          1105  1 !           a large enough digit and scale factor to avoid overflows if
;  983          1106  1 !           possible.
;  984          1107  1 !       2.  If one operand is Decimal and one is integer, the following
;  985          1108  1 !           integer --> decimal coversions occur (in BASIC):
;  986          1109  1 !               BYTE --> DECIMAL(3,0)
;  987          1110  1 !               WORD --> DECIMAL(5,0)
;  988          1111  1 !               LONG --> DECIMAL(10,0)
;  989          1112  1 !           The debugger converts them all to DECIMAL(31,0).
;  990          1113  1 !       3.  If one operand is Decimal(d,s) and one is floating-point, the
;  991          1114  1 !           following decimal --> floating-point conversions occur:
;  992          1115  1 !               if range of d is <=1  thru <=6  --> SINGLE
;  993          1116  1 !               if range of d is <=7  thru <=15 --> DOUBLE----¦
;  994          1117  1 !                                                    GFLOAT   ¦ <-- depends on floating-point operand
;  995          1118  1 !                                                    HFLOAT__¦
;  996          1119  1 !               if range of d is  =16           --> DOUBLE
;  997          1120  1 !               if range of d is <=17 thru <=31 --> HFLOAT
;  998          1121  1 !           The debugger employs this same scheme.
;  999          1122  1 !
; 1000          1123  1
; 1001          1124  1
; 1002          1125  1 ! Define the Type Conversion Information Table for BASIC.
; 1003          1126  1 ! There is no CVT_TABLE specifying exceptions to the DBG$CVT_DX_DX rules.
; 1004          1127  1 !
; 1005        P 1128  1 CONVERSION_INFO_TABLE (BASIC_CVTINFO_TABLE,
; 1006          1129  1     CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
; 1007          1130  1
; 1008          1131  1
; 1009          1132  1 ! Define the Type Hierarchy Table for BASIC.
; 1010          1133  1 ! This table is described above when we talk about conversion rules.
; 1011          1134  1 ! Leaving out the G edges, the graph specified by this table is:
; 1012          1135  1 !
; 1013          1136  1 !                 B -> W -> L -> F -> D -> H
; 1014          1137  1 !                             \
```

```
;1015          1138  1 !                                P
;1016          1139  1 ! The case of converting packed to float is handled by the routine
;1017          1140  1 ! MAP_PACKED in DBGEVALOP.
;1018          1141  1 !
;1019          1142  1
;1020          1143  1
;1021          1144  1 ! Define a Type Hierarchy Table for BASIC.
;1022          1145  1 !
;1023        P 1146  1 TYPE_HIERARCHY_TABLE (BASIC_HIER1_TABLE,
;1024        P 1147  1     TYPE_GRAPH_EDGE (B, W),
;1025        P 1148  1     TYPE_GRAPH_EDGE (W, L),
;1026        P 1149  1     TYPE_GRAPH_EDGE (L, F),
;1027        P 1150  1     TYPE_GRAPH_EDGE (L, P),
;1028        P 1151  1     TYPE_GRAPH_EDGE (F, D),
;1029        P 1152  1     TYPE_GRAPH_EDGE (F, G),
;1030        P 1153  1     TYPE_GRAPH_EDGE (D, H),
;1031        P 1154  1     TYPE_GRAPH_EDGE (G, H),
;1032          1155  1     0);
;1033          1156  1
;1034          1157  1
;1035          1158  1 ! Define another Type Hierarchy Table for BASIC.
;1036          1159  1 ! This is a subset of the HIER1 table. It is used for those operators that
;1037          1160  1 ! only accept integer types. It would also be OK to use the HIER1 table,
;1038          1161  1 ! but providing a smaller table speeds up the code.
;1039          1162  1 !
;1040        P 1163  1 TYPE_HIERARCHY_TABLE (BASIC_HIER2_TABLE,
;1041        P 1164  1     TYPE_GRAPH_EDGE (B, W),
;1042        P 1165  1     TYPE_GRAPH_EDGE (W, L),
;1043          1166  1     0);
;1044          1167  1
;1045          1168  1 ! Define the Type Hierarchy Table for BASIC deposit.
;1046          1169  1 ! This is a circular table which includes all types except T.
;1047          1170  1 ! This means that any of the numeric types are convertible to any
;1048          1171  1 ! of the other numeric types on a DEPOSIT.
;1049          1172  1 !
;1050        P 1173  1 TYPE_HIERARCHY_TABLE (BASIC_HIERD_TABLE,
;1051        P 1174  1     TYPE_GRAPH_EDGE (B, W),
;1052        P 1175  1     TYPE_GRAPH_EDGE (W, L),
;1053        P 1176  1     TYPE_GRAPH_EDGE (L, P),
;1054        P 1177  1     TYPE_GRAPH_EDGE (P, F),
;1055        P 1178  1     TYPE_GRAPH_EDGE (F, D),
;1056        P 1179  1     TYPE_GRAPH_EDGE (D, G),
;1057        P 1180  1     TYPE_GRAPH_EDGE (G, H),
;1058        P 1181  1     TYPE_GRAPH_EDGE (H, B),
;1059          1182  1     0);
;1060          1183  1
;1061          1184  1
;1062          1185  1 !++
;1063          1186  1 ! Most of the arithmetic routines operate on two arguments of the same type.
;1064          1187  1 ! That type may be B, W, L, F, D, G, H, P, so we provide all of those case
;1065          1188  1 ! indices.
;1066          1189  1 !--
;1067          1190  1
;1068          1191  1 ! Define the Operator Routine Table for BASIC addition.
;1069          1192  1 ! Note that addition of text strings is actually concatenation.
;1070          1193  1 !
;1071        P 1194  1 OPERATOR_ROUTINE_TABLE (BASIC_ADD_TABLE,
```

```
;  1072        P 1195  1     OPERATOR_ROUTINE (B, B, B, ADD_B_B),
;  1073        P 1196  1     OPERATOR_ROUTINE (W, W, W, ADD_W_W),
;  1074        P 1197  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;  1075        P 1198  1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  1076        P 1199  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;  1077        P 1200  1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
;  1078        P 1201  1     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
;  1079        P 1202  1     OPERATOR_ROUTINE (P, P, P, ADD_P_P),
;  1080          1203  1     OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
;  1081          1204  1
;  1082          1205  1
;  1083          1206  1 ! Define the Operator Routine Table for BASIC subtraction.
;  1084          1207  1 !
;  1085        P 1208  1 OPERATOR_ROUTINE_TABLE (BASIC_SUB_TABLE,
;  1086        P 1209  1     OPERATOR_ROUTINE (B, B, B, SUB_B_B),
;  1087        P 1210  1     OPERATOR_ROUTINE (W, W, W, SUB_W_W),
;  1088        P 1211  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  1089        P 1212  1     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  1090        P 1213  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  1091        P 1214  1     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  1092        P 1215  1     OPERATOR_ROUTINE (H, H, H, SUB_H_H),
;  1093          1216  1     OPERATOR_ROUTINE (P, P, P, SUB_P_P));
;  1094          1217  1
;  1095          1218  1
;  1096          1219  1 ! Define the Operator Routine Table for BASIC multiplication.
;  1097          1220  1 !
;  1098        P 1221  1 OPERATOR_ROUTINE_TABLE (BASIC_MUL_TABLE,
;  1099        P 1222  1     OPERATOR_ROUTINE (B, B, B, MUL_B_B),
;  1100        P 1223  1     OPERATOR_ROUTINE (W, W, W, MUL_W_W),
;  1101        P 1224  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;  1102        P 1225  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  1103        P 1226  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;  1104        P 1227  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;  1105        P 1228  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H),
;  1106          1229  1     OPERATOR_ROUTINE (P, P, P, MUL_P_P));
;  1107          1230  1
;  1108          1231  1
;  1109          1232  1 ! Define the Operator Routine Table for BASIC division.
;  1110          1233  1 !
;  1111        P 1234  1 OPERATOR_ROUTINE_TABLE (BASIC_DIV_TABLE,
;  1112        P 1235  1     OPERATOR_ROUTINE (B, B, B, DIV_B_B),
;  1113        P 1236  1     OPERATOR_ROUTINE (W, W, W, DIV_W_W),
;  1114        P 1237  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;  1115        P 1238  1     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
;  1116        P 1239  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
;  1117        P 1240  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;  1118        P 1241  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H),
;  1119          1242  1     OPERATOR_ROUTINE (P, P, P, DIV_P_P));
;  1120          1243  1
;  1121          1244  1
;  1122          1245  1 ! Define the Operator Routine Table for BASIC unary plus.
;  1123          1246  1 !
;  1124        P 1247  1 OPERATOR_ROUTINE_TABLE (BASIC_UNARY_PLUS_TABLE,
;  1125        P 1248  1
;  1126        P 1249  1 ! The following are not language dependent types.  This is needed for DEBUG
;  1127        P 1250  1 ! types.  For example, DEP/QUAD L= +1.
;  1128        P 1251  1 !
```

```
;  1129        P 1252  1      OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  1130        P 1253  1      OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  1131        P 1254  1
;  1132        P 1255  1      OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  1133        P 1256  1      OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  1134        P 1257  1      OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  1135        P 1258  1      OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  1136        P 1259  1      OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  1137        P 1260  1      OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  1138        P 1261  1      OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  1139          1262  1      OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P);
;  1140          1263  1
;  1141          1264  1
;  1142          1265  1  ! Define the Operator Routine Table for BASIC unary minus.
;  1143          1266  1  !
;  1144        P 1267  1  OPERATOR_ROUTINE_TABLE (BASIC_UNARY_MINUS_TABLE,
;  1145        P 1268  1  ! The following are not language dependent types.  This is needed for DEBUG
;  1146        P 1269  1  ! types.  For example, DEP/QUAD L= +1.
;  1147        P 1270  1  !
;  1148        P 1271  1      OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  1149        P 1272  1      OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  1150        P 1273  1
;  1151        P 1274  1      OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  1152        P 1275  1      OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  1153        P 1276  1      OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  1154        P 1277  1      OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  1155        P 1278  1      OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  1156        P 1279  1      OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  1157        P 1280  1      OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  1158          1281  1      OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P);
;  1159          1282  1
;  1160          1283  1
;  1161          1284  1  ! Define the Operator Routine Table for BASIC exponentiation.
;  1162          1285  1  ! Exponentiation has some mixed forms. For example, if you raise a
;  1163          1286  1  ! floating number to an integer power, you do not necessarily
;  1164          1287  1  ! want to first convert the int to float. Instead, a special
;  1165          1288  1  ! routine indices such as  POWER_F_L are provided to do the right thing here.
;  1166          1289  1  !
;  1167        P 1290  1  OPERATOR_ROUTINE_TABLE (BASIC_POWER_TABLE,
;  1168        P 1291  1      OPERATOR_ROUTINE (W, W, W, POWER_W_W),
;  1169        P 1292  1      OPERATOR_ROUTINE (L, L, L, POWER_L_L),
;  1170        P 1293  1      OPERATOR_ROUTINE (F, L, F, POWER_F_L),
;  1171        P 1294  1      OPERATOR_ROUTINE (D, L, D, POWER_D_L),
;  1172        P 1295  1      OPERATOR_ROUTINE (G, L, G, POWER_G_L),
;  1173        P 1296  1      OPERATOR_ROUTINE (H, L, H, POWER_H_L),
;  1174        P 1297  1      OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;  1175        P 1298  1      OPERATOR_ROUTINE (D, F, D, POWER_D_F),
;  1176        P 1299  1      OPERATOR_ROUTINE (F, D, D, POWER_F_D),
;  1177        P 1300  1      OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;  1178        P 1301  1      OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;  1179          1302  1      OPERATOR_ROUTINE (H, H, H, POWER_H_H);
;  1180          1303  1
;  1181          1304  1
;  1182          1305  1  ! Define the Operator Routine Table for BASIC equal.
;  1183          1306  1  ! This operator can be done on strings as well as all the numeric
;  1184          1307  1  ! types.
;  1185          1308  1  !
```

```
;  1186        P 1309  1 OPERATOR_ROUTINE_TABLE (BASIC_EQL_TABLE,
;  1187        P 1310  1     OPERATOR_ROUTINE (RFA, RFA, L, EQL_RFA_RFA),
;  1188        P 1311  1     OPERATOR_ROUTINE (T, T, L, EQL_T_T),
;  1189        P 1312  1     OPERATOR_ROUTINE (B, B, L, EQL_B_B),
;  1190        P 1313  1     OPERATOR_ROUTINE (W, W, L, EQL_W_W),
;  1191        P 1314  1     OPERATOR_ROUTINE (L, L, L, EQL_L_L),
;  1192        P 1315  1     OPERATOR_ROUTINE (F, F, L, EQL_F_F),
;  1193        P 1316  1     OPERATOR_ROUTINE (D, D, L, EQL_D_D),
;  1194        P 1317  1     OPERATOR_ROUTINE (G, G, L, EQL_G_G),
;  1195        P 1318  1     OPERATOR_ROUTINE (H, H, L, EQL_H_H),
;  1196          1319  1     OPERATOR_ROUTINE (P, P, L, EQL_P_P));
;  1197          1320  1
;  1198          1321  1
;  1199          1322  1 ! Define the Operator Routine Table for BASIC not equal.
;  1200          1323  1 ! This operator can be done on strings as well as all the numeric types.
;  1201          1324  1 !
;  1202        P 1325  1 OPERATOR_ROUTINE_TABLE (BASIC_NEQ_TABLE,
;  1203        P 1326  1     OPERATOR_ROUTINE (RFA, RFA, L, NEQ_RFA_RFA),
;  1204        P 1327  1     OPERATOR_ROUTINE (T, T, L, NEQ_T_T),
;  1205        P 1328  1     OPERATOR_ROUTINE (B, B, L, NEQ_B_B),
;  1206        P 1329  1     OPERATOR_ROUTINE (W, W, L, NEQ_W_W),
;  1207        P 1330  1     OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
;  1208        P 1331  1     OPERATOR_ROUTINE (F, F, L, NEQ_F_F),
;  1209        P 1332  1     OPERATOR_ROUTINE (D, D, L, NEQ_D_D),
;  1210        P 1333  1     OPERATOR_ROUTINE (G, G, L, NEQ_G_G),
;  1211        P 1334  1     OPERATOR_ROUTINE (H, H, L, NEQ_H_H),
;  1212          1335  1     OPERATOR_ROUTINE (P, P, L, NEQ_P_P));
;  1213          1336  1
;  1214          1337  1 ! In the tables for the comparison operators,
;  1215          1338  1 ! we allow strings to be compared, and also all the numeric types.
;  1216          1339  1 ! Define the Operator Routine Table for BASIC greater than.
;  1217          1340  1 !
;  1218        P 1341  1 OPERATOR_ROUTINE_TABLE (BASIC_GTR_TABLE,
;  1219        P 1342  1     OPERATOR_ROUTINE (T, T, L, GTR_T_T),
;  1220        P 1343  1     OPERATOR_ROUTINE (B, B, L, GTR_B_B),
;  1221        P 1344  1     OPERATOR_ROUTINE (W, W, L, GTR_W_W),
;  1222        P 1345  1     OPERATOR_ROUTINE (L, L, L, GTR_L_L),
;  1223        P 1346  1     OPERATOR_ROUTINE (F, F, L, GTR_F_F),
;  1224        P 1347  1     OPERATOR_ROUTINE (D, D, L, GTR_D_D),
;  1225        P 1348  1     OPERATOR_ROUTINE (G, G, L, GTR_G_G),
;  1226        P 1349  1     OPERATOR_ROUTINE (H, H, L, GTR_H_H),
;  1227          1350  1     OPERATOR_ROUTINE (P, P, L, GTR_P_P));
;  1228          1351  1
;  1229          1352  1
;  1230          1353  1 ! Define the Operator Routine Table for BASIC greater than or equal to.
;  1231          1354  1 !
;  1232        P 1355  1 OPERATOR_ROUTINE_TABLE (BASIC_GEQ_TABLE,
;  1233        P 1356  1     OPERATOR_ROUTINE (T, T, L, GEQ_T_T),
;  1234        P 1357  1     OPERATOR_ROUTINE (B, B, L, GEQ_B_B),
;  1235        P 1358  1     OPERATOR_ROUTINE (W, W, L, GEQ_W_W),
;  1236        P 1359  1     OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
;  1237        P 1360  1     OPERATOR_ROUTINE (F, F, L, GEQ_F_F),
;  1238        P 1361  1     OPERATOR_ROUTINE (D, D, L, GEQ_D_D),
;  1239        P 1362  1     OPERATOR_ROUTINE (G, G, L, GEQ_G_G),
;  1240        P 1363  1     OPERATOR_ROUTINE (H, H, L, GEQ_H_H),
;  1241          1364  1     OPERATOR_ROUTINE (P, P, L, GEQ_P_P));
;  1242          1365  1
```

```
;  1243          1366  1
;  1244          1367  1 ! Define the Operator Routine Table for BASIC less than.
;  1245          1368  1 !
;  1246        P 1369  1 OPERATOR_ROUTINE_TABLE (BASIC_LSS_TABLE,
;  1247        P 1370  1     OPERATOR_ROUTINE (T, T, L, LSS_T_T),
;  1248        P 1371  1     OPERATOR_ROUTINE (B, B, L, LSS_B_B),
;  1249        P 1372  1     OPERATOR_ROUTINE (W, W, L, LSS_W_W),
;  1250        P 1373  1     OPERATOR_ROUTINE (L, L, L, LSS_L_L),
;  1251        P 1374  1     OPERATOR_ROUTINE (F, F, L, LSS_F_F),
;  1252        P 1375  1     OPERATOR_ROUTINE (D, D, L, LSS_D_D),
;  1253        P 1376  1     OPERATOR_ROUTINE (G, G, L, LSS_G_G),
;  1254        P 1377  1     OPERATOR_ROUTINE (H, H, L, LSS_H_H),
;  1255          1378  1     OPERATOR_ROUTINE (P, P, L, LSS_P_P));
;  1256          1379  1
;  1257          1380  1
;  1258          1381  1 ! Define the Operator Routine Table for BASIC less than or equal to.
;  1259          1382  1 !
;  1260        P 1383  1 OPERATOR_ROUTINE_TABLE (BASIC_LEQ_TABLE,
;  1261        P 1384  1     OPERATOR_ROUTINE (T, T, L, LEQ_T_T),
;  1262        P 1385  1     OPERATOR_ROUTINE (B, B, L, LEQ_B_B),
;  1263        P 1386  1     OPERATOR_ROUTINE (W, W, L, LEQ_W_W),
;  1264        P 1387  1     OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
;  1265        P 1388  1     OPERATOR_ROUTINE (F, F, L, LEQ_F_F),
;  1266        P 1389  1     OPERATOR_ROUTINE (D, D, L, LEQ_D_D),
;  1267        P 1390  1     OPERATOR_ROUTINE (G, G, L, LEQ_G_G),
;  1268        P 1391  1     OPERATOR_ROUTINE (H, H, L, LEQ_H_H),
;  1269          1392  1     OPERATOR_ROUTINE (P, P, L, LEQ_P_P));
;  1270          1393  1
;  1271          1394  1
;  1272          1395  1 ! The logical operators .AND., .OR., .EQV., .NEQV., .NOT. can be applied
;  1273          1396  1 ! only to integer data types.
;  1274          1397  1
;  1275          1398  1 ! Define the Operator Routine Table for BASIC not.
;  1276          1399  1 !
;  1277        P 1400  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_NOT_TABLE,
;  1278        P 1401  1     OPERATOR_ROUTINE (B, B, B, BIT_NOT_B),
;  1279        P 1402  1     OPERATOR_ROUTINE (W, W, W, BIT_NOT_W),
;  1280          1403  1     OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
;  1281          1404  1
;  1282          1405  1
;  1283          1406  1 ! Define the Operator Routine Table for BASIC and.
;  1284          1407  1 !
;  1285        P 1408  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_AND_TABLE,
;  1286        P 1409  1     OPERATOR_ROUTINE (B, B, B, BIT_AND_B_B),
;  1287        P 1410  1     OPERATOR_ROUTINE (W, W, W, BIT_AND_W_W),
;  1288          1411  1     OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
;  1289          1412  1
;  1290          1413  1
;  1291          1414  1 ! Define the Operator Routine Table for BASIC or.
;  1292          1415  1 !
;  1293        P 1416  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_OR_TABLE,
;  1294        P 1417  1     OPERATOR_ROUTINE (B, B, B, BIT_OR_B_B),
;  1295        P 1418  1     OPERATOR_ROUTINE (W, W, W, BIT_OR_W_W),
;  1296          1419  1     OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
;  1297          1420  1
;  1298          1421  1
;  1299          1422  1 ! Define the Operator Routine Table for BASIC xor, neqv
```

```
;1300          1423  1 !
;1301        P 1424  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_XOR_TABLE,
;1302        P 1425  1     OPERATOR_ROUTINE (B, B, B, BIT_XOR_B_B),
;1303        P 1426  1     OPERATOR_ROUTINE (W, W, W, BIT_XOR_W_W),
;1304          1427  1     OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
;1305          1428  1
;1306          1429  1
;1307          1430  1 ! Define the Operator Routine Table for BASIC eqv.
;1308          1431  1 !
;1309        P 1432  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_EQV_TABLE,
;1310        P 1433  1     OPERATOR_ROUTINE (B, B, B, BIT_EQV_B_B),
;1311        P 1434  1     OPERATOR_ROUTINE (W, W, W, BIT_EQV_W_W),
;1312          1435  1     OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
;1313          1436  1
;1314          1437  1
;1315          1438  1 ! Define the Operator Routine Table for BASIC imp.
;1316          1439  1 !
;1317        P 1440  1 OPERATOR_ROUTINE_TABLE (BASIC_BIT_IMP_TABLE,
;1318        P 1441  1     OPERATOR_ROUTINE (B, B, B, BIT_IMP_B_B),
;1319        P 1442  1     OPERATOR_ROUTINE (W, W, W, BIT_IMP_W_W),
;1320          1443  1     OPERATOR_ROUTINE (L, L, L, BIT_IMP_L_L));
;1321          1444  1
;1322          1445  1
;1323          1446  1 ! Define the Operator Information Table for BASIC.
;1324          1447  1 !
;1325        P 1448  1 OPERATOR_INFO_TABLE (BASIC_OPINFO_TABLE,
;1326        P 1449  1
;1327        P 1450  1     ! The following are arithmetic tables that accept all numeric data types,
;1328        P 1451  1     ! including complex. They thus go through the larger HIER1 table, and
;1329        P 1452  1     ! need to specify an incompatibility table.
;1330        P 1453  1     !
;1331        P 1454  1     OPERATOR_INFO_ENTRY (ADD, BASIC_ADD_TABLE, BASIC_HIER1_TABLE,
;1332        P 1455  1         TABLEBASE),
;1333        P 1456  1     OPERATOR_INFO_ENTRY (SUBTRACT, BASIC_SUB_TABLE, BASIC_HIER1_TABLE,
;1334        P 1457  1         TABLEBASE),
;1335        P 1458  1     OPERATOR_INFO_ENTRY (MULTIPLY, BASIC_MUL_TABLE, BASIC_HIER1_TABLE,
;1336        P 1459  1         TABLEBASE),
;1337        P 1460  1     OPERATOR_INFO_ENTRY (DIVIDE, BASIC_DIV_TABLE, BASIC_HIER1_TABLE,
;1338        P 1461  1         TABLEBASE),
;1339        P 1462  1     OPERATOR_INFO_ENTRY (UNARY_PLUS, BASIC_UNARY_PLUS_TABLE,
;1340        P 1463  1         BASIC_HIER1_TABLE, TABLEBASE),
;1341        P 1464  1     OPERATOR_INFO_ENTRY (UNARY_MINUS, BASIC_UNARY_MINUS_TABLE,
;1342        P 1465  1         BASIC_HIER1_TABLE, TABLEBASE),
;1343        P 1466  1     OPERATOR_INFO_ENTRY (POWER_OF, BASIC_POWER_TABLE, BASIC_HIER1_TABLE,
;1344        P 1467  1         TABLEBASE),
;1345        P 1468  1
;1346        P 1469  1     ! The relationals accept all numeric types and thus need the larger
;1347        P 1470  1     ! hierarchy table.  There is no incompatibility table.
;1348        P 1471  1     !
;1349        P 1472  1     OPERATOR_INFO_ENTRY (EQUAL, BASIC_EQL_TABLE, BASIC_HIER1_TABLE,
;1350        P 1473  1         TABLEBASE),
;1351        P 1474  1     OPERATOR_INFO_ENTRY (NOT_EQUAL, BASIC_NEQ_TABLE, BASIC_HIER1_TABLE,
;1352        P 1475  1         TABLEBASE),
;1353        P 1476  1     OPERATOR_INFO_ENTRY (GTR_THAN, BASIC_GTR_TABLE, BASIC_HIER1_TABLE,
;1354        P 1477  1         TABLEBASE),
;1355        P 1478  1     OPERATOR_INFO_ENTRY (GTR_EQUAL, BASIC_GEQ_TABLE, BASIC_HIER1_TABLE,
;1356        P 1479  1         TABLEBASE),
```

```
;  1357          P 1480  1     OPERATOR_INFO_ENTRY (LSS_THAN, BASIC_LSS_TABLE, BASIC_HIER1_TABLE,
;  1358          P 1481  1         TABLEBASE),
;  1359          P 1482  1     OPERATOR_INFO_ENTRY (LSS_EQUAL, BASIC_LEQ_TABLE, BASIC_HIER1_TABLE,
;  1360          P 1483  1         TABLEBASE),
;  1361          P 1484  1
;  1362          P 1485  1     ! The logical operators accept only integer quantities so they can
;  1363          P 1486  1     ! use the smallest hierarchy table. They also do not need an
;  1364          P 1487  1     ! incompatibility table.
;  1365          P 1488  1     !
;  1366          P 1489  1     OPERATOR_INFO_ENTRY (BIT_NOT, BASIC_BIT_NOT_TABLE, BASIC_HIER2_TABLE,
;  1367          P 1490  1         TABLEBASE),
;  1368          P 1491  1     OPERATOR_INFO_ENTRY (BIT_AND, BASIC_BIT_AND_TABLE, BASIC_HIER2_TABLE,
;  1369          P 1492  1         TABLEBASE),
;  1370          P 1493  1     OPERATOR_INFO_ENTRY (BIT_OR, BASIC_BIT_OR_TABLE, BASIC_HIER2_TABLE,
;  1371          P 1494  1         TABLEBASE),
;  1372          P 1495  1     OPERATOR_INFO_ENTRY (BIT_XOR, BASIC_BIT_XOR_TABLE, BASIC_HIER2_TABLE,
;  1373          P 1496  1         TABLEBASE),
;  1374          P 1497  1     OPERATOR_INFO_ENTRY (BIT_EQV, BASIC_BIT_EQV_TABLE, BASIC_HIER2_TABLE,
;  1375          P 1498  1         TABLEBASE),
;  1376          P 1499  1     OPERATOR_INFO_ENTRY (BIT_IMP, BASIC_BIT_IMP_TABLE, BASIC_HIER2_TABLE,
;  1377          P 1500  1         TABLEBASE),
;  1378          P 1501  1
;  1379          P 1502  1     ! The CONVERT operator gets called to convert subscripts to integer
;  1380          P 1503  1     ! type and to convert expressions in FOR, IF, WHILE, REPEAT statements
;  1381          P 1504  1     ! to integer type. It can thus use the smaller HIER2 table to specify
;  1382          P 1505  1     ! the rules for conversion to integer.
;  1383          P 1506  1     !
;  1384          P 1507  1     OPERATOR_INFO_ENTRY (CONVERT, TABLEBASE, BASIC_HIER2_TABLE,
;  1385          P 1508  1         TABLEBASE),
;  1386          P 1509  1
;  1387          P 1510  1     ! The DEPOSIT operator gets called on the DEPOSIT command. It has
;  1388          P 1511  1     ! its own hierarchy table which allows any numeric type to be
;  1389          P 1512  1     ! converted to any other numeric type. The incompatibility table,
;  1390          P 1513  1     ! however, still prevents depositing D types into G types and
;  1391          P 1514  1     ! vice versa.
;  1392          P 1515  1     !
;  1393          P 1516  1     OPERATOR_INFO_ENTRY (DEPOSIT, TABLEBASE, BASIC_HIERD_TABLE,
;  1394          P 1517  1         TABLEBASE),
;  1395          P 1518  1
;  1396          P 1519  1
;  1397          P 1520  1     ! The IDENTITY operator gets called at the end of an EVALUATE command
;  1398          P 1521  1     ! to apply the PRIM_TO_VAL routine and then apply the appropriate
;  1399          P 1522  1     ! type mappings. This will ensure that EV BU will print as a signed integer,
;  1400          P 1523  1     ! for example.
;  1401          P 1524  1     !
;  1402          P 1525  1     ! The identity operator does not require any tables.
;  1403          P 1526  1     !
;  1404          P 1527  1     OPERATOR_INFO_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)
;  1405            1528  1     );
;  1406            1529  1
```

```
;  1408          1530  1 !
;  1409          1531  1 !       B L I S S   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  1410          1532  1 !
;  1411          1533  1 !
;  1412          1534  1 ! This section contains the Operator Routine and Type tables needed to
;  1413          1535  1 ! evaluate expressions in BLISS.
;  1414          1536  1 !
;  1415          1537  1
;  1416          1538  1
;  1417          1539  1 ! Here is the list of types that we may get back from DBG$PRIM_TO_VAL
;  1418          1540  1 ! when we pass it a BLISS primary:
;  1419          1541  1 !
;  1420          1542  1 !       B       - byte signed
;  1421          1543  1 !       BU      - byte unsigned
;  1422          1544  1 !       W       - word signed
;  1423          1545  1 !       WU      - word unsigned
;  1424          1546  1 !       L       - longword signed
;  1425          1547  1 !       LU      - longword unsigned
;  1426          1548  1 !       V       - aligned bitstring. e.g., the primary X<0,5,0>
;  1427          1549  1 !       SV      - signed aligned bitstring. e.g., X<0,5,1>
;  1428          1550  1 !       VU      - unaligned bitstring. e.g.,    X<1,5,0>
;  1429          1551  1 !       SVU     - signed unaligned bitstring, e.g., X<1,5,1>
;  1430          1552  1 !       PTR     - we get this back for REF items
;  1431          1553  1 !       Z       - "unknown" - we get this back for BLISS field names
;  1432          1554  1 !                   e.g., X[fieldname]. No operations are allowed on
;  1433          1555  1 !                   fieldnames so this type does not appear in the
;  1434          1556  1 !                   tables.
;  1435          1557  1 !       ZI      - instruction, %LINE 10
;  1436          1558  1 !       ZEM     - entry mask, e.g., EV ROUT_NAME
;  1437          1559  1 !
;  1438          1560  1 ! DBG$PRIM_TO_ADDR always returns type L.
;  1439          1561  1
;  1440          1562  1
;  1441          1563  1 ! Define a table that maps PTR type into integer. Type PTR can come
;  1442          1564  1 ! back from REF objects.
;  1443          1565  1 !
;  1444        P 1566  1 TYPE_MAPPING_TABLE (BLISS_MAP_TABLE,
;  1445          1567  1     TYPE_GRAPH_EDGE (PTR,L));
;  1446          1568  1
;  1447          1569  1 ! Define the Type Conversion Information Table for BLISS.
;  1448          1570  1 ! This points to the above mapping table.
;  1449          1571  1 !
;  1450        P 1572  1 CONVERSION_INFO_TABLE (BLISS_CVTINFO_TABLE,
;  1451          1573  1     CONVERSION_INFO_ENTRY (BLISS_MAP_TABLE, TABLEBASE));
;  1452          1574  1
;  1453          1575  1
;  1454          1576  1 ! Define the Type Hierarchy Table. All operations are done on signed longwords.
;  1455          1577  1 ! Thus we provide a path for all types to be converted to signed longwords.
;  1456          1578  1 !
;  1457        P 1579  1 TYPE_HIERARCHY_TABLE (BLISS_HIER_TABLE,
;  1458        P 1580  1     TYPE_GRAPH_EDGE (BU, WU),
;  1459        P 1581  1     TYPE_GRAPH_EDGE (WU, LU),
;  1460        P 1582  1     TYPE_GRAPH_EDGE (LU,  L),
;  1461        P 1583  1     TYPE_GRAPH_EDGE (B ,  W),
;  1462        P 1584  1     TYPE_GRAPH_EDGE (W ,  L),
;  1463          1585  1     0);
;  1464          1586  1
```

```
;  1465          1587  1
;  1466          1588  1 ! Define the Type Hierarchy Table for DEPOSIT.
;  1467          1589  1 ! This is a circular graph which includes all types that can be obtained
;  1468          1590  1 ! from calling PRIM_TO_VAL on a BLISS primary. What this means is that
;  1469          1591  1 ! any type is convertible to any other type on a DEPOSIT.
;  1470          1592  1 !
;  1471        P 1593  1 TYPE_HIERARCHY_TABLE (BLISS_HIERD_TABLE,
;  1472        P 1594  1     TYPE_GRAPH_EDGE (B, BU),
;  1473        P 1595  1     TYPE_GRAPH_EDGE (BU, W),
;  1474        P 1596  1     TYPE_GRAPH_EDGE (W, WU),
;  1475        P 1597  1     TYPE_GRAPH_EDGE (WU, L),
;  1476        P 1598  1     TYPE_GRAPH_EDGE (L, LU),
;  1477        P 1599  1     TYPE_GRAPH_EDGE (LU, VU),
;  1478        P 1600  1     TYPE_GRAPH_EDGE (VU, SVU),
;  1479        P 1601  1     TYPE_GRAPH_EDGE (SVU, SV),
;  1480        P 1602  1     TYPE_GRAPH_EDGE (SV, V),
;  1481        P 1603  1     TYPE_GRAPH_EDGE (V, B),
;  1482          1604  1     0);
;  1483          1605  1
;  1484          1606  1 !++
;  1485          1607  1 ! All of the arithmetic operations below are defined to work on signed
;  1486          1608  1 ! longwords.
;  1487          1609  1 !
;  1488          1610  1 ! Note that only the fetch operator (.) does an implicit fetch for BLISS.
;  1489          1611  1 ! This means, for example, that
;  1490          1612  1 !       EVAL A+B
;  1491          1613  1 ! will add the addresses of A and B, not the values. Addresses are obtained
;  1492          1614  1 ! from the DBG$PRIM_TO_ADDR, and their type is always L.
;  1493          1615  1 !
;  1494          1616  1 ! If the fetch is done, the DBG$PRIM_TO_VAL routine gets called, and the value
;  1495          1617  1 ! is pulled from that value descriptor and put into the result value
;  1496          1618  1 ! descriptor, which is of type longword. Thus the result of the fetch
;  1497          1619  1 ! operator is always longword. Any extraction of byte, word, bitfield, etc.
;  1498          1620  1 ! values is done inside of DBG$PRIM_TO_VAL.
;  1499          1621  1 !
;  1500          1622  1 ! For example,
;  1501          1623  1 !       EVAL .W + .BU  ! W is word signed, BU is byte unsigned
;  1502          1624  1 ! The .W operator is first applied. DBG$PRIM_TO_VAL is called and it
;  1503          1625  1 ! extracts the word quantity for W, sign extends it to a longword,
;  1504          1626  1 ! and places the longword value in the Value Descriptor. We copy that
;  1505          1627  1 ! value into the Value Descriptor containing the result of the .W operation.
;  1506          1628  1 ! Similarly we obtain a type L descriptor with the value of BU, properly
;  1507          1629  1 ! zero-extended. These are then added as longwords, and the result is
;  1508          1630  1 ! a longword.
;  1509          1631  1 !
;  1510          1632  1 ! So the upshot of all this is that none of the operations except FETCH
;  1511          1633  1 ! ever see anything except dtype L.
;  1512          1634  1 !--
;  1513          1635  1
;  1514          1636  1 ! Define the Operator Routine Table for BLISS addition.
;  1515          1637  1 !
;  1516        P 1638  1 OPERATOR_ROUTINE_TABLE (BLISS_ADD_TABLE,
;  1517          1639  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L));
;  1518          1640  1
;  1519          1641  1 ! Define the Operator Routine Table for BLISS subtraction.
;  1520          1642  1 !
;  1521        P 1643  1 OPERATOR_ROUTINE_TABLE (BLISS_SUB_TABLE,
```

```
;  1522          1644  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L));
;  1523          1645  1
;  1524          1646  1 ! Define the Operator Routine Table for BLISS Multiplication.
;  1525          1647  1 !
;  1526        P 1648  1 OPERATOR_ROUTINE_TABLE (BLISS_MUL_TABLE,
;  1527          1649  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L));
;  1528          1650  1
;  1529          1651  1 ! Define the Operator Routine Table for BLISS Division.
;  1530          1652  1 !
;  1531        P 1653  1 OPERATOR_ROUTINE_TABLE (BLISS_DIV_TABLE,
;  1532          1654  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L));
;  1533          1655  1
;  1534          1656  1 ! Define the Operator Routine Table for BLISS Modulus.
;  1535          1657  1 ! The BLISS modulus function is actually a remainder function.
;  1536          1658  1 !
;  1537        P 1659  1 OPERATOR_ROUTINE_TABLE (BLISS_MOD_TABLE,
;  1538          1660  1     OPERATOR_ROUTINE (L, L, L, REM_L_L));
;  1539          1661  1
;  1540          1662  1 ! Define the Operator Routine Table for BLISS arithmetic shift.
;  1541          1663  1 ! There is only one shift operation in BLISS. It uses the "^" symbol.
;  1542          1664  1 ! A positive right argument indicates left shift and a negative right
;  1543          1665  1 ! argument indicates right shift. This is how the SHIFT_LEFT_L_L routine
;  1544          1666  1 ! behaves.
;  1545          1667  1 !
;  1546        P 1668  1 OPERATOR_ROUTINE_TABLE (BLISS_SHIFT_TABLE,
;  1547          1669  1     OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L));
;  1548          1670  1
;  1549          1671  1 ! Define the Operator Routine Table for BLISS Equal.
;  1550          1672  1 !
;  1551        P 1673  1 OPERATOR_ROUTINE_TABLE (BLISS_EQUAL_TABLE,
;  1552          1674  1     OPERATOR_ROUTINE (L, L, L, EQL_L_L));
;  1553          1675  1
;  1554          1676  1 ! Define the Operator Routine Table for BLISS Not Equal.
;  1555          1677  1 !
;  1556        P 1678  1 OPERATOR_ROUTINE_TABLE (BLISS_NOT_EQUAL_TABLE,
;  1557          1679  1     OPERATOR_ROUTINE (L, L, L, NEQ_L_L));
;  1558          1680  1
;  1559          1681  1 ! Define the Operator Routine Table for BLISS Less Than.
;  1560          1682  1 !
;  1561        P 1683  1 OPERATOR_ROUTINE_TABLE (BLISS_LSS_THAN_TABLE,
;  1562          1684  1     OPERATOR_ROUTINE (L, L, L, LSS_L_L));
;  1563          1685  1
;  1564          1686  1 ! Define the Operator Routine Table for BLISS Less Than Unsigned.
;  1565          1687  1 !
;  1566        P 1688  1 OPERATOR_ROUTINE_TABLE (BLISS_LSSU_THAN_TABLE,
;  1567          1689  1     OPERATOR_ROUTINE (L, L, L, LSS_LU_LO));
;  1568          1690  1
;  1569          1691  1 ! Define the Operator Routine Table for BLISS Greater Than.
;  1570          1692  1 !
;  1571        P 1693  1 OPERATOR_ROUTINE_TABLE (BLISS_GTR_THAN_TABLE,
;  1572          1694  1     OPERATOR_ROUTINE (L, L, L, GTR_L_L));
;  1573          1695  1
;  1574          1696  1 ! Define the Operator Routine Table for BLISS Greater Than Unsigned.
;  1575          1697  1 !
;  1576        P 1698  1 OPERATOR_ROUTINE_TABLE (BLISS_GTRU_THAN_TABLE,
;  1577          1699  1     OPERATOR_ROUTINE (L, L, L, GTR_LU_LO));
;  1578          1700  1
```

```
;  1579          1701  1 ! Define the Operator Routine Table for BLISS Less Than or Equal.
;  1580          1702  1 !
;  1581        P 1703  1 OPERATOR_ROUTINE_TABLE (BLISS_LSS_EQUAL_TABLE,
;  1582          1704  1     OPERATOR_ROUTINE (L, L, L, LEQ_L_L));
;  1583          1705  1
;  1584          1706  1 ! Define the Operator Routine Table for BLISS Less Than or Equal Unsigned.
;  1585          1707  1 !
;  1586        P 1708  1 OPERATOR_ROUTINE_TABLE (BLISS_LSSU_EQUAL_TABLE,
;  1587          1709  1     OPERATOR_ROUTINE (L, L, L, LEQ_LU_LU));
;  1588          1710  1
;  1589          1711  1 ! Define the Operator Routine Table for BLISS Greater Than or Equal.
;  1590          1712  1 !
;  1591        P 1713  1 OPERATOR_ROUTINE_TABLE (BLISS_GTR_EQUAL_TABLE,
;  1592          1714  1     OPERATOR_ROUTINE (L, L, L, GEQ_L_L));
;  1593          1715  1
;  1594          1716  1 ! Define the Operator Routine Table for BLISS Greater Than or Equal Unsigned.
;  1595          1717  1 !
;  1596        P 1718  1 OPERATOR_ROUTINE_TABLE (BLISS_GTRU_EQUAL_TABLE,
;  1597          1719  1     OPERATOR_ROUTINE (L, L, L, GEQ_LU_LU));
;  1598          1720  1
;  1599          1721  1 ! Define the Operator Routine Table for BLISS Bitwise And.
;  1600          1722  1 !
;  1601        P 1723  1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_AND_TABLE,
;  1602          1724  1     OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
;  1603          1725  1
;  1604          1726  1 ! Define the Operator Routine Table for BLISS Bitwise Or.
;  1605          1727  1 !
;  1606        P 1728  1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_OR_TABLE,
;  1607          1729  1     OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
;  1608          1730  1
;  1609          1731  1 ! Define the Operator Routine Table for BLISS Bitwise Xor.
;  1610          1732  1 !
;  1611        P 1733  1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_XOR_TABLE,
;  1612          1734  1     OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
;  1613          1735  1
;  1614          1736  1 ! Define the Operator Routine Table for BLISS Bitwise Eqv.
;  1615          1737  1 !
;  1616        P 1738  1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_EQV_TABLE,
;  1617          1739  1     OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
;  1618          1740  1
;  1619          1741  1 ! Define the Operator Routine Table for BLISS Unary Plus.
;  1620          1742  1 !
;  1621        P 1743  1 OPERATOR_ROUTINE_TABLE (BLISS_UNARY_PLUS_TABLE,
;  1622        P 1744  1
;  1623        P 1745  1 ! The following are not language dependent types.  This is needed for DEBUG
;  1624        P 1746  1 ! types.  For example, DEP/QUAD L= +1.
;  1625        P 1747  1 !
;  1626        P 1748  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  1627        P 1749  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  1628        P 1750  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  1629        P 1751  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  1630        P 1752  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  1631        P 1753  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  1632        P 1754  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  1633        P 1755  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  1634        P 1756  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  1635        P 1757  1
```

```
;  1636          1758  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L));
;  1637          1759  1
;  1638          1760  1 ! Define the Operator Routine Table for BLISS Unary Minus.
;  1639          1761  1 !
;  1640        P 1762  1 OPERATOR_ROUTINE_TABLE (BLISS_UNARY_MINUS_TABLE,
;  1641        P 1763  1
;  1642        P 1764  1 ! The following are not language dependent types.  This is needed for DEBUG
;  1643        P 1765  1 ! types.  For example, DEP/QUAD L= +1.
;  1644        P 1766  1 !
;  1645        P 1767  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  1646        P 1768  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  1647        P 1769  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  1648        P 1770  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  1649        P 1771  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  1650        P 1772  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  1651        P 1773  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  1652        P 1774  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  1653        P 1775  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  1654        P 1776  1
;  1655          1777  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L));
;  1656          1778  1
;  1657          1779  1 ! Define the Operator Routine Table for BLISS Bitwise Not.
;  1658          1780  1 !
;  1659        P 1781  1 OPERATOR_ROUTINE_TABLE (BLISS_BIT_NOT_TABLE,
;  1660          1782  1     OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
;  1661          1783  1
;  1662          1784  1 ! Define the Operator Routine Table for BLISS bit selection.
;  1663          1785  1 ! The bit-select operator X<p,s,e> can be applied to any
;  1664          1786  1 ! BLISS expression. It goes through the DBG$BLISS_BITSELECT
;  1665          1787  1 ! routine in DBGEVALOP.
;  1666          1788  1 !
;  1667          1789  1 ! The <p,s,e> operation can be viewed as just modifying the
;  1668          1790  1 ! address given by X. That is, X can be any BLISS expression,
;  1669          1791  1 ! and the result of all BLISS expressions is a longword.
;  1670          1792  1 ! If the X<p,s,e> expression does not have a fetch associated with
;  1671          1793  1 ! it, the value of X<p,s,e> is just X+p/8
;  1672          1794  1 !
;  1673          1795  1 ! If there is a fetch, then what happens is:
;  1674          1796  1 ! (1) The <p,s,e> operator is done first. The DBG$BLISS_BITSELECT routine just
;  1675          1797  1 !     squirels away the information inside of DBGEVALOP.
;  1676          1798  1 !     Nothing is done with it until the fetch.
;  1677          1799  1 ! (2) the extraction of the BLISS field is
;  1678          1800  1 !     done at the evaluation of the fetch operator. (DBG$BLISS_INDIRECTION
;  1679          1801  1 !     in the DBGEVALOP module.)
;  1680          1802  1 !
;  1681        P 1803  1 OPERATOR_ROUTINE_TABLE (BLISS_BITSELECT_TABLE,
;  1682          1804  1     OPERATOR_ROUTINE (L, L, L, BITSELECT));
;  1683          1805  1
;  1684          1806  1 ! Define the Operator Routine Table for BLISS indirection.
;  1685          1807  1 ! Indirection is the only operator that call DBG$PRIM_TO_VAL to do
;  1686          1808  1 ! the implicit fetch, so it is the only one that may see all the possible
;  1687          1809  1 ! dtypes that we may get back from a BLISS primary. We thus include
;  1688          1810  1 ! those dtypes where it is legal to do a fetch.(E.g., .ROUT-NAME is not
;  1689          1811  1 ! legal, so that is not here).
;  1690          1812  1 !
;  1691        P 1813  1 OPERATOR_ROUTINE_TABLE (BLISS_INDIRECT_TABLE,
;  1692        P 1814  1     OPERATOR_ROUTINE (B, B, L, INDIRECT_LU),
```

```
;  1693        P 1815  1     OPERATOR_ROUTINE (W, W, L, INDIRECT_LU),
;  1694        P 1816  1     OPERATOR_ROUTINE (BU, BU, L, INDIRECT_LU),
;  1695        P 1817  1     OPERATOR_ROUTINE (WU, WU, L, INDIRECT_LU),
;  1696        P 1818  1     OPERATOR_ROUTINE (LU, LU, L, INDIRECT_LU),
;  1697        P 1819  1     OPERATOR_ROUTINE (V, V, L, INDIRECT_LU),
;  1698        P 1820  1     OPERATOR_ROUTINE (VU, VU, L, INDIRECT_LU),
;  1699        P 1821  1     OPERATOR_ROUTINE (SV, SV, L, INDIRECT_LU),
;  1700        P 1822  1     OPERATOR_ROUTINE (SVU, SVU, L, INDIRECT_LU),
;  1701          1823  1     OPERATOR_ROUTINE (L, L, L, INDIRECT_LU);
;  1702          1824  1
;  1703          1825  1
;  1704          1826  1 ! Define the Operator Information Table for BLISS.
;  1705          1827  1 !
;  1706        P 1828  1 OPERATOR_INFO_TABLE (BLISS_OPINFO_TABLE,
;  1707        P 1829  1
;  1708        P 1830  1     ! All of the arithmetic operators use the same hierarchy table,
;  1709        P 1831  1     ! BLISS_HIER_TABLE. They all have the fetch flag turned off, meaning
;  1710        P 1832  1     ! they do address arithmetic in the absence of an explicit fetch.
;  1711        P 1833  1     ! There is not incompatibility table for any of the BLISS operators.
;  1712        P 1834  1     !
;  1713        P 1835  1     OPERATOR_INFO_ENTRY
;  1714        P 1836  1         (UNARY_PLUS, BLISS_UNARY_PLUS_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1715        P 1837  1     OPERATOR_INFO_ENTRY
;  1716        P 1838  1         (UNARY_MINUS, BLISS_UNARY_MINUS_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1717        P 1839  1     OPERATOR_INFO_ENTRY
;  1718        P 1840  1         (BIT_NOT, BLISS_BIT_NOT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1719        P 1841  1     OPERATOR_INFO_ENTRY
;  1720        P 1842  1         (ADD, BLISS_ADD_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1721        P 1843  1     OPERATOR_INFO_ENTRY
;  1722        P 1844  1         (SUBTRACT, BLISS_SUB_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1723        P 1845  1     OPERATOR_INFO_ENTRY
;  1724        P 1846  1         (MULTIPLY, BLISS_MUL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1725        P 1847  1     OPERATOR_INFO_ENTRY
;  1726        P 1848  1         (DIVIDE, BLISS_DIV_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1727        P 1849  1     OPERATOR_INFO_ENTRY
;  1728        P 1850  1         (REMAINDER, BLISS_MOD_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1729        P 1851  1     OPERATOR_INFO_ENTRY
;  1730        P 1852  1         (LEFT_SHIFT, BLISS_SHIFT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1731        P 1853  1     OPERATOR_INFO_ENTRY
;  1732        P 1854  1         (EQUAL, BLISS_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1733        P 1855  1     OPERATOR_INFO_ENTRY
;  1734        P 1856  1         (NOT_EQUAL, BLISS_NOT_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1735        P 1857  1     OPERATOR_INFO_ENTRY
;  1736        P 1858  1         (GTR_THAN, BLISS_GTR_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1737        P 1859  1     OPERATOR_INFO_ENTRY
;  1738        P 1860  1         (GTR_THAN_U, BLISS_GTRU_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1739        P 1861  1     OPERATOR_INFO_ENTRY
;  1740        P 1862  1         (LSS_THAN, BLISS_LSS_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1741        P 1863  1     OPERATOR_INFO_ENTRY
;  1742        P 1864  1         (LSS_THAN_U, BLISS_LSSU_THAN_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1743        P 1865  1     OPERATOR_INFO_ENTRY
;  1744        P 1866  1         (GTR_EQUAL, BLISS_GTR_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1745        P 1867  1     OPERATOR_INFO_ENTRY
;  1746        P 1868  1         (GTR_EQUAL_U, BLISS_GTRU_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1747        P 1869  1     OPERATOR_INFO_ENTRY
;  1748        P 1870  1         (LSS_EQUAL, BLISS_LSS_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1749        P 1871  1     OPERATOR_INFO_ENTRY
```

```
;  1750        P 1872  1          (LSS_EQUAL_U, BLISS_LSSU_EQUAL_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1751        P 1873  1      OPERATOR_INFO_ENTRY
;  1752        P 1874  1          (BIT_AND, BLISS_BIT_AND_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1753        P 1875  1      OPERATOR_INFO_ENTRY
;  1754        P 1876  1          (BIT_OR, BLISS_BIT_OR_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1755        P 1877  1      OPERATOR_INFO_ENTRY
;  1756        P 1878  1          (BIT_XOR, BLISS_BIT_XOR_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1757        P 1879  1      OPERATOR_INFO_ENTRY
;  1758        P 1880  1          (BIT_EQV, BLISS_BIT_EQV_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1759        P 1881  1      OPERATOR_INFO_ENTRY
;  1760        P 1882  1          (BITSELECT, BLISS_BITSELECT_TABLE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1761        P 1883  1
;  1762        P 1884  1      ! CONVERT gets called to convert subscripts to integer type.
;  1763        P 1885  1      ! It also gets called to convert expressions in FOR loops or
;  1764        P 1886  1      ! REPEAT counts, or expressions in WHILE or IF statements, to
;  1765        P 1887  1      ! integer type. As far as I know, conversion to type L is the
;  1766        P 1888  1      ! only conversion we see for BLISS. The normal hierarchy table
;  1767        P 1889  1      ! should thus be adequate.
;  1768        P 1890  1      !
;  1769        P 1891  1      OPERATOR_INFO_ENTRY
;  1770        P 1892  1          (CONVERT, TABLEBASE, BLISS_HIER_TABLE, TABLEBASE, FALSE),
;  1771        P 1893  1
;  1772        P 1894  1      ! The fetch operator is the one that has the fetch flag set to TRUE.
;  1773        P 1895  1      !
;  1774        P 1896  1      OPERATOR_INFO_ENTRY
;  1775        P 1897  1          (INDIRECT, BLISS_INDIRECT_TABLE, BLISS_HIER_TABLE, TABLEBASE, TRUE),
;  1776        P 1898  1
;  1777        P 1899  1      ! DEPOSIT uses its own hierarchy table, allowing DEPOSIT of any type
;  1778        P 1900  1      ! into any type. The fetch_flag is still false, indicating that
;  1779        P 1901  1      ! no implicit fetch is done on the right-hand-side of the deposit.
;  1780        P 1902  1      ! E.g., DEP X = Y  will put the address of Y into X; DEP X = .Y
;  1781        P 1903  1      ! will put the value of Y into X.
;  1782        P 1904  1      !
;  1783        P 1905  1      OPERATOR_INFO_ENTRY
;  1784        P 1906  1          (DEPOSIT, TABLEBASE, BLISS_HIERD_TABLE, TABLEBASE, FALSE),
;  1785        P 1907  1
;  1786        P 1908  1      ! The identity operator is called at the end of an evaluate if
;  1787        P 1909  1      ! we still have a primary, e.g., EVAL X  will call DBG$EVAL_LANG_OPERATOR
;  1788        P 1910  1      ! with "IDENTITY" and we can then do our thing. In the BLISS case,
;  1789        P 1911  1      ! "doing our thing" means calling DBG$PRIM_TO_ADDR, and returning
;  1790        P 1912  1      ! that descriptor.
;  1791        P 1913  1      !
;  1792        P 1914  1      ! The reason for an "identity" operator is to ensure that EVAL A
;  1793        P 1915  1      ! will go through the same code paths as, say, EVAL A+0 or EVAL +A
;  1794        P 1916  1      !
;  1795        P 1917  1      ! The identity operator uses no tables.
;  1796        P 1918  1      !
;  1797        P 1919  1      OPERATOR_INFO_ENTRY
;  1798        P 1920  1          (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE, FALSE)
;  1799          1921  1      );
;  1800          1922  1
```

```
;  1802          1923  1 ;
;  1803          1924  1 ;       C  O P E R A T O R  I N F O R M A T I O N  T A B L E S
;  1804          1925  1 ;
;  1805          1926  1 ;
;  1806          1927  1 ; This section contains the Operator Routine and Type tables needed to
;  1807          1928  1 ; evaluate expressions in the C language.
;  1808          1929  1 ;
;  1809          1930  1 ; C constants:
;  1810          1931  1 ;       Integer: decimal, octal, hexdecimal.  Double, String
;  1811          1932  1 ;
;  1812          1933  1 ; C Data Types:
;  1813          1934  1 ;       . Integer (signed, unsigned)
;  1814          1935  1 ;         char ( 8-bit byte), short (16-bit integer), int (32-bit integer)
;  1815          1936  1 ;       . Floating-point numbers
;  1816          1937  1 ;         float (F_float), double (D_float)
;  1817          1938  1 ;       . Enum values
;  1818          1939  1 ;         Scalars of a user-defined type
;  1819          1940  1 ;       . Pointers (typed)
;  1820          1941  1 ;         32-bit addresses of other variables
;  1821          1942  1 ;
;  1822          1943  1 ; C aggregates:
;  1823          1944  1 ;       . Array
;  1824          1945  1 ;       . Structure
;  1825          1946  1 ;       . Union
;  1826          1947  1 ;
;  1827          1948  1 ; Expressions:
;  1828          1949  1 ;       . Primary
;  1829          1950  1 ;
;  1830          1951  1 ;       . Negating Arithmetic (-E, TC applied)
;  1831          1952  1 ;         E: an E of any arithmetic type.
;  1832          1953  1 ;         Result: the arithmetic negative of the expression.  (The negative of
;  1833          1954  1 ;         an unsigned quantity is computed by subtracting its value from 2**32)
;  1834          1955  1 ;
;  1835          1956  1 ;       . Negating Logical (!E, TC applied)
;  1836          1957  1 ;         E: pointer (or other address-valued E, array), or an E of any
;  1837          1958  1 ;         arithmetic type.  Result: the logical negative of the expression,
;  1838          1959  1 ;         result type is int.
;  1839          1960  1 ;
;  1840          1961  1 ;       . Incrementing and Decrementing Variables (--, ++)
;  1841          1962  1 ;
;  1842          1963  1 ;       . Computing address (&lvalue)
;  1843          1964  1 ;         Result: the address of the object to which the lvalue refers.
;  1844          1965  1 ;         (The & may not be applied to register or to bit fields in structure
;  1845          1966  1 ;          or union).
;  1846          1967  1 ;
;  1847          1968  1 ;       . Dereferencing Pointers (*E)
;  1848          1969  1 ;         E: Pointer or other address-valued E.
;  1849          1970  1 ;         Result: a reference to the object to which the expression points,
;  1850          1971  1 ;         the type of the addressed object is the type of the result.
;  1851          1972  1 ;
;  1852          1973  1 ;       . One's complement (~E, TC applied)
;  1853          1974  1 ;         E: integer or character
;  1854          1975  1 ;
;  1855          1976  1 ;       . Additive - (+, -, TC applied)
;  1856          1977  1 ;         1. Operand: Address of an array element and a value of any integral
;  1857          1978  1 ;            type can be added (integer is converted to an address offset by
;  1858          1979  1 ;            by integer * length of the addressed object).  Result: the address
```

```
;1859          1980  1 !           of an object of the same type.
;1860          1981  1 !         2. Operand: A value of any integral type may be subtracted from a
;1861          1982  1 !            pointer or address.
;1862          1983  1 !         3. Operand: enum + enum, or enum - enum.  Result: int
;1863          1984  1 !         4. If two addresses of objects of the same type are substracted,
;1864          1985  1 !            the result is int.
;1865          1986  1 !
;1866          1987  1 !       . Multiplicative - (*, /, %: mod, TC applied)
;1867          1988  1 !         Operand: integral mod integral.
;1868          1989  1 !
;1869          1990  1 !       . Equality - (==, !=, TC applied)  Result: type int.
;1870          1991  1 !         1. Two pointers or addresses (if they identify the same storage
;1871          1992  1 !            location then they are equal).
;1872          1993  1 !         2. A pointer or address can be compared with an integer.
;1873          1994  1 !
;1874          1995  1 !       . Relational - (<, >, <=, >=, TC applied)  Result: type int.
;1875          1996  1 !         same as above.
;1876          1997  1 !
;1877          1998  1 !       . Bitwise - (&: and, ^: xor, |:or, TC applied)
;1878          1999  1 !         Operand: both must be integrals.
;1879          2000  1 !
;1880          2001  1 !       . Logical - (&&: and, ||: or)  Result: type int.
;1881          2002  1 !         Operand: fundamentatal types or a pointer, or address-valued E.
;1882          2003  1 !
;1883          2004  1 !       . Shift - (<<, >>, TC applied)
;1884          2005  1 !         Operand: both must be inegral.  the right-hand operand --> int, and
;1885          2006  1 !         the type of the result is the type of the left operand.
;1886          2007  1 !         E1 << E2: the value of E1 shifted to the left by E2 bits, vacated
;1887          2008  1 !                   bits are cleard.
;1888          2009  1 !         E1 >> E2: the value of E1 shifted to the right by E2 bits.  Vacated
;1889          2010  1 !                   bits are cleared if E1 is unsigned, else, vacated bits
;1890          2011  1 !                   are filled with a copy of E1's sign bit.
;1891          2012  1 !         (The result of the shift is undefined if E2 is negative, or the
;1892          2013  1 !          value of E2 > 32 bits)
;1893          2014  1 !
;1894          2015  1 !       . Assignment, conditional, and comma (we do not support)
;1895          2016  1 !
;1896          2017  1 ! Type Conversion: (operands of different types appear in an expression)
;1897          2018  1 !       . Arithmetic
;1898          2019  1 !         1. char or short (signed or unsigned) --> int (signed, or unsigned)
;1899          2020  1 !            f_float --> d_float.  Char is treated as signed.
;1900          2021  1 !         2. If either operand is double, the other --> double.  Result: double.
;1901          2022  1 !         3. If either operand is unsigned, the other --> unsigned.  Result: unsigned.
;1902          2023  1 !         4. Otherwise, both operands must be int.  Result: integer.
;1903          2024  1 !         5. Whenever an unsigned integer and a signed integer are combined,
;1904          2025  1 !            signed --> signed int --> unsigned.  Result: unsigned.
;1905          2026  1 !         6. For some operators require integers as operands, f or d --> int.
;1906          2027  1 !
;1907          2028  1
;1908          2029  1
;1909          2030  1 ! Define the Type Conversion Information Table for C.
;1910          2031  1 ! C has no special rules for type conversions, so we do not have a language
;1911          2032  1 ! specific type conversion table.
;1912          2033  1 !
;1913        P 2034  1 CONVERSION_INFO_TABLE (C_CVTINFO_TABLE,
;1914          2035  1     CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
;1915          2036  1
```

```
;  1916          2037  1
;  1917          2038  1 ! Define the Type Hierarchy Table for C.
;  1918          2039  1 ! This table defines what is refered to as the "usual type conversion rules"
;  1919          2040  1 ! in the C manual. These rules state, basically, that:
;  1920          2041  1 !
;  1921          2042  1 ! char -> longword integer
;  1922          2043  1 ! short integer -> longword integer (with the same sign attribute)
;  1923          2044  1 ! signed integer -> unsigned
;  1924          2045  1 ! integer -> float
;  1925          2046  1 ! float -> double float
;  1926          2047  1 !
;  1927          2048  1 ! The first edge, T->B, is there so that variables declared as CHAR can
;  1928          2049  1 ! be treated as integers, as C allows. The conversion should check that
;  1929          2050  1 ! the length of the char string is 1.
;  1930          2051  1 !
;  1931          2052  1 ! We also include an edge for ENUM->L. This will allow any arithmetic with
;  1932          2053  1 ! enumeration types that is also allowed for integers. This may be a more
;  1933          2054  1 ! permissive implementation than the language allows.
;  1934          2055  1 !
;  1935          2056  1 ! We also include V -> LU, VU -> LU, SV -> L, and SVU -> L.
;  1936          2057  1 ! C declares components of a packed record to be of type V, VU, SV, or SVU,
;  1937          2058  1 ! but operations on these are just integer operations. So we convert
;  1938          2059  1 ! these to integer.
;  1939          2060  1 !
;  1940        P 2061  1 TYPE_HIERARCHY_TABLE (C_HIER_TABLE,
;  1941        P 2062  1     TYPE_GRAPH_EDGE (T, B),
;  1942        P 2063  1
;  1943        P 2064  1     TYPE_GRAPH_EDGE (ENUM, L),
;  1944        P 2065  1
;  1945        P 2066  1     TYPE_GRAPH_EDGE (V,  LU),
;  1946        P 2067  1     TYPE_GRAPH_EDGE (VU, LU),
;  1947        P 2068  1     TYPE_GRAPH_EDGE (SV, L),
;  1948        P 2069  1     TYPE_GRAPH_EDGE (SVU, L),
;  1949        P 2070  1
;  1950        P 2071  1     TYPE_GRAPH_EDGE (B , W),
;  1951        P 2072  1     TYPE_GRAPH_EDGE (W , L),
;  1952        P 2073  1     TYPE_GRAPH_EDGE (BU, WU),
;  1953        P 2074  1     TYPE_GRAPH_EDGE (WU, LU),
;  1954        P 2075  1     TYPE_GRAPH_EDGE (L, LU),
;  1955        P 2076  1     TYPE_GRAPH_EDGE (L , D),
;  1956        P 2077  1     TYPE_GRAPH_EDGE (LU, D),
;  1957        P 2078  1     TYPE_GRAPH_EDGE (F , D),
;  1958          2079  1     0);
;  1959          2080  1
;  1960          2081  1
;  1961          2082  1 ! The HIERD table defines what pairs are legal in a DEPOSIT.
;  1962          2083  1 ! Give a circular graph which allows DEPOSIT any-any.
;  1963          2084  1 !
;  1964        P 2085  1 TYPE_HIERARCHY_TABLE (C_HIERD_TABLE,
;  1965        P 2086  1     TYPE_GRAPH_EDGE (T, B),
;  1966        P 2087  1     TYPE_GRAPH_EDGE (B , BU),
;  1967        P 2088  1     TYPE_GRAPH_EDGE (BU, W),
;  1968        P 2089  1     TYPE_GRAPH_EDGE (W , WU),
;  1969        P 2090  1     TYPE_GRAPH_EDGE (WU, LU),
;  1970        P 2091  1     TYPE_GRAPH_EDGE (LU, L),
;  1971        P 2092  1     TYPE_GRAPH_EDGE (L, F),
;  1972        P 2093  1     TYPE_GRAPH_EDGE (F , D),
```

```
;1973        P 2094  1      TYPE_GRAPH_EDGE (D, ENUM),
;1974        P 2095  1      TYPE_GRAPH_EDGE (ENUM, TPTR),
;1975        P 2096  1      TYPE_GRAPH_EDGE (TPTR, V),
;1976        P 2097  1      TYPE_GRAPH_EDGE (V, SV),
;1977        P 2098  1      TYPE_GRAPH_EDGE (SV, VU),
;1978        P 2099  1      TYPE_GRAPH_EDGE (VU, SVU),
;1979        P 2100  1      TYPE_GRAPH_EDGE (SVU, T),
;1980          2101  1      0);
;1981          2102  1
;1982          2103  1
;1983          2104  1 ! Add, subtract, multiply, divide, unary minus
;1984          2105  1 ! These accept all numeric types.
;1985          2106  1
;1986          2107  1 ! Define the Operator Routine Table for C Addition.
;1987          2108  1 ! Add also has special cases for TPTR + L, which is a special kind
;1988          2109  1 ! of addition in which the integer is scaled to the size of the
;1989          2110  1 ! pointed-to object.
;1990          2111  1 !
;1991        P 2112  1 OPERATOR_ROUTINE_TABLE (C_ADD_TABLE,
;1992        P 2113  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;1993        P 2114  1     OPERATOR_ROUTINE (LU, LU, LU, ADD_LU_LU),
;1994        P 2115  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;1995        P 2116  1     OPERATOR_ROUTINE (TPTR, LU, TPTR, ADD_TPTR_L),
;1996          2117  1     OPERATOR_ROUTINE (LU, TPTR, TPTR, ADD_TPTR_L));
;1997          2118  1
;1998          2119  1
;1999          2120  1 ! Define the Operator Routine Table for C Subtraction.
;2000          2121  1 ! There is a special case routine for TPTR - L, which is a  special kind
;2001          2122  1 ! of subtraction in which the integer is scaled to the size of the
;2002          2123  1 ! pointed-to object. Also, TPTR-TPTR is another special case in which
;2003          2124  1 ! the result is scaled.
;2004          2125  1 !
;2005        P 2126  1 OPERATOR_ROUTINE_TABLE (C_SUB_TABLE,
;2006        P 2127  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;2007        P 2128  1     OPERATOR_ROUTINE (LU, LU, LU, SUB_LU_LU),
;2008        P 2129  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;2009        P 2130  1     OPERATOR_ROUTINE (TPTR, LU, TPTR, SUB_TPTR_L),
;2010          2131  1     OPERATOR_ROUTINE (TPTR, TPTR, L, SUB_TPTR_TPTR));
;2011          2132  1
;2012          2133  1
;2013          2134  1 ! Define the Operator Routine Table for C Multiplication.
;2014          2135  1 !
;2015        P 2136  1 OPERATOR_ROUTINE_TABLE (C_MUL_TABLE,
;2016        P 2137  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;2017        P 2138  1     OPERATOR_ROUTINE (LU, LU, LU, MUL_LU_LU),
;2018          2139  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D));
;2019          2140  1
;2020          2141  1
;2021          2142  1 ! Define the Operator Routine Table for C Division.
;2022          2143  1 !
;2023        P 2144  1 OPERATOR_ROUTINE_TABLE (C_DIV_TABLE,
;2024        P 2145  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;2025        P 2146  1     OPERATOR_ROUTINE (LU, LU, LU, DIV_LU_LU),
;2026          2147  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D));
;2027          2148  1
;2028          2149  1
;2029          2150  1 ! Define the Operator Routine Table for C Unary Minus (Negating Arithmetic).
```

```
;2030          2151  1 !
;2031        P 2152  1 OPERATOR_ROUTINE_TABLE (C_UNARY_MINUS_TABLE,
;2032        P 2153  1
;2033        P 2154  1 ! The following are not language dependent types.  This is needed for DEBUG
;2034        P 2155  1 ! types.  For example, DEP/QUAD L= +1.
;2035        P 2156  1 !
;2036        P 2157  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;2037        P 2158  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;2038        P 2159  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;2039        P 2160  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;2040        P 2161  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;2041        P 2162  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;2042        P 2163  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;2043        P 2164  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;2044        P 2165  1
;2045        P 2166  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;2046        P 2167  1     OPERATOR_ROUTINE (LU, LU, LU, UNARY_MINUS_LU),
;2047          2168  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D));
;2048          2169  1
;2049          2170  1
;2050          2171  1 ! This table is needed for the +/- constant.
;2051          2172  1 !
;2052        P 2173  1 OPERATOR_ROUTINE_TABLE (C_UNARY_PLUS_TABLE,
;2053        P 2174  1
;2054        P 2175  1 ! The following are not language dependent types.  This is needed for DEBUG
;2055        P 2176  1 ! types.  For example, DEP/QUAD L= +1.
;2056        P 2177  1 !
;2057        P 2178  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;2058        P 2179  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;2059        P 2180  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;2060        P 2181  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;2061        P 2182  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;2062        P 2183  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;2063        P 2184  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;2064        P 2185  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;2065        P 2186  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;2066          2187  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O));
;2067          2188  1
;2068          2189  1
;2069          2190  1 ! Define the Operator Routine Table for C Modulus.
;2070          2191  1 ! Modulus is only defined for integer types.
;2071          2192  1 ! Note: The C Modulus function is really the remainder function.
;2072          2193  1 !
;2073        P 2194  1 OPERATOR_ROUTINE_TABLE (C_MOD_TABLE,
;2074        P 2195  1     OPERATOR_ROUTINE (L, L, L, REM_L_L),
;2075          2196  1     OPERATOR_ROUTINE (LU, LU, LU, REM_LU_LU));
;2076          2197  1
;2077          2198  1
;2078          2199  1 ! Relational operators. These accept all three numeric types,
;2079          2200  1 ! and also pointer type (which is just treated as integer).
;2080          2201  1
;2081          2202  1 ! Define the Operator Routine Table for C Equal.
;2082          2203  1 ! Signed and usigned equal are the same so they both use the EQL_L_L
;2083          2204  1 ! routine.
;2084          2205  1 !
;2085        P 2206  1 OPERATOR_ROUTINE_TABLE (C_EQL_TABLE,
;2086        P 2207  1     OPERATOR_ROUTINE (TPTR, LO, L, EQL_L_L),
```

```
;  2087          P 2208  1      OPERATOR_ROUTINE (LU, TPTR, L, EQL_L_L),
;  2088          P 2209  1      OPERATOR_ROUTINE (TPTR, TPTR, L, EQL_L_L),
;  2089          P 2210  1      OPERATOR_ROUTINE (L, L, L, EQL_L_L),
;  2090          P 2211  1      OPERATOR_ROUTINE (LU, LU, L, EQL_L_L),
;  2091            2212  1      OPERATOR_ROUTINE (D, D, L, EQL_D_D));
;  2092            2213  1
;  2093            2214  1
;  2094            2215  1  ! Define the Operator Routine Table for C Not Equal.
;  2095            2216  1  ! Signed and usigned not-equal are the same so they both use the EQL_L_L
;  2096            2217  1  ! routine.
;  2097            2218  1  !
;  2098          P 2219  1  OPERATOR_ROUTINE_TABLE (C_NEQ_TABLE,
;  2099          P 2220  1      OPERATOR_ROUTINE (TPTR, LU, L, NEQ_L_L),
;  2100          P 2221  1      OPERATOR_ROUTINE (LU, TPTR, L, NEQ_L_L),
;  2101          P 2222  1      OPERATOR_ROUTINE (TPTR, TPTR, L, NEQ_L_L),
;  2102          P 2223  1      OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
;  2103          P 2224  1      OPERATOR_ROUTINE (LU, LU, L, NEQ_L_L),
;  2104            2225  1      OPERATOR_ROUTINE (D, D, L, NEQ_D_D));
;  2105            2226  1
;  2106            2227  1
;  2107            2228  1  ! Define the Operator Routine Table for C Less Than.
;  2108            2229  1  ! Unsigned less than used a different routine than signed less than.
;  2109            2230  1  !
;  2110          P 2231  1  OPERATOR_ROUTINE_TABLE (C_LSS_TABLE,
;  2111          P 2232  1      OPERATOR_ROUTINE (TPTR, L, L, LSS_L_L),
;  2112          P 2233  1      OPERATOR_ROUTINE (TPTR, LU, L, LSS_[U_LU),
;  2113          P 2234  1      OPERATOR_ROUTINE (L, TPTR, L, LSS_[_L),
;  2114          P 2235  1      OPERATOR_ROUTINE (LU, TPTR, L, LSS_[U_LU),
;  2115          P 2236  1      OPERATOR_ROUTINE (TPTR, TPTR, L, LSS_[_L),
;  2116          P 2237  1      OPERATOR_ROUTINE (L, L, L, LSS_L_L),
;  2117          P 2238  1      OPERATOR_ROUTINE (LU, LU, L, LSS_LU_LU),
;  2118            2239  1      OPERATOR_ROUTINE (D, D, L, LSS_D_D));
;  2119            2240  1
;  2120            2241  1
;  2121            2242  1  ! Define the Operator Routine Table for C Greater Than.
;  2122            2243  1  ! Unsigned greater than uses a different routine than signed greater than.
;  2123            2244  1  !
;  2124          P 2245  1  OPERATOR_ROUTINE_TABLE (C_GTR_TABLE,
;  2125          P 2246  1      OPERATOR_ROUTINE (TPTR, L, L, GTR_L_L),
;  2126          P 2247  1      OPERATOR_ROUTINE (TPTR, LU, L, GTR_[U_LU),
;  2127          P 2248  1      OPERATOR_ROUTINE (L, TPTR, L, GTR_[_L),
;  2128          P 2249  1      OPERATOR_ROUTINE (LU, TPTR, L, GTR_[U_LU),
;  2129          P 2250  1      OPERATOR_ROUTINE (TPTR, TPTR, L, GTR_[_L),
;  2130          P 2251  1      OPERATOR_ROUTINE (L, L, L, GTR_L_L),
;  2131          P 2252  1      OPERATOR_ROUTINE (LU, LU, L, GTR_LU_LU),
;  2132            2253  1      OPERATOR_ROUTINE (D, D, L, GTR_D_D));
;  2133            2254  1
;  2134            2255  1
;  2135            2256  1  ! Define the Operator Routine Table for C Less Than or Equal To.
;  2136            2257  1  ! Unsigned less than/equal to uses a different routine than signed
;  2137            2258  1  ! less than/equal to.
;  2138            2259  1  !
;  2139          P 2260  1  OPERATOR_ROUTINE_TABLE (C_LEQ_TABLE,
;  2140          P 2261  1      OPERATOR_ROUTINE (TPTR, L, L, LEQ_L_L),
;  2141          P 2262  1      OPERATOR_ROUTINE (TPTR, LU, L, LEQ_[U_LU),
;  2142          P 2263  1      OPERATOR_ROUTINE (L, TPTR, L, LEQ_[_L),
;  2143          P 2264  1      OPERATOR_ROUTINE (LU, TPTR, L, LEQ_[U_LU),
```

```
;  2144        P 2265  1      OPERATOR_ROUTINE (TPTR, TPTR, L, LEQ_L_L),
;  2145        P 2266  1      OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
;  2146        P 2267  1      OPERATOR_ROUTINE (LU, LU, L, LEQ_LU_LU),
;  2147          2268  1      OPERATOR_ROUTINE (D, D, L, LEQ_D_D));
;  2148          2269  1
;  2149          2270  1
;  2150          2271  1  ! Define the Operator Routine Table for C Greater Than or Equal.
;  2151          2272  1  ! Unsigned greater than or equal to uses a different routine than
;  2152          2273  1  ! signed greater than or equal to.
;  2153          2274  1  !
;  2154        P 2275  1  OPERATOR_ROUTINE_TABLE (C_GEQ_TABLE,
;  2155        P 2276  1      OPERATOR_ROUTINE (TPTR, L, L, GEQ_L_L),
;  2156        P 2277  1      OPERATOR_ROUTINE (TPTR, LU, L, GEQ_LU_LU),
;  2157        P 2278  1      OPERATOR_ROUTINE (L, TPTR, L, GEQ_L_L),
;  2158        P 2279  1      OPERATOR_ROUTINE (LU, TPTR, L, GEQ_LU_LU),
;  2159        P 2280  1      OPERATOR_ROUTINE (TPTR, TPTR, L, GEQ_L_L),
;  2160        P 2281  1      OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
;  2161        P 2282  1      OPERATOR_ROUTINE (LU, LU, L, GEQ_LU_LU),
;  2162          2283  1      OPERATOR_ROUTINE (D, D, L, GEQ_D_D));
;  2163          2284  1
;  2164          2285  1
;  2165          2286  1  ! Bitwise operators.
;  2166          2287  1  ! These accept only integer data types (no float).
;  2167          2288  1
;  2168          2289  1  ! Define the Operator Routine Table for C Bitwise And.
;  2169          2290  1  !
;  2170        P 2291  1  OPERATOR_ROUTINE_TABLE (C_BIT_AND_TABLE,
;  2171        P 2292  1      OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L),
;  2172          2293  1      OPERATOR_ROUTINE (LU, LU, LU, BIT_AND_L_L));
;  2173          2294  1
;  2174          2295  1
;  2175          2296  1  ! Define the Operator Routine Table for C Bitwise Or.
;  2176          2297  1  !
;  2177        P 2298  1  OPERATOR_ROUTINE_TABLE (C_BIT_OR_TABLE,
;  2178        P 2299  1      OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L),
;  2179          2300  1      OPERATOR_ROUTINE (LU, LU, LU, BIT_OR_L_L));
;  2180          2301  1
;  2181          2302  1
;  2182          2303  1  ! Define the Operator Routine Table for C Bitwise Xor.
;  2183          2304  1  !
;  2184        P 2305  1  OPERATOR_ROUTINE_TABLE (C_BIT_XOR_TABLE,
;  2185        P 2306  1      OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L),
;  2186          2307  1      OPERATOR_ROUTINE (LU, LU, LU, BIT_XOR_L_L));
;  2187          2308  1
;  2188          2309  1  ! Define the Operator Routine Table for C Bitwise Not (One's Complement).
;  2189          2310  1  !
;  2190        P 2311  1  OPERATOR_ROUTINE_TABLE (C_BIT_NOT_TABLE,
;  2191        P 2312  1      OPERATOR_ROUTINE (L, L, L, BIT_NOT_L),
;  2192          2313  1      OPERATOR_ROUTINE (LU, LU, LU, BIT_NOT_L));
;  2193          2314  1
;  2194          2315  1
;  2195          2316  1  ! Logical operations.
;  2196          2317  1  ! These just do the Boolean operations with TRUE <-> not zero,
;  2197          2318  1  ! FALSE <-> zero.
;  2198          2319  1  ! For mixed int-float logical operations, we convert both to float.
;  2199          2320  1  ! This is not identical to what the compiler does. But hopefully,
;  2200          2321  1  ! conversion to float should preserve the zero/notzero characteristic.
```

```
;  2201          2322  1 ! so we should get the same final answer except in obscure boundary
;  2202          2323  1 ! conditions.
;  2203          2324  1 ! We also allow pointers, which are just treated as integer.
;  2204          2325  1
;  2205          2326  1 ! Define the Operator Routine Table for C Logical And.
;  2206          2327  1 !
;  2207        P 2328  1 OPERATOR_ROUTINE_TABLE (C_AND_TABLE,
;  2208        P 2329  1     OPERATOR_ROUTINE (TPTR, TPTR, L, AND_L_L),
;  2209        P 2330  1     OPERATOR_ROUTINE (TPTR, LU, L, AND_L_L),
;  2210        P 2331  1     OPERATOR_ROUTINE (LU, TPTR, L, AND_L_L),
;  2211        P 2332  1     OPERATOR_ROUTINE (L, L, L, AND_L_L),
;  2212        P 2333  1     OPERATOR_ROUTINE (LU, LU, L, AND_L_L),
;  2213          2334  1     OPERATOR_ROUTINE (D, D, L, AND_D_D));
;  2214          2335  1
;  2215          2336  1
;  2216          2337  1 ! Define the Operator Routine Table for C Logical Or.
;  2217          2338  1 !
;  2218        P 2339  1 OPERATOR_ROUTINE_TABLE (C_OR_TABLE,
;  2219        P 2340  1     OPERATOR_ROUTINE (TPTR, TPTR, L, OR_L_L),
;  2220        P 2341  1     OPERATOR_ROUTINE (TPTR, LU, L, OR_L_L),
;  2221        P 2342  1     OPERATOR_ROUTINE (LU, TPTR, L, OR_L_L),
;  2222        P 2343  1     OPERATOR_ROUTINE (L, L, L, OR_L_L),
;  2223        P 2344  1     OPERATOR_ROUTINE (LU, LU, L, OR_L_L),
;  2224          2345  1     OPERATOR_ROUTINE (D, D, L, OR_D_D));
;  2225          2346  1
;  2226          2347  1
;  2227          2348  1 ! Define the Operator Routine Table for C Logical Not.
;  2228          2349  1 !
;  2229        P 2350  1 OPERATOR_ROUTINE_TABLE (C_NOT_TABLE,
;  2230        P 2351  1     OPERATOR_ROUTINE (TPTR, TPTR, L, NOT_L),
;  2231        P 2352  1     OPERATOR_ROUTINE (L, L, L, NOT_L),
;  2232        P 2353  1     OPERATOR_ROUTINE (LU, LU, L, NOT_L),
;  2233          2354  1     OPERATOR_ROUTINE (D, D, L, NOT_D));
;  2234          2355  1
;  2235          2356  1
;  2236          2357  1 ! Shift operators.
;  2237          2358  1 ! These accept only integer types.
;  2238          2359  1
;  2239          2360  1 ! Define the Operator Routine Table for C Left Shift.
;  2240          2361  1 !
;  2241        P 2362  1 OPERATOR_ROUTINE_TABLE (C_SHIFT_LEFT_TABLE,
;  2242        P 2363  1     OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L),
;  2243          2364  1     OPERATOR_ROUTINE (LU, LU, LU, SHIFT_LEFT_L_L));
;  2244          2365  1
;  2245          2366  1
;  2246          2367  1 ! Define the Operator Routine Table for C Right Shift.
;  2247          2368  1 ! Unsigned right shift is different from signed. For unsigned right
;  2248          2369  1 ! shift, we always shift in zeros. For signed right shift, we shift
;  2249          2370  1 ! in copies of the sign bit.
;  2250          2371  1 !
;  2251        P 2372  1 OPERATOR_ROUTINE_TABLE (C_SHIFT_RT_TABLE,
;  2252        P 2373  1     OPERATOR_ROUTINE (L, L, L, SHIFT_RT_L_L),
;  2253          2374  1     OPERATOR_ROUTINE (LU, LU, LU, SHIFT_RT_LU_LU));
;  2254          2375  1
;  2255          2376  1
;  2256          2377  1 ! Define Operator Routine Tables for ++X  X++ --X  X--
;  2257          2378  1 !
```

```
;  2258        P 2379  1 OPERATOR_ROUTINE_TABLE (C_PRE_INCR_TABLE,
;  2259        P 2380  1     OPERATOR_ROUTINE (L, L, L, PRE_INCR_L),
;  2260        P 2381  1     OPERATOR_ROUTINE (LU, LU, LU, PRE_INCR_LU),
;  2261        P 2382  1     OPERATOR_ROUTINE (D, D, D, PRE_INCR_D),
;  2262          2383  1     OPERATOR_ROUTINE (TPTR, TPTR, TPTR, PRE_INCR_TPTR));
;  2263          2384  1
;  2264        P 2385  1 OPERATOR_ROUTINE_TABLE (C_POST_INCR_TABLE,
;  2265        P 2386  1     OPERATOR_ROUTINE (L, L, L, POST_INCR_L),
;  2266        P 2387  1     OPERATOR_ROUTINE (LU, LU, LU, POST_INCR_LU),
;  2267        P 2388  1     OPERATOR_ROUTINE (D, D, D, POST_INCR_D),
;  2268          2389  1     OPERATOR_ROUTINE (TPTR, TPTR, TPTR, POST_INCR_TPTR));
;  2269          2390  1
;  2270        P 2391  1 OPERATOR_ROUTINE_TABLE (C_PRE_DECR_TABLE,
;  2271        P 2392  1     OPERATOR_ROUTINE (L, L, L, PRE_DECR_L),
;  2272        P 2393  1     OPERATOR_ROUTINE (LU, LU, LU, PRE_DECR_LU),
;  2273        P 2394  1     OPERATOR_ROUTINE (D, D, D, PRE_DECR_D),
;  2274          2395  1     OPERATOR_ROUTINE (TPTR, TPTR, TPTR, PRE_DECR_TPTR));
;  2275          2396  1
;  2276        P 2397  1 OPERATOR_ROUTINE_TABLE (C_POST_DECR_TABLE,
;  2277        P 2398  1     OPERATOR_ROUTINE (L, L, L, POST_DECR_L),
;  2278        P 2399  1     OPERATOR_ROUTINE (LU, LU, LU, POST_DECR_LU),
;  2279        P 2400  1     OPERATOR_ROUTINE (D, D, D, POST_DECR_D),
;  2280          2401  1     OPERATOR_ROUTINE (TPTR, TPTR, TPTR, POST_DECR_TPTR));
;  2281          2402  1
;  2282          2403  1
;  2283          2404  1 ! Define the Operator Routine Table for C Address Of.
;  2284          2405  1 ! The address-of operator will have the FETCH_FLAG off in the operator
;  2285          2406  1 ! information table. This means that PRIM_TO_ADDR will get called.
;  2286          2407  1 ! The value of the result will thus be the address of the operand.
;  2287          2408  1 ! The type of the result is "pointer to xxx", where "xxx" is the type
;  2288          2409  1 ! of the operand. This means that the the routine which is called to
;  2289          2410  1 ! do "address_of" must construct a typeid for the result.
;  2290          2411  1 !
;  2291        P 2412  1 OPERATOR_ROUTINE_TABLE (C_ADDRESS_TABLE,
;  2292          2413  1     OPERATOR_ROUTINE (L, L, TPTR, ADDRESS_L));
;  2293          2414  1
;  2294          2415  1
;  2295          2416  1 ! Define the Operator Routine Table for the size of operator.
;  2296          2417  1 ! Size-of will look up the typeid in the symbol table and determine
;  2297          2418  1 ! the declared size of the object. The FETCH_FLAG will be FALSE, so
;  2298          2419  1 ! PRIM_TO_ADDR will be called instead of PRIM_TO_VAL. This is
;  2299          2420  1 ! because we do not need the value of the object.
;  2300          2421  1 !
;  2301        P 2422  1 OPERATOR_ROUTINE_TABLE (C_SIZEOF_TABLE,
;  2302          2423  1     OPERATOR_ROUTINE (L, L, L, SIZEOF_L));
;  2303          2424  1
;  2304          2425  1
;  2305          2426  1 ! Define the Operator Routine Table for C Indirection (Dereferencing Pointers).
;  2306          2427  1 ! The indirection operator (*) will do a pointer dereference of
;  2307          2428  1 ! its argument. The type of the returned object depends on the
;  2308          2429  1 ! type of the pointer. (We will have to do a symbol table lookup
;  2309          2430  1 ! on the TYPEID).
;  2310          2431  1 !
;  2311        P 2432  1 OPERATOR_ROUTINE_TABLE (C_INDIRECT_TABLE,
;  2312          2433  1     OPERATOR_ROUTINE (TPTR, TPTR, UNKNOWN, INDIRECT_TPTR));
;  2313          2434  1
;  2314          2435  1
```

```
;  2315          2436  1 ! Define the Operator Information Table for C.
;  2316          2437  1 !
;  2317        P 2438  1 OPERATOR_INFO_TABLE (C_OPINFO_TABLE,
;  2318        P 2439  1
;  2319        P 2440  1     ! All C operators use the same hierarchy table, and have no
;  2320        P 2441  1     ! type incompatibility table.
;  2321        P 2442  1
;  2322        P 2443  1     ! Arithmetic operators.
;  2323        P 2444  1     !
;  2324        P 2445  1     OPERATOR_INFO_ENTRY
;  2325        P 2446  1         (ADD, C_ADD_TABLE, C_HIER_TABLE, TABLEBASE),
;  2326        P 2447  1     OPERATOR_INFO_ENTRY
;  2327        P 2448  1         (SUBTRACT, C_SUB_TABLE, C_HIER_TABLE, TABLEBASE),
;  2328        P 2449  1     OPERATOR_INFO_ENTRY
;  2329        P 2450  1         (MULTIPLY, C_MUL_TABLE, C_HIER_TABLE, TABLEBASE),
;  2330        P 2451  1     OPERATOR_INFO_ENTRY
;  2331        P 2452  1         (DIVIDE, C_DIV_TABLE, C_HIER_TABLE, TABLEBASE),
;  2332        P 2453  1     OPERATOR_INFO_ENTRY
;  2333        P 2454  1         (REMAINDER, C_MOD_TABLE, C_HIER_TABLE, TABLEBASE),
;  2334        P 2455  1     OPERATOR_INFO_ENTRY
;  2335        P 2456  1         (UNARY_MINUS, C_UNARY_MINUS_TABLE, C_HIER_TABLE, TABLEBASE),
;  2336        P 2457  1
;  2337        P 2458  1     ! This is needed for /Qualifier for depositing the +/- constant.
;  2338        P 2459  1     !
;  2339        P 2460  1     OPERATOR_INFO_ENTRY
;  2340        P 2461  1         (UNARY_PLUS, C_UNARY_PLUS_TABLE, C_HIER_TABLE, TABLEBASE),
;  2341        P 2462  1
;  2342        P 2463  1     ! Relational operators.
;  2343        P 2464  1     !
;  2344        P 2465  1     OPERATOR_INFO_ENTRY
;  2345        P 2466  1         (EQUAL, C_EQL_TABLE, C_HIER_TABLE, TABLEBASE),
;  2346        P 2467  1     OPERATOR_INFO_ENTRY
;  2347        P 2468  1         (NOT_EQUAL, C_NEQ_TABLE, C_HIER_TABLE, TABLEBASE),
;  2348        P 2469  1     OPERATOR_INFO_ENTRY
;  2349        P 2470  1         (LSS_THAN, C_LSS_TABLE, C_HIER_TABLE, TABLEBASE),
;  2350        P 2471  1     OPERATOR_INFO_ENTRY
;  2351        P 2472  1         (GTR_THAN, C_GTR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2352        P 2473  1     OPERATOR_INFO_ENTRY
;  2353        P 2474  1         (LSS_EQUAL, C_LEQ_TABLE, C_HIER_TABLE, TABLEBASE),
;  2354        P 2475  1     OPERATOR_INFO_ENTRY
;  2355        P 2476  1         (GTR_EQUAL, C_GEQ_TABLE, C_HIER_TABLE, TABLEBASE),
;  2356        P 2477  1
;  2357        P 2478  1     ! Bitwise logical operators.
;  2358        P 2479  1     !
;  2359        P 2480  1     OPERATOR_INFO_ENTRY
;  2360        P 2481  1         (BIT_AND, C_BIT_AND_TABLE, C_HIER_TABLE, TABLEBASE),
;  2361        P 2482  1     OPERATOR_INFO_ENTRY
;  2362        P 2483  1         (BIT_OR, C_BIT_OR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2363        P 2484  1     OPERATOR_INFO_ENTRY
;  2364        P 2485  1         (BIT_XOR, C_BIT_XOR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2365        P 2486  1     OPERATOR_INFO_ENTRY
;  2366        P 2487  1         (BIT_NOT, C_BIT_NOT_TABLE, C_HIER_TABLE, TABLEBASE),
;  2367        P 2488  1
;  2368        P 2489  1     ! Logical operators.
;  2369        P 2490  1     !
;  2370        P 2491  1     OPERATOR_INFO_ENTRY
;  2371        P 2492  1         (SHORT_AND, C_AND_TABLE, C_HIER_TABLE, TABLEBASE),
```

```
;  2372          P 2493  1     OPERATOR_INFO_ENTRY
;  2373          P 2494  1         (SHORT_OR, C_OR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2374          P 2495  1     OPERATOR_INFO_ENTRY
;  2375          P 2496  1         (NOT, C_NOT_TABLE, C_HIER_TABLE, TABLEBASE),
;  2376          P 2497  1
;  2377          P 2498  1     ! Shift operators.
;  2378          P 2499  1     !
;  2379          P 2500  1     OPERATOR_INFO_ENTRY
;  2380          P 2501  1         (LEFT_SHIFT, C_SHIFT_LEFT_TABLE, C_HIER_TABLE, TABLEBASE),
;  2381          P 2502  1     OPERATOR_INFO_ENTRY
;  2382          P 2503  1         (RIGHT_SHIFT, C_SHIFT_RT_TABLE, C_HIER_TABLE, TABLEBASE),
;  2383          P 2504  1
;  2384          P 2505  1     ! Operators with side effects.
;  2385          P 2506  1     !
;  2386          P 2507  1     OPERATOR_INFO_ENTRY
;  2387          P 2508  1         (PRE_INCR, C_PRE_INCR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2388          P 2509  1     OPERATOR_INFO_ENTRY
;  2389          P 2510  1         (POST_INCR, C_POST_INCR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2390          P 2511  1     OPERATOR_INFO_ENTRY
;  2391          P 2512  1         (PRE_DECR, C_PRE_DECR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2392          P 2513  1     OPERATOR_INFO_ENTRY
;  2393          P 2514  1         (POST_DECR, C_POST_DECR_TABLE, C_HIER_TABLE, TABLEBASE),
;  2394          P 2515  1
;  2395          P 2516  1     ! Operators that work on the SYMID or TYPEID.
;  2396          P 2517  1     !
;  2397          P 2518  1     OPERATOR_INFO_ENTRY
;  2398          P 2519  1         (ADDRESS_OF, C_ADDRESS_TABLE, C_HIER_TABLE, TABLEBASE, FALSE),
;  2399          P 2520  1     OPERATOR_INFO_ENTRY
;  2400          P 2521  1         (SIZEOF, C_SIZEOF_TABLE, C_HIER_TABLE, TABLEBASE, FALSE),
;  2401          P 2522  1
;  2402          P 2523  1     ! Indirection
;  2403          P 2524  1     !
;  2404          P 2525  1     OPERATOR_INFO_ENTRY
;  2405          P 2526  1         (INDIRECT, C_INDIRECT_TABLE, C_HIER_TABLE, TABLEBASE),
;  2406          P 2527  1
;  2407          P 2528  1     ! DEPOSIT is used to implement the DEPOSIT command.
;  2408          P 2529  1     !
;  2409          P 2530  1     OPERATOR_INFO_ENTRY
;  2410          P 2531  1         (DEPOSIT, TABLEBASE, C_HIERD_TABLE, TABLEBASE),
;  2411          P 2532  1
;  2412          P 2533  1     ! CONVERT is used for things like converting subscripts.
;  2413          P 2534  1     !
;  2414          P 2535  1     OPERATOR_INFO_ENTRY
;  2415          P 2536  1         (CONVERT, TABLEBASE, C_HIER_TABLE, TABLEBASE),
;  2416          P 2537  1
;  2417          P 2538  1     ! Identity is called at the end of an EVALUATE if we still have
;  2418          P 2539  1     ! a Primary.
;  2419          P 2540  1     !
;  2420          P 2541  1     OPERATOR_INFO_ENTRY
;  2421            2542  1         (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
;  2422            2543  1
```

```
;  2424          2544  1 ;
;  2425          2545  1 ;       C O B O L   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  2426          2546  1 ;
;  2427          2547  1 ;
;  2428          2548  1 ; This section contains the Operator Routine and Type tables needed to
;  2429          2549  1 ; evaluate expressions in the COBOL language.
;  2430          2550  1 ;
;  2431          2551  1 ;
;  2432          2552  1 ; The following summarizes the information in the COBOL manual, and from
;  2433          2553  1 ; our old cobol support about data types, type conversions and operators.
;  2434          2554  1 ; There is further documentation within the tables below, describing
;  2435          2555  1 ; exactly how we translate this into the DEBUG tables.
;  2436          2556  1 ;
;  2437          2557  1 ; Cobol Data Types:
;  2438          2558  1 ;
;  2439          2559  1 ;       .Elementary
;  2440          2560  1 ;        Alphabetic, Alphanumeric:          (Text string)
;  2441          2561  1 ;        Numeric:
;  2442          2562  1 ;          Binary
;  2443          2563  1 ;            PIC 9      -- PIC 9(4)  COMP   (Scaled W, WU)
;  2444          2564  1 ;            PIC 9(5)   -- PIC 9(9)  COMP   (Scaled L, LU)
;  2445          2565  1 ;            PIC 9(10) -- PIC 9(18) COMP   (Scaled Q, QU)
;  2446          2566  1 ;            INDEX                         (L)
;  2447          2567  1 ;          Floating
;  2448          2568  1 ;            COMP-1                        (Floating)
;  2449          2569  1 ;            COMP-2                        (Double)
;  2450          2570  1 ;          Packed-Decimal
;  2451          2571  1 ;            COMP-3                        (Packed with/without Scaled)
;  2452          2572  1 ;          Decimal
;  2453          2573  1 ;            PIC 99...                     (Scaled NU)
;  2454          2574  1 ;            PIC S9... LEADING SEPARATE    (Scaled NL)
;  2455          2575  1 ;            PIC S9... LEADING             (Scaled NLO)
;  2456          2576  1 ;            PIC S9... TRAILING SEPARATE   (Scaled NR)
;  2457          2577  1 ;            PIC S9... TRAILING            (Scaled NRO)
;  2458          2578  1 ;
;  2459          2579  1 ;       .Group
;  2460          2580  1 ;        Alphanumeric                      (Text String)
;  2461          2581  1 ;
;  2462          2582  1 ;       .Edited data                       (Text String)
;  2463          2583  1 ;
;  2464          2584  1 ; Expressions:
;  2465          2585  1 ;       .Arithmetic
;  2466          2586  1 ;        operator: + - * / (** - not support) unary+ unary-
;  2467          2587  1 ;        operand:  numeric literal or numeric elementary identifier
;  2468          2588  1 ;
;  2469          2589  1 ;       .Conditional (TRUE, FALSE)
;  2470          2590  1 ;        operator: =, NOT =, >, NOT >, <, NOT <
;  2471          2591  1 ;        operand:  elementary identifier, literal, or AE
;  2472          2592  1 ;
;  2473          2593  1 ;       .Complex Conditional (TRUE, FALSE)
;  2474          2594  1 ;        operator: NOT, AND, OR
;  2475          2595  1 ;        operand:  conditions
;  2476          2596  1 ;
;  2477          2597  1 ;       Note: abbreviated forms are not supported.
;  2478          2598  1 ;
;  2479          2599  1 ; Type Conversion:
;  2480          2600  1 ;
```

```
;  2481          2601  1 ! SD     WU==W ----------------+
;  2482          2602  1 ! SD     LU==L ----------------|
;  2483          2603  1 ! SD     QU==Q ----------------|
;  2484          2604  1 !          NU, NL, NLO, NR, NRO-|--> P
;  2485          2605  1 !          F--------------------|
;  2486          2606  1 !          D--------------------+
;  2487          2607  1 !
;  2488          2608  1 ! 1.       All the integer and float constants are picked up as packed decimal if
;  2489          2609  1 !          one of the operand is SD descriptor type.
;  2490          2610  1 !
;  2491          2611  1 ! 2.       If one of the operand is Scaled Descriptor or Packed decimal
;  2492          2612  1 !          data type the operation is always done in Packed Form.  The
;  2493          2613  1 !          operation even includes the deposit.
;  2494          2614  1 !
;  2495          2615  1 !          So in COBOL, one will have the following case:
;  2496          2616  1 !            DEP SC2V2=F
;  2497          2617  1 !            F converts to P, make a packed decimal place holder for SC2V2
;  2498          2618  1 !            (which is word scaled -2), convert P to this place holder,
;  2499          2619  1 !            final conversion will convert this place holder to SC2V2 data type.
;  2500          2620  1 !
;  2501          2621  1 !            EV F+F
;  2502          2622  1 !            the operation is done by adding F to F.
;  2503          2623  1 !
;  2504          2624  1 !            EV F+P
;  2505          2625  1 !            the operation is done by adding P to P.
;  2506          2626  1 !
;  2507          2627  1 !            DEP P=123.123
;  2508          2628  1 !            in here 123.123 is picked up as packed decimal constant instead of
;  2509          2629  1 !            float.
;  2510          2630  1 !
;  2511          2631  1 !            DEP (SD, WU) = -123
;  2512          2632  1 !            the value we got is the absolute value.
;  2513          2633  1 !
;  2514          2634  1 !            EV  1.1+1
;  2515          2635  1 !            the operation is done in Packed form.
;  2516          2636  1 !
;  2517          2637  1 ! Define the COBOL Specific Type Conversion Table.  This allows proper
;  2518          2638  1 ! handling of COBOL Edited Data type.
;  2519          2639  1 !
;  2520        P 2640  1 LANG_CVT_TABLE (COBOL_CVT_TABLE,
;  2521        P 2641  1     LANG_CVT_ENTRY (COB_PICT, PICT, ANY),
;  2522          2642  1     0);
;  2523          2643  1
;  2524          2644  1
;  2525          2645  1 ! Define the Type Conversion Information Table for COBOL.  No rounding
;  2526          2646  1 ! takes place in COBOL.
;  2527          2647  1 !
;  2528        P 2648  1 CONVERSION_INFO_TABLE (COBOL_CVTINFO_TABLE,
;  2529          2649  1     CONVERSION_INFO_ENTRY (TABLEBASE, COBOL_CVT_TABLE));
;  2530          2650  1
;  2531          2651  1 ! Define Type Hierarchy Table for COBOL.
;  2532          2652  1 !
;  2533        P 2653  1 TYPE_HIERARCHY_TABLE (COBOL_HIER_TABLE,
;  2534        P 2654  1     TYPE_GRAPH_EDGE (WU, P),
;  2535        P 2655  1     TYPE_GRAPH_EDGE (W, P),
;  2536        P 2656  1     TYPE_GRAPH_EDGE (LU, P),
;  2537        P 2657  1     TYPE_GRAPH_EDGE (L, P),
```

```
;  2538        P 2658  1      TYPE_GRAPH_EDGE (QU, P),
;  2539        P 2659  1      TYPE_GRAPH_EDGE (Q, P),
;  2540        P 2660  1      TYPE_GRAPH_EDGE (NU, P),
;  2541        P 2661  1      TYPE_GRAPH_EDGE (NL, P),
;  2542        P 2662  1      TYPE_GRAPH_EDGE (NLO, P),
;  2543        P 2663  1      TYPE_GRAPH_EDGE (NR, P),
;  2544        P 2664  1      TYPE_GRAPH_EDGE (NRO, P),
;  2545        P 2665  1      TYPE_GRAPH_EDGE (F, D),
;  2546        P 2666  1      TYPE_GRAPH_EDGE (D, P),
;  2547          2667  1      0);
;  2548          2668  1
;  2549          2669  1  ! Define the Type Hierarchy Table for COBOL deposit.
;  2550          2670  1  !
;  2551        P 2671  1  TYPE_HIERARCHY_TABLE (COBOL_HIERD_TABLE,
;  2552        P 2672  1      TYPE_GRAPH_EDGE (PICT, QU),
;  2553        P 2673  1      TYPE_GRAPH_EDGE (WU, W),
;  2554        P 2674  1      TYPE_GRAPH_EDGE (W, LU),
;  2555        P 2675  1      TYPE_GRAPH_EDGE (LU, L),
;  2556        P 2676  1      TYPE_GRAPH_EDGE (L, QU),
;  2557        P 2677  1      TYPE_GRAPH_EDGE (QU, Q),
;  2558        P 2678  1      TYPE_GRAPH_EDGE (Q, NU),
;  2559        P 2679  1      TYPE_GRAPH_EDGE (NU, NL),
;  2560        P 2680  1      TYPE_GRAPH_EDGE (NL, NLO),
;  2561        P 2681  1      TYPE_GRAPH_EDGE (NLO, NR),
;  2562        P 2682  1      TYPE_GRAPH_EDGE (NR, NRO),
;  2563        P 2683  1      TYPE_GRAPH_EDGE (NRO, P),
;  2564        P 2684  1      TYPE_GRAPH_EDGE (P, F),
;  2565        P 2685  1      TYPE_GRAPH_EDGE (F, D),
;  2566        P 2686  1      TYPE_GRAPH_EDGE (D, PICT),
;  2567          2687  1      0);
;  2568          2688  1
;  2569          2689  1  ! Define the Operator Routine Table for COBOL addition.
;  2570          2690  1  !
;  2571        P 2691  1  OPERATOR_ROUTINE_TABLE (COBOL_ADD_TABLE,
;  2572        P 2692  1      OPERATOR_ROUTINE (P, P, P, ADD_P_P),
;  2573        P 2693  1      OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  2574          2694  1      OPERATOR_ROUTINE (D, D, D, ADD_D_D));
;  2575          2695  1
;  2576          2696  1  ! Define the Operator Routine Table for COBOL subtraction.
;  2577          2697  1  !
;  2578        P 2698  1  OPERATOR_ROUTINE_TABLE (COBOL_SUB_TABLE,
;  2579        P 2699  1      OPERATOR_ROUTINE (P, P, P, SUB_P_P),
;  2580        P 2700  1      OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  2581          2701  1      OPERATOR_ROUTINE (D, D, D, SUB_D_D));
;  2582          2702  1
;  2583          2703  1  ! Define the Operator Routine Table for COBOL multiplication.
;  2584          2704  1  !
;  2585        P 2705  1  OPERATOR_ROUTINE_TABLE (COBOL_MUL_TABLE,
;  2586        P 2706  1      OPERATOR_ROUTINE (P, P, P, MUL_P_P),
;  2587        P 2707  1      OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  2588          2708  1      OPERATOR_ROUTINE (D, D, D, MUL_D_D));
;  2589          2709  1
;  2590          2710  1  ! Define the Operator Routine Table for COBOL division.
;  2591          2711  1  !
;  2592        P 2712  1  OPERATOR_ROUTINE_TABLE (COBOL_DIV_TABLE,
;  2593        P 2713  1      OPERATOR_ROUTINE (P, P, P, DIV_P_P),
;  2594        P 2714  1      OPERATOR_ROUTINE (F, F, F, DIV_F_F),
```

```
;  2595          2715  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D));
;  2596          2716  1
;  2597          2717  1 ! Define the Operator Routine Table for COBOL unary plus.
;  2598          2718  1 !
;  2599        P 2719  1 OPERATOR_ROUTINE_TABLE (COBOL_UNARY_PLUS_TABLE,
;  2600        P 2720  1
;  2601        P 2721  1 ! The following are not language dependent types.  This is needed for DEBUG
;  2602        P 2722  1 ! types.  For example, DEP/QUAD L= +1.
;  2603        P 2723  1 !
;  2604        P 2724  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  2605        P 2725  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  2606        P 2726  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  2607        P 2727  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  2608        P 2728  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  2609        P 2729  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  2610        P 2730  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  2611        P 2731  1
;  2612        P 2732  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  2613        P 2733  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  2614          2734  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D));
;  2615          2735  1
;  2616          2736  1 ! Define the Operator Routine Table for COBOL unary minus.
;  2617          2737  1 !
;  2618        P 2738  1 OPERATOR_ROUTINE_TABLE (COBOL_UNARY_MINUS_TABLE,
;  2619        P 2739  1
;  2620        P 2740  1 ! The following are not language dependent types.  This is needed for DEBUG
;  2621        P 2741  1 ! types.  For example, DEP/QUAD L= +1.
;  2622        P 2742  1 !
;  2623        P 2743  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  2624        P 2744  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  2625        P 2745  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  2626        P 2746  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  2627        P 2747  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  2628        P 2748  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  2629        P 2749  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  2630        P 2750  1
;  2631        P 2751  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  2632        P 2752  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  2633          2753  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D));
;  2634          2754  1
;  2635          2755  1 ! Define the Operator Routine Table for COBOL =.
;  2636          2756  1 !
;  2637        P 2757  1 OPERATOR_ROUTINE_TABLE (COBOL_EQL_TABLE,
;  2638        P 2758  1     OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
;  2639        P 2759  1     OPERATOR_ROUTINE (P, P, TF, EQL_P_P),
;  2640        P 2760  1     OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
;  2641          2761  1     OPERATOR_ROUTINE (D, D, TF, EQL_D_D));
;  2642          2762  1
;  2643          2763  1 ! Define the Operator Routine Table for COBOL NOT =.
;  2644          2764  1 !
;  2645        P 2765  1 OPERATOR_ROUTINE_TABLE (COBOL_NEQ_TABLE,
;  2646        P 2766  1     OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
;  2647        P 2767  1     OPERATOR_ROUTINE (P, P, TF, NEQ_P_P),
;  2648        P 2768  1     OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
;  2649          2769  1     OPERATOR_ROUTINE (D, D, TF, NEQ_D_D));
;  2650          2770  1
;  2651          2771  1 ! Define the Operator Routine Table for COBOL >.
```

```
;  2652          2772  1 !
;  2653        P 2773  1 OPERATOR_ROUTINE_TABLE (COBOL_GTR_TABLE,
;  2654        P 2774  1     OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
;  2655        P 2775  1     OPERATOR_ROUTINE (P, P, TF, GTR_P_P),
;  2656        P 2776  1     OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
;  2657          2777  1     OPERATOR_ROUTINE (D, D, TF, GTR_D_D));
;  2658          2778  1
;  2659          2779  1 ! Define the Operator Routine Table for COBOL NOT <.
;  2660          2780  1 !
;  2661        P 2781  1 OPERATOR_ROUTINE_TABLE (COBOL_GEQ_TABLE,
;  2662        P 2782  1     OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
;  2663        P 2783  1     OPERATOR_ROUTINE (P, P, TF, GEQ_P_P),
;  2664        P 2784  1     OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
;  2665          2785  1     OPERATOR_ROUTINE (D, D, TF, GEQ_D_D));
;  2666          2786  1
;  2667          2787  1 ! Define the Operator Routine Table for COBOL <.
;  2668          2788  1 !
;  2669        P 2789  1 OPERATOR_ROUTINE_TABLE (COBOL_LSS_TABLE,
;  2670        P 2790  1     OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
;  2671        P 2791  1     OPERATOR_ROUTINE (P, P, TF, LSS_P_P),
;  2672        P 2792  1     OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
;  2673          2793  1     OPERATOR_ROUTINE (D, D, TF, LSS_D_D));
;  2674          2794  1
;  2675          2795  1 ! Define the Operator Routine Table for COBOL NOT >.
;  2676          2796  1 !
;  2677        P 2797  1 OPERATOR_ROUTINE_TABLE (COBOL_LEQ_TABLE,
;  2678        P 2798  1     OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
;  2679        P 2799  1     OPERATOR_ROUTINE (P, P, TF, LEQ_P_P),
;  2680        P 2800  1     OPERATOR_ROUTINE (F, F, TF, LE  F_F),
;  2681          2801  1     OPERATOR_ROUTINE (D, D, TF, LEQ_D_D));
;  2682          2802  1
;  2683          2803  1 ! Define the Operator Routine Table for COBOL NOT.
;  2684          2804  1 !
;  2685        P 2805  1 OPERATOR_ROUTINE_TABLE (COBOL_NOT_TABLE,
;  2686          2806  1     OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
;  2687          2807  1
;  2688          2808  1 ! Define the Operator Routine Table for COBOL AND.
;  2689          2809  1 !
;  2690        P 2810  1 OPERATOR_ROUTINE_TABLE (COBOL_AND_TABLE,
;  2691          2811  1     OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
;  2692          2812  1
;  2693          2813  1 ! Define the Operator Routine Table for COBOL OR.
;  2694          2814  1 !
;  2695        P 2815  1 OPERATOR_ROUTINE_TABLE (COBOL_OR_TABLE,
;  2696          2816  1     OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
;  2697          2817  1
;  2698          2818  1
;  2699          2819  1 ! Define the Operator Information Table for COBOL.
;  2700          2820  1 !
;  2701        P 2821  1 OPERATOR_INFO_TABLE (COBOL_OPINFO_TABLE,
;  2702        P 2822  1     OPERATOR_INFO_ENTRY
;  2703        P 2823  1         (ADD, COBOL_ADD_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2704        P 2824  1     OPERATOR_INFO_ENTRY
;  2705        P 2825  1         (SUBTRACT, COBOL_SUB_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2706        P 2826  1     OPERATOR_INFO_ENTRY
;  2707        P 2827  1         (MULTIPLY, COBOL_MUL_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2708        P 2828  1     OPERATOR_INFO_ENTRY
```

```
;  2709          P 2829  1          (DIVIDE, COBOL_DIV_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2710          P 2830  1      OPERATOR_INFO_ENTRY
;  2711          P 2831  1          (UNARY_PLUS, COBOL_UNARY_PLUS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2712          P 2832  1      OPERATOR_INFO_ENTRY
;  2713          P 2833  1          (UNARY_MINUS, COBOL_UNARY_MINUS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2714          P 2834  1      OPERATOR_INFO_ENTRY
;  2715          P 2835  1          (EQUAL, COBOL_EQL_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2716          P 2836  1      OPERATOR_INFO_ENTRY
;  2717          P 2837  1          (NOT_EQUAL, COBOL_NEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2718          P 2838  1      OPERATOR_INFO_ENTRY
;  2719          P 2839  1          (GTR_THAN, COBOL_GTR_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2720          P 2840  1      OPERATOR_INFO_ENTRY
;  2721          P 2841  1          (GTR_EQUAL, COBOL_GEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2722          P 2842  1      OPERATOR_INFO_ENTRY
;  2723          P 2843  1          (LSS_THAN, COBOL_LSS_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2724          P 2844  1      OPERATOR_INFO_ENTRY
;  2725          P 2845  1          (LSS_EQUAL, COBOL_LEQ_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2726          P 2846  1      OPERATOR_INFO_ENTRY
;  2727          P 2847  1          (NOT, COBOL_NOT_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2728          P 2848  1      OPERATOR_INFO_ENTRY
;  2729          P 2849  1          (AND, COBOL_AND_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2730          P 2850  1      OPERATOR_INFO_ENTRY
;  2731          P 2851  1          (OR, COBOL_OR_TABLE, COBOL_HIER_TABLE, TABLEBASE),
;  2732          P 2852  1      OPERATOR_INFO_ENTRY
;  2733          P 2853  1          (CONVERT, TABLEBASE, COBOL_HIERD_TABLE, TABLEBASE),
;  2734          P 2854  1      OPERATOR_INFO_ENTRY
;  2735          P 2855  1          (DEPOSIT, TABLEBASE, COBOL_HIERD_TABLE, TABLEBASE),
;  2736          P 2856  1      OPERATOR_INFO_ENTRY
;  2737            2857  1          (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
;  2738            2858  1
```

```
;  2740          2859  1 !
;  2741          2860  1 !  F O R T R A N   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  2742          2861  1 !
;  2743          2862  1 ! This section contains the Operator Routine and Type tables needed to
;  2744          2863  1 ! evaluate expressions in the FORTRAN language.
;  2745          2864  1 !
;  2746          2865  1 !
;  2747          2866  1 ! The following summarizes the information in the FORTRAN manual about
;  2748          2867  1 ! data types, type conversions, and operators. There is further documentation
;  2749          2868  1 ! within the tables below, describing exactly how we translate this into
;  2750          2869  1 ! the DEBUG tables.
;  2751          2870  1 !
;  2752          2871  1 ! Fortran Data:
;  2753          2872  1 !       Constants, Integer, Real, Character variables, Integer, Real, Character
;  2754          2873  1 !       Array
;  2755          2874  1 !
;  2756          2875  1 ! Fortran Data Types:
;  2757          2876  1 !       . Logical
;  2758          2877  1 !         logical*1 (byte), logical*2 (word), logical*4 (long)
;  2759          2878  1 !       . Integer
;  2760          2879  1 !         integer*2 (word), integer*4 (long)
;  2761          2880  1 !       . Real
;  2762          2881  1 !         real*4 (f_float), real*8 (d_float, g_float), real*16 (h_float)
;  2763          2882  1 !       . Complex
;  2764          2883  1 !         complex*8 (f_float), complex*16 (d_float, g_float)
;  2765          2884  1 !       . Character
;  2766          2885  1 !
;  2767          2886  1 ! Expressions:
;  2768          2887  1 !       . Arithmetic
;  2769          2888  1 !         Operand: numeric (logical, integer, real).  Result: numeric values.
;  2770          2889  1 !         Operator: ** (exponentiation), * (multiplication), / (division),
;  2771          2890  1 !         + (addition, unary plus), - (subtraction, unary minus)
;  2772          2891  1 !
;  2773          2892  1 !       . Character
;  2774          2893  1 !         Operand: character.  Result: character values.
;  2775          2894  1 !         Operator: // (concatenation)
;  2776          2895  1 !
;  2777          2896  1 !       . Relational
;  2778          2897  1 !         Operand: numerics or characters.  Result: logical values.
;  2779          2898  1 !         Operator: .LT. (less than), .LE. (less than or equal to),
;  2780          2899  1 !         .EQ. (equal to), .NE. (not equal to), .GT. (greater than),
;  2781          2900  1 !         .GE. (greater than or equal to)
;  2782          2901  1 !
;  2783          2902  1 !       . Logical
;  2784          2903  1 !         Operand: integer or logical.  Result: logical values.
;  2785          2904  1 !         Operator: .AND., .OR., .XOR., .NEQV. (.XOR.), .EQV., .NOT.
;  2786          2905  1 !
;  2787          2906  1 ! Type Conversion:
;  2788          2907  1 !       . Arithmetic
;  2789          2908  1 !         1. Operands are of the same data type, the result is also of that
;  2790          2909  1 !            data type.
;  2791          2910  1 !
;  2792          2911  1 !         2. Otherwise, Logical --> integer*2 --> integer*4 --> real*4 -->
;  2793          2912  1 !            real*8 --> real*16 --> complex*8 --> complex*16.
;  2794          2913  1 !
;  2795          2914  1 !               *** Note - although a chart such as the one given above
;  2796          2915  1 !               *** does appear in the FORTRAN manual, that is not
```

```
;  2797          2916  1 !           *** exactly how things are really done. A more precise graph
;  2798          2917  1 !           *** is:
;  2799          2918  1 !              B -> W -> L -> F -> D -> H
;  2800          2919  1 !                              \    \   /
;  2801          2920  1 !                               FC -> DC
;  2802          2921  1 !
;  2803          2922  1 !           That is, things become non-linear when complex is involved.
;  2804          2923  1 !           For example, if a real*16 is added to a complex*8 then
;  2805          2924  1 !           both are converted up to complex*16
;  2806          2925  1 !
;  2807          2926  1 !           This second diagram is our hierarchy table.
;  2808          2927  1 !
;  2809          2928  1 !           If the user specified /G_FLOAT then the diagram is the same
;  2810          2929  1 !           except that G is substituted for D.
;  2811          2930  1 !
;  2812          2931  1 !         3. An operation involving a complex*8 and a real*16
;  2813          2932  1 !            produces complex*16 result.
;  2814          2933  1 !
;  2815          2934  1 !         4. Integer operations are performed on integers. (Logical is treated
;  2816          2935  1 !            as integers)
;  2817          2936  1 !
;  2818          2937  1 !         5. Real operations are perfomred only on reals or combinations of
;  2819          2938  1 !            real, integer, and logcal.  Integer --> real before the operation.
;  2820          2939  1 !
;  2821          2940  1 !         6. Real*8, Real*16 operations.  Elements --> higher-precision (see 2)
;  2822          2941  1 !
;  2823          2942  1 !         7. Complex operations.  (see 2, 4, 5, 6, and 3)
;  2824          2943  1 !
;  2825          2944  1 !       .Relational
;  2826          2945  1 !         1. Complex can be related only by the .EQ. and .NE.
;  2827          2946  1 !         2. Lower-ranked data type --> higher-ranked data type before the
;  2828          2947  1 !            the comparison is made.
;  2829          2948  1 !
;  2830          2949  1 !       .Logical
;  2831          2950  1 !         1. Logical operator operates on logical, the result is logical.
;  2832          2951  1 !         2. Logical operator operates on integer, the operation is carried
;  2833          2952  1 !            out bit-by-bit on the corresponding bits of the internal of the
;  2834          2953  1 !            integer, the result is integer.
;  2835          2954  1 !         3. Logical operator operates on combined logical, integer, logical
;  2836          2955  1 !            --> integer, then (see 2).
;  2837          2956  1 !
;  2838          2957  1
;  2839          2958  1
;  2840          2959  1 ! Define the Type Mapping Table for FORTRAN
;  2841          2960  1 ! In FORTRAN, data that was declared as logical*1, logical*2, or logical*4
;  2842          2961  1 ! appears in the DST as BU, WU, and LU. However, for our purposes, we
;  2843          2962  1 ! want to treat it as B, W, or L. We thus "map" the dtypes.
;  2844          2963  1 !
;  2845          2964  1 ! For example, if BU contains -1, and L contains 1, then we
;  2846          2965  1 ! want EV BU+L to give 0, not 256.
;  2847          2966  1 ! The type mapping table is thus needed to turn BU->B, WU->W, and LU->L
;  2848          2967  1 !
;  2849        P 2968  1 TYPE_MAPPING_TABLE (FORTRAN_MAP_TABLE,
;  2850        P 2969  1     TYPE_GRAPH_EDGE (BU, B),
;  2851        P 2970  1     TYPE_GRAPH_EDGE (WU, W),
;  2852        P 2971  1     TYPE_GRAPH_EDGE (LU, L),
;  2853          2972  1     0);
```

```
;  2854          2973  1
;  2855          2974  1
;  2856          2975  1 ! Define the Type Conversion Information Table for FORTRAN.
;  2857          2976  1 ! This points to the mapping table define above. There is no CVT_TABLE
;  2858          2977  1 ! specifying exceptions to the DBG$CVT_DX_DX rules.
;  2859          2978  1 !
;  2860        P 2979  1 CONVERSION_INFO_TABLE (FORTRAN_CVTINFO_TABLE,
;  2861          2980  1     CONVERSION_INFO_ENTRY (FORTRAN_MAP_TABLE, TABLEBASE));
;  2862          2981  1
;  2863          2982  1
;  2864          2983  1 ! Define the Type Hierarchy Table for FORTRAN.
;  2865          2984  1 ! This table is described above when we talk about conversion rules.
;  2866          2985  1 ! Note that the Incompatibility Table prevents mixing D with G types.
;  2867          2986  1 ! Leaving out the G edges, the graph specified by this table is:
;  2868          2987  1 !
;  2869          2988  1 !                 B -> W -> L -> F -> D -> H
;  2870          2989  1 !                                \    \   /
;  2871          2990  1 !                                 FC -> DC
;  2872          2991  1 !
;  2873        P 2992  1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER1_TABLE,
;  2874        P 2993  1     TYPE_GRAPH_EDGE (B, W),
;  2875        P 2994  1     TYPE_GRAPH_EDGE (W, L),
;  2876        P 2995  1     TYPE_GRAPH_EDGE (L, F),
;  2877        P 2996  1     TYPE_GRAPH_EDGE (F, D),
;  2878        P 2997  1     TYPE_GRAPH_EDGE (F, G),
;  2879        P 2998  1     TYPE_GRAPH_EDGE (F, FC),
;  2880        P 2999  1     TYPE_GRAPH_EDGE (D, H),
;  2881        P 3000  1     TYPE_GRAPH_EDGE (D, DC),
;  2882        P 3001  1     TYPE_GRAPH_EDGE (G, H),
;  2883        P 3002  1     TYPE_GRAPH_EDGE (G, GC),
;  2884        P 3003  1     TYPE_GRAPH_EDGE (H, DC),
;  2885        P 3004  1     TYPE_GRAPH_EDGE (H, GC),
;  2886        P 3005  1     TYPE_GRAPH_EDGE (FC, DC),
;  2887        P 3006  1     TYPE_GRAPH_EDGE (FC, GC),
;  2888          3007  1     0);
;  2889          3008  1
;  2890          3009  1
;  2891          3010  1 ! Define a Type Hierarchy Table for FORTRAN
;  2892          3011  1 ! This is a subset of the HIER1 table, which is used for operators that
;  2893          3012  1 ! do not accept complex types. It would also be OK to use the HIER1 table
;  2894          3013  1 ! for those operators, but giving a smaller table makes the code run
;  2895          3014  1 ! faster.
;  2896          3015  1 !
;  2897        P 3016  1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER2_TABLE,
;  2898        P 3017  1     TYPE_GRAPH_EDGE (B, W),
;  2899        P 3018  1     TYPE_GRAPH_EDGE (W, L),
;  2900        P 3019  1     TYPE_GRAPH_EDGE (L, F),
;  2901        P 3020  1     TYPE_GRAPH_EDGE (F, D),
;  2902        P 3021  1     TYPE_GRAPH_EDGE (F, G),
;  2903        P 3022  1     TYPE_GRAPH_EDGE (D, H),
;  2904        P 3023  1     TYPE_GRAPH_EDGE (G, H),
;  2905          3024  1     0);
;  2906          3025  1
;  2907          3026  1
;  2908          3027  1 ! Define a Type Hierarchy Table for FORTRAN
;  2909          3028  1 ! This is a subset of the HIER2 and HIER1 tables. It is used for those
;  2910          3029  1 ! operators that only accept integer types. It would also be OK to
```

```
; 2911          3030  1 ! use the HIER1 table but providing a smaller table speeds up the code.
; 2912          3031  1 !
; 2913        P 3032  1 TYPE_HIERARCHY_TABLE (FORTRAN_HIER3_TABLE,
; 2914        P 3033  1     TYPE_GRAPH_EDGE (B, W),
; 2915        P 3034  1     TYPE_GRAPH_EDGE (W, L),
; 2916          3035  1     0);
; 2917          3036  1
; 2918          3037  1 ! Define the Type Hierarchy Table for FORTRAN deposit.
; 2919          3038  1 ! This is a circular table which includes all types except T.
; 2920          3039  1 ! This means that any of the numeric types are convertible to any
; 2921          3040  1 ! of the other numeric types on a DEPOSIT.
; 2922          3041  1 !
; 2923        P 3042  1 TYPE_HIERARCHY_TABLE (FORTRAN_HIERD_TABLE,
; 2924        P 3043  1     TYPE_GRAPH_EDGE (B, W),
; 2925        P 3044  1     TYPE_GRAPH_EDGE (W, L),
; 2926        P 3045  1     TYPE_GRAPH_EDGE (L, F),
; 2927        P 3046  1     TYPE_GRAPH_EDGE (F, D),
; 2928        P 3047  1     TYPE_GRAPH_EDGE (D, G),
; 2929        P 3048  1     TYPE_GRAPH_EDGE (G, H),
; 2930        P 3049  1     TYPE_GRAPH_EDGE (H, FC),
; 2931        P 3050  1     TYPE_GRAPH_EDGE (FC, DC),
; 2932        P 3051  1     TYPE_GRAPH_EDGE (DC, GC),
; 2933        P 3052  1     TYPE_GRAPH_EDGE (GC, B),
; 2934          3053  1     0);
; 2935          3054  1
; 2936          3055  1 ! Define the Type Incompatibility Table for FORTRAN.
; 2937          3056  1 ! This prevents the user from mixing D and G types in an expression.
; 2938          3057  1 !
; 2939        P 3058  1 TYPE_INCOMP_TABLE (FORTRAN_INCOMP_TABLE,
; 2940        P 3059  1     TYPE_GRAPH_EDGE (D, G),
; 2941        P 3060  1     TYPE_GRAPH_EDGE (D, GC),
; 2942        P 3061  1     TYPE_GRAPH_EDGE (G, DC),
; 2943        P 3062  1     TYPE_GRAPH_EDGE (DC, GC),
; 2944          3063  1     0);
; 2945          3064  1
; 2946          3065  1 !++
; 2947          3066  1 ! Most of the arithmetic routines operate on two arguments of the same type.
; 2948          3067  1 ! That type may be B, W, L, F, D, G, H, FC, DC, GC, so we provide all
; 2949          3068  1 ! of those case indices.
; 2950          3069  1 !--
; 2951          3070  1
; 2952          3071  1 ! Define the Operator Routine Table for FORTRAN addition.
; 2953          3072  1 !
; 2954        P 3073  1 OPERATOR_ROUTINE_TABLE (FORTRAN_ADD_TABLE,
; 2955        P 3074  1     OPERATOR_ROUTINE (B, B, B, ADD_B_B),
; 2956        P 3075  1     OPERATOR_ROUTINE (W, W, W, ADD_W_W),
; 2957        P 3076  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
; 2958        P 3077  1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
; 2959        P 3078  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
; 2960        P 3079  1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
; 2961        P 3080  1     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
; 2962        P 3081  1     OPERATOR_ROUTINE (FC, FC, FC, ADD_FC_FC),
; 2963        P 3082  1     OPERATOR_ROUTINE (DC, DC, DC, ADD_DC_DC),
; 2964          3083  1     OPERATOR_ROUTINE (GC, GC, GC, ADD_GC_GC));
; 2965          3084  1
; 2966          3085  1
; 2967          3086  1 ! Define the Operator Routine Table for FORTRAN subtraction.
```

```
;  2968          3087  1 !
;  2969        P 3088  1 OPERATOR_ROUTINE_TABLE (FORTRAN_SUB_TABLE,
;  2970        P 3089  1     OPERATOR_ROUTINE (B, B, B, SUB_B_B),
;  2971        P 3090  1     OPERATOR_ROUTINE (W, W, W, SUB_W_W),
;  2972        P 3091  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  2973        P 3092  1     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  2974        P 3093  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  2975        P 3094  1     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  2976        P 3095  1     OPERATOR_ROUTINE (H, H, H, SUB_H_H),
;  2977        P 3096  1     OPERATOR_ROUTINE (FC, FC, FC, SUB_FC_FC),
;  2978        P 3097  1     OPERATOR_ROUTINE (DC, DC, DC, SUB_DC_DC),
;  2979          3098  1     OPERATOR_ROUTINE (GC, GC, GC, SUB_GC_GC));
;  2980          3099  1
;  2981          3100  1
;  2982          3101  1 ! Define the Operator Routine Table for FORTRAN multiplication.
;  2983          3102  1 !
;  2984        P 3103  1 OPERATOR_ROUTINE_TABLE (FORTRAN_MUL_TABLE,
;  2985        P 3104  1     OPERATOR_ROUTINE (B, B, B, MUL_B_B),
;  2986        P 3105  1     OPERATOR_ROUTINE (W, W, W, MUL_W_W),
;  2987        P 3106  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;  2988        P 3107  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  2989        P 3108  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;  2990        P 3109  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;  2991        P 3110  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H),
;  2992        P 3111  1     OPERATOR_ROUTINE (FC, FC, FC, MUL_FC_FC),
;  2993        P 3112  1     OPERATOR_ROUTINE (DC, DC, DC, MUL_DC_DC),
;  2994          3113  1     OPERATOR_ROUTINE (GC, GC, GC, MUL_GC_GC));
;  2995          3114  1
;  2996          3115  1
;  2997          3116  1 ! Define the Operator Routine Table for FORTRAN division.
;  2998          3117  1 !
;  2999        P 3118  1 OPERATOR_ROUTINE_TABLE (FORTRAN_DIV_TABLE,
;  3000        P 3119  1     OPERATOR_ROUTINE (B, B, B, DIV_B_B),
;  3001        P 3120  1     OPERATOR_ROUTINE (W, W, W, DIV_W_W),
;  3002        P 3121  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;  3003        P 3122  1     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
;  3004        P 3123  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
;  3005        P 3124  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;  3006        P 3125  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H),
;  3007        P 3126  1     OPERATOR_ROUTINE (FC, FC, FC, DIV_FC_FC),
;  3008        P 3127  1     OPERATOR_ROUTINE (DC, DC, DC, DIV_DC_DC),
;  3009          3128  1     OPERATOR_ROUTINE (GC, GC, GC, DIV_GC_GC));
;  3010          3129  1
;  3011          3130  1
;  3012          3131  1 ! Define the Operator Routine Table for FORTRAN unary plus.
;  3013          3132  1 !
;  3014        P 3133  1 OPERATOR_ROUTINE_TABLE (FORTRAN_UNARY_PLUS_TABLE,
;  3015        P 3134  1
;  3016        P 3135  1 ! The following are not language dependent types.  This is needed for DEBUG
;  3017        P 3136  1 ! types.  For example, DEP/QUAD L= +1.
;  3018        P 3137  1 !
;  3019        P 3138  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  3020        P 3139  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  3021        P 3140  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  3022        P 3141  1
;  3023        P 3142  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  3024        P 3143  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
```

```
;  3025        P 3144  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  3026        P 3145  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  3027        P 3146  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  3028        P 3147  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  3029        P 3148  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  3030        P 3149  1     OPERATOR_ROUTINE (FC, FC, FC, UNARY_PLUS_FC),
;  3031        P 3150  1     OPERATOR_ROUTINE (DC, DC, DC, UNARY_PLUS_DC),
;  3032          3151  1     OPERATOR_ROUTINE (GC, GC, GC, UNARY_PLUS_GC));
;  3033          3152  1
;  3034          3153  1
;  3035          3154  1 ! Define the Operator Routine Table for FORTRAN unary minus.
;  3036          3155  1 !
;  3037        P 3156  1 OPERATOR_ROUTINE_TABLE (FORTRAN_UNARY_MINUS_TABLE,
;  3038        P 3157  1
;  3039        P 3158  1 ! The following are not language dependent types.  This is needed for DEBUG
;  3040        P 3159  1 ! types.  For example, DEP/QUAD L= +1.
;  3041        P 3160  1 !
;  3042        P 3161  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  3043        P 3162  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  3044        P 3163  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  3045        P 3164  1
;  3046        P 3165  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  3047        P 3166  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  3048        P 3167  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  3049        P 3168  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  3050        P 3169  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  3051        P 3170  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  3052        P 3171  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  3053        P 3172  1     OPERATOR_ROUTINE (FC, FC, FC, UNARY_MINUS_FC),
;  3054        P 3173  1     OPERATOR_ROUTINE (DC, DC, DC, UNARY_MINUS_DC),
;  3055          3174  1     OPERATOR_ROUTINE (GC, GC, GC, UNARY_MINUS_GC));
;  3056          3175  1
;  3057          3176  1
;  3058          3177  1 ! Define the Operator Routine Table for FORTRAN exponentiation.
;  3059          3178  1 ! Exponentiation has some mixed forms. For example, if you raise a
;  3060          3179  1 ! floating number to an integer power, you do not necessarily
;  3061          3180  1 ! want to first convert the int to float. Instead, a special
;  3062          3181  1 ! routine indices such as  POWER_F_L are provided to do the right thing here.
;  3063          3182  1 !
;  3064        P 3183  1 OPERATOR_ROUTINE_TABLE (FORTRAN_POWER_TABLE,
;  3065        P 3184  1     OPERATOR_ROUTINE (W, W, W, POWER_W_W),
;  3066        P 3185  1     OPERATOR_ROUTINE (L, L, L, POWER_L_L),
;  3067        P 3186  1     OPERATOR_ROUTINE (F, L, F, POWER_F_L),
;  3068        P 3187  1     OPERATOR_ROUTINE (D, L, D, POWER_D_L),
;  3069        P 3188  1     OPERATOR_ROUTINE (G, L, G, POWER_G_L),
;  3070        P 3189  1     OPERATOR_ROUTINE (H, L, H, POWER_H_L),
;  3071        P 3190  1     OPERATOR_ROUTINE (FC, L, FC, POWER_FC_L),
;  3072        P 3191  1     OPERATOR_ROUTINE (DC, L, DC, POWER_DC_L),
;  3073        P 3192  1     OPERATOR_ROUTINE (GC, L, GC, POWER_GC_L),
;  3074        P 3193  1     OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;  3075        P 3194  1     OPERATOR_ROUTINE (D, F, D, POWER_D_F),
;  3076        P 3195  1     OPERATOR_ROUTINE (F, D, D, POWER_F_D),
;  3077        P 3196  1     OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;  3078        P 3197  1     OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;  3079        P 3198  1     OPERATOR_ROUTINE (H, H, H, POWER_H_H),
;  3080        P 3199  1     OPERATOR_ROUTINE (FC, FC, FC, POWER_FC_FC),
;  3081        P 3200  1     OPERATOR_ROUTINE (DC, DC, DC, POWER_DC_DC),
```

```
;  3082          3201  1     OPERATOR_ROUTINE (GC, GC, GC, POWER_GC_GC));
;  3083          3202  1
;  3084          3203  1
;  3085          3204  1 ! Define the Operator Routine Table for FORTRAN concatenate.
;  3086          3205  1 ! Concatenate can only be done on strings.
;  3087          3206  1 !
;  3088        P 3207  1 OPERATOR_ROUTINE_TABLE (FORTRAN_CONCAT_TABLE,
;  3089          3208  1     OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
;  3090          3209  1
;  3091          3210  1
;  3092          3211  1 ! Define the Operator Routine Table for FORTRAN equal.
;  3093          3212  1 ! This operator can be done on strings as well as all the numeric
;  3094          3213  1 ! types.
;  3095          3214  1 !
;  3096        P 3215  1 OPERATOR_ROUTINE_TABLE (FORTRAN_EQL_TABLE,
;  3097        P 3216  1     OPERATOR_ROUTINE (T, T, L, EQL_T_T),
;  3098        P 3217  1     OPERATOR_ROUTINE (B, B, L, EQL_B_B),
;  3099        P 3218  1     OPERATOR_ROUTINE (W, W, L, EQL_W_W),
;  3100        P 3219  1     OPERATOR_ROUTINE (L, L, L, EQL_L_L),
;  3101        P 3220  1     OPERATOR_ROUTINE (F, F, L, EQL_F_F),
;  3102        P 3221  1     OPERATOR_ROUTINE (D, D, L, EQL_D_D),
;  3103        P 3222  1     OPERATOR_ROUTINE (G, G, L, EQL_G_G),
;  3104        P 3223  1     OPERATOR_ROUTINE (H, H, L, EQL_H_H),
;  3105        P 3224  1     OPERATOR_ROUTINE (FC, FC, L, EQL_FC_FC),
;  3106        P 3225  1     OPERATOR_ROUTINE (DC, DC, L, EQL_DC_DC),
;  3107          3226  1     OPERATOR_ROUTINE (GC, GC, L, EQL_GC_GC));
;  3108          3227  1
;  3109          3228  1
;  3110          3229  1 ! Define the Operator Routine Table for FORTRAN not equal.
;  3111          3230  1 ! This operator can be done on strings as well as all the numeric types.
;  3112          3231  1 !
;  3113        P 3232  1 OPERATOR_ROUTINE_TABLE (FORTRAN_NEQ_TABLE,
;  3114        P 3233  1     OPERATOR_ROUTINE (T, T, L, NEQ_T_T),
;  3115        P 3234  1     OPERATOR_ROUTINE (B, B, L, NEQ_B_B),
;  3116        P 3235  1     OPERATOR_ROUTINE (W, W, L, NEQ_W_W),
;  3117        P 3236  1     OPERATOR_ROUTINE (L, L, L, NEQ_L_L),
;  3118        P 3237  1     OPERATOR_ROUTINE (F, F, L, NEQ_F_F),
;  3119        P 3238  1     OPERATOR_ROUTINE (D, D, L, NEQ_D_D),
;  3120        P 3239  1     OPERATOR_ROUTINE (G, G, L, NEQ_G_G),
;  3121        P 3240  1     OPERATOR_ROUTINE (H, H, L, NEQ_H_H),
;  3122        P 3241  1     OPERATOR_ROUTINE (FC, FC, L, NEQ_FC_FC),
;  3123        P 3242  1     OPERATOR_ROUTINE (DC, DC, L, NEQ_DC_DC),
;  3124          3243  1     OPERATOR_ROUTINE (GC, GC, L, NEQ_GC_GC));
;  3125          3244  1
;  3126          3245  1 ! In the tables for the comparison operators,
;  3127          3246  1 ! we allow strings to be compared, and also all the numeric types except
;  3128          3247  1 ! for the complex floats.
;  3129          3248  1
;  3130          3249  1 ! Define the Operator Routine Table for FORTRAN greater than.
;  3131          3250  1 !
;  3132        P 3251  1 OPERATOR_ROUTINE_TABLE (FORTRAN_GTR_TABLE,
;  3133        P 3252  1     OPERATOR_ROUTINE (T, T, L, GTR_T_T),
;  3134        P 3253  1     OPERATOR_ROUTINE (B, B, L, GTR_B_B),
;  3135        P 3254  1     OPERATOR_ROUTINE (W, W, L, GTR_W_W),
;  3136        P 3255  1     OPERATOR_ROUTINE (L, L, L, GTR_L_L),
;  3137        P 3256  1     OPERATOR_ROUTINE (F, F, L, GTR_F_F),
;  3138        P 3257  1     OPERATOR_ROUTINE (D, D, L, GTR_D_D),
```

```
;  3139        P 3258  1     OPERATOR_ROUTINE (G, G, L, GTR_G_G),
;  3140          3259  1     OPERATOR_ROUTINE (H, H, L, GTR_H_H));
;  3141          3260  1
;  3142          3261  1
;  3143          3262  1 ! Define the Operator Routine Table for FORTRAN greater than or equal to.
;  3144          3263  1 !
;  3145        P 3264  1 OPERATOR_ROUTINE_TABLE (FORTRAN_GEQ_TABLE,
;  3146        P 3265  1     OPERATOR_ROUTINE (T, T, L, GEQ_T_T),
;  3147        P 3266  1     OPERATOR_ROUTINE (B, B, L, GEQ_B_B),
;  3148        P 3267  1     OPERATOR_ROUTINE (W, W, L, GEQ_W_W),
;  3149        P 3268  1     OPERATOR_ROUTINE (L, L, L, GEQ_L_L),
;  3150        P 3269  1     OPERATOR_ROUTINE (F, F, L, GEQ_F_F),
;  3151        P 3270  1     OPERATOR_ROUTINE (D, D, L, GEQ_D_D),
;  3152        P 3271  1     OPERATOR_ROUTINE (G, G, L, GEQ_G_G),
;  3153          3272  1     OPERATOR_ROUTINE (H, H, L, GEQ_H_H));
;  3154          3273  1
;  3155          3274  1
;  3156          3275  1 ! Define the Operator Routine Table for FORTRAN less than.
;  3157          3276  1 !
;  3158        P 3277  1 OPERATOR_ROUTINE_TABLE (FORTRAN_LSS_TABLE,
;  3159        P 3278          OPERATOR_ROUTINE (T, T, L, LSS_T_T),
;  3160        P 3279  1       OPERATOR_ROUTINE (B, B, L, LSS_B_B),
;  3161        P 3280  1       OPERATOR_ROUTINE (W, W, L, LSS_W_W),
;  3162        P 3281  1       OPERATOR_ROUTINE (L, L, L, LSS_L_L),
;  3163        P 3282  1       OPERATOR_ROUTINE (F, F, L, LSS_F_F),
;  3164        P 3283  1       OPERATOR_ROUTINE (D, D, L, LSS_D_D),
;  3165        P 3284  1       OPERATOR_ROUTINE (G, G, L, LSS_G_G),
;  3166          3285  1       OPERATOR_ROUTINE (H, H, L, LSS_H_H));
;  3167          3286  1
;  3168          3287  1
;  3169          3288  1 ! Define the Operator Routine Table for FORTRAN less than or equal to.
;  3170          3289  1 !
;  3171        P 3290  1 OPERATOR_ROUTINE_TABLE (FORTRAN_LEQ_TABLE,
;  3172        P 3291  1       OPERATOR_ROUTINE (T, T, L, LEQ_T_T),
;  3173        P 3292  1       OPERATOR_ROUTINE (B, B, L, LEQ_B_B),
;  3174        P 3293  1       OPERATOR_ROUTINE (W, W, L, LEQ_W_W),
;  3175        P 3294  1       OPERATOR_ROUTINE (L, L, L, LEQ_L_L),
;  3176        P 3295  1       OPERATOR_ROUTINE (F, F, L, LEQ_F_F),
;  3177        P 3296  1       OPERATOR_ROUTINE (D, D, L, LEQ_D_D),
;  3178        P 3297  1       OPERATOR_ROUTINE (G, G, L, LEQ_G_G),
;  3179          3298  1       OPERATOR_ROUTINE (H, H, L, LEQ_H_H));
;  3180          3299  1
;  3181          3300  1
;  3182          3301  1 ! The logical operators .AND., .OR., .EQV., .NEQV., .NOT. can be applied
;  3183          3302  1 ! only to integer data types.
;  3184          3303  1
;  3185          3304  1 ! Define the Operator Routine Table for FORTRAN not.
;  3186          3305  1 !
;  3187        P 3306  1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_NOT_TABLE,
;  3188        P 3307  1       OPERATOR_ROUTINE (B, B, B, BIT_NOT_B),
;  3189        P 3308  1       OPERATOR_ROUTINE (W, W, W, BIT_NOT_W),
;  3190          3309  1       OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
;  3191          3310  1
;  3192          3311  1
;  3193          3312  1 ! Define the Operator Routine Table for FORTRAN and.
;  3194          3313  1 !
;  3195        P 3314  1 OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_AND_TABLE,
```

```
;  3196        P 3315  1      OPERATOR_ROUTINE (B, B, B, BIT_AND_B_B),
;  3197        P 3316  1      OPERATOR_ROUTINE (W, W, W, BIT_AND_W_W),
;  3198          3317  1      OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
;  3199          3318  1
;  3200          3319  1
;  3201          3320  1  ! Define the Operator Routine Table for FORTRAN or.
;  3202          3321  1  !
;  3203        P 3322  1  OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_OR_TABLE,
;  3204        P 3323  1      OPERATOR_ROUTINE (B, B, B, BIT_OR_B_B),
;  3205        P 3324  1      OPERATOR_ROUTINE (W, W, W, BIT_OR_W_W),
;  3206          3325  1      OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
;  3207          3326  1
;  3208          3327  1
;  3209          3328  1  ! Define the Operator Routine Table for FORTRAN xor, neqv
;  3210          3329  1  !
;  3211        P 3330  1  OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_XOR_TABLE,
;  3212        P 3331  1      OPERATOR_ROUTINE (B, B, B, BIT_XOR_B_B),
;  3213        P 3332  1      OPERATOR_ROUTINE (W, W, W, BIT_XOR_W_W),
;  3214          3333  1      OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
;  3215          3334  1
;  3216          3335  1
;  3217          3336  1  ! Define the Operator Routine Table for FORTRAN eqv.
;  3218          3337  1  !
;  3219        P 3338  1  OPERATOR_ROUTINE_TABLE (FORTRAN_BIT_EQV_TABLE,
;  3220        P 3339  1      OPERATOR_ROUTINE (B, B, B, BIT_EQV_B_B),
;  3221        P 3340  1      OPERATOR_ROUTINE (W, W, W, BIT_EQV_W_W),
;  3222          3341  1      OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
;  3223          3342  1
;  3224          3343  1
;  3225          3344  1  ! Define the Operator Information Table for FORTRAN.
;  3226          3345  1  !
;  3227        P 3346  1  OPERATOR_INFO_TABLE (FORTRAN_OPINFO_TABLE,
;  3228        P 3347  1
;  3229        P 3348  1      ! The following are arithmetic tables that accept all numeric data types,
;  3230        P 3349  1      ! including complex. They thus go through the larger HIER1 table, and
;  3231        P 3350  1      ! need to specify an incompatibility table.
;  3232        P 3351  1      !
;  3233        P 3352  1      OPERATOR_INFO_ENTRY (ADD, FORTRAN_ADD_TABLE, FORTRAN_HIER1_TABLE,
;  3234        P 3353  1          FORTRAN_INCOMP_TABLE),
;  3235        P 3354  1      OPERATOR_INFO_ENTRY (SUBTRACT, FORTRAN_SUB_TABLE, FORTRAN_HIER1_TABLE,
;  3236        P 3355  1          FORTRAN_INCOMP_TABLE),
;  3237        P 3356  1      OPERATOR_INFO_ENTRY (MULTIPLY, FORTRAN_MUL_TABLE, FORTRAN_HIER1_TABLE,
;  3238        P 3357  1          FORTRAN_INCOMP_TABLE),
;  3239        P 3358  1      OPERATOR_INFO_ENTRY (DIVIDE, FORTRAN_DIV_TABLE, FORTRAN_HIER1_TABLE,
;  3240        P 3359  1          FORTRAN_INCOMP_TABLE),
;  3241        P 3360  1      OPERATOR_INFO_ENTRY (UNARY_PLUS, FORTRAN_UNARY_PLUS_TABLE,
;  3242        P 3361  1          FORTRAN_HIER1_TABLE, FORTRAN_INCOMP_TABLE),
;  3243        P 3362  1      OPERATOR_INFO_ENTRY (UNARY_MINUS, FORTRAN_UNARY_MINUS_TABLE,
;  3244        P 3363  1          FORTRAN_HIER1_TABLE, FORTRAN_INCOMP_TABLE),
;  3245        P 3364  1      OPERATOR_INFO_ENTRY (POWER_OF, FORTRAN_POWER_TABLE, FORTRAN_HIER1_TABLE,
;  3246        P 3365  1          FORTRAN_INCOMP_TABLE),
;  3247        P 3366  1
;  3248        P 3367  1      ! Equal/Notequal can accept all numeric types including complex and
;  3249        P 3368  1      ! thus also need the larger hierarchy table and the incompatibility
;  3250        P 3369  1      ! table.
;  3251        P 3370  1      !
;  3252        P 3371  1      OPERATOR_INFO_ENTRY (EQUAL, FORTRAN_EQL_TABLE, FORTRAN_HIER1_TABLE,
```

```
;  3253        P 3372  1          FORTRAN_INCOMP_TABLE),
;  3254        P 3373  1      OPERATOR_INFO_ENTRY (NOT_EQUAL, FORTRAN_NEQ_TABLE, FORTRAN_HIER1_TABLE,
;  3255        P 3374  1          FORTRAN_INCOMP_TABLE),
;  3256        P 3375  1
;  3257        P 3376  1      ! The other relationals do not accept complex so they can get by with
;  3258        P 3377  1      ! the smaller hierarachy table.
;  3259        P 3378  1      !
;  3260        P 3379  1      OPERATOR_INFO_ENTRY (GTR_THAN, FORTRAN_GTR_TABLE, FORTRAN_HIER2_TABLE,
;  3261        P 3380  1          FORTRAN_INCOMP_TABLE),
;  3262        P 3381  1      OPERATOR_INFO_ENTRY (GTR_EQUAL, FORTRAN_GEQ_TABLE, FORTRAN_HIER2_TABLE,
;  3263        P 3382  1          FORTRAN_INCOMP_TABLE),
;  3264        P 3383  1      OPERATOR_INFO_ENTRY (LSS_THAN, FORTRAN_LSS_TABLE, FORTRAN_HIER2_TABLE,
;  3265        P 3384  1          FORTRAN_INCOMP_TABLE),
;  3266        P 3385  1      OPERATOR_INFO_ENTRY (LSS_EQUAL, FORTRAN_LEQ_TABLE, FORTRAN_HIER2_TABLE,
;  3267        P 3386  1          FORTRAN_INCOMP_TABLE),
;  3268        P 3387  1
;  3269        P 3388  1      ! The logical operators accept only integer quantities so they can
;  3270        P 3389  1      ! use the smallest hierarchy table. They also do not need an
;  3271        P 3390  1      ! incompatibility table.
;  3272        P 3391  1      !
;  3273        P 3392  1      OPERATOR_INFO_ENTRY (NOT, FORTRAN_BIT_NOT_TABLE, FORTRAN_HIER3_TABLE,
;  3274        P 3393  1          TABLEBASE),
;  3275        P 3394  1      OPERATOR_INFO_ENTRY (AND, FORTRAN_BIT_AND_TABLE, FORTRAN_HIER3_TABLE,
;  3276        P 3395  1          TABLEBASE),
;  3277        P 3396  1      OPERATOR_INFO_ENTRY (OR, FORTRAN_BIT_OR_TABLE, FORTRAN_HIER3_TABLE,
;  3278        P 3397  1          TABLEBASE),
;  3279        P 3398  1      OPERATOR_INFO_ENTRY (XOR, FORTRAN_BIT_XOR_TABLE, FORTRAN_HIER3_TABLE,
;  3280        P 3399  1          TABLEBASE),
;  3281        P 3400  1      OPERATOR_INFO_ENTRY (EQV, FORTRAN_BIT_EQV_TABLE, FORTRAN_HIER3_TABLE,
;  3282        P 3401  1          TABLEBASE),
;  3283        P 3402  1
;  3284        P 3403  1      ! Concatenate accepts only string types and there are no conversions
;  3285        P 3404  1      ! so we do not need a hierarchy table or an incompatibility table.
;  3286        P 3405  1      !
;  3287        P 3406  1      OPERATOR_INFO_ENTRY (CONCATENATE, FORTRAN_CONCAT_TABLE, TABLEBASE,
;  3288        P 3407  1          TABLEBASE),
;  3289        P 3408  1
;  3290        P 3409  1      ! The CONVERT operator gets called to convert subscripts to integer
;  3291        P 3410  1      ! type and to convert expressions in FOR, IF, WHILE, REPEAT statements
;  3292        P 3411  1      ! to integer type. It can thus use the smaller HIER3 table to specify
;  3293        P 3412  1      ! the rules for conversion to integer.
;  3294        P 3413  1      !
;  3295        P 3414  1      OPERATOR_INFO_ENTRY (CONVERT, TABLEBASE, FORTRAN_HIER3_TABLE,
;  3296        P 3415  1          TABLEBASE),
;  3297        P 3416  1
;  3298        P 3417  1      ! The DEPOSIT operator gets called on the DEPOSIT command. It has
;  3299        P 3418  1      ! its own hierarchy table which allows any numeric type to be
;  3300        P 3419  1      ! converted to any other numeric type. The incompatibility table,
;  3301        P 3420  1      ! however, still prevents depositing D types into G types and
;  3302        P 3421  1      ! vice versa.
;  3303        P 3422  1      !
;  3304        P 3423  1      OPERATOR_INFO_ENTRY (DEPOSIT, TABLEBASE, FORTRAN_HIERD_TABLE,
;  3305        P 3424  1          FORTRAN_INCOMP_TABLE),
;  3306        P 3425  1
;  3307        P 3426  1
;  3308        P 3427  1      ! The IDENTITY operator gets called at the end of an EVALUATE command
;  3309        P 3428  1      ! to apply the PRIM_TO_VAL routine and then apply the appropriate
```

```
;  3310        P 3429  1     ! type mappings. This will ensure that EV BU will print as a signed integer,
;  3311        P 3430  1     ! for example.
;  3312        P 3431  1     !
;  3313        P 3432  1     ! The identity operator does not require any tables.
;  3314        P 3433  1     !
;  3315        P 3434  1     OPERATOR_INFO_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)
;  3316          3435  1     );
;  3317          3436  1
```

```
;  3319          3437  1 !
;  3320          3438  1 !       M A C R O   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  3321          3439  1 !
;  3322          3440  1 !
;  3323          3441  1 ! This section contains the Operator Routine and Type tables needed to
;  3324          3442  1 ! evaluate expressions in the MACRO language.
;  3325          3443  1 !
;  3326          3444  1
;  3327          3445  1 ! Although MACRO does not contain run-time expressions in the language,
;  3328          3446  1 ! we provide a set of operators much the same as that provided for BLISS:
;  3329          3447  1 !
;  3330          3448  1 ! Arithmetic: + - * / @ MOD     (binary @ is the shift operator)
;  3331          3449  1 ! Relational: EQL NEQ LSS LEQ GTR GEQ LSSU LEQU GTRU GEQU
;  3332          3450  1 ! Logical: AND OR XOR EQV NOT
;  3333          3451  1 ! Bit selection: X<p,s,e>
;  3334          3452  1 ! Indirection: @
;  3335          3453  1
;  3336          3454  1 ! Define a Type Mapping Table for MACRO.
;  3337          3455  1 ! MACRO declares all its data to be labels, which means we get type ZI,
;  3338          3456  1 ! even though we really want to treat it as integer. To simplify things,
;  3339          3457  1 ! we just map ZI to integer here.
;  3340          3458  1 !
;  3341        P 3459  1 TYPE_MAPPING_TABLE (MACRO_MAP_TABLE,
;  3342        P 3460  1     TYPE_GRAPH_EDGE (ZI, L),
;  3343          3461  1     0);
;  3344          3462  1
;  3345          3463  1 ! Define the Type Conversion Information Table for MACRO.
;  3346          3464  1 ! We specify the Mapping Table defined above. There is no language
;  3347          3465  1 ! specific table for conversions - we just use the rules in DBG$CVT_DX_DX
;  3348          3466  1 !
;  3349        P 3467  1 CONVERSION_INFO_TABLE (MACRO_CVTINFO_TABLE,
;  3350          3468  1     CONVERSION_INFO_ENTRY (MACRO_MAP_TABLE, TABLEBASE));
;  3351          3469  1
;  3352          3470  1
;  3353          3471  1 ! Define the Type Hierarchy Table. All operations are done on signed longwords.
;  3354          3472  1 ! Thus we provide a path for all types to be converted to signed longwords.
;  3355          3473  1 ! Note - even though we do not get dtypes B, W, BU, WU, LU from the
;  3356          3474  1 ! MACRO DST, we provide conversion paths here, in case the user is debugging
;  3357          3475  1 ! a module produced by another language but with language set to MACRO.
;  3358          3476  1 !
;  3359        P 3477  1 TYPE_HIERARCHY_TABLE (MACRO_HIER_TABLE,
;  3360        P 3478  1     TYPE_GRAPH_EDGE (BU, WU),
;  3361        P 3479  1     TYPE_GRAPH_EDGE (WU, LU),
;  3362        P 3480  1     TYPE_GRAPH_EDGE (LU,  L),
;  3363        P 3481  1     TYPE_GRAPH_EDGE (B ,  W),
;  3364        P 3482  1     TYPE_GRAPH_EDGE (W ,  L),
;  3365          3483  1     0);
;  3366          3484  1
;  3367          3485  1 ! Define the Type Hierarchy Table for DEPOSIT.
;  3368          3486  1 ! This is a circular graph which includes all types that can be obtained
;  3369          3487  1 ! from calling PRIM_TO_VAL on a MACRO primary.  (Plus some other types as
;  3370          3488  1 ! well - see above note).
;  3371          3489  1 ! What this means is that
;  3372          3490  1 ! any type is convertible to any other type on a DEPOSIT.
;  3373          3491  1 !
;  3374        P 3492  1 TYPE_HIERARCHY_TABLE (MACRO_HIERD_TABLE,
;  3375        P 3493  1     TYPE_GRAPH_EDGE (B, BU),
```

```
;  3376        P 3494  1     TYPE_GRAPH_EDGE (BU, W),
;  3377        P 3495  1     TYPE_GRAPH_EDGE (W, WU),
;  3378        P 3496  1     TYPE_GRAPH_EDGE (WU, L),
;  3379        P 3497  1     TYPE_GRAPH_EDGE (L, LU),
;  3380        P 3498  1     TYPE_GRAPH_EDGE (LU, VU),
;  3381        P 3499  1     TYPE_GRAPH_EDGE (VU, SVU),
;  3382        P 3500  1     TYPE_GRAPH_EDGE (SVU, SV),
;  3383        P 3501  1     TYPE_GRAPH_EDGE (SV, V),
;  3384        P 3502  1     TYPE_GRAPH_EDGE (V, ZI),
;  3385        P 3503  1     TYPE_GRAPH_EDGE (ZI, ZEM),
;  3386        P 3504  1     TYPE_GRAPH_EDGE (ZEM, B),
;  3387          3505  1     0);
;  3388          3506  1
;  3389          3507  1 ! All of the arithmetic, relational, and logical
;  3390          3508  1 ! operators just operate on longword quantities.
;  3391          3509  1
;  3392          3510  1 ! Define the Operator Routine Table for MACRO addition.
;  3393          3511  1 !
;  3394        P 3512  1 OPERATOR_ROUTINE_TABLE (MACRO_ADD_TABLE,
;  3395          3513  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L));
;  3396          3514  1
;  3397          3515  1 ! Define the Operator Routine Table for MACRO subtraction.
;  3398          3516  1 !
;  3399        P 3517  1 OPERATOR_ROUTINE_TABLE(MACRO_SUB_TABLE,
;  3400          3518  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L));
;  3401          3519  1
;  3402          3520  1 ! Define the Operator Routine Table for MACRO multiplication.
;  3403          3521  1 !
;  3404        P 3522  1 OPERATOR_ROUTINE_TABLE (MACRO_MUL_TABLE,
;  3405          3523  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L));
;  3406          3524  1
;  3407          3525  1 ! Define the Operator Routine Table for MACRO Division.
;  3408          3526  1 !
;  3409        P 3527  1 OPERATOR_ROUTINE_TABLE (MACRO_DIV_TABLE,
;  3410          3528  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L));
;  3411          3529  1
;  3412          3530  1 ! Define the Operator Routine Table for MACRO arithmetic shift.
;  3413          3531  1 !
;  3414        P 3532  1 OPERATOR_ROUTINE_TABLE (MACRO_SHIFT_TABLE,
;  3415          3533  1     OPERATOR_ROUTINE (L, L, L, SHIFT_LEFT_L_L));
;  3416          3534  1
;  3417          3535  1 ! Define the Operator Routine Table for MACRO modulus.
;  3418          3536  1 !
;  3419        P 3537  1 OPERATOR_ROUTINE_TABLE (MACRO_MOD_TABLE,
;  3420          3538  1     OPERATOR_ROUTINE (L, L, L, MOD_L_L));
;  3421          3539  1
;  3422          3540  1 ! Define the Operator Routine Table for MACRO Unary Plus.
;  3423          3541  1 !
;  3424        P 3542  1 OPERATOR_ROUTINE_TABLE (MACRO_UNARY_PLUS_TABLE,
;  3425        P 3543  1
;  3426        P 3544  1 ! The following are not language dependent types.  This is needed for DEBUG
;  3427        P 3545  1 ! types.  For example, DEP/QUAD L= +1.
;  3428        P 3546  1 !
;  3429        P 3547  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  3430        P 3548  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  3431        P 3549  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  3432        P 3550  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
```

```
;  3433        P 3551  1      OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  3434        P 3552  1      OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  3435        P 3553  1      OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  3436        P 3554  1      OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  3437        P 3555  1      OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  3438        P 3556  1
;  3439          3557  1      OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L));
;  3440          3558  1
;  3441          3559  1  ! Define the Operator Routine Table for MACRO Unary Minus.
;  3442          3560  1  !
;  3443        P 3561  1  OPERATOR_ROUTINE_TABLE (MACRO_UNARY_MINUS_TABLE,
;  3444        P 3562  1
;  3445        P 3563  1  ! The following are not language dependent types.  This is needed for DEBUG
;  3446        P 3564  1  ! types.  For example, DEP/QUAD L= +1.
;  3447        P 3565  1  !
;  3448        P 3566  1      OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  3449        P 3567  1      OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  3450        P 3568  1      OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  3451        P 3569  1      OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  3452        P 3570  1      OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  3453        P 3571  1      OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  3454        P 3572  1      OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  3455        P 3573  1      OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  3456        P 3574  1      OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  3457        P 3575  1
;  3458          3576  1      OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L));
;  3459          3577  1
;  3460          3578  1  ! Define the Operator Routine Table for MACRO Equal.
;  3461          3579  1  !
;  3462        P 3580  1  OPERATOR_ROUTINE_TABLE (MACRO_EQUAL_TABLE,
;  3463          3581  1      OPERATOR_ROUTINE (L, L, L, EQL_L_L));
;  3464          3582  1
;  3465          3583  1  ! Define the Operator Routine Table for MACRO Not Equal.
;  3466          3584  1  !
;  3467        P 3585  1  OPERATOR_ROUTINE_TABLE (MACRO_NOT_EQUAL_TABLE,
;  3468          3586  1      OPERATOR_ROUTINE (L, L, L, NEQ_L_L));
;  3469          3587  1
;  3470          3588  1  ! Define the Operator Routine Table for MACRO Less Than.
;  3471          3589  1  !
;  3472        P 3590  1  OPERATOR_ROUTINE_TABLE (MACRO_LSS_THAN_TABLE,
;  3473          3591  1      OPERATOR_ROUTINE (L, L, L, LSS_L_L));
;  3474          3592  1
;  3475          3593  1  ! Define the Operator Routine Table for MACRO Greater Than.
;  3476          3594  1  !
;  3477        P 3595  1  OPERATOR_ROUTINE_TABLE (MACRO_GTR_THAN_TABLE,
;  3478          3596  1      OPERATOR_ROUTINE (L, L, L, GTR_L_L));
;  3479          3597  1
;  3480          3598  1  ! Define the Operator Routine Table for MACRO Less Than or Equal.
;  3481          3599  1  !
;  3482        P 3600  1  OPERATOR_ROUTINE_TABLE (MACRO_LSS_EQUAL_TABLE,
;  3483          3601  1      OPERATOR_ROUTINE (L, L, L, LEQ_L_L));
;  3484          3602  1
;  3485          3603  1  ! Define the Operator Routine Table for MACRO Greater Than or Equal.
;  3486          3604  1  !
;  3487        P 3605  1  OPERATOR_ROUTINE_TABLE (MACRO_GTR_EQUAL_TABLE,
;  3488          3606  1      OPERATOR_ROUTINE (L, L, L, GEQ_L_L));
;  3489          3607  1
```

```
;  3490          3608  1 ! Define the Operator Routine Table for MACRO Less Than Unsigned.
;  3491          3609  1 !
;  3492        P 3610  1 OPERATOR_ROUTINE_TABLE (MACRO_LSSU_THAN_TABLE,
;  3493          3611  1     OPERATOR_ROUTINE (L, L, L, LSS_LU_LU));
;  3494          3612  1
;  3495          3613  1 ! Define the Operator Routine Table for MACRO Greater Than Unsigned.
;  3496          3614  1 !
;  3497        P 3615  1 OPERATOR_ROUTINE_TABLE (MACRO_GTRU_THAN_TABLE,
;  3498          3616  1     OPERATOR_ROUTINE (L, L, L, GTR_LU_LU));
;  3499          3617  1
;  3500          3618  1 ! Define the Operator Routine Table for MACRO Less Than or Equal Unsigned.
;  3501          3619  1 !
;  3502        P 3620  1 OPERATOR_ROUTINE_TABLE (MACRO_LSSU_EQUAL_TABLE,
;  3503          3621  1     OPERATOR_ROUTINE (L, L, L, LEQ_LU_LU));
;  3504          3622  1
;  3505          3623  1 ! Define the Operator Routine Table for MACRO Greater Than or Equal Unsigned.
;  3506          3624  1 !
;  3507        P 3625  1 OPERATOR_ROUTINE_TABLE (MACRO_GTRU_EQUAL_TABLE,
;  3508          3626  1     OPERATOR_ROUTINE (L, L, L, GEQ_LU_LU));
;  3509          3627  1
;  3510          3628  1 ! Define the Operator Routine Table for MACRO Bitwise And.
;  3511          3629  1 !
;  3512        P 3630  1 OPERATOR_ROUTINE_TABLE (MACRO_BIT_AND_TABLE,
;  3513          3631  1     OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
;  3514          3632  1
;  3515          3633  1 ! Define the Operator Routine Table for MACRO Bitwise Or.
;  3516          3634  1 !
;  3517        P 3635  1 OPERATOR_ROUTINE_TABLE (MACRO_BIT_OR_TABLE,
;  3518          3636  1     OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
;  3519          3637  1
;  3520          3638  1 ! Define the Operator Routine Table for MACRO Bitwise Xor.
;  3521          3639  1 !
;  3522        P 3640  1 OPERATOR_ROUTINE_TABLE (MACRO_BIT_XOR_TABLE,
;  3523          3641  1     OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
;  3524          3642  1
;  3525          3643  1 ! Define the Operator Routine Table for MACRO Bitwise Eqv.
;  3526          3644  1 !
;  3527        P 3645  1 OPERATOR_ROUTINE_TABLE (MACRO_BIT_EQV_TABLE,
;  3528          3646  1     OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
;  3529          3647  1
;  3530          3648  1 ! Define the Operator Routine Table for MACRO Bitwise Not.
;  3531          3649  1 !
;  3532        P 3650  1 OPERATOR_ROUTINE_TABLE (MACRO_BIT_NOT_TABLE,
;  3533          3651  1     OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
;  3534          3652  1
;  3535          3653  1 ! Define the Operator Routine Table for MACRO bit selection.
;  3536          3654  1 ! Bit selection is a little unusual. It basically just modifies
;  3537          3655  1 ! the descriptor for the address.
;  3538          3656  1 !
;  3539          3657  1 ! When not combined with a fetch, X<p,s,e> is equivalent to X+p/8
;  3540          3658  1 ! When combined with a fetch, @X<p,s,e> extracts s bits starting from
;  3541          3659  1 ! position p, with sign extension in e is 1.
;  3542          3660  1 !
;  3543          3661  1 ! See the routines DBG$BLISS_BITSELECT and DBG$BLISS_INDIRECTION,
;  3544          3662  1 ! and also the comments in the BLISS tables.
;  3545          3663  1 !
;  3546        P 3664  1 OPERATOR_ROUTINE_TABLE (MACRO_BITSELECT_TABLE,
```

```
;  3547          3665  1     OPERATOR_ROUTINE (L, L, L, BITSELECT));
;  3548          3666  1
;  3549          3667  1 ! Define the Operator Routine Table for MACRO indirection.
;  3550          3668  1 ! This operator is the only one that fetches its arguments, so
;  3551          3669  1 ! it is the only one that may see all of the possible data types.
;  3552          3670  1 ! The result of the fetch is always a longword.
;  3553          3671  1 !
;  3554        P 3672  1 OPERATOR_ROUTINE_TABLE (MACRO_INDIRECT_TABLE,
;  3555        P 3673  1     OPERATOR_ROUTINE (B, B, L, INDIRECT_LU),
;  3556        P 3674  1     OPERATOR_ROUTINE (W, W, L, INDIRECT_LU),
;  3557        P 3675  1     OPERATOR_ROUTINE (BU,BU,L, INDIRECT_LU),
;  3558        P 3676  1     OPERATOR_ROUTINE (WU,WU,L, INDIRECT_LU),
;  3559        P 3677  1     OPERATOR_ROUTINE (LU,LU,L, INDIRECT_LU),
;  3560        P 3678  1     OPERATOR_ROUTINE (V, V, L, INDIRECT_LU),
;  3561        P 3679  1     OPERATOR_ROUTINE (VU,VU,L, INDIRECT_LU),
;  3562        P 3680  1     OPERATOR_ROUTINE (SV,SV,L, INDIRECT_LU),
;  3563        P 3681  1     OPERATOR_ROUTINE (SVU,SVU,L, INDIRECT_LU),
;  3564          3682  1     OPERATOR_ROUTINE (L, L, L, INDIRECT_LO));
;  3565          3683  1
;  3566          3684  1 ! Define the Operator Information Table for MACRO.
;  3567          3685  1 !
;  3568        P 3686  1 OPERATOR_INFO_TABLE (MACRO_OPINFO_TABLE,
;  3569        P 3687  1
;  3570        P 3688  1     ! All of the arithmetic, logical, and relational operators use the
;  3571        P 3689  1     ! same hierarchy table. The FALSE fetch flag means that, like BLISS,
;  3572        P 3690  1     ! if the operands are not explicitly fetched, address arithmetic
;  3573        P 3691  1     ! is done. E.g.,
;  3574        P 3692  1     ! EVAL A+B  will add the addresses of A and B
;  3575        P 3693  1     ! EVAL @A+@B  will add the contents of A and B
;  3576        P 3694  1     !
;  3577        P 3695  1     OPERATOR_INFO_ENTRY
;  3578        P 3696  1         (UNARY_PLUS, MACRO_UNARY_PLUS_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3579        P 3697  1     OPERATOR_INFO_ENTRY
;  3580        P 3698  1         (UNARY_MINUS, MACRO_UNARY_MINUS_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3581        P 3699  1     OPERATOR_INFO_ENTRY
;  3582        P 3700  1         (BIT_NOT, MACRO_BIT_NOT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3583        P 3701  1     OPERATOR_INFO_ENTRY
;  3584        P 3702  1         (ADD, MACRO_ADD_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3585        P 3703  1     OPERATOR_INFO_ENTRY
;  3586        P 3704  1         (SUBTRACT, MACRO_SUB_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3587        P 3705  1     OPERATOR_INFO_ENTRY
;  3588        P 3706  1         (MULTIPLY, MACRO_MUL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3589        P 3707  1     OPERATOR_INFO_ENTRY
;  3590        P 3708  1         (DIVIDE, MACRO_DIV_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3591        P 3709  1     OPERATOR_INFO_ENTRY
;  3592        P 3710  1         (LEFT_SHIFT, MACRO_SHIFT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3593        P 3711  1     OPERATOR_INFO_ENTRY
;  3594        P 3712  1         (EQUAL, MACRO_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3595        P 3713  1     OPERATOR_INFO_ENTRY
;  3596        P 3714  1         (NOT_EQUAL, MACRO_NOT_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3597        P 3715  1     OPERATOR_INFO_ENTRY
;  3598        P 3716  1         (GTR_THAN, MACRO_GTR_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3599        P 3717  1     OPERATOR_INFO_ENTRY
;  3600        P 3718  1         (LSS_THAN, MACRO_LSS_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3601        P 3719  1     OPERATOR_INFO_ENTRY
;  3602        P 3720  1         (GTR_EQUAL, MACRO_GTR_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3603        P 3721  1     OPERATOR_INFO_ENTRY
```

```
;  3604          P 3722  1          (LSS_EQUAL, MACRO_LSS_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3605          P 3723  1      OPERATOR_INFO_ENTRY
;  3606          P 3724  1          (REMAINDER, MACRO_MOD_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3607          P 3725  1      OPERATOR_INFO_ENTRY
;  3608          P 3726  1          (GTR_THAN_U, MACRO_GTRU_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3609          P 3727  1      OPERATOR_INFO_ENTRY
;  3610          P 3728  1          (LSS_THAN_U, MACRO_LSSU_THAN_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3611          P 3729  1      OPERATOR_INFO_ENTRY
;  3612          P 3730  1          (GTR_EQUAL_U, MACRO_GTRU_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3613          P 3731  1      OPERATOR_INFO_ENTRY
;  3614          P 3732  1          (LSS_EQUAL_U, MACRO_LSSU_EQUAL_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3615          P 3733  1      OPERATOR_INFO_ENTRY
;  3616          P 3734  1          (BIT_AND, MACRO_BIT_AND_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3617          P 3735  1      OPERATOR_INFO_ENTRY
;  3618          P 3736  1          (BIT_OR, MACRO_BIT_OR_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3619          P 3737  1      OPERATOR_INFO_ENTRY
;  3620          P 3738  1          (BIT_XOR, MACRO_BIT_XOR_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3621          P 3739  1      OPERATOR_INFO_ENTRY
;  3622          P 3740  1          (BIT_EQV, MACRO_BIT_EQV_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3623          P 3741  1      OPERATOR_INFO_ENTRY
;  3624          P 3742  1          (BITSELECT, MACRO_BITSELECT_TABLE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3625          P 3743  1
;  3626          P 3744  1      ! CONVERT gets used to convert value descriptors to integer type.
;  3627          P 3745  1      !
;  3628          P 3746  1      OPERATOR_INFO_ENTRY
;  3629          P 3747  1          (CONVERT, TABLEBASE, MACRO_HIER_TABLE, TABLEBASE, FALSE),
;  3630          P 3748  1
;  3631          P 3749  1
;  3632          P 3750  1      ! The INDIRECT operator (@) is the only one that does a fetch, so the
;  3633          P 3751  1      ! fetch flag is set to TRUE for this operator.
;  3634          P 3752  1      !
;  3635          P 3753  1      OPERATOR_INFO_ENTRY
;  3636          P 3754  1          (INDIRECT, MACRO_INDIRECT_TABLE, MACRO_HIER_TABLE, TABLEBASE, TRUE),
;  3637          P 3755  1
;  3638          P 3756  1      ! DEPOSIT uses its own hierarchy table which allows any dtype to be
;  3639          P 3757  1      ! deposited into any other. The fetch flag is false, meaning no implicit
;  3640          P 3758  1      ! fetch is done on the right-hand-side of the deposit.
;  3641          P 3759  1      ! E.g.  DEP A = B  deposits the address of B into A;
;  3642          P 3760  1      !       DEP A = @B  deposits the contents of B into A.
;  3643          P 3761  1      !
;  3644          P 3762  1      OPERATOR_INFO_ENTRY
;  3645          P 3763  1          (DEPOSIT, TABLEBASE, MACRO_HIERD_TABLE, TABLEBASE, FALSE),
;  3646          P 3764  1
;  3647          P 3765  1      ! The IDENTITY operator is used to turn primaries into values at
;  3648          P 3766  1      ! the end of an evaluate command. This ensures that EVAL A will
;  3649          P 3767  1      ! do exactly the same thing (go through the same code path)
;  3650          P 3768  1      ! as EVAL A+0  or  EVAL +A
;  3651          P 3769  1      !
;  3652          P 3770  1      OPERATOR_INFO_ENTRY
;  3653          P 3771  1          (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE, FALSE)
;  3654            3772  1      );
;  3655            3773  1
```

```
;  3657          3774  1 !
;  3658          3775  1 ! P A S C A L   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  3659          3776  1 !
;  3660          3777  1 !
;  3661          3778  1 ! This section contains the Operator Routine and Type tables needed to
;  3662          3779  1 ! evaluate expressions in the PASCAL language.
;  3663          3780  1 !
;  3664          3781  1 ! The following summarizes the information from the PASCAL manual.
;  3665          3782  1 !
;  3666          3783  1 ! PASCAL Data Types:
;  3667          3784  1 !
;  3668          3785  1 !       . Ordinal (Scalar)
;  3669          3786  1 !         .Integer (signed L, -2**31+1 to 2**31-1, unsigned L, 0 to 2**32-1)
;  3670          3787  1 !          in binary, octal, and hexdecimal.  Note: Integer value > MAXINT,
;  3671          3788  1 !          that value is treated unsigned, otherwise is always treated as
;  3672          3789  1 !          signed.
;  3673          3790  1 !         .Character (single character).
;  3674          3791  1 !         .Boolean (TRUE, FALSE).
;  3675          3792  1 !         .Enumerated (an ordered set of constant values denoted by 1 to 65535
;  3676          3793  1 !          identifiers).
;  3677          3794  1 !         .Subrange (a limited portion of another ordinal type for use as a
;  3678          3795  1 !          distinct type).
;  3679          3796  1 !
;  3680          3797  1 !       . Real (Scalar)
;  3681          3798  1 !         F_float -0.29e-38, -1.70e38, 0.29e-38, 1.70e38
;  3682          3799  1 !         D_float -0.29d-38, -1.70d38, 0.29d-38, 1.70d38
;  3683          3800  1 !         G_float -0.56d-308, -0.90d308, 0.56d-308, 0 90d308
;  3684          3801  1 !         H_float -0.84q-4932, -0.59q4932, 0.84q-4932  0.59q4932
;  3685          3802  1 !
;  3686          3803  1 !       . Structured
;  3687          3804  1 !         RECORD, ARRAY, VARYING OF CHAR, SET, and FILE
;  3688          3805  1 !         Note:
;  3689          3806  1 !          1. use constructor to set constant value, except FILE
;  3690          3807  1 !          2. an item of any structured type can be packed, except VARYING
;  3691          3808  1 !         RECORD, RECORD with Variants, ARRAY, VARYING OF CHAR, SET, FILE
;  3692          3809  1 !         (of ordinal, real, structured and pointer).
;  3693          3810  1 !
;  3694          3811  1 !       . Pointer
;  3695          3812  1 !         dynamic variable, NIL.
;  3696          3813  1 !
;  3697          3814  1 ! Compatibility:
;  3698          3815  1 !       . Structural
;  3699          3816  1 !       . Assignment
;  3700          3817  1 !
;  3701          3818  1 ! Type Conversions:
;  3702          3819  1 !       .Arithmetic Types
;  3703          3820  1 !        INTEGER                    (lowest)
;  3704          3821  1 !        UNSIGNED
;  3705          3822  1 !        REAL
;  3706          3823  1 !        DOUBLE
;  3707          3824  1 !        QUARD                      (highest)
;  3708          3825  1 !
;  3709          3826  1 !       .Character Types
;  3710          3827  1 !        CHAR                       (lowest)
;  3711          3828  1 !        PACKED ARRAY OF CHAR
;  3712          3829  1 !        VARYING OF CHAR            (highest)
;  3713          3830  1 !
```

```
;  3714          3831  1 ! Operators
;  3715          3832  1 !       .Arithmetic (+ - * ** / DIV REM MOD)
;  3716          3833  1 !        + - * ** operate on Arithmetic Types and produces a result of the
;  3717          3834  1 !          same type as the values.
;  3718          3835  1 !        / operates on Arithmetic Types and produces a Real result.
;  3719          3836  1 !        DIV REM MOD operate on integer and unsigned and produces an integer or
;  3720          3837  1 !          unsigned result.
;  3721          3838  1 !
;  3722          3839  1 !       .Relational (= <> < <= > >=)
;  3723          3840  1 !        Test Ordinal, Real, String or Set expressions and return a Boolean
;  3724          3841  1 !          result.
;  3725          3842  1 !
;  3726          3843  1 !       .Logical (AND OR NOT)
;  3727          3844  1 !        Operate on Boolean expressions and produce a Boolean Value.
;  3728          3845  1 !
;  3729          3846  1 !       .String (+ = <> < <= > >=)
;  3730          3847  1 !        Operate on String expressions.
;  3731          3848  1 !
;  3732          3849  1 !       .Set (+ * - = <> <= >= IN)
;  3733          3850  1 !        Form the union, intersection, or difference of two sets, compares two
;  3734          3851  1 !          sets, or tests an ordinal value for inclusion in a set.
;  3735          3852  1 !
;  3736          3853  1
;  3737          3854  1
;  3738          3855  1 ! Define a Mapping Table for PASCAL.
;  3739          3856  1 ! The compiler gives us a dtype of DST$K_BOOL.
;  3740          3857  1 ! We've defined a DSC$K_DTYPE_BOOL = DST$K_BOOL. But we then
;  3741          3858  1 ! map this dtype into our new DSC$K_DTYPE_TF. This keeps the dtype codes
;  3742          3859  1 !  dense and also compatible across langauges.
;  3743          3860  1
;  3744        P 3861  1 TYPE_MAPPING_TABLE (PASCAL_MAP_TABLE,
;  3745        P 3862  1     TYPE_GRAPH_EDGE (TPTR, LU),
;  3746          3863  1     TYPE_GRAPH_EDGE (BOOL, TF));
;  3747          3864  1
;  3748          3865  1 ! Define the Type Conversion Information Table for PASCAL.
;  3749          3866  1 ! PASCAL has a mapping table but no language-specific table of
;  3750          3867  1 ! exceptions. It thus uses the DEBUG conversions given in DBG$CVT_DX_DX.
;  3751          3868  1 !
;  3752        P 3869  1 CONVERSION_INFO_TABLE (PASCAL_CVTINFO_TABLE,
;  3753          3870  1     CONVERSION_INFO_ENTRY (PASCAL_MAP_TABLE, TABLEBASE, TRUE));
;  3754          3871  1
;  3755          3872  1 ! Define the Type Hierarchy Table for PASCAL.
;  3756          3873  1 ! This specifies that, for numeric types,
;  3757          3874  1 ! implicit conversions always go up along
;  3758          3875  1 ! the hierarchy
;  3759          3876  1 !
;  3760          3877  1 ! B -> W -> L -> LU -> F -> D -> H
;  3761          3878  1 !
;  3762          3879  1 ! If /G_FLOAT was specified then substitute G for D in the above.
;  3763          3880  1 ! Note - the PASCAL compiler never generates B or W, but we allow
;  3764          3881  1 ! a conversion path so that such data can be accessed if your language
;  3765          3882  1 ! is set to PASCAL in a mixed-language environment.
;  3766          3883  1 !
;  3767          3884  1 ! Text can be converted to varying text for string operations.
;  3768          3885  1 !
;  3769        P 3886  1 TYPE_HIERARCHY_TABLE (PASCAL_HIER_TABLE,
;  3770        P 3887  1     TYPE_GRAPH_EDGE (T, VT),
```

```
;3771        P 3888  1      TYPE_GRAPH_EDGE (B, W),
;3772        P 3889  1      TYPE_GRAPH_EDGE (W, L),
;3773        P 3890  1      TYPE_GRAPH_EDGE (L, LU),
;3774        P 3891  1      TYPE_GRAPH_EDGE (LU, F),
;3775        P 3892  1      TYPE_GRAPH_EDGE (F, D),
;3776        P 3893  1      TYPE_GRAPH_EDGE (F, G),
;3777        P 3894  1      TYPE_GRAPH_EDGE (D, H),
;3778        P 3895  1      TYPE_GRAPH_EDGE (G, H),
;3779           3896  1      0);
;3780           3897  1
;3781           3898  1  ! Define a smaller Type Hierarchy Table for PASCAL.
;3782           3899  1  ! This is a subset of the above table, and can be used for operators
;3783           3900  1  ! that only allow integer types.
;3784           3901  1  !
;3785        P 3902  1  TYPE_HIERARCHY_TABLE (PASCAL_HIER1_TABLE,
;3786        P 3903  1      TYPE_GRAPH_EDGE (B, W),
;3787        P 3904  1      TYPE_GRAPH_EDGE (W, L),
;3788        P 3905  1      TYPE_GRAPH_EDGE (L, LU),
;3789           3906  1      0);
;3790           3907  1
;3791           3908  1  ! Define the Type Hierarchy Table for PASCAL deposit.
;3792           3909  1  ! This specifies what we will allow on a DEPOSIT command. That is,
;3793           3910  1  ! if there is a path from one dtype to another in the table below
;3794           3911  1  ! then we will allow the first to be deposited into the second.
;3795           3912  1  !
;3796           3913  1  ! We have adopted similar rules as the PASCAL compiler; that is, in
;3797           3914  1  ! general, if the compiler allows the assignment we will allow the deposit.
;3798           3915  1  ! There may be a few cases where we are more permissive than the compiler.
;3799           3916  1  !
;3800        P 3917  1  TYPE_HIERARCHY_TABLE (PASCAL_HIERD_TABLE,
;3801        P 3918  1      TYPE_GRAPH_EDGE (L, LU),
;3802        P 3919  1
;3803        P 3920  1      ! Allow integers to be deposited into any of SUBRNG, or ENUM.
;3804        P 3921  1      ! The type converter will call a TYPEID_CHECK routine which will
;3805        P 3922  1      ! give an informational if the language does not allow the
;3806        P 3923  1      ! assignment. We do, however, do the operation anyway.
;3807        P 3924  1      !
;3808        P 3925  1      TYPE_GRAPH_EDGE (LU, SUBRNG),
;3809        P 3926  1      TYPE_GRAPH_EDGE (LU, ENUM),
;3810        P 3927  1      TYPE_GRAPH_EDGE (LU, SET),
;3811        P 3928  1
;3812        P 3929  1      ! Allow assignments in an upward direction.
;3813        P 3930  1      !
;3814        P 3931  1      TYPE_GRAPH_EDGE (LU, F),
;3815        P 3932  1      TYPE_GRAPH_EDGE (F, D),
;3816        P 3933  1      TYPE_GRAPH_EDGE (F, G),
;3817        P 3934  1      TYPE_GRAPH_EDGE (D, H),
;3818        P 3935  1      TYPE_GRAPH_EDGE (G, H),
;3819        P 3936  1
;3820        P 3937  1      ! Text can be assigned into a VT variable.
;3821        P 3938  1      !
;3822        P 3939  1      TYPE_GRAPH_EDGE (T, VT),
;3823        P 3940  1
;3824        P 3941  1      ! Allow assignments of T, ENUM, and TF to subrange. Subrange
;3825        P 3942  1      ! can be such things as 'a'..'z', red..green, or false..true
;3826        P 3943  1      !
;3827        P 3944  1      TYPE_GRAPH_EDGE (T, SET),
```

```
;  3828        P 3945  1      TYPE_GRAPH_EDGE (T, SUBRNG),
;  3829        P 3946  1      TYPE_GRAPH_EDGE (ENUM, SET),
;  3830        P 3947  1      TYPE_GRAPH_EDGE (ENUM, SUBRNG),
;  3831        P 3948  1      TYPE_GRAPH_EDGE (TF, SET),
;  3832          3949  1      TYPE_GRAPH_EDGE (TF, SUBRNG));
;  3833          3950  1
;  3834          3951  1
;  3835          3952  1  ! Define the Type Incompatibility Table for PASCAL
;  3836          3953  1  ! We do not allow types D and G to be mixed in arithmetic statements.
;  3837          3954  1  !
;  3838        P 3955  1  TYPE_INCOMP_TABLE (PASCAL_INCOMP_TABLE,
;  3839        P 3956  1      TYPE_GRAPH_EDGE (D, G),
;  3840          3957  1      0);
;  3841          3958  1
;  3842          3959  1
;  3843          3960  1  ! Define the Operator Routine Table for PASCAL Addition.
;  3844          3961  1  ! We can add any of the numeric types. "+" can also be applied to sets,
;  3845          3962  1  ! in which case it means set union. It can be applied to varying strings
;  3846          3963  1  ! and in this case it means string concatenation.
;  3847          3964  1  ! The additional specifier for strings and sets refers to a typeid check
;  3848          3965  1  ! routine that checks that the arguments are really of compatible type.
;  3849          3966  1  ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
;  3850          3967  1  !
;  3851        P 3968  1  OPERATOR_ROUTINE_TABLE (PASCAL_ADD_TABLE,
;  3852        P 3969  1      OPERATOR_ROUTINE (VT, VT, VT, CONCAT_T_T),
;  3853        P 3970  1      OPERATOR_ROUTINE (SET, SET, SET, UNION_SET_SET, SET_SET),
;  3854        P 3971  1      OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;  3855        P 3972  1      OPERATOR_ROUTINE (LU, LU, LU, ADD_LU_LU),
;  3856        P 3973  1      OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  3857        P 3974  1      OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;  3858        P 3975  1      OPERATOR_ROUTINE (G, G, G, ADD_G_G),
;  3859          3976  1      OPERATOR_ROUTINE (H, H, H, ADD_H_H));
;  3860          3977  1
;  3861          3978  1
;  3862          3979  1  ! Define the Operator Routine Table for PASCAL Subtraction.
;  3863          3980  1  ! Subtraction can be done to any of the numeric type.
;  3864          3981  1  ! "-" can also be applied to sets, in which case it means set difference.
;  3865          3982  1  ! The additional specifier for sets refers to a typeid check
;  3866          3983  1  ! routine that checks that the arguments are really of compatible type.
;  3867          3984  1  ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
;  3868          3985  1  !
;  3869        P 3986  1  OPERATOR_ROUTINE_TABLE (PASCAL_SUB_TABLE,
;  3870        P 3987  1      OPERATOR_ROUTINE (SET, SET, SET, DIFFERENCE_SET_SET, SET_SET),
;  3871        P 3988  1      OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  3872        P 3989  1      OPERATOR_ROUTINE (LU, LU, LU, SUB_LU_LU),
;  3873        P 3990  1      OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  3874        P 3991  1      OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  3875        P 3992  1      OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  3876          3993  1      OPERATOR_ROUTINE (H, H, H, SUB_H_H));
;  3877          3994  1
;  3878          3995  1
;  3879          3996  1  ! Define the Operator Routine Table for PASCAL Multiplication.
;  3880          3997  1  ! Multiplication can be applied to any of the numeric types.
;  3881          3998  1  ! "*" can also be applied to sets, in which case it means set difference.
;  3882          3999  1  ! The additional specifier for sets refers to a typeid check
;  3883          4000  1  ! routine that checks that the arguments are really of compatible type.
;  3884          4001  1  ! See the routine DBG$PERFORM_TYPEID_CHECK in DBGEVALOP.
```

```
;3885          4002  1 !
;3886        P 4003  1 OPERATOR_ROUTINE_TABLE (PASCAL_MUL_TABLE,
;3887        P 4004  1     OPERATOR_ROUTINE (SET, SET, SET, INTERSECT_SET_SET, SET_SET),
;3888        P 4005  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;3889        P 4006  1     OPERATOR_ROUTINE (LU, LU, LU, MUL_LU_LU),
;3890        P 4007  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;3891        P 4008  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;3892        P 4009  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;3893          4010  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H));
;3894          4011  1
;3895          4012  1
;3896          4013  1 ! Define the Operator Routine Table for PASCAL Exponentiation.
;3897          4014  1 ! This can be applied to any of the numeric types.
;3898          4015  1 !
;3899        P 4016  1 OPERATOR_ROUTINE_TABLE (PASCAL_POWER_TABLE,
;3900        P 4017  1     OPERATOR_ROUTINE (L, L, L, POWER_L_L),
;3901        P 4018  1     OPERATOR_ROUTINE (LU, LU, LU, POWER_L_L),
;3902        P 4019  1     OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;3903        P 4020  1     OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;3904        P 4021  1     OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;3905          4022  1     OPERATOR_ROUTINE (H, H, H, POWER_H_H));
;3906          4023  1
;3907          4024  1
;3908          4025  1 ! Define the Operator Routine Table for PASCAL Division.
;3909          4026  1 ! PASCAL "/" is only for floating division, "DIV" is used for integer
;3910          4027  1 ! division.
;3911          4028  1 !
;3912        P 4029  1 OPERATOR_ROUTI  _TABLE (PASCAL_DIV_TABLE,
;3913        P 4030  1     OPERATOR_  TINE (F, F, F, DIV_F_F),
;3914        P 4031  1     OPERATOR_  TINE (D, D, D, DIV_D_D),
;3915        P 4032  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;3916          4033  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H));
;3917          4034  1
;3918          4035  1 ! Unary Plus and Minus can be applied to any of the numeric types.
;3919          4036  1
;3920          4037  1 ! Define the Operator Routine Table for PASCAL Unary plus.
;3921          4038  1 !
;3922        P 4039  1 OPERATOR_ROUTINE_TABLE (PASCAL_UNARY_PLUS_TABLE,
;3923        P 4040  1
;3924        P 4041  1 ! The following are not language dependent types.  This is needed for DEBUG
;3925        P 4042  1 ! types.  For example, DEP/QUAD L= +1.
;3926        P 4043  1 !
;3927        P 4044  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;3928        P 4045  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;3929        P 4046  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;3930        P 4047  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;3931        P 4048  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;3932        P 4049  1
;3933        P 4050  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;3934        P 4051  1     OPERATOR_ROUTINE (LU, LU, LU, UNARY_PLUS_L),
;3935        P 4052  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;3936        P 4053  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;3937        P 4054  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;3938          4055  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H));
;3939          4056  1
;3940          4057  1
;3941          4058  1 ! Define the Operator Routine Table for PASCAL unary minus.
```

```
;3942          4059  1 !
;3943        P 4060  1 OPERATOR_ROUTINE_TABLE (PASCAL_UNARY_MINUS_TABLE,
;3944        P 4061  1
;3945        P 4062  1 ! The following are not language dependent types.  This is needed for DEBUG
;3946        P 4063  1 ! types.  For example, DEP/QUAD L= +1.
;3947        P 4064  1 !
;3948        P 4065  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;3949        P 4066  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;3950        P 4067  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;3951        P 4068  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;3952        P 4069  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;3953        P 4070  1
;3954        P 4071  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;3955        P 4072  1     OPERATOR_ROUTINE (LU, LU, LU, UNARY_MINUS_L),
;3956        P 4073  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;3957        P 4074  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;3958        P 4075  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;3959          4076  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H));
;3960          4077  1
;3961          4078  1
;3962          4079  1 ! Define the Operator Routine Table for PASCAL Integer Division.
;3963          4080  1 ! PASCAL has a DIV operator for integer divide.
;3964          4081  1 !
;3965        P 4082  1 OPERATOR_ROUTINE_TABLE (PASCAL_INTDIV_TABLE,
;3966        P 4083  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;3967          4084  1     OPERATOR_ROUTINE (LU, LU, LU, DIV_LU_LU));
;3968          4085  1
;3969          4086  1 ! MOD and REM can only be applied to integer arguments.
;3970          4087  1
;3971          4088  1 ! Define the Operator Routine Table for PASCAL Modulus.
;3972          4089  1 !
;3973        P 4090  1 OPERATOR_ROUTINE_TABLE (PASCAL_MOD_TABLE,
;3974        P 4091  1     OPERATOR_ROUTINE (L, L, L, MOD_L_L),
;3975          4092  1     OPERATOR_ROUTINE (LU, LU, LU, MOD_LU_LU));
;3976          4093  1
;3977          4094  1
;3978          4095  1 ! Define the Operator Routine Table for PASCAL Remainder.
;3979          4096  1 !
;3980        P 4097  1 OPERATOR_ROUTINE_TABLE (PASCAL_REM_TABLE,
;3981        P 4098  1     OPERATOR_ROUTINE (L, L, L, REM_L_L),
;3982          4099  1     OPERATOR_ROUTINE (LU, LU, LU, REM_LU_LU));
;3983          4100  1
;3984          4101  1
;3985          4102  1 ! Equal/Notequal can be applied to numeric types, sets, enumeration types,
;3986          4103  1 ! strings, pointers, and Booleans.
;3987          4104  1 ! The SET_SET, and ENUM_ENUM parameters below
;3988          4105  1 ! specify a typeid check to be done to ensure that operands being compared
;3989          4106  1 ! are really of compatible types. See the DBG$PERFORM_TYPEID_CHECK routine
;3990          4107  1 ! in DBGEVALOP.
;3991          4108  1
;3992          4109  1 ! Define the Operator Routine Table for PASCAL equal.
;3993          4110  1 !
;3994        P 4111  1 OPERATOR_ROUTINE_TABLE (PASCAL_EQL_TABLE,
;3995        P 4112  1     OPERATOR_ROUTINE (SET, SET, TF, EQL_SET_SET, SET_SET),
;3996        P 4113  1     OPERATOR_ROUTINE (TF, TF, TF, EQL_L_L),
;3997        P 4114  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, EQL_L_L, ENUM_ENUM),
;3998        P 4115  1     OPERATOR_ROUTINE (VT, VT, TF, EQL_VT_VT),
```

```
;3999        P 4116  1     OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
;4000        P 4117  1     OPERATOR_ROUTINE (LU, LU, TF, EQL_L_L),
;4001        P 4118  1     OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
;4002        P 4119  1     OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
;4003        P 4120  1     OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
;4004          4121  1     OPERATOR_ROUTINE (H, H, TF, EQL_H_H));
;4005          4122  1
;4006          4123  1
;4007          4124  1 ! Define the Operator Routine Table for PASCAL not equal.
;4008          4125  1 !
;4009        P 4126  1 OPERATOR_ROUTINE_TABLE (PASCAL_NEQ_TABLE,
;4010        P 4127  1     OPERATOR_ROUTINE (SET, SET, TF, NEQ_SET_SET, SET_SET),
;4011        P 4128  1     OPERATOR_ROUTINE (TF, TF, TF, NEQ_L_L),
;4012        P 4129  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, NEQ_L_L, ENUM_ENUM),
;4013        P 4130  1     OPERATOR_ROUTINE (VT, VT, TF, NEQ_VT_VT),
;4014        P 4131  1     OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
;4015        P 4132  1     OPERATOR_ROUTINE (LU, LU, TF, NEQ_L_L),
;4016        P 4133  1     OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
;4017        P 4134  1     OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
;4018        P 4135  1     OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
;4019          4136  1     OPERATOR_ROUTINE (H, H, TF, NEQ_H_H));
;4020          4137  1
;4021          4138  1
;4022          4139  1 ! Define the Operator Routine Table for PASCAL greater than or equal to.
;4023          4140  1 ! Greater/equal can be applied to numeric types, sets, enumeration types,
;4024          4141  1 ! strings, and Booleans.
;4025          4142  1 ! On sets, it means "contains".
;4026          4143  1 !
;4027        P 4144  1 OPERATOR_ROUTINE_TABLE (PASCAL_GEQ_TABLE,
;4028        P 4145  1     OPERATOR_ROUTINE (SET, SET, TF, GEQ_SET_SET, SET_SET),
;4029        P 4146  1     OPERATOR_ROUTINE (TF, TF, TF, GEQ_L_L),
;4030        P 4147  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, GEQ_L_L, ENUM_ENUM),
;4031        P 4148  1     OPERATOR_ROUTINE (VT, VT, TF, GEQ_VT_VT),
;4032        P 4149  1     OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
;4033        P 4150  1     OPERATOR_ROUTINE (LU, LU, TF, GEQ_LU_LU),
;4034        P 4151  1     OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
;4035        P 4152  1     OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
;4036        P 4153  1     OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
;4037          4154  1     OPERATOR_ROUTINE (H, H, TF, GEQ_H_H));
;4038          4155  1
;4039          4156  1
;4040          4157  1 ! Define the Operator Routine Table for PASCAL great than.
;4041          4158  1 ! Greater than can be applied to numeric types, enumeration types,
;4042          4159  1 ! strings, and Booleans.
;4043          4160  1 !
;4044        P 4161  1 OPERATOR_ROUTINE_TABLE (PASCAL_GTR_TABLE,
;4045        P 4162  1     OPERATOR_ROUTINE (TF, TF, TF, GTR_L_L),
;4046        P 4163  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, GTR_L_L, ENUM_ENUM),
;4047        P 4164  1     OPERATOR_ROUTINE (VT, VT, TF, GTR_VT_VT),
;4048        P 4165  1     OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
;4049        P 4166  1     OPERATOR_ROUTINE (LU, LU, TF, GTR_LU_LU),
;4050        P 4167  1     OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
;4051        P 4168  1     OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
;4052        P 4169  1     OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
;4053          4170  1     OPERATOR_ROUTINE (H, H, TF, GTR_H_H));
;4054          4171  1
;4055          4172  1
```

```
;  4056          4173  1
;  4057          4174  1
;  4058          4175  1 ! Define the Operator Routine Table for PASCAL less than.
;  4059          4176  1 ! Less than can be applied to numeric types, enumeration types,
;  4060          4177  1 ! strings, and Booleans.
;  4061          4178  1 !
;  4062        P 4179  1 OPERATOR_ROUTINE_TABLE (PASCAL_LSS_TABLE,
;  4063        P 4180  1     OPERATOR_ROUTINE (TF, TF, TF, LSS_L_L),
;  4064        P 4181  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, LSS_L_L, ENUM_ENUM),
;  4065        P 4182  1     OPERATOR_ROUTINE (VT, VT, TF, LSS_VT_VT),
;  4066        P 4183  1     OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
;  4067        P 4184  1     OPERATOR_ROUTINE (LU, LU, TF, LSS_LU_LU),
;  4068        P 4185  1     OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
;  4069        P 4186  1     OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
;  4070        P 4187  1     OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
;  4071          4188  1     OPERATOR_ROUTINE (H, H, TF, LSS_H_H));
;  4072          4189  1
;  4073          4190  1
;  4074          4191  1 ! Define the Operator Routine Table for PASCAL less than or equal to.
;  4075          4192  1 ! Less/equal can be applied to numeric types, sets, enumeration types,
;  4076          4193  1 ! strings, and Booleans.
;  4077          4194  1 ! On sets it means "is contained in".
;  4078          4195  1 !
;  4079        P 4196  1 OPERATOR_ROUTINE_TABLE (PASCAL_LEQ_TABLE,
;  4080        P 4197  1     OPERATOR_ROUTINE (SET, SET, TF, LEQ_SET_SET, SET_SET),
;  4081        P 4198  1     OPERATOR_ROUTINE (TF, TF, TF, LEQ_L_L),
;  4082        P 4199  1     OPERATOR_ROUTINE (ENUM, ENUM, TF, LEQ_L_L, ENUM_ENUM),
;  4083        P 4200  1     OPERATOR_ROUTINE (VT, VT, TF, LEQ_VT_VT),
;  4084        P 4201  1     OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
;  4085        P 4202  1     OPERATOR_ROUTINE (LU, LU, TF, LEQ_LU_LU),
;  4086        P 4203  1     OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
;  4087        P 4204  1     OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
;  4088        P 4205  1     OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
;  4089          4206  1     OPERATOR_ROUTINE (H, H, TF, LEQ_H_H));
;  4090          4207  1
;  4091          4208  1
;  4092          4209  1 ! The logical operators can only be applied to Booleans.
;  4093          4210  1
;  4094          4211  1 ! Define the Operator Routine Table for PASCAL not.
;  4095          4212  1 !
;  4096        P 4213  1 OPERATOR_ROUTINE_TABLE (PASCAL_NOT_TABLE,
;  4097          4214  1     OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
;  4098          4215  1
;  4099          4216  1
;  4100          4217  1 ! Define the Operator Routine Table for PASCAL and.
;  4101          4218  1 !
;  4102        P 4219  1 OPERATOR_ROUTINE_TABLE (PASCAL_AND_TABLE,
;  4103          4220  1     OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
;  4104          4221  1
;  4105          4222  1
;  4106          4223  1 ! Define the Operator Routine Table for PASCAL or.
;  4107          4224  1 !
;  4108        P 4225  1 OPERATOR_ROUTINE_TABLE (PASCAL_OR_TABLE,
;  4109          4226  1     OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
;  4110          4227  1
;  4111          4228  1
;  4112          4229  1 ! Define the Operator Routine Table for PASCAL Set IN.
```

```
;  4113          4230  1 ! Sets can be composed of integers, characters, Booleans, or
;  4114          4231  1 ! enumeration types, so we allow the inquiry whether an element
;  4115          4232  1 ! of one of those types is in the set. The typeid_check routine
;  4116          4233  1 ! is present to check that the type of the left arg matches the
;  4117          4234  1 ! element type of the set.
;  4118          4235  1 !
;  4119        P 4236  1 OPERATOR_ROUTINE_TABLE (PASCAL_IN_TABLE,
;  4120        P 4237  1     OPERATOR_ROUTINE (_, SET, TF, IN_SET_SET, SET_SET),
;  4121        P 4238  1     OPERATOR_ROUTINE (LU, SET, TF, IN_SET_SET, SET_SET),
;  4122        P 4239  1     OPERATOR_ROUTINE (T, SET, TF, IN_SET_SET, SET_SET),
;  4123        P 4240  1     OPERATOR_ROUTINE (TF, SET, TF, IN_SET_SET, SET_SET),
;  4124          4241  1     OPERATOR_ROUTINE (ENUM, SET, TF, IN_SET_SET, SET_SET));
;  4125          4242  1
;  4126          4243  1
;  4127          4244  1 ! Define the Operator Routine Table for the PASCAL built-in function SUCC.
;  4128          4245  1 !
;  4129        P 4246  1 OPERATOR_ROUTINE_TABLE (PASCAL_SUCCESSOR_TABLE,
;  4130          4247  1     OPERATOR_ROUTINE (ENUM, ENUM, ENUM, SUCC_ENUM, ENUM_ENUM));
;  4131          4248  1
;  4132          4249  1
;  4133          4250  1 ! Define the Operator Routine Table for the PASCAL built-in function PRED.
;  4134          4251  1 !
;  4135        P 4252  1 OPERATOR_ROUTINE_TABLE (PASCAL_PREDECESSOR_TABLE,
;  4136          4253  1     OPERATOR_ROUTINE (ENUM, ENUM, ENUM, PRED_ENUM, ENUM_ENUM));
;  4137          4254  1
;  4138          4255  1
;  4139          4256  1 ! Define the Operator Information Table for PASCAL.
;  4140          4257  1 !
;  4141        P 4258  1 OPERATOR_INFO_TABLE (PASCAL_OPINFO_TABLE,
;  4142        P 4259  1
;  4143        P 4260  1     ! The following arithmetic operators use the normal hierarchy table and
;  4144        P 4261  1     ! the incompatibility table.
;  4145        P 4262  1     !
;  4146        P 4263  1     OPERATOR_INFO_ENTRY
;  4147        P 4264  1         (ADD, PASCAL_ADD_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4148        P 4265  1     OPERATOR_INFO_ENTRY
;  4149        P 4266  1         (SUBTRACT, PASCAL_SUB_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4150        P 4267  1     OPERATOR_INFO_ENTRY
;  4151        P 4268  1         (MULTIPLY, PASCAL_MUL_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4152        P 4269  1     OPERATOR_INFO_ENTRY
;  4153        P 4270  1         (POWER_OF, PASCAL_POWER_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4154        P 4271  1     OPERATOR_INFO_ENTRY
;  4155        P 4272  1         (DIVIDE, PASCAL_DIV_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4156        P 4273  1     OPERATOR_INFO_ENTRY
;  4157        P 4274  1         (UNARY_PLUS, PASCAL_UNARY_PLUS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4158        P 4275  1     OPERATOR_INFO_ENTRY
;  4159        P 4276  1         (UNARY_MINUS, PASCAL_UNARY_MINUS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4160        P 4277  1
;  4161        P 4278  1     ! DIV, MOD, and REM operate only on integers and thus use the smaller
;  4162        P 4279  1     ! hierarchy table HIER1. It would also be OK to use the HIER table
;  4163        P 4280  1     ! but a smaller table makes the code run faster.
;  4164        P 4281  1     !
;  4165        P 4282  1     OPERATOR_INFO_ENTRY
;  4166        P 4283  1         (INT_DIVIDE, PASCAL_INTDIV_TABLE, PASCAL_HIER1_TABLE, TABLEBASE),
;  4167        P 4284  1     OPERATOR_INFO_ENTRY
;  4168        P 4285  1         (MODULUS, PASCAL_MOD_TABLE, PASCAL_HIER1_TABLE, TABLEBASE),
;  4169        P 4286  1     OPERATOR_INFO_ENTRY
```

```
;  4170        P 4287  1          (REMAINDER, PASCAL_REM_TABLE, PASCAL_HIER1_TABLE, PASCAL_INCOMP_TABLE),
;  4171        P 4288  1
;  4172        P 4289  1      ! The relational operators all use the normal HIER table and the
;  4173        P 4290  1      ! normal incompatibility table.
;  4174        P 4291  1      !
;  4175        P 4292  1      OPERATOR_INFO_ENTRY
;  4176        P 4293  1          (EQUAL, PASCAL_EQL_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4177        P 4294  1      OPERATOR_INFO_ENTRY
;  4178        P 4295  1          (NOT_EQUAL, PASCAL_NEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4179        P 4296  1      OPERATOR_INFO_ENTRY
;  4180        P 4297  1          (LSS_THAN, PASCAL_LSS_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4181        P 4298  1      OPERATOR_INFO_ENTRY
;  4182        P 4299  1          (GTR_THAN, PASCAL_GTR_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4183        P 4300  1      OPERATOR_INFO_ENTRY
;  4184        P 4301  1          (LSS_EQUAL, PASCAL_LEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4185        P 4302  1      OPERATOR_INFO_ENTRY
;  4186        P 4303  1          (GTR_EQUAL, PASCAL_GEQ_TABLE, PASCAL_HIER_TABLE, PASCAL_INCOMP_TABLE),
;  4187        P 4304  1
;  4188        P 4305  1      ! The logical operators only operate on the type TF. They thus need no
;  4189        P 4306  1      ! hierarchy table and no incompatibility table.
;  4190        P 4307  1      !
;  4191        P 4308  1      OPERATOR_INFO_ENTRY
;  4192        P 4309  1          (AND, PASCAL_AND_TABLE, TABLEBASE, TABLEBASE),
;  4193        P 4310  1      OPERATOR_INFO_ENTRY
;  4194        P 4311  1          (OR, PASCAL_OR_TABLE, TABLEBASE, TABLEBASE),
;  4195        P 4312  1      OPERATOR_INFO_ENTRY
;  4196        P 4313  1          (NOT, PASCAL_NOT_TABLE, TABLEBASE, TABLEBASE),
;  4197        P 4314  1
;  4198        P 4315  1      ! Set member is only allowed between the pairs given in the operator
;  4199        P 4316  1      ! table above, with no implicit conversions.
;  4200        P 4317  1      !
;  4201        P 4318  1      OPERATOR_INFO_ENTRY
;  4202        P 4319  1          (SET_MEMBER, PASCAL_IN_TABLE, TABLEBASE, TABLEBASE),
;  4203        P 4320  1
;  4204        P 4321  1      ! CONVERT gets used to convert subscripts to the appropriate type.
;  4205        P 4322  1      ! We specify the HIERD table to allow the same conversions
;  4206        P 4323  1      ! that we allow on a DEPOSIT.
;  4207        P 4324  1      !
;  4208        P 4325  1      OPERATOR_INFO_ENTRY
;  4209        P 4326  1          (CONVERT, TABLEBASE, PASCAL_HIERD_TABLE, PASCAL_INCOMP_TABLE),
;  4210        P 4327  1
;  4211        P 4328  1      ! DEPOSIT gets used in the DEPOSIT operator. The HIERD table allows,
;  4212        P 4329  1      ! for the most part, the same combinations that the PASCAL compiler
;  4213        P 4330  1      ! allows on assignment.
;  4214        P 4331  1      !
;  4215        P 4332  1      OPERATOR_INFO_ENTRY
;  4216        P 4333  1          (DEPOSIT, TABLEBASE, PASCAL_HIERD_TABLE, PASCAL_INCOMP_TABLE),
;  4217        P 4334  1
;  4218        P 4335  1      ! The identity operator is called at the end of an EVALUATE command
;  4219        P 4336  1      ! if we still have a Primary and we need to convert it to a Value
;  4220        P 4337  1      ! Descriptor.
;  4221        P 4338  1      !
;  4222        P 4339  1      OPERATOR_INFO_ENTRY
;  4223        P 4340  1          (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE),
;  4224        P 4341  1
;  4225        P 4342  1      ! Built-in functions use only the tables required by the particular
;  4226        P 4343  1      ! function.  Most need not use the hierarchy table or the incompatibility
```

```
;  4227        P 4344  1     ! table.
;  4228        P 4345  1     !
;  4229        P 4346  1     OPERATOR_INFO_ENTRY
;  4230        P 4347  1         (SUCCESSOR, PASCAL_SUCCESSOR_TABLE, TABLEBASE, TABLEBASE),
;  4231        P 4348  1
;  4232        P 4349  1     OPERATOR_INFO_ENTRY
;  4233        P 4350  1         (PREDECESSOR, PASCAL_PREDECESSOR_TABLE, TABLEBASE, TABLEBASE)
;  4234          4351  1     );
;  4235          4352  1
```

```
;  4237     4353  1 !
;  4238     4354  1 !       P L I   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  4239     4355  1 !
;  4240     4356  1 !
;  4241     4357  1 ! This section contains the Operator Routine and Type tables needed to
;  4242     4358  1 ! evaluate expressions in the PLI language.
;  4243     4359  1 !
;  4244     4360  1 ! Summary of Data Types
;  4245     4361  1 !
;  4246     4362  1 ! .Computational data types
;  4247     4363  1 !
;  4248     4364  1 !  .Arithmetic data types
;  4249     4365  1 !
;  4250     4366  1 !   - fix-point binary    (integers)
;  4251     4367  1 !     FIXED [BINARY] [(p)]              1 <=p<=7  (byte)
;  4252     4368  1 !                                       8 <=p<=15 (word)
;  4253     4369  1 !                                       16<=p<=31 (longword)
;  4254     4370  1 !                                       default=(31)
;  4255     4371  1 !     Note: all integer constants are represented as fixed-point decimal.
;  4256     4372  1 !
;  4257     4373  1 !   - fixed-point decimal (decimal integers and fractions)
;  4258     4374  1 !     [FIXED] DECIMAL [(p[,q])]         1<=p<=31
;  4259     4375  1 !                                       0<=q<=p
;  4260     4376  1 !                                       default=(10,0)
;  4261     4377  1 !
;  4262     4378  1 !   - floating-point      (binary and decimal)
;  4263     4379  1 !     FLOAT [DECIMAL/BINARY] [(p)]
;  4264     4380  1 !            BINARY: default            1<=p<=113      p: no. of bits in mantissa
;  4265     4381  1 !                                       default=(24)
;  4266     4382  1 !            DECIMAL                    1<=p<=34       p: no. of digits in mantissa
;  4267     4383  1 !                                       default=(7)
;  4268     4384  1 !
;  4269     4385  1 !     floating point values    sign bits    exponent bits  fractional bits
;  4270     4386  1 !     .F (1<=p<=7)/(1<=p<=24)     1             8              24
;  4271     4387  1 !     .D (8<=p<=15)/(25<=p<=53)   1             8              53
;  4272     4388  1 !     .G (same as above)          1            11              53
;  4273     4389  1 !     .H (16<=p<=34)/(54<=p<=113)1             15             113
;  4274     4390  1 !
;  4275     4391  1 !   - pictured             (fixed-point data stored in character form)
;  4276     4392  1 !     PICTURE 'picture'
;  4277     4393  1 !
;  4278     4394  1 ! .Character-string data
;  4279     4395  1 !  CHARACTER [(n)] [VARYING]          n<=32767
;  4280     4396  1 !                                     default=(1)
;  4281     4397  1 !
;  4282     4398  1 ! .Bit-string data
;  4283     4399  1 !  BIT [(n)] [ALIGNED]                n<=32767 bits
;  4284     4400  1 !                                     default=(1)
;  4285     4401  1 !
;  4286     4402  1 !  PLI pads the bit-string value in the direction of least significance with
;  4287     4403  1 !  zeros or truncates the least significant bits from the bit-string value.
;  4288     4404  1 !
;  4289     4405  1 !  note: DECLARE ABIT BIT (10);
;  4290     4406  1 !        ABIT = '1011'B;
;  4291     4407  1 !        (high memory) 0000001101 (low memory)
;  4292     4408  1 !
;  4293     4409  1 !        'character-string'Bn is not supported.
```

```
;4294   4410  1 !         (n is with bases other than 2. B/B1:0,1;B2:0,1,2,3
;4295   4411  1 !                                        B3:0...7;B4:0...F)
;4296   4412  1 !
;4297   4413  1 ! .Noncomputational data types
;4298   4414  1 !
;4299   4415  1 !  .Entry constants and variables
;4300   4416  1 !  .Label constants and variables
;4301   4417  1 !  .File constants and variables
;4302   4418  1 !  .Pointers
;4303   4419  1 !  .Areas and offsets
;4304   4420  1 !
;4305   4421  1 ! Aggregates
;4306   4422  1 ! .Arrays
;4307   4423  1 ! .Structures
;4308   4424  1 !
;4309   4425  1 ! Operators and Operands
;4310   4426  1 ! .Arithmetic  (+ - / * **)              Arithmetic operands
;4311   4427  1 ! .Relational  (> < ^> ^< >= <=         computational operands only
;4312   4428  1 !                 = ^=)                 two operands of the same type of all
;4313   4429  1 ! .Logical     (^ & !)                   Bit-string operands
;4314   4430  1 ! .Concatenation (!!)                   Bit-string/Character-string
;4315   4431  1 !
;4316   4432  1 ! Note: / should not be used to divide two fixed-point binary operand.
;4317   4433  1 !
;4318   4434  1 ! Precedence:
;4319   4435  1 ! ** +(unary) -(unary) ^ * / + - !! > < ^> ^< = ^= <= >= & !
;4320   4436  1 !
;4321   4437  1 ! Conversion:
;4322   4438  1 ! .If any operand has BINARY, the derived base is BINARY.  Otherwise, the
;4323   4439  1 !  derived base is DECIMAL.
;4324   4440  1 ! .If any operand has FLOAT, the derived scale is FLOAT.  Otherwise, the
;4325   4441  1 !  derived base is FIXED.
;4326   4442  1 !
;4327   4443  1 ! Conversion and arithmetic operation:
;4328   4444  1 !  fixed-point decimal integer --> fixed-point binary (with zero scale) if
;4329   4445  1 !  one operand is fixed-point binary.
;4330   4446  1 !
;4331   4447  1 !  note:
;4332   4448  1 !   ev fixbin/2   (in PLI is not allowed, we perform the operation)
;4333   4449  1 !   ev fixbin+2.2 (in PLI is not allowed, we perform the operation)
;4334   4450  1 !
;4335   4451  1 !  fixed-point binary/fixed-point decimal --> fixed-point float.
;4336   4452  1 !
;4337   4453  1 !  picture --> fixed decimal with precision and scale.
;4338   4454  1 !
;4339   4455  1 !  Offsets and Pointers:  ptr <- ptr, offset <- offset,
;4340   4456  1 !                        ptr <- offset, offset <- ptr (offset must have been
;4341   4457  1 !                                                      declared with an area)
;4342   4458  1 !
;4343   4459  1 ! Rules for conversion of data
;4344   4460  1 !
;4345   4461  1 ! target\source  ¦  Arithmetic  ¦  Pictured  ¦  Bit String  ¦  Character String
;4346   4462  1 ! ---------------------------------------------------------------------------
;4347   4463  1 ! Arithmetic
;4348   4464  1 ! FIXED BINARY    (fixedoverflow)  Pictured     Nonnegative    Arithmetic
;4349   4465  1 ! FIXED DECIMAL    src p > dst q   values have  FIXED BINARY   constant
;4350   4466  1 !                                 FIXED      (fixedoverflow)  (error)
```

```
;4351        4467  1 ! FIXED BINARY    (fixedoverflow)  DECIMAL     src value >     invalid numeric
;4352        4468  1 !                   src value >     format,     2**31,          characters
;4353        4469  1 !                   dst storage     followed    followed        -> dst type
;4354        4470  1 !                                   the same    the same        (fixedoverflow)
;4355        4471  1 ! FIXED DECIMAL    (truncated)      rules as    rules as        src p is too
;4356        4472  1 !                   src q > dst q   arith. ->   arith. ->       large
;4357        4473  1 !                  (padded 0 right) arith.      arith.          (truncated)
;4358        4474  1 !                   src q < dst q                               src q is large
;4359        4475  1 !
;4360        4476  1 ! FLOAT             (overflow)                   null is 0       null/spaces is 0
;4361        4477  1 !                   src !value! >
;4362        4478  1 !                   dst flt value
;4363        4479  1 !                  (underflow)
;4364        4480  1 !                   src !value!
;4365        4481  1 !                   is too small
;4366        4482  1 ! ------------------------------------------------------------------------------
;4367        4483  1 ! Bit String        takes absolute   same as     (padded 0)      character string
;4368        4484  1 !                   value ->         arith.      src n <         of 0s and 1s ->
;4369        4485  1 !                   FIXED BINARY                 dst n           bit-string
;4370        4486  1 !                   (31)                         (truncated)     (error)
;4371        4487  1 !                   (fixedoverflow)              src n >         any other char.
;4372        4488  1 !                   src value >                  dst n
;4373        4489  1 !                   dst value
;4374        4490  1 !                   sign and q are
;4375        4491  1 !                   lost during
;4376        4492  1 !                   the conversion
;4377        4493  1 ! ------------------------------------------------------------------------------
;4378        4494  1 ! Char. String      FIXED BINARY ->  character    bit string ->   (padded trailing
;4379        4495  1 !                   FIXED DECIMAL    string       char. string     spaces)
;4380        4496  1 !                   FLOAT BINARY ->  represent.                   intmed. >
;4381        4497  1 !                   FLOAT DECIMAL    is used                      dst (not varying)
;4382        4498  1 !                                                                 (truacted)
;4383        4499  1 !                                                                 dst > intmed.
;4384        4500  1 ! ------------------------------------------------------------------------------
;4385        4501  1 ! Pictured          src -> fixed decimal
;4386        4502  1 ! FIXED DECIMAL
;4387        4503  1 !
;4388        4504  1 !
;4389        4505  1 !
;4390        4506  1 ! This section contains the Operator Routine and Type tables needed to
;4391        4507  1 ! evaluate expressions in the PASCAL language.
;4392        4508  1 !
;4393        4509  1 !       P L I   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;4394        4510  1 !
;4395        4511  1 !
;4396        4512  1 ! This section contains the Operator Routine and Type tables needed to
;4397        4513  1 ! evaluate expressions in the PLI language.
;4398        4514  1 !
;4399        4515  1 !
;4400        4516  1
;4401        4517  1
;4402        4518  1 ! Define a Type Mapping Table for PL/I.
;4403        4519  1 !
;4404      P 4520  1 TYPE_MAPPING_TABLE (PLI_MAP_TABLE,
;4405      P 4521  1     TYPE_GRAPH_EDGE (PTR, LT),
;4406      P 4522  1     TYPE_GRAPH_EDGE (TF, V),
;4407        4523  1     0);
```

```
;  4408          4524  1
;  4409          4525  1
;  4410          4526  1 ! Define the Type Incompatibility Table for PL/I.  This prevents mixing
;  4411          4527  1 ! D and G types in an expression or deposit.
;  4412          4528  1 !
;  4413        P 4529  1 TYPE_INCOMP_TABLE (PLI_INCOMP_TABLE,
;  4414        P 4530  1     TYPE_GRAPH_EDGE (D, G),
;  4415          4531  1     0);
;  4416          4532  1
;  4417          4533  1
;  4418          4534  1 ! Define the PL/I Specific Type Conversion Table.  This allows proper
;  4419          4535  1 ! handling of PL/I bit-strings.
;  4420          4536  1 !
;  4421        P 4537  1 LANG_CVT_TABLE (PLI_CVT_TABLE,
;  4422        P 4538  1     LANG_CVT_ENTRY (PLI_CVT, ANY, V),
;  4423        P 4539  1     LANG_CVT_ENTRY (PLI_CVT, V, ANY),
;  4424        P 4540  1     LANG_CVT_ENTRY (PLI_CVT, ANY, VU),
;  4425        P 4541  1     LANG_CVT_ENTRY (PLI_CVT, VU, ANY),
;  4426        P 4542  1     LANG_CVT_ENTRY (PLI_CVT, ANY, T),
;  4427        P 4543  1     LANG_CVT_ENTRY (PLI_CVT, T, ANY),
;  4428        P 4544  1     LANG_CVT_ENTRY (PLI_CVT, ANY, VT),
;  4429        P 4545  1     LANG_CVT_ENTRY (PLI_CVT, VT, ANY),
;  4430        P 4546  1     LANG_CVT_ENTRY (PLI_CVT, PICT, ANY),
;  4431        P 4547  1     LANG_CVT_ENTRY (PLI_CVT, ANY, PICT),
;  4432          4548  1     0);
;  4433          4549  1
;  4434          4550  1
;  4435          4551  1 ! Define the Type Conversion Information Table for PLI.  PL/I bit-strings
;  4436          4552  1 ! require PL/I specific conversions (they are stored in reverse order).
;  4437          4553  1 !
;  4438        P 4554  1 CONVERSION_INFO_TABLE (PLI_CVTINFO_TABLE,
;  4439          4555  1     CONVERSION_INFO_ENTRY (PLI_MAP_TABLE, PLI_CVT_TABLE));
;  4440          4556  1
;  4441          4557  1
;  4442          4558  1 ! Define the Type Hierarchy Table for PLI.
;  4443          4559  1 !
;  4444          4560  1
;  4445          4561  1
;  4446          4562  1 ! Define a Type Hierarachy Table for PLI.
;  4447          4563  1 !
;  4448        P 4564  1 TYPE_HIERARCHY_TABLE (PLI_HIER1_TABLE,
;  4449        P 4565  1
;  4450        P 4566  1 ! The following 4 entries are made to validate the valid operand.  To
;  4451        P 4567  1 ! decide the target type, there is additional code to help.
;  4452        P 4568  1 !
;  4453        P 4569  1     TYPE_GRAPH_EDGE (T, P),
;  4454        P 4570  1     TYPE_GRAPH_EDGE (VT, P),
;  4455        P 4571  1     TYPE_GRAPH_EDGE (V, P),
;  4456        P 4572  1     TYPE_GRAPH_EDGE (VU, P),
;  4457        P 4573  1
;  4458        P 4574  1     TYPE_GRAPH_EDGE (PICT, P),
;  4459        P 4575  1     TYPE_GRAPH_EDGE (B, W),
;  4460        P 4576  1     TYPE_GRAPH_EDGE (W, P),
;  4461        P 4577  1     TYPE_GRAPH_EDGE (P, L),
;  4462        P 4578  1     TYPE_GRAPH_EDGE (L, F),
;  4463        P 4579  1     TYPE_GRAPH_EDGE (L, FIXED),
;  4464        P 4580  1     TYPE_GRAPH_EDGE (F, D),
```

```
;  4465        P 4581  1     TYPE_GRAPH_EDGE (F, G),
;  4466        P 4582  1     TYPE_GRAPH_EDGE (FIXED, H),
;  4467        P 4583  1     TYPE_GRAPH_EDGE (D, H),
;  4468        P 4584  1     TYPE_GRAPH_EDGE (G, H),
;  4469          4585  1     0);
;  4470          4586  1
;  4471          4587  1
;  4472          4588  1 ! Define a Type Hierarachy Table for PLI. (Relational)
;  4473          4589  1 !
;  4474        P 4590  1 TYPE_HIERARCHY_TABLE (PLI_HIER2_TABLE,
;  4475        P 4591  1     TYPE_GRAPH_EDGE (VT, P),
;  4476        P 4592  1     TYPE_GRAPH_EDGE (VT, T),
;  4477        P 4593  1     TYPE_GRAPH_EDGE (T, P),
;  4478        P 4594  1     TYPE_GRAPH_EDGE (VU, P),
;  4479        P 4595  1     TYPE_GRAPH_EDGE (VU, T),
;  4480        P 4596  1     TYPE_GRAPH_EDGE (VU, VT),
;  4481        P 4597  1     TYPE_GRAPH_EDGE (VU, V),
;  4482        P 4598  1     TYPE_GRAPH_EDGE (V, P),
;  4483        P 4599  1     TYPE_GRAPH_EDGE (V, T),
;  4484        P 4600  1     TYPE_GRAPH_EDGE (V, VT),
;  4485        P 4601  1
;  4486        P 4602  1     TYPE_GRAPH_EDGE (PICT, P),
;  4487        P 4603  1     TYPE_GRAPH_EDGE (B, W),
;  4488        P 4604  1     TYPE_GRAPH_EDGE (W, P),
;  4489        P 4605  1     TYPE_GRAPH_EDGE (P, L),
;  4490        P 4606  1     TYPE_GRAPH_EDGE (L, F),
;  4491        P 4607  1     TYPE_GRAPH_EDGE (L, FIXED),
;  4492        P 4608  1     TYPE_GRAPH_EDGE (F, D),
;  4493        P 4609  1     TYPE_GRAPH_EDGE (F, G),
;  4494        P 4610  1     TYPE_GRAPH_EDGE (FIXED, H),
;  4495        P 4611  1     TYPE_GRAPH_EDGE (D, H),
;  4496        P 4612  1     TYPE_GRAPH_EDGE (G, H),
;  4497          4613  1     0);
;  4498          4614  1
;  4499          4615  1
;  4500          4616  1 ! Define a Type Hierarachy Table for PLI. (Logical)
;  4501          4617  1 !
;  4502        P 4618  1 TYPE_HIERARCHY_TABLE (PLI_HIER3_TABLE,
;  4503        P 4619  1     TYPE_GRAPH_EDGE (B, V),
;  4504        P 4620  1     TYPE_GRAPH_EDGE (W, V),
;  4505        P 4621  1     TYPE_GRAPH_EDGE (L, V),
;  4506        P 4622  1     TYPE_GRAPH_EDGE (P, V),
;  4507        P 4623  1     TYPE_GRAPH_EDGE (F, V),
;  4508        P 4624  1     TYPE_GRAPH_EDGE (D, V),
;  4509        P 4625  1     TYPE_GRAPH_EDGE (G, V),
;  4510        P 4626  1     TYPE_GRAPH_EDGE (H, V),
;  4511        P 4627  1     TYPE_GRAPH_EDGE (PICT, V),
;  4512        P 4628  1     TYPE_GRAPH_EDGE (T, V),
;  4513        P 4629  1     TYPE_GRAPH_EDGE (VT, V),
;  4514        P 4630  1     TYPE_GRAPH_EDGE (VU, V),
;  4515          4631  1     0);
;  4516          4632  1
;  4517          4633  1
;  4518          4634  1 ! Define a Type Hierarachy Table for PLI. (Concatination)
;  4519          4635  1 !
;  4520        P 4636  1 TYPE_HIERARCHY_TABLE (PLI_HIER4_TABLE,
;  4521        P 4637  1     TYPE_GRAPH_EDGE (B, T),
```

```
;  4522        P 4638  1     TYPE_GRAPH_EDGE (W, T),
;  4523        P 4639  1     TYPE_GRAPH_EDGE (L, T),
;  4524        P 4640  1     TYPE_GRAPH_EDGE (P, T),
;  4525        P 4641  1     TYPE_GRAPH_EDGE (F, T),
;  4526        P 4642  1     TYPE_GRAPH_EDGE (D, T),
;  4527        P 4643  1     TYPE_GRAPH_EDGE (G, T),
;  4528        P 4644  1     TYPE_GRAPH_EDGE (H, T),
;  4529        P 4645  1     TYPE_GRAPH_EDGE (PICT, T),
;  4530        P 4646  1     TYPE_GRAPH_EDGE (VT, T),
;  4531        P 4647  1     TYPE_GRAPH_EDGE (VU, V),
;  4532        P 4648  1     TYPE_GRAPH_EDGE (V, T),
;  4533          4649  1     0);
;  4534          4650  1
;  4535          4651  1
;  4536          4652  1 ! Define a Type Hierarachy Table for PLI. (Unary +/-)
;  4537          4653  1 !
;  4538        P 4654  1 TYPE_HIERARCHY_TABLE (PLI_HIER5_TABLE,
;  4539        P 4655  1     TYPE_GRAPH_EDGE (T, P),
;  4540        P 4656  1     TYPE_GRAPH_EDGE (VT, P),
;  4541        P 4657  1     TYPE_GRAPH_EDGE (V, L),
;  4542        P 4658  1     TYPE_GRAPH_EDGE (VU, L),
;  4543        P 4659  1     TYPE_GRAPH_EDGE (PICT, P),
;  4544        P 4660  1     TYPE_GRAPH_EDGE (B, W),
;  4545        P 4661  1     TYPE_GRAPH_EDGE (W, P),
;  4546        P 4662  1     TYPE_GRAPH_EDGE (P, L),
;  4547        P 4663  1     TYPE_GRAPH_EDGE (L, F),
;  4548        P 4664  1     TYPE_GRAPH_EDGE (L, FIXED),
;  4549        P 4665  1     TYPE_GRAPH_EDGE (F, D),
;  4550        P 4666  1     TYPE_GRAPH_EDGE (F, G),
;  4551        P 4667  1     TYPE_GRAPH_EDGE (FIXED, H),
;  4552        P 4668  1     TYPE_GRAPH_EDGE (D, H),
;  4553        P 4669  1     TYPE_GRAPH_EDGE (G, H),
;  4554          4670  1     0);
;  4555          4671  1
;  4556          4672  1
;  4557          4673  1 ! For PLI DEPOSIT.
;  4558          4674  1 !
;  4559        P 4675  1 TYPE_HIERARCHY_TABLE (PLI_HIERD_TABLE,
;  4560        P 4676  1     TYPE_GRAPH_EDGE (B, W),
;  4561        P 4677  1     TYPE_GRAPH_EDGE (W, L),
;  4562        P 4678  1     TYPE_GRAPH_EDGE (L, P),
;  4563        P 4679  1     TYPE_GRAPH_EDGE (P, FIXED),
;  4564        P 4680  1     TYPE_GRAPH_EDGE (FIXED, F),
;  4565        P 4681  1     TYPE_GRAPH_EDGE (F, D),
;  4566        P 4682  1     TYPE_GRAPH_EDGE (D, G),
;  4567        P 4683  1     TYPE_GRAPH_EDGE (G, H),
;  4568        P 4684  1     TYPE_GRAPH_EDGE (H, V),
;  4569        P 4685  1     TYPE_GRAPH_EDGE (V, VU),
;  4570        P 4686  1     TYPE_GRAPH_EDGE (VU, T),
;  4571        P 4687  1     TYPE_GRAPH_EDGE (T, VT),
;  4572        P 4688  1     TYPE_GRAPH_EDGE (VT, PICT),
;  4573        P 4689  1     TYPE_GRAPH_EDGE (PICT, B),
;  4574          4690  1     0);
;  4575          4691  1
;  4576          4692  1
;  4577          4693  1 ! Most of the arithmetic routines operate on two arguments of the same type.
;  4578          4694  1 ! That type may be B, W, L, F, D, G, H, P, so we provide all of those case
```

```
;  4579          4695  1 ! indices.
;  4580          4696  1 !
;  4581          4697  1 ! Define the Operator Routine Table for PLI addition.
;  4582          4698  1 !
;  4583        P 4699  1 OPERATOR_ROUTINE_TABLE (PLI_ADD_TABLE,
;  4584        P 4700  1     OPERATOR_ROUTINE (B, B, B, ADD_B_B),
;  4585        P 4701  1     OPERATOR_ROUTINE (W, W, W, ADD_W_W),
;  4586        P 4702  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;  4587        P 4703  1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  4588        P 4704  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;  4589        P 4705  1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
;  4590        P 4706  1     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
;  4591        P 4707  1     OPERATOR_ROUTINE (P, P, P, ADD_P_P),
;  4592          4708  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, ADD_FIXED_FIXED));
;  4593          4709  1
;  4594          4710  1
;  4595          4711  1 ! Define the Operator Routine Table for PLI subtraction.
;  4596          4712  1 !
;  4597        P 4713  1 OPERATOR_ROUTINE_TABLE (PLI_SUB_TABLE,
;  4598        P 4714  1     OPERATOR_ROUTINE (B, B, B, SUB_B_B),
;  4599        P 4715  1     OPERATOR_ROUTINE (W, W, W, SUB_W_W),
;  4600        P 4716  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  4601        P 4717  1     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  4602        P 4718  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  4603        P 4719  1     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  4604        P 4720  1     OPERATOR_ROUTINE (H, H, H, SUB_H_H),
;  4605        P 4721  1     OPERATOR_ROUTINE (P, P, P, SUB_P_P),
;  4606          4722  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, SUB_FIXED_FIXED));
;  4607          4723  1
;  4608          4724  1
;  4609          4725  1 ! Define the Operator Routine Table for PLI multiplication.
;  4610          4726  1 !
;  4611        P 4727  1 OPERATOR_ROUTINE_TABLE (PLI_MUL_TABLE,
;  4612        P 4728  1     OPERATOR_ROUTINE (B, B, B, MUL_B_B),
;  4613        P 4729  1     OPERATOR_ROUTINE (W, W, W, MUL_W_W),
;  4614        P 4730  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;  4615        P 4731  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  4616        P 4732  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;  4617        P 4733  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;  4618        P 4734  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H),
;  4619        P 4735  1     OPERATOR_ROUTINE (P, P, P, MUL_P_P),
;  4620          4736  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, MUL_FIXED_FIXED));
;  4621          4737  1
;  4622          4738  1
;  4623          4739  1 ! Define the Operator Routine Table for PLI division.
;  4624          4740  1 !
;  4625        P 4741  1 OPERATOR_ROUTINE_TABLE (PLI_DIV_TABLE,
;  4626        P 4742  1     OPERATOR_ROUTINE (B, B, B, DIV_B_B),
;  4627        P 4743  1     OPERATOR_ROUTINE (W, W, W, DIV_W_W),
;  4628        P 4744  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;  4629        P 4745  1     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
;  4630        P 4746  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
;  4631        P 4747  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;  4632        P 4748  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H),
;  4633        P 4749  1     OPERATOR_ROUTINE (P, P, P, DIV_P_P),
;  4634          4750  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, DIV_FIXED_FIXED));
;  4635          4751  1
```

```
;  4636          4752  1
;  4637          4753  1 ! Define the Operator Routine Table for PLI unary plus.
;  4638          4754  1 !
;  4639        P 4755  1 OPERATOR_ROUTINE_TABLE (PLI_UNARY_PLUS_TABLE,
;  4640        P 4756  1
;  4641        P 4757  1 ! The following are not language dependent types.  This is needed for DEBUG
;  4642        P 4758  1 ! types.  For example, DEP/QUAD L= +1.
;  4643        P 4759  1 !
;  4644        P 4760  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  4645        P 4761  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  4646        P 4762  1
;  4647        P 4763  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  4648        P 4764  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  4649        P 4765  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  4650        P 4766  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  4651        P 4767  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  4652        P 4768  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  4653        P 4769  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  4654        P 4770  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  4655          4771  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_PLUS_FIXED));
;  4656          4772  1
;  4657          4773  1
;  4658          4774  1 ! Define the Operator Routine Table for PLI unary minus.
;  4659          4775  1 !
;  4660        P 4776  1 OPERATOR_ROUTINE_TABLE (PLI_UNARY_MINUS_TABLE,
;  4661        P 4777  1
;  4662        P 4778  1 ! The following are not language dependent types.  This is needed for DEBUG
;  4663        P 4779  1 ! types.  For example, DEP/QUAD L= +1.
;  4664        P 4780  1 !
;  4665        P 4781  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  4666        P 4782  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  4667        P 4783  1
;  4668        P 4784  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  4669        P 4785  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  4670        P 4786  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  4671        P 4787  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  4672        P 4788  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  4673        P 4789  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  4674        P 4790  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  4675        P 4791  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  4676          4792  1     OPERATOR_ROUTINE (FIXED, FIXED, FIXED, UNARY_MINUS_FIXED));
;  4677          4793  1
;  4678          4794  1
;  4679          4795  1 ! Define the Operator Routine Table for PLI exponentiation.
;  4680          4796  1 ! Exponentiation has some mixed forms. For example, if you raise a
;  4681          4797  1 ! floating number to an integer power, you do not necessarily
;  4682          4798  1 ! want to first convert the int to float. Instead, a special
;  4683          4799  1 ! routine indices such as  POWER_F_L are provided to do the right thing here.
;  4684          4800  1 !
;  4685        P 4801  1 OPERATOR_ROUTINE_TABLE (PLI_POWER_TABLE,
;  4686        P 4802  1     OPERATOR_ROUTINE (F, L, F, POWER_F_L),
;  4687        P 4803  1     OPERATOR_ROUTINE (D, L, D, POWER_D_L),
;  4688        P 4804  1     OPERATOR_ROUTINE (G, L, G, POWER_G_L),
;  4689        P 4805  1     OPERATOR_ROUTINE (H, L, H, POWER_H_L),
;  4690        P 4806  1     OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;  4691        P 4807  1     OPERATOR_ROUTINE (D, F, D, POWER_D_F),
;  4692        P 4808  1     OPERATOR_ROUTINE (F, D, D, POWER_F_D),
```

```
;  4693        P 4809  1     OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;  4694        P 4810  1     OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;  4695          4811  1     OPERATOR_ROUTINE (H, H, H, POWER_H_H));
;  4696          4812  1
;  4697          4813  1
;  4698          4814  1 ! Define the Operator Routine Table for PLI concatenation.
;  4699          4815  1 ! Concatenate can only be done on character or bit strings.
;  4700          4816  1 !
;  4701        P 4817  1 OPERATOR_ROUTINE_TABLE (PLI_CONCAT_TABLE,
;  4702        P 4818  1     OPERATOR_ROUTINE (T, T, T, CONCAT_T_T),
;  4703          4819  1     OPERATOR_ROUTINE (V, V, V, CONCAT_TF_TF));
;  4704          4820  1
;  4705          4821  1
;  4706          4822  1 ! In the tables for the comparison operators, we allow strings to be
;  4707          4823  1 ! compared, and also all the numeric types.
;  4708          4824  1 !
;  4709          4825  1 ! Define the Operator Routine Table for PLI equal.
;  4710          4826  1 !
;  4711        P 4827  1 OPERATOR_ROUTINE_TABLE (PLI_EQL_TABLE,
;  4712        P 4828  1     OPERATOR_ROUTINE (B, B, TF, EQL_B_B),
;  4713        P 4829  1     OPERATOR_ROUTINE (W, W, TF, EQL_W_W),
;  4714        P 4830  1     OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
;  4715        P 4831  1     OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
;  4716        P 4832  1     OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
;  4717        P 4833  1     OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
;  4718        P 4834  1     OPERATOR_ROUTINE (H, H, TF, EQL_H_H),
;  4719        P 4835  1     OPERATOR_ROUTINE (P, P, TF, EQL_P_P),
;  4720        P 4836  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, EQL_FIXED_FIXED),
;  4721        P 4837  1     OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
;  4722        P 4838  1     OPERATOR_ROUTINE (VT, VT, TF, EQL_VT_VT),
;  4723        P 4839  1     OPERATOR_ROUTINE (V, V, TF, EQL_TF_TF),
;  4724        P 4840  1     OPERATOR_ROUTINE (VU, VU, TF, EQL_TF_TF),
;  4725        P 4841  1 !
;  4726        P 4842  1 ! This one should have non-computational data item as well.
;  4727        P 4843  1 !
;  4728        P 4844  1 !   entry, label, file, pointer (which we map into L), area and offsets
;  4729        P 4845  1 !
;  4730          4846  1     );
;  4731          4847  1
;  4732          4848  1
;  4733          4849  1 ! Define the Operator Routine Table for PLI not equal.
;  4734          4850  1 !
;  4735        P 4851  1 OPERATOR_ROUTINE_TABLE (PLI_NEQ_TABLE,
;  4736        P 4852  1     OPERATOR_ROUTINE (B, B, TF, NEQ_B_B),
;  4737        P 4853  1     OPERATOR_ROUTINE (W, W, TF, NEQ_W_W),
;  4738        P 4854  1     OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
;  4739        P 4855  1     OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
;  4740        P 4856  1     OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
;  4741        P 4857  1     OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
;  4742        P 4858  1     OPERATOR_ROUTINE (H, H, TF, NEQ_H_H),
;  4743        P 4859  1     OPERATOR_ROUTINE (P, P, TF, NEQ_P_P),
;  4744        P 4860  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, NEQ_FIXED_FIXED),
;  4745        P 4861  1     OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
;  4746        P 4862  1     OPERATOR_ROUTINE (VT, VT, TF, NEQ_VT_VT),
;  4747        P 4863  1     OPERATOR_ROUTINE (V, V, TF, NEQ_TF_TF),
;  4748        P 4864  1     OPERATOR_ROUTINE (VU, VU, TF, NEQ_TF_TF),
;  4749        P 4865  1
```

```
;4750        P 4866  1 !
;4751        P 4867  1 ! This one should have non-computational data item as well.
;4752        P 4868  1 !
;4753        P 4869  1 !   entry, label, file, pointer (which we map into L), area and offsets
;4754        P 4870  1 !
;4755          4871  1     );
;4756          4872  1
;4757          4873  1
;4758          4874  1 ! Define the Operator Routine Table for PLI greater than.
;4759          4875  1 !
;4760        P 4876  1 OPERATOR_ROUTINE_TABLE (PLI_GTR_TABLE,
;4761        P 4877  1     OPERATOR_ROUTINE (B, B, TF, GTR_B_B),
;4762        P 4878  1     OPERATOR_ROUTINE (W, W, TF, GTR_W_W),
;4763        P 4879  1     OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
;4764        P 4880  1     OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
;4765        P 4881  1     OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
;4766        P 4882  1     OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
;4767        P 4883  1     OPERATOR_ROUTINE (H, H, TF, GTR_H_H),
;4768        P 4884  1     OPERATOR_ROUTINE (P, P, TF, GTR_P_P),
;4769        P 4885  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, GTR_FIXED_FIXED),
;4770        P 4886  1     OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
;4771        P 4887  1     OPERATOR_ROUTINE (VT, VT, TF, GTR_VT_VT),
;4772        P 4888  1     OPERATOR_ROUTINE (VU, VU, TF, GTR_TF_TF),
;4773          4889  1     OPERATOR_ROUTINE (V, V, TF, GTR_TF_TF));
;4774          4890  1
;4775          4891  1
;4776          4892  1 ! Define the Operator Routine Table for PLI greater than or equal to.
;4777          4893  1 !
;4778        P 4894  1 OPERATOR_ROUTINE_TABLE (PLI_GEQ_TABLE,
;4779        P 4895  1     OPERATOR_ROUTINE (B, B, TF, GEQ_B_B),
;4780        P 4896  1     OPERATOR_ROUTINE (W, W, TF, GEQ_W_W),
;4781        P 4897  1     OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
;4782        P 4898  1     OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
;4783        P 4899  1     OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
;4784        P 4900  1     OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
;4785        P 4901  1     OPERATOR_ROUTINE (H, H, TF, GEQ_H_H),
;4786        P 4902  1     OPERATOR_ROUTINE (P, P, TF, GEQ_P_P),
;4787        P 4903  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, GEQ_FIXED_FIXED),
;4788        P 4904  1     OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
;4789        P 4905  1     OPERATOR_ROUTINE (VT, VT, TF, GEQ_VT_VT),
;4790        P 4906  1     OPERATOR_ROUTINE (VU, VU, TF, GEQ_TF_TF),
;4791          4907  1     OPERATOR_ROUTINE (V, V, TF, GEQ_TF_TF));
;4792          4908  1
;4793          4909  1
;4794          4910  1 ! Define the Operator Routine Table for PLI less than.
;4795          4911  1 !
;4796        P 4912  1 OPERATOR_ROUTINE_TABLE (PLI_LSS_TABLE,
;4797        P 4913  1     OPERATOR_ROUTINE (B, B, TF, LSS_B_B),
;4798        P 4914  1     OPERATOR_ROUTINE (W, W, TF, LSS_W_W),
;4799        P 4915  1     OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
;4800        P 4916  1     OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
;4801        P 4917  1     OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
;4802        P 4918  1     OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
;4803        P 4919  1     OPERATOR_ROUTINE (H, H, TF, LSS_H_H),
;4804        P 4920  1     OPERATOR_ROUTINE (P, P, TF, LSS_P_P),
;4805        P 4921  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, LSS_FIXED_FIXED),
;4806        P 4922  1     OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
```

```
;  4807        P 4923  1     OPERATOR_ROUTINE (VT, VT, TF, LSS_VT_VT),
;  4808        P 4924  1     OPERATOR_ROUTINE (VU, VU, TF, LSS_TF_TF),
;  4809          4925  1     OPERATOR_ROUTINE (V, V, TF, LSS_TF_TF));
;  4810          4926  1
;  4811          4927  1
;  4812          4928  1 ! Define the Operator Routine Table for PLI less than or equal to.
;  4813          4929  1 !
;  4814        P 4930  1 OPERATOR_ROUTINE_TABLE (PLI_LEQ_TABLE,
;  4815        P 4931  1     OPERATOR_ROUTINE (B, B, TF, LEQ_B_B),
;  4816        P 4932  1     OPERATOR_ROUTINE (W, W, TF, LEQ_W_W),
;  4817        P 4933  1     OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
;  4818        P 4934  1     OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
;  4819        P 4935  1     OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
;  4820        P 4936  1     OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
;  4821        P 4937  1     OPERATOR_ROUTINE (H, H, TF, LEQ_H_H),
;  4822        P 4938  1     OPERATOR_ROUTINE (P, P, TF, LEQ_P_P),
;  4823        P 4939  1     OPERATOR_ROUTINE (FIXED, FIXED, TF, LEQ_FIXED_FIXED),
;  4824        P 4940  1     OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
;  4825        P 4941  1     OPERATOR_ROUTINE (VT, VT, TF, LEQ_VT_VT),
;  4826        P 4942  1     OPERATOR_ROUTINE (VU, VU, TF, LEQ_TF_TF),
;  4827          4943  1     OPERATOR_ROUTINE (V, V, TF, LEQ_TF_TF));
;  4828          4944  1
;  4829          4945  1
;  4830          4946  1 ! The logical operators AND, OR, and NOT can be applied only to
;  4831          4947  1 ! bit-string data types.
;  4832          4948  1 !
;  4833          4949  1 ! Define the Operator Routine Table for PLI NOT.
;  4834          4950  1 !
;  4835        P 4951  1 OPERATOR_ROUTINE_TABLE (PLI_BIT_NOT_TABLE,
;  4836          4952  1     OPERATOR_ROUTINE (V, V, V, BIT_NOT_TF));
;  4837          4953  1
;  4838          4954  1
;  4839          4955  1 ! Define the Operator Routine Table for PLI AND.
;  4840          4956  1 !
;  4841        P 4957  1 OPERATOR_ROUTINE_TABLE (PLI_BIT_AND_TABLE,
;  4842          4958  1     OPERATOR_ROUTINE (V, V, V, BIT_AND_TF));
;  4843          4959  1
;  4844          4960  1
;  4845          4961  1 ! Define the Operator Routine Table for PLI OR.
;  4846          4962  1 !
;  4847        P 4963  1 OPERATOR_ROUTINE_TABLE (PLI_BIT_OR_TABLE,
;  4848          4964  1     OPERATOR_ROUTINE (V, V, V, BIT_OR_TF));
;  4849          4965  1
;  4850          4966  1
;  4851          4967  1 ! Define the Operator Information Table for PLI.
;  4852          4968  1 !
;  4853        P 4969  1 OPERATOR_INFO_TABLE (PLI_OPINFO_TABLE,
;  4854        P 4970  1
;  4855        P 4971  1
;  4856        P 4972  1     ! The following are arithmetic tables that accept all numeric data types.
;  4857        P 4973  1     ! They thus need to specify an incompatibility table.
;  4858        P 4974  1     !
;  4859        P 4975  1     OPERATOR_INFO_ENTRY (ADD, PLI_ADD_TABLE, PLI_HIER1_TABLE,
;  4860        P 4976  1         PLI_INCOMP_TABLE),
;  4861        P 4977  1     OPERATOR_INFO_ENTRY (SUBTRACT, PLI_SUB_TABLE, PLI_HIER1_TABLE,
;  4862        P 4978  1         PLI_INCOMP_TABLE),
;  4863        P 4979  1     OPERATOR_INFO_ENTRY (MULTIPLY, PLI_MUL_TABLE, PLI_HIER1_TABLE,
```

```
;  4864          P 4980  1          PLI_INCOMP_TABLE),
;  4865          P 4981  1      OPERATOR_INFO_ENTRY (DIVIDE, PLI_DIV_TABLE, PLI_HIER1_TABLE,
;  4866          P 4982  1          PLI_INCOMP_TABLE),
;  4867          P 4983  1      OPERATOR_INFO_ENTRY (UNARY_PLUS, PLI_UNARY_PLUS_TABLE,
;  4868          P 4984  1          PLI_HIER5_TABLE, PLI_INCOMP_TABLE),
;  4869          P 4985  1      OPERATOR_INFO_ENTRY (UNARY_MINUS, PLI_UNARY_MINUS_TABLE,
;  4870          P 4986  1          PLI_HIER5_TABLE, PLI_INCOMP_TABLE),
;  4871          P 4987  1      OPERATOR_INFO_ENTRY (POWER_OF, PLI_POWER_TABLE, PLI_HIER1_TABLE,
;  4872          P 4988  1          PLI_INCOMP_TABLE),
;  4873          P 4989  1
;  4874          P 4990  1
;  4875          P 4991  1      ! The relationals accept all numeric and bit-string types.
;  4876          P 4992  1      ! There is no incompatibility table.
;  4877          P 4993  1      !
;  4878          P 4994  1      OPERATOR_INFO_ENTRY (EQUAL, PLI_EQL_TABLE, PLI_HIER2_TABLE,
;  4879          P 4995  1          PLI_INCOMP_TABLE),
;  4880          P 4996  1      OPERATOR_INFO_ENTRY (NOT_EQUAL, PLI_NEQ_TABLE, PLI_HIER2_TABLE,
;  4881          P 4997  1          PLI_INCOMP_TABLE),
;  4882          P 4998  1      OPERATOR_INFO_ENTRY (GTR_THAN, PLI_GTR_TABLE, PLI_HIER2_TABLE,
;  4883          P 4999  1          PLI_INCOMP_TABLE),
;  4884          P 5000  1      OPERATOR_INFO_ENTRY (GTR_EQUAL, PLI_GEQ_TABLE, PLI_HIER2_TABLE,
;  4885          P 5001  1          PLI_INCOMP_TABLE),
;  4886          P 5002  1      OPERATOR_INFO_ENTRY (LSS_THAN, PLI_LSS_TABLE, PLI_HIER2_TABLE,
;  4887          P 5003  1          PLI_INCOMP_TABLE),
;  4888          P 5004  1      OPERATOR_INFO_ENTRY (LSS_EQUAL, PLI_LEQ_TABLE, PLI_HIER2_TABLE,
;  4889          P 5005  1          PLI_INCOMP_TABLE),
;  4890          P 5006  1
;  4891          P 5007  1
;  4892          P 5008  1      ! The logical operators accept only bit-string quantities so they do
;  4893          P 5009  1      ! not need a hierarchy table. They also do not need an incompatibility
;  4894          P 5010  1      ! table.
;  4895          P 5011  1      !
;  4896          P 5012  1      OPERATOR_INFO_ENTRY (BIT_NOT, PLI_BIT_NOT_TABLE, PLI_HIER3_TABLE,
;  4897          P 5013  1          PLI_INCOMP_TABLE),
;  4898          P 5014  1      OPERATOR_INFO_ENTRY (BIT_AND, PLI_BIT_AND_TABLE, PLI_HIER3_TABLE,
;  4899          P 5015  1          PLI_INCOMP_TABLE),
;  4900          P 5016  1      OPERATOR_INFO_ENTRY (BIT_OR, PLI_BIT_OR_TABLE, PLI_HIER3_TABLE,
;  4901          P 5017  1          PLI_INCOMP_TABLE),
;  4902          P 5018  1
;  4903          P 5019  1
;  4904          P 5020  1      ! Concatenation allows only character or bit string operands, so it does
;  4905          P 5021  1      ! not need a hierarchy table. The incompatibility table is not required,
;  4906          P 5022  1      ! either.
;  4907          P 5023  1      !
;  4908          P 5024  1      OPERATOR_INFO_ENTRY (CONCATENATE, PLI_CONCAT_TABLE, PLI_HIER4_TABLE,
;  4909          P 5025  1          PLI_INCOMP_TABLE),
;  4910          P 5026  1
;  4911          P 5027  1      OPERATOR_INFO_ENTRY (CONVERT, TABLEBASE, PLI_HIERD_TABLE,
;  4912          P 5028  1          TABLEBASE),
;  4913          P 5029  1
;  4914          P 5030  1
;  4915          P 5031  1      ! The DEPOSIT operator gets called on the DEPOSIT command. It has
;  4916          P 5032  1      ! its own hierarchy table which allows any numeric type to be
;  4917          P 5033  1      ! converted to any other numeric type. The incompatibility table,
;  4918          P 5034  1      ! however, still prevents depositing D types into G types and
;  4919          P 5035  1      ! vice versa.
;  4920          P 5036  1      !
```

```
;  4921          P 5037  1     OPERATOR_INFO_ENTRY (DEPOSIT, TABLEBASE, PLI_HIERD_TABLE,
;  4922          P 5038  1         PLI_INCOMP_TABLE),
;  4923          P 5039  1
;  4924          P 5040  1
;  4925          P 5041  1     ! The IDENTITY operator gets called at the end of an EVALUATE command
;  4926          P 5042  1     ! to apply the PRIM_TO_VAL routine and then apply the appropriate
;  4927          P 5043  1     ! type mappings. This will ensure that EV BU will print as a signed integer,
;  4928          P 5044  1     ! for example.
;  4929          P 5045  1     !
;  4930          P 5046  1     ! The identity operator does not require any tables.
;  4931          P 5047  1     !
;  4932          P 5048  1     OPERATOR_INFO_ENTRY (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE)
;  4933            5049  1     );
;  4934            5050  1
```

```
;4936        5051  1 !
;4937        5052  1 !       R P G   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;4938        5053  1 !
;4939        5054  1 !
;4940        5055  1 ! This section contains the Operator Routine and Type tables needed to
;4941        5056  1 ! evaluate expressions in the RPG language.
;4942        5057  1 !
;4943        5058  1 !
;4944        5059  1 ! The following summarizes the information in the RPG-11 manual,
;4945        5060  1 ! RPGV1DPLN by Treggiari.
;4946        5061  1 !
;4947        5062  1 ! There is further documentation within the tables below, describing
;4948        5063  1 ! exactly how we translate this into the DEBUG tables.
;4949        5064  1 !
;4950        5065  1 ! RPG Data Types:
;4951        5066  1 !
;4952        5067  1 !       Trailing overpunched numeric (scaled)
;4953        5068  1 !       Packed numeric               (scaled)
;4954        5069  1 !       word                         (scaled)
;4955        5070  1 !       long-word                    (scaled)
;4956        5071  1 !       character
;4957        5072  1 !
;4958        5073  1 !       1-dimensional array of the above types
;4959        5074  1 !       table data type
;4960        5075  1 !       special registers which begin with an asterisk
;4961        5076  1 !
;4962        5077  1 ! Expressions:
;4963        5078  1 !       .Arithmetic
;4964        5079  1 !        operator: + - * / unary+ unary-
;4965        5080  1 !        operand:  numeric literal/identifier
;4966        5081  1 !
;4967        5082  1 !       .Conditional (TRUE, FALSE)
;4968        5083  1 !        operator: =, NOT =, >, NOT >, <, NOT <
;4969        5084  1 !        operand:  identifier, literal, or AE
;4970        5085  1 !
;4971        5086  1 ! Type Conversion:
;4972        5087  1 !
;4973        5088  1 !       W ----------------+
;4974        5089  1 !       L ----------------|
;4975        5090  1 !       NRO---------------|--> P
;4976        5091  1 !
;4977        5092  1 !       If one of the operand is Scaled Descriptor or Packed decimal
;4978        5093  1 !       data type the operation is always done in Packed Form.  The
;4979        5094  1 !       operation even includes the deposit.
;4980        5095  1 !
;4981        5096  1 !
;4982        5097  1 !
;4983        5098  1 !
;4984        5099  1 ! Define the Type Conversion Information Table for RPG.  No rounding
;4985        5100  1 ! takes place in RPG.
;4986        5101  1 !
;4987      P 5102  1 CONVERSION_INFO_TABLE (RPG_CVTINFO_TABLE,
;4988        5103  1     CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
;4989        5104  1
;4990        5105  1 ! Define Type Hierarchy Table for RPG.
;4991        5106  1 !
;4992      P 5107  1 TYPE_HIERARCHY_TABLE (RPG_HIER_TABLE,
```

```
;  4993        P 5108  1      TYPE_GRAPH_EDGE (W, P),
;  4994        P 5109  1      TYPE_GRAPH_EDGE (L, P),
;  4995        P 5110  1      TYPE_GRAPH_EDGE (NRO, P),
;  4996          5111  1      0);
;  4997          5112  1
;  4998          5113  1  ! Define the Type Hierarchy Table for RPG deposit.
;  4999          5114  1  !
;  5000        P 5115  1  TYPE_HIERARCHY_TABLE (RPG_HIERD_TABLE,
;  5001        P 5116  1      TYPE_GRAPH_EDGE (W, L),
;  5002        P 5117  1      TYPE_GRAPH_EDGE (L, NRO),
;  5003        P 5118  1      TYPE_GRAPH_EDGE (NRO, P),
;  5004        P 5119  1      TYPE_GRAPH_EDGE (P, W),
;  5005          5120  1      0);
;  5006          5121  1
;  5007          5122  1  ! Define the Operator Routine Table for RPG addition.
;  5008          5123  1  !
;  5009        P 5124  1  OPERATOR_ROUTINE_TABLE (RPG_ADD_TABLE,
;  5010          5125  1      OPERATOR_ROUTINE (P, P, P, ADD_P_P));
;  5011          5126  1
;  5012          5127  1  ! Define the Operator Routine Table for RPG subtraction.
;  5013          5128  1  !
;  5014        P 5129  1  OPERATOR_ROUTINE_TABLE (RPG_SUB_TABLE,
;  5015          5130  1      OPERATOR_ROUTINE (P, P, P, SUB_P_P));
;  5016          5131  1
;  5017          5132  1  ! Define the Operator Routine Table for RPG multiplication.
;  5018          5133  1  !
;  5019        P 5134  1  OPERATOR_ROUTINE_TABLE (RPG_MUL_TABLE,
;  5020          5135  1      OPERATOR_ROUTINE (P, P, P, MUL_P_P));
;  5021          5136  1
;  5022          5137  1  ! Define the Operator Routine Table for RPG division.
;  5023          5138  1  !
;  5024        P 5139  1  OPERATOR_ROUTINE_TABLE (RPG_DIV_TABLE,
;  5025          5140  1      OPERATOR_ROUTINE (P, P, P, DIV_P_P));
;  5026          5141  1
;  5027          5142  1  ! Define the Operator Routine Table for RPG unary plus.
;  5028          5143  1  !
;  5029        P 5144  1  OPERATOR_ROUTINE_TABLE (RPG_UNARY_PLUS_TABLE,
;  5030        P 5145  1
;  5031        P 5146  1  ! The following are not language dependent types.  This is needed for DEBUG
;  5032        P 5147  1  ! types.  For example, DEP/QUAD L= +1.
;  5033        P 5148  1  !
;  5034        P 5149  1      OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  5035        P 5150  1      OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  5036        P 5151  1      OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  5037        P 5152  1      OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  5038        P 5153  1      OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  5039        P 5154  1      OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  5040        P 5155  1      OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  5041        P 5156  1      OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  5042        P 5157  1      OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  5043        P 5158  1
;  5044          5159  1      OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P));
;  5045          5160  1
;  5046          5161  1  ! Define the Operator Routine Table for RPG unary minus.
;  5047          5162  1  !
;  5048        P 5163  1  OPERATOR_ROUTINE_TABLE (RPG_UNARY_MINUS_TABLE,
;  5049        P 5164  1
```

```
;  5050        P 5165  1 ! The following are not language dependent types.  This is needed for DEBUG
;  5051        P 5166  1 ! types.  for example, DEP/QUAD L= +1.
;  5052        P 5167  1 !
;  5053        P 5168  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  5054        P 5169  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  5055        P 5170  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  5056        P 5171  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  5057        P 5172  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  5058        P 5173  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  5059        P 5174  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  5060        P 5175  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  5061        P 5176  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  5062        P 5177  1
;  5063          5178  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P));
;  5064          5179  1
;  5065          5180  1 ! Define the Operator Routine Table for RPG =.
;  5066          5181  1 !
;  5067        P 5182  1 OPERATOR_ROUTINE_TABLE (RPG_EQL_TABLE,
;  5068        P 5183  1     OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
;  5069          5184  1     OPERATOR_ROUTINE (P, P, TF, EQL_P_P));
;  5070          5185  1
;  5071          5186  1 ! Define the Operator Routine Table for RPG NOT =.
;  5072          5187  1 !
;  5073        P 5188  1 OPERATOR_ROUTINE_TABLE (RPG_NEQ_TABLE,
;  5074        P 5189  1     OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
;  5075          5190  1     OPERATOR_ROUTINE (P, P, TF, NEQ_P_P));
;  5076          5191  1
;  5077          5192  1 ! Define the Operator Routine Table for RPG >.
;  5078          5193  1 !
;  5079        P 5194  1 OPERATOR_ROUTINE_TABLE (RPG_GTR_TABLE,
;  5080        P 5195  1     OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
;  5081          5196  1     OPERATOR_ROUTINE (P, P, TF, GTR_P_P));
;  5082          5197  1
;  5083          5198  1 ! Define the Operator Routine Table for RPG NOT <.
;  5084          5199  1 !
;  5085        P 5200  1 OPERATOR_ROUTINE_TABLE (RPG_GEQ_TABLE,
;  5086        P 5201  1     OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
;  5087          5202  1     OPERATOR_ROUTINE (P, P, TF, GEQ_P_P));
;  5088          5203  1
;  5089          5204  1 ! Define the Operator Routine Table for RPG <.
;  5090          5205  1 !
;  5091        P 5206  1 OPERATOR_ROUTINE_TABLE (RPG_LSS_TABLE,
;  5092        P 5207  1     OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
;  5093          5208  1     OPERATOR_ROUTINE (P, P, TF, LSS_P_P));
;  5094          5209  1
;  5095          5210  1 ! Define the Operator Routine Table for RPG NOT >.
;  5096          5211  1 !
;  5097        P 5212  1 OPERATOR_ROUTINE_TABLE (RPG_LEQ_TABLE,
;  5098        P 5213  1     OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
;  5099          5214  1     OPERATOR_ROUTINE (P, P, TF, LEQ_P_P));
;  5100          5215  1
;  5101          5216  1 ! Define the Operator Routine Table for RPG NOT.
;  5102          5217  1 !
;  5103        P 5218  1 OPERATOR_ROUTINE_TABLE (RPG_NOT_TABLE,
;  5104          5219  1     OPERATOR_ROUTINE (TF, TF, TF, NOT_L));
;  5105          5220  1
;  5106          5221  1 ! Define the Operator Routine Table for RPG AND.
```

```
;  5107          5222  1 !
;  5108        P 5223  1 OPERATOR_ROUTINE_TABLE (RPG_AND_TABLE,
;  5109          5224  1     OPERATOR_ROUTINE (TF, TF, TF, AND_L_L));
;  5110          5225  1
;  5111          5226  1 ! Define the Operator Routine Table for RPG OR.
;  5112          5227  1 !
;  5113        P 5228  1 OPERATOR_ROUTINE_TABLE (RPG_OR_TABLE,
;  5114          5229  1     OPERATOR_ROUTINE (TF, TF, TF, OR_L_L));
;  5115          5230  1
;  5116          5231  1
;  5117          5232  1 ! Define the Operator Information Table for RPG.
;  5118          5233  1 !
;  5119        P 5234  1 OPERATOR_INFO_TABLE (RPG_OPINFO_TABLE,
;  5120        P 5235  1     OPERATOR_INFO_ENTRY
;  5121        P 5236  1         (ADD, RPG_ADD_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5122        P 5237  1     OPERATOR_INFO_ENTRY
;  5123        P 5238  1         (SUBTRACT, RPG_SUB_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5124        P 5239  1     OPERATOR_INFO_ENTRY
;  5125        P 5240  1         (MULTIPLY, RPG_MUL_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5126        P 5241  1     OPERATOR_INFO_ENTRY
;  5127        P 5242  1         (DIVIDE, RPG_DIV_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5128        P 5243  1     OPERATOR_INFO_ENTRY
;  5129        P 5244  1         (UNARY_PLUS, RPG_UNARY_PLUS_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5130        P 5245  1     OPERATOR_INFO_ENTRY
;  5131        P 5246  1         (UNARY_MINUS, RPG_UNARY_MINUS_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5132        P 5247  1     OPERATOR_INFO_ENTRY
;  5133        P 5248  1         (EQUAL, RPG_EQL_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5134        P 5249  1     OPERATOR_INFO_ENTRY
;  5135        P 5250  1         (NOT_EQUAL, RPG_NEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5136        P 5251  1     OPERATOR_INFO_ENTRY
;  5137        P 5252  1         (GTR_THAN, RPG_GTR_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5138        P 5253  1     OPERATOR_INFO_ENTRY
;  5139        P 5254  1         (GTR_EQUAL, RPG_GEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5140        P 5255  1     OPERATOR_INFO_ENTRY
;  5141        P 5256  1         (LSS_THAN, RPG_LSS_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5142        P 5257  1     OPERATOR_INFO_ENTRY
;  5143        P 5258  1         (LSS_EQUAL, RPG_LEQ_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5144        P 5259  1     OPERATOR_INFO_ENTRY
;  5145        P 5260  1         (NOT, RPG_NOT_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5146        P 5261  1     OPERATOR_INFO_ENTRY
;  5147        P 5262  1         (AND, RPG_AND_TABLE, RPG_HIER_TABLE, TABLEBASE),
;  5148        P 5263  1     OPERATOR_INFO_ENTRY
;  5149        P 5264  1         (OR,  RPG_OR_TABLE,  RPG_HIER_TABLE, TABLEBASE),
;  5150        P 5265  1     OPERATOR_INFO_ENTRY
;  5151        P 5266  1         (CONVERT, TABLEBASE, RPG_HIERD_TABLE, TABLEBASE),
;  5152        P 5267  1     OPERATOR_INFO_ENTRY
;  5153        P 5268  1         (DEPOSIT, TABLEBASE, RPG_HIERD_TABLE, TABLEBASE),
;  5154        P 5269  1     OPERATOR_INFO_ENTRY
;  5155          5270  1         (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
;  5156          5271  1
```

```
;  5158          5272  1 !
;  5159          5273  1 !       U N K N O W N   O P E R A T O R   I N F O R M A T I O N   T A B L E S
;  5160          5274  1 !
;  5161          5275  1 !
;  5162          5276  1 ! This section contains the Operator Routine and Type tables needed to
;  5163          5277  1 ! evaluate expressions in the UNKNOWN language.
;  5164          5278  1 !
;  5165          5279  1 ! The purpose of "UNKNOWN" is to provide some level of support for other
;  5166          5280  1 ! languages by allowing SET LANG UNKNOWN. E.g., if somebody is writing a
;  5167          5281  1 ! compiler for a language other than one which DEBUG supports such as
;  5168          5282  1 ! JOVIAL, HAL, SIMULA, ..., they can get some level of DEBUG support
;  5169          5283  1 ! by generating DST and saying SET LANG UNK
;  5170          5284  1 !
;  5171          5285  1 ! So, the guiding principal below is to allow as many data types as possible
;  5172          5286  1 ! and be very permissive about conversions between data types. This will
;  5173          5287  1 ! increase the chance of covering the allowed operations and data types
;  5174          5288  1 ! in a given language. (Although it probably means we will be allowing
;  5175          5289  1 ! many things that the language does not allow).
;  5176          5290  1 !
;  5177          5291  1
;  5178          5292  1 ! Define the Type Conversion Information Table for UNKNOWN.
;  5179          5293  1 ! There are no initial type mappings or exceptions to language-specific
;  5180          5294  1 ! conversion rules.
;  5181          5295  1 !
;  5182        P 5296  1 CONVERSION_INFO_TABLE (UNKNOWN_CVTINFO_TABLE,
;  5183          5297  1     CONVERSION_INFO_ENTRY (TABLEBASE, TABLEBASE));
;  5184          5298  1
;  5185          5299  1
;  5186          5300  1 ! Define the Type Hierarchy Table for UNKNOWN.
;  5187          5301  1 ! This allows most integer-based types to be converted to L,
;  5188          5302  1 ! and L can then be converted up to F, D, G, and H.
;  5189          5303  1 ! The complex types are not included here - they increase the size
;  5190          5304  1 ! of all the tables, and most languages do not support complex
;  5191          5305  1 ! arithmetic, so it was not deemed worth it.
;  5192          5306  1 !
;  5193        P 5307  1 TYPE_HIERARCHY_TABLE (UNKNOWN_HIER_TABLE,
;  5194        P 5308  1     TYPE_GRAPH_EDGE (TF, L),
;  5195        P 5309  1     TYPE_GRAPH_EDGE (PTR, L),
;  5196        P 5310  1     TYPE_GRAPH_EDGE (TPTR, L),
;  5197        P 5311  1     TYPE_GRAPH_EDGE (ENUM, L),
;  5198        P 5312  1     TYPE_GRAPH_EDGE (BU, WU),
;  5199        P 5313  1     TYPE_GRAPH_EDGE (WU, LU),
;  5200        P 5314  1     TYPE_GRAPH_EDGE (LU, L),
;  5201        P 5315  1     TYPE_GRAPH_EDGE (B, W),
;  5202        P 5316  1     TYPE_GRAPH_EDGE (W, L),
;  5203        P 5317  1     TYPE_GRAPH_EDGE (L, F),
;  5204        P 5318  1     TYPE_GRAPH_EDGE (F , D),
;  5205        P 5319  1     TYPE_GRAPH_EDGE (F, G),
;  5206        P 5320  1     TYPE_GRAPH_EDGE (F, FC),
;  5207        P 5321  1     TYPE_GRAPH_EDGE (D, H),
;  5208        P 5322  1     TYPE_GRAPH_EDGE (D, DC),
;  5209        P 5323  1     TYPE_GRAPH_EDGE (G, H),
;  5210        P 5324  1     TYPE_GRAPH_EDGE (G, GC),
;  5211        P 5325  1     TYPE_GRAPH_EDGE (FC, DC),
;  5212        P 5326  1     TYPE_GRAPH_EDGE (FC, GC),
;  5213          5327  1     0);
;  5214          5328  1
```

```
;  5215          5329  1
;  5216          5330  1 ! Define the Type Hierarchy Table for UNKNOWN deposit.
;  5217          5331  1 ! This is a circular graph which allows any type to be deposited
;  5218          5332  1 ! into any other (even though DBG$CVT_DX_DX may not allow all
;  5219          5333  1 ! of the combinations.)
;  5220          5334  1 !
;  5221        P 5335  1 TYPE_HIERARCHY_TABLE (UNKNOWN_HIERD_TABLE,
;  5222        P 5336  1     TYPE_GRAPH_EDGE (BU, B),
;  5223        P 5337  1     TYPE_GRAPH_EDGE (B, WU),
;  5224        P 5338  1     TYPE_GRAPH_EDGE (WU, W),
;  5225        P 5339  1     TYPE_GRAPH_EDGE (W, LU),
;  5226        P 5340  1     TYPE_GRAPH_EDGE (LU, L),
;  5227        P 5341  1     TYPE_GRAPH_EDGE (L, F),
;  5228        P 5342  1     TYPE_GRAPH_EDGE (F ,D),
;  5229        P 5343  1     TYPE_GRAPH_EDGE (F, G),
;  5230        P 5344  1     TYPE_GRAPH_EDGE (D, H),
;  5231        P 5345  1     TYPE_GRAPH_EDGE (G, H),
;  5232        P 5346  1     TYPE_GRAPH_EDGE (H, FC),
;  5233        P 5347  1     TYPE_GRAPH_EDGE (FC, DC),
;  5234        P 5348  1     TYPE_GRAPH_EDGE (FC, GC),
;  5235        P 5349  1     TYPE_GRAPH_EDGE (DC, ENUM),
;  5236        P 5350  1     TYPE_GRAPH_EDGE (GC, ENUM),
;  5237        P 5351  1     TYPE_GRAPH_EDGE (ENUM, TPTR),
;  5238        P 5352  1     TYPE_GRAPH_EDGE (TPTR, PTR),
;  5239        P 5353  1     TYPE_GRAPH_EDGE (PTR, TF),
;  5240        P 5354  1     TYPE_GRAPH_EDGE (TF, BU),
;  5241          5355  1     0);
;  5242          5356  1
;  5243          5357  1 ! Define the Type Incompatibility Table for language UNKNOWN.
;  5244          5358  1 ! D and G arithmetic is always incompatible.
;  5245          5359  1 !
;  5246        P 5360  1 TYPE_INCOMP_TABLE (UNKNOWN_INCOMP_TABLE,
;  5247        P 5361  1     TYPE_GRAPH_EDGE (D, G),
;  5248        P 5362  1     TYPE_GRAPH_EDGE (D, GC),
;  5249        P 5363  1     TYPE_GRAPH_EDGE (G, DC),
;  5250        P 5364  1     TYPE_GRAPH_EDGE (DC, GC),
;  5251          5365  1     0);
;  5252          5366  1
;  5253          5367  1
;  5254          5368  1 ! Allow arithmetic on any of the numeric types.
;  5255          5369  1 ! Other types (PTR, TPTR, ENUM, unsigned types) can be converted to L before doing
;  5256          5370  1 ! any arithmetic.
;  5257          5371  1
;  5258          5372  1 ! Define the Operator Routine Table for UNKNOWN addition.
;  5259          5373  1 !
;  5260        P 5374  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_ADD_TABLE,
;  5261        P 5375  1     OPERATOR_ROUTINE (L, L, L, ADD_L_L),
;  5262        P 5376  1     OPERATOR_ROUTINE (F, F, F, ADD_F_F),
;  5263        P 5377  1     OPERATOR_ROUTINE (D, D, D, ADD_D_D),
;  5264        P 5378  1     OPERATOR_ROUTINE (G, G, G, ADD_G_G),
;  5265        P 5379  1     OPERATOR_ROUTINE (H, H, H, ADD_H_H),
;  5266        P 5380  1     OPERATOR_ROUTINE (FC, FC, FC, ADD_FC_FC),
;  5267        P 5381  1     OPERATOR_ROUTINE (DC, DC, DC, ADD_DC_DC),
;  5268          5382  1     OPERATOR_ROUTINE (GC, GC, GC, ADD_GC_GC));
;  5269          5383  1
;  5270          5384  1
;  5271          5385  1 ! Define the Operator Routine Table for UNKNOWN subtraction.
```

```
;  5272          5386  1 !
;  5273        P 5387  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_SUB_TABLE,
;  5274        P 5388  1     OPERATOR_ROUTINE (L, L, L, SUB_L_L),
;  5275        P 5389  1     OPERATOR_ROUTINE (F, F, F, SUB_F_F),
;  5276        P 5390  1     OPERATOR_ROUTINE (D, D, D, SUB_D_D),
;  5277        P 5391  1     OPERATOR_ROUTINE (G, G, G, SUB_G_G),
;  5278        P 5392  1     OPERATOR_ROUTINE (H, H, H, SUB_H_H),
;  5279        P 5393  1     OPERATOR_ROUTINE (FC, FC, FC, SUB_FC_FC),
;  5280        P 5394  1     OPERATOR_ROUTINE (DC, DC, DC, SUB_DC_DC),
;  5281          5395  1     OPERATOR_ROUTINE (GC, GC, GC, SUB_GC_GC));
;  5282          5396  1
;  5283          5397  1
;  5284          5398  1 ! Define the Operator Routine Table for UNKNOWN multiplication.
;  5285          5399  1 !
;  5286        P 5400  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_MUL_TABLE,
;  5287        P 5401  1     OPERATOR_ROUTINE (L, L, L, MUL_L_L),
;  5288        P 5402  1     OPERATOR_ROUTINE (F, F, F, MUL_F_F),
;  5289        P 5403  1     OPERATOR_ROUTINE (D, D, D, MUL_D_D),
;  5290        P 5404  1     OPERATOR_ROUTINE (G, G, G, MUL_G_G),
;  5291        P 5405  1     OPERATOR_ROUTINE (H, H, H, MUL_H_H),
;  5292        P 5406  1     OPERATOR_ROUTINE (FC, FC, FC, MUL_FC_FC),
;  5293        P 5407  1     OPERATOR_ROUTINE (DC, DC, DC, MUL_DC_DC),
;  5294          5408  1     OPERATOR_ROUTINE (GC, GC, GC, MUL_GC_GC));
;  5295          5409  1
;  5296          5410  1
;  5297          5411  1 ! Define the Operator Routine Table for UNKNOWN division.
;  5298          5412  1 !
;  5299        P 5413  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_DIV_TABLE,
;  5300        P 5414  1     OPERATOR_ROUTINE (L, L, L, DIV_L_L),
;  5301        P 5415  1     OPERATOR_ROUTINE (F, F, F, DIV_F_F),
;  5302        P 5416  1     OPERATOR_ROUTINE (D, D, D, DIV_D_D),
;  5303        P 5417  1     OPERATOR_ROUTINE (G, G, G, DIV_G_G),
;  5304        P 5418  1     OPERATOR_ROUTINE (H, H, H, DIV_H_H),
;  5305        P 5419  1     OPERATOR_ROUTINE (FC, FC, FC, DIV_FC_FC),
;  5306        P 5420  1     OPERATOR_ROUTINE (DC, DC, DC, DIV_DC_DC),
;  5307          5421  1     OPERATOR_ROUTINE (GC, GC, GC, DIV_GC_GC));
;  5308          5422  1
;  5309          5423  1
;  5310          5424  1 ! Define the Operator Routine Table for UNKNOWN unary plus.
;  5311          5425  1 !
;  5312        P 5426  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_UNARY_PLUS_TABLE,
;  5313        P 5427  1
;  5314        P 5428  1 ! The following are not language dependent types.  This is needed for DEBUG
;  5315        P 5429  1 ! types.  For example, DEP/QUAD L= +1.
;  5316        P 5430  1 !
;  5317        P 5431  1     OPERATOR_ROUTINE (B, B, B, UNARY_PLUS_B),
;  5318        P 5432  1     OPERATOR_ROUTINE (W, W, W, UNARY_PLUS_W),
;  5319        P 5433  1     OPERATOR_ROUTINE (P, P, P, UNARY_PLUS_P),
;  5320        P 5434  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_PLUS_Q),
;  5321        P 5435  1     OPERATOR_ROUTINE (O, O, O, UNARY_PLUS_O),
;  5322        P 5436  1
;  5323        P 5437  1     OPERATOR_ROUTINE (L, L, L, UNARY_PLUS_L),
;  5324        P 5438  1     OPERATOR_ROUTINE (F, F, F, UNARY_PLUS_F),
;  5325        P 5439  1     OPERATOR_ROUTINE (D, D, D, UNARY_PLUS_D),
;  5326        P 5440  1     OPERATOR_ROUTINE (G, G, G, UNARY_PLUS_G),
;  5327        P 5441  1     OPERATOR_ROUTINE (H, H, H, UNARY_PLUS_H),
;  5328        P 5442  1     OPERATOR_ROUTINE (FC, FC, FC, UNARY_PLUS_FC),
```

```
;  5329        P 5443  1     OPERATOR_ROUTINE (DC, DC, DC, UNARY_PLUS_DC),
;  5330          5444  1     OPERATOR_ROUTINE (GC, GC, GC, UNARY_PLUS_GC));
;  5331          5445  1
;  5332          5446  1
;  5333          5447  1 ! Define the Operator Routine Table for UNKNOWN unary minus.
;  5334          5448  1 !
;  5335        P 5449  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_UNARY_MINUS_TABLE,
;  5336        P 5450  1
;  5337        P 5451  1 ! The following are not language dependent types.  This is needed for DEBUG
;  5338        P 5452  1 ! types.  For example, DEP/QUAD L= +1.
;  5339        P 5453  1 !
;  5340        P 5454  1     OPERATOR_ROUTINE (B, B, B, UNARY_MINUS_B),
;  5341        P 5455  1     OPERATOR_ROUTINE (W, W, W, UNARY_MINUS_W),
;  5342        P 5456  1     OPERATOR_ROUTINE (P, P, P, UNARY_MINUS_P),
;  5343        P 5457  1     OPERATOR_ROUTINE (Q, Q, Q, UNARY_MINUS_Q),
;  5344        P 5458  1     OPERATOR_ROUTINE (O, O, O, UNARY_MINUS_O),
;  5345        P 5459  1
;  5346        P 5460  1     OPERATOR_ROUTINE (L, L, L, UNARY_MINUS_L),
;  5347        P 5461  1     OPERATOR_ROUTINE (F, F, F, UNARY_MINUS_F),
;  5348        P 5462  1     OPERATOR_ROUTINE (D, D, D, UNARY_MINUS_D),
;  5349        P 5463  1     OPERATOR_ROUTINE (G, G, G, UNARY_MINUS_G),
;  5350        P 5464  1     OPERATOR_ROUTINE (H, H, H, UNARY_MINUS_H),
;  5351        P 5465  1     OPERATOR_ROUTINE (FC, FC, FC, UNARY_MINUS_FC),
;  5352        P 5466  1     OPERATOR_ROUTINE (DC, DC, DC, UNARY_MINUS_DC),
;  5353          5467  1     OPERATOR_ROUTINE (GC, GC, GC, UNARY_MINUS_GC));
;  5354          5468  1
;  5355          5469  1
;  5356          5470  1 ! Define the Operator Routine Table for UNKNOWN exponentiation.
;  5357          5471  1 ! We special-case the mixed cases F**L, D**L, G**L, and H**L because
;  5358          5472  1 ! we do not want the exponent converted to float there. Note - this
;  5359          5473  1 ! is the same table as for FORTRAN.
;  5360          5474  1 !
;  5361        P 5475  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_POWER_TABLE,
;  5362        P 5476  1     OPERATOR_ROUTINE (W, W, W, POWER_W_W),
;  5363        P 5477  1     OPERATOR_ROUTINE (L, L, L, POWER_L_L),
;  5364        P 5478  1     OPERATOR_ROUTINE (F, L, F, POWER_F_L),
;  5365        P 5479  1     OPERATOR_ROUTINE (D, L, D, POWER_D_L),
;  5366        P 5480  1     OPERATOR_ROUTINE (G, L, G, POWER_G_L),
;  5367        P 5481  1     OPERATOR_ROUTINE (H, L, H, POWER_H_L),
;  5368        P 5482  1     OPERATOR_ROUTINE (FC, L, FC, POWER_FC_L),
;  5369        P 5483  1     OPERATOR_ROUTINE (DC, L, DC, POWER_DC_L),
;  5370        P 5484  1     OPERATOR_ROUTINE (GC, L, GC, POWER_GC_L),
;  5371        P 5485  1     OPERATOR_ROUTINE (F, F, F, POWER_F_F),
;  5372        P 5486  1     OPERATOR_ROUTINE (D, F, D, POWER_D_F),
;  5373        P 5487  1     OPERATOR_ROUTINE (F, D, D, POWER_F_D),
;  5374        P 5488  1     OPERATOR_ROUTINE (D, D, D, POWER_D_D),
;  5375        P 5489  1     OPERATOR_ROUTINE (G, G, G, POWER_G_G),
;  5376        P 5490  1     OPERATOR_ROUTINE (H, H, H, POWER_H_H),
;  5377        P 5491  1     OPERATOR_ROUTINE (FC, FC, FC, POWER_FC_FC),
;  5378        P 5492  1     OPERATOR_ROUTINE (DC, DC, DC, POWER_DC_DC),
;  5379          5493  1     OPERATOR_ROUTINE (GC, GC, GC, POWER_GC_GC));
;  5380          5494  1
;  5381          5495  1
;  5382          5496  1 ! The relationals are defined on strings and on numeric types.
;  5383          5497  1 ! We do not specially handle types like ENUM, TPTR, and so on, as
;  5384          5498  1 ! in PASCAL - here we just convert them to integer and do integer
;  5385          5499  1 ! comparison. The intent is to be looser about type-checking
```

```
;  5386          5500  1 ! than is PASCAL.
;  5387          5501  1
;  5388          5502  1 ! Define the Operator Routine Table for UNKNOWN equal.
;  5389          5503  1 !
;  5390        P 5504  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_EQL_TABLE,
;  5391        P 5505  1     OPERATOR_ROUTINE (T, T, TF, EQL_T_T),
;  5392        P 5506  1     OPERATOR_ROUTINE (L, L, TF, EQL_L_L),
;  5393        P 5507  1     OPERATOR_ROUTINE (F, F, TF, EQL_F_F),
;  5394        P 5508  1     OPERATOR_ROUTINE (D, D, TF, EQL_D_D),
;  5395        P 5509  1     OPERATOR_ROUTINE (G, G, TF, EQL_G_G),
;  5396        P 5510  1     OPERATOR_ROUTINE (H, H, TF, EQL_H_H),
;  5397        P 5511  1     OPERATOR_ROUTINE (FC, FC, TF, EQL_FC_FC),
;  5398        P 5512  1     OPERATOR_ROUTINE (DC, DC, TF, EQL_DC_DC),
;  5399          5513  1     OPERATOR_ROUTINE (GC, GC, TF, EQL_GC_GC));
;  5400          5514  1
;  5401          5515  1 ! Define the Operator Routine Table for UNKNOWN not-equal.
;  5402          5516  1 !
;  5403        P 5517  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_NEQ_TABLE,
;  5404        P 5518  1     OPERATOR_ROUTINE (T, T, TF, NEQ_T_T),
;  5405        P 5519  1     OPERATOR_ROUTINE (L, L, TF, NEQ_L_L),
;  5406        P 5520  1     OPERATOR_ROUTINE (F, F, TF, NEQ_F_F),
;  5407        P 5521  1     OPERATOR_ROUTINE (D, D, TF, NEQ_D_D),
;  5408        P 5522  1     OPERATOR_ROUTINE (G, G, TF, NEQ_G_G),
;  5409        P 5523  1     OPERATOR_ROUTINE (H, H, TF, NEQ_H_H),
;  5410        P 5524  1     OPERATOR_ROUTINE (FC, FC, TF, NEQ_FC_FC),
;  5411        P 5525  1     OPERATOR_ROUTINE (DC, DC, TF, NEQ_DC_DC),
;  5412          5526  1     OPERATOR_ROUTINE (GC, GC, TF, NEQ_GC_GC));
;  5413          5527  1
;  5414          5528  1
;  5415          5529  1 ! Define the Operator Routine Table for UNKNOWN less than.
;  5416          5530  1 !
;  5417        P 5531  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_LSS_THAN_TABLE,
;  5418        P 5532  1     OPERATOR_ROUTINE (T, T, TF, LSS_T_T),
;  5419        P 5533  1     OPERATOR_ROUTINE (L, L, TF, LSS_L_L),
;  5420        P 5534  1     OPERATOR_ROUTINE (F, F, TF, LSS_F_F),
;  5421        P 5535  1     OPERATOR_ROUTINE (D, D, TF, LSS_D_D),
;  5422        P 5536  1     OPERATOR_ROUTINE (G, G, TF, LSS_G_G),
;  5423          5537  1     OPERATOR_ROUTINE (H, H, TF, LSS_H_H));
;  5424          5538  1
;  5425          5539  1 ! Define the Operator Routine Table for UNKNOWN greater than.
;  5426          5540  1 !
;  5427        P 5541  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_GTR_THAN_TABLE,
;  5428        P 5542  1     OPERATOR_ROUTINE (T, T, TF, GTR_T_T),
;  5429        P 5543  1     OPERATOR_ROUTINE (L, L, TF, GTR_L_L),
;  5430        P 5544  1     OPERATOR_ROUTINE (F, F, TF, GTR_F_F),
;  5431        P 5545  1     OPERATOR_ROUTINE (D, D, TF, GTR_D_D),
;  5432        P 5546  1     OPERATOR_ROUTINE (G, G, TF, GTR_G_G),
;  5433          5547  1     OPERATOR_ROUTINE (H, H, TF, GTR_H_H));
;  5434          5548  1
;  5435          5549  1
;  5436          5550  1 ! Define the Operator Routine Table for UNKNOWN less than or equal.
;  5437          5551  1 !
;  5438        P 5552  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_LEQ_TABLE,
;  5439        P 5553  1     OPERATOR_ROUTINE (T, T, TF, LEQ_T_T),
;  5440        P 5554  1     OPERATOR_ROUTINE (L, L, TF, LEQ_L_L),
;  5441        P 5555  1     OPERATOR_ROUTINE (F, F, TF, LEQ_F_F),
;  5442        P 5556  1     OPERATOR_ROUTINE (D, D, TF, LEQ_D_D),
```

```
;  5443        P 5557  1     OPERATOR_ROUTINE (G, G, TF, LEQ_G_G),
;  5444          5558  1     OPERATOR_ROUTINE (H, H, TF, LEQ_H_H));
;  5445          5559  1
;  5446          5560  1 ! Define the Operator Routine Table for UNKNOWN greater than or equal.
;  5447          5561  1 !
;  5448        P 5562  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_GEQ_TABLE,
;  5449        P 5563  1     OPERATOR_ROUTINE (T, T, TF, GEQ_T_T),
;  5450        P 5564  1     OPERATOR_ROUTINE (L, L, TF, GEQ_L_L),
;  5451        P 5565  1     OPERATOR_ROUTINE (F, F, TF, GEQ_F_F),
;  5452        P 5566  1     OPERATOR_ROUTINE (D, D, TF, GEQ_D_D),
;  5453        P 5567  1     OPERATOR_ROUTINE (G, G, TF, GEQ_G_G),
;  5454          5568  1     OPERATOR_ROUTINE (H, H, TF, GEQ_H_H));
;  5455          5569  1
;  5456          5570  1 ! The logical operators AND, OR, NOT, EQV and XOR are defined only
;  5457          5571  1 ! on integers. Other types such as TF can be converted to Boolean.
;  5458          5572  1
;  5459          5573  1 ! Define the Operator Routine Table for UNKNOWN AND.
;  5460          5574  1 !
;  5461        P 5575  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_AND_TABLE,
;  5462          5576  1     OPERATOR_ROUTINE (L, L, L, BIT_AND_L_L));
;  5463          5577  1
;  5464          5578  1 ! Define the Operator Routine Table for UNKNOWN OR.
;  5465          5579  1 !
;  5466        P 5580  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_OR_TABLE,
;  5467          5581  1     OPERATOR_ROUTINE (L, L, L, BIT_OR_L_L));
;  5468          5582  1
;  5469          5583  1 ! Define the Operator Routine Table for UNKNOWN XOR.
;  5470          5584  1 !
;  5471        P 5585  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_XOR_TABLE,
;  5472          5586  1     OPERATOR_ROUTINE (L, L, L, BIT_XOR_L_L));
;  5473          5587  1
;  5474          5588  1 ! Define the Operator Routine Table for UNKNOWN EQV.
;  5475          5589  1 !
;  5476        P 5590  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_EQV_TABLE,
;  5477          5591  1     OPERATOR_ROUTINE (L, L, L, BIT_EQV_L_L));
;  5478          5592  1
;  5479          5593  1 ! Define the Operator Routine Table for UNKNOWN NOT.
;  5480          5594  1 !
;  5481        P 5595  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_NOT_TABLE,
;  5482          5596  1     OPERATOR_ROUTINE (L, L, L, BIT_NOT_L));
;  5483          5597  1
;  5484          5598  1 ! Define the Operator Routine Table for UNKNOWN concatenate.
;  5485          5599  1 !
;  5486        P 5600  1 OPERATOR_ROUTINE_TABLE (UNKNOWN_CONCATENATE_TABLE,
;  5487          5601  1     OPERATOR_ROUTINE (T, T, T, CONCAT_T_T));
;  5488          5602  1
;  5489          5603  1 ! Define the Operator Information Table for UNKNOWN.
;  5490          5604  1 !
;  5491        P 5605  1 OPERATOR_INFO_TABLE (UNKNOWN_OPINFO_TABLE,
;  5492        P 5606  1
;  5493        P 5607  1     ! All of the arithmetic, logical, and relational operators use
;  5494        P 5608  1     ! the same hierarchy table and the same incompatibility table.
;  5495        P 5609  1
;  5496        P 5610  1     ! Arithmetic.
;  5497        P 5611  1     !
;  5498        P 5612  1     OPERATOR_INFO_ENTRY
;  5499        P 5613  1         (ADD, UNKNOWN_ADD_TABLE,
```

```
;  5500        P 5614  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5501        P 5615  1      OPERATOR_INFO_ENTRY
;  5502        P 5616  1          (SUBTRACT, UNKNOWN_SUB_TABLE,
;  5503        P 5617  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5504        P 5618  1      OPERATOR_INFO_ENTRY
;  5505        P 5619  1          (MULTIPLY, UNKNOWN_MUL_TABLE,
;  5506        P 5620  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5507        P 5621  1      OPERATOR_INFO_ENTRY
;  5508        P 5622  1          (DIVIDE, UNKNOWN_DIV_TABLE,
;  5509        P 5623  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5510        P 5624  1      OPERATOR_INFO_ENTRY
;  5511        P 5625  1          (UNARY_PLUS, UNKNOWN_UNARY_PLUS_TABLE,
;  5512        P 5626  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5513        P 5627  1      OPERATOR_INFO_ENTRY
;  5514        P 5628  1          (UNARY_MINUS, UNKNOWN_UNARY_MINUS_TABLE,
;  5515        P 5629  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5516        P 5630  1      OPERATOR_INFO_ENTRY
;  5517        P 5631  1          (POWER_OF, UNKNOWN_POWER_TABLE,
;  5518        P 5632  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5519        P 5633  1
;  5520        P 5634  1      ! Relationals
;  5521        P 5635  1      !
;  5522        P 5636  1      OPERATOR_INFO_ENTRY
;  5523        P 5637  1          (EQUAL, UNKNOWN_EQL_TABLE,
;  5524        P 5638  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5525        P 5639  1      OPERATOR_INFO_ENTRY
;  5526        P 5640  1          (NOT_EQUAL, UNKNOWN_NEQ_TABLE,
;  5527        P 5641  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5528        P 5642  1      OPERATOR_INFO_ENTRY
;  5529        P 5643  1          (LSS_THAN, UNKNOWN_LSS_THAN_TABLE,
;  5530        P 5644  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5531        P 5645  1      OPERATOR_INFO_ENTRY
;  5532        P 5646  1          (GTR_THAN, UNKNOWN_GTR_THAN_TABLE,
;  5533        P 5647  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5534        P 5648  1      OPERATOR_INFO_ENTRY
;  5535        P 5649  1          (LSS_EQUAL, UNKNOWN_LEQ_TABLE,
;  5536        P 5650  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5537        P 5651  1      OPERATOR_INFO_ENTRY
;  5538        P 5652  1          (GTR_EQUAL, UNKNOWN_GEQ_TABLE,
;  5539        P 5653  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5540        P 5654  1
;  5541        P 5655  1      ! Logicals.
;  5542        P 5656  1      !
;  5543        P 5657  1      OPERATOR_INFO_ENTRY
;  5544        P 5658  1          (AND, UNKNOWN_AND_TABLE,
;  5545        P 5659  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5546        P 5660  1      OPERATOR_INFO_ENTRY
;  5547        P 5661  1          (OR, UNKNOWN_OR_TABLE,
;  5548        P 5662  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5549        P 5663  1      OPERATOR_INFO_ENTRY
;  5550        P 5664  1          (XOR, UNKNOWN_XOR_TABLE,
;  5551        P 5665  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5552        P 5666  1      OPERATOR_INFO_ENTRY
;  5553        P 5667  1          (EQV, UNKNOWN_EQV_TABLE,
;  5554        P 5668  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5555        P 5669  1      OPERATOR_INFO_ENTRY
;  5556        P 5670  1          (NOT, UNKNOWN_NOT_TABLE,
```

```
;  5557        P 5671  1          UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5558        P 5672  1
;  5559        P 5673  1      ! String operations.
;  5560        P 5674  1      !
;  5561        P 5675  1      OPERATOR_INFO_ENTRY
;  5562        P 5676  1          (CONCATENATE, UNKNOWN_CONCATENATE_TABLE,
;  5563        P 5677  1           UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5564        P 5678  1
;  5565        P 5679  1      ! CONVERT is used for things like converting subscripts.
;  5566        P 5680  1      !
;  5567        P 5681  1      OPERATOR_INFO_ENTRY
;  5568        P 5682  1          (CONVERT, TABLEBASE, UNKNOWN_HIER_TABLE, UNKNOWN_INCOMP_TABLE),
;  5569        P 5683  1
;  5570        P 5684  1      ! DEPOSIT is used in the DEPOSIT command.
;  5571        P 5685  1      !
;  5572        P 5686  1      OPERATOR_INFO_ENTRY
;  5573        P 5687  1          (DEPOSIT, TABLEBASE, UNKNOWN_HIERD_TABLE, UNKNOWN_INCOMP_TABLE),
;  5574        P 5688  1
;  5575        P 5689  1      ! IDENTITY is used at the end of an EVALUATE to turn the primary
;  5576        P 5690  1      ! into a value descriptor.
;  5577        P 5691  1      !
;  5578        P 5692  1      OPERATOR_INFO_ENTRY
;  5579          5693  1          (IDENTITY, TABLEBASE, TABLEBASE, TABLEBASE));
;  5580          5694  1
```

```
;  5582          5695  1 ! The following data structure holds a type pair in a word and is used
;  5583          5696  1 ! to search for the type pair in the Operator Routine Table.
;  5584          5697  1 !
;  5585          5698  1 FIELD
;  5586          5699  1     TYPE_PAIR_FLDS =
;  5587          5700  1         SET
;  5588          5701  1         B$_RIGHT_TYPE = [0,B0_],
;  5589          5702  1         B$_LEFT_TYPE  = [0,B1_],
;  5590          5703  1         W$_TYPE_PAIR  = [0,W0_]
;  5591          5704  1         TES;
;  5592          5705  1
;  5593          5706  1 MACRO
;  5594          5707  1     TYPE$PAIR = BLOCK [1, WORD] FIELD (TYPE_PAIR_FLDS) %;
;  5595          5708  1
```

```
;  5597          5709  1 ROUTINE AAA_DUMMY =
;  5598          5710  1 !
;  5599          5711  1 ! FUNCTION
;  5600          5712  1 !       This routine just causes the above tables to be dumped in the
;  5601          5713  1 !       listing.
;  5602          5714  1 !
;  5603          5715  2     BEGIN
;  5604          5716  2     0
;  5605          5717  1     END;

                                                                .TITLE  DBGEVALOP
                                                                .IDENT  \V04-000\

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

                                                  45  53  41  42  00000 P.AAA:  .ASCII  \BASE\                                ;
                                              00000000  00000000  00004 ADA_CVTINFO_TABLE:
                                                                        .LONG   0, 0                                          ;
                                                        00  0000C         .BYTE   0                                           ;
                                                            0000D         .BLKB   3                                           ;
                                                 00000007  00010          .LONG   7                                           ;
1C2B  1B0A  0B0A  2B08  0A08  0804  0803  0807  0802  0806  00014 P.AAB:   .WORD   2054, 2050, 2055, 2051, 2052, 2568, -       ;
                                                0000  1C1B  1C0B  00028                 11016, 2826, 6922, 7211, 7179, 7195, 0      ;
                                                            0002E         .BLKB   2                                           ;
                                                 0000000A  00030          .LONG   10                                          ;
1C1B  1B0B  0B0A  0A2B  2B04  0408  0803  0307  0702  0206  00034 P.AAC:   .WORD   518, 1794, 775, 2051, 1032, 11012, 2603, -  ;
0000  341C  341B  340B  340A  342B  3408  340E  342F  061C  00048                  2826, 6923, 7195, 1564, 13359, 13326, -     ;
                                                                                   13320, 13355, 13322, 13323, 13339, 13340, -  ;
                                                                                   0                                          ;
                                                 00000016  0005C          .LONG   22                                          ;
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06  00060 P.AAD:   .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -  ;
00  00  01  0A  09  09  00  15  00  00  00  F6  15  15  00  0006F                  0, 0, 7, 0, 21, 21, -10, 0, 0, 0, 21, 0, -  ;
00  00  BF  08  08  00  1A  00  00  01  0C  1A  1A  00  09  0007E                  9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -   ;
00  C0  0A  0A  00  2B  00  00  01  10  2B  2B  00  08  00  0008D                  0, 0, 26, 0, 8, 8, -65, 0, 0, 0, 8, 0, -    ;
C2  1B  1B  00  0B  00  00  00  C1  0B  0B  00  0A  00  00  0009C                  43, 43, 16, 1, 0, 0, 43, 0, 10, 10, -64, -  ;
        00  1C  00  00  00  C3  1C  1C  00  1B  00  00  00  000AB                  0, 0, 0, 10, 0, 11, 11, -63, 0, 0, 0, 11, - ;
                                                                                   0, 27, 27, -62, 0, 0, 0, 27, 0, 28, 28, -   ;
                                                                                   -61, 0, 0, 0, 28, 0                         ;
                                                 00000016  000B8          .LONG   22                                          ;
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06  000BC P.AAE:   .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -  ;
00  00  01  0B  09  09  00  15  00  00  00  F7  15  15  00  000CB                  0, 0, 7, 0, 21, 21, -9, 0, 0, 0, 21, 0, -   ;
00  00  B3  08  08  00  1A  00  00  01  0D  1A  1A  00  09  000DA                  9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -   ;
00  B5  0A  0A  00  2B  00  00  01  11  2B  2B  00  08  00  000E9                  0, 0, 26, 0, 8, 8, -77, 0, 0, 0, 8, 0, -    ;
B7  1B  1B  00  0B  00  00  00  B6  0B  0B  00  0A  00  00  000F8                  43, 43, 17, 1, 0, 0, 43, 0, 10, 10, -75, -  ;
        00  1C  00  00  00  B8  1C  1C  00  1B  00  00  00  00107                  0, 0, 0, 10, 0, 11, 11, -74, 0, 0, 0, 11, - ;
                                                                                   0, 27, 27, -73, 0, 0, 0, 27, 0, 28, 28, -   ;
                                                                                   -72, 0, 0, 0, 28, 0                         ;
                                                 0000000C  00114          .LONG   12                                          ;
2B  00  00  01  12  2B  2B  00  08  00  00  00  CA  08  08  00118 P.AAF:   .BYTE   8, 8, -54, 0, 0, 0, 8, 0, 43, 43, 18, 1, -  ;
00  00  00  CC  0B  0B  00  0A  00  00  00  CB  0A  0A  00  00127                  0, 0, 43, 0, 10, 10, -53, 0, 0, 0, 10, 0, -  ;
00  00  CE  1C  1C  00  1B  00  00  00  CD  1B  1B  00  0B  00136                  11, 11, -52, 0, 0, 0, 11, 0, 27, 27, -51, - ;
                                                    00  1C  00  00145            0, 0, 0, 27, 0, 28, 28, -50, 0, 0, 0, 28, - ;
                                                                                   0                                          ;
                                                 0000000C  00148          .LONG   12                                          ;
2B  00  00  01  13  2B  2B  00  08  00  00  00  05  08  08  0014C P.AAG:   .BYTE   8, 8, 5, 0, 0, 0, 8, 0, 43, 43, 19, 1, 0, - ;
00  00  00  08  0B  0B  00  0A  00  00  00  07  0A  0A  00  0015B                  0, 43, 0, 10, 10, 7, 0, 0, 0, 10, 0, 11, -  ;
```

```
00 00 0A 1C 1C 00 1B 00 00 00 09 1B 1B 00 0B 0016A                          11, 8, 0, 0, 0, 11, 0, 27, 27, 9, 0, 0, -
                                    00 1C 00 00179                          0, 27, 0, 28, 28, 10, 0, 0, 0, 28, 0
                                       0000000C 0017C           .LONG       12
2B 00 00 01 14 2B 2B 00 08 00 00 00 13 08 08 00180 P.AAH:   .BYTE    8, 8, 19, 0, 0, 0, 8, 0, 43, 43, 20, 1, -
00 00 00 16 0B 0B 00 0A 00 00 00 15 0A 0A 00 0018F                          0, 0, 43, 0, 10, 10, 21, 0, 0, 0, 10, 0, -
00 00 18 1C 1C 00 1B 00 00 00 17 1B 1B 00 0B 0019E                          11, 11, 22, 0, 0, 0, 11, 0, 27, 27, 23, -
                                    00 1C 00 001AD                          0, 0, 0, 27, 0, 28, 28, 24, 0, 0, 0, 28, 0
                                       0000000C 001B0           .LONG       12
2B 00 00 01 15 2B 2B 00 08 00 00 00 2F 08 08 001B4 P.AAI:   .BYTE    8, 8, 47, 0, 0, 0, 8, 0, 43, 43, 21, 1, -
00 00 00 32 0B 0B 00 0A 00 00 00 31 0A 0A 00 001C3                          0, 0, 43, 0, 10, 10, 49, 0, 0, 0, 10, 0, -
00 00 34 1C 1C 00 1B 00 00 00 33 1B 1B 00 0B 001D2                          11, 11, 50, 0, 0, 0, 11, 0, 27, 27, 51, -
                                    00 1C 00 001E1                          0, 0, 0, 27, 0, 28, 28, 52, 0, 0, 0, 28, 0
                                       0000000C 001E4           .LONG       12
2B 00 00 01 16 2B 2B 00 08 00 00 00 21 08 08 001E8 P.AAJ:   .BYTE    8, 8, 33, 0, 0, 0, 8, 0, 43, 43, 22, 1, -
00 00 00 24 0B 0B 00 0A 00 00 00 23 0A 0A 00 001F7                          0, 0, 43, 0, 10, 10, 35, 0, 0, 0, 10, 0, -
00 00 26 1C 1C 00 1B 00 00 00 25 1B 1B 00 0B 00206                          11, 11, 36, 0, 0, 0, 11, 0, 27, 27, 37, -
                                    00 1C 00 00215                          0, 0, 0, 27, 0, 28, 28, 38, 0, 0, 0, 28, 0
                                       00000002 00218           .LONG       2
                        00 08 00 00 00 39 08 08 0021C P.AAK:   .BYTE    8, 8, 57, 0, 0, 0, 8, 0
                                       00000002 00224           .LONG       2
                        00 08 00 00 00 3B 08 08 00228 P.AAL:   .BYTE    8, 8, 59, 0, 0, 0, 8, 0
                                       0000000A 00230           .LONG       10
0A 00 00 00 49 0A 0A 00 08 00 00 00 41 08 08 00234 P.AAM:   .BYTE    8, 8, 65, 0, 0, 0, 8, 0, 10, 10, 73, 0, -
00 00 00 4D 1B 1B 00 0B 00 00 00 4C 0B 0B 00 00243                          0, 0, 10, 0, 11, 11, 76, 0, 0, 0, 11, 0, -
                  00 1C 00 00 00 4E 1C 1C 00 1B 00252                       27, 27, 77, 0, 0, 0, 27, 0, 28, 28, 78, -
                                                                            0, 0, 0, 28, 0
                                       00000002 0025C           .LONG       2
                        00 28 00 00 00 AA 28 28 00260 P.AAN:   .BYTE    40, 40, -86, 0, 0, 0, 40, 0
                                       00000002 00268           .LONG       2
                        00 28 00 00 00 A6 28 28 0026C P.AAO:   .BYTE    40, 40, -90, 0, 0, 0, 40, 0
                                       00000002 00274           .LONG       2
                        00 28 00 00 00 AE 28 28 00278 P.AAP:   .BYTE    40, 40, -82, 0, 0, 0, 40, 0
                                       00000002 00280           .LONG       2
                        00 28 00 00 00 B0 28 28 00284 P.AAQ:   .BYTE    40, 40, -80, 0, 0, 0, 40, 0
                                       00000010 0028C           .LONG       16
28 00 00 00 61 08 08 00 28 00 01 00 61 2F 2F 00290 P.AAR:   .BYTE    47, 47, 97, 0, 1, 0, 40, 0, 8, 8, 97, 0, -
00 00 00 62 0A 0A 00 28 00 00 01 17 2B 2B 00 0029F                          0, 0, 40, 0, 43, 43, 23, 1, 0, 0, 40, 0, -
00 00 64 1B 1B 00 28 00 00 00 63 0B 0B 00 28 002AE                          10, 10, 98, 0, 0, 0, 40, 0, 11, 11, 99, -
00 FF 28 28 00 28 00 00 00 65 1C 1C 00 28 00 002BD                          0, 0, 0, 40, 0, 27, 27, 100, 0, 0, 0, 40, -
                                    00 28 00 00 002CC                       0, 28, 28, 101, 0, 0, 0, 40, 0, 40, 40, -
                                                                            -1, 0, 0, 0, 40, 0
                                       00000010 002D0           .LONG       16
28 00 00 00 6C 08 08 00 28 00 01 00 6C 2F 2F 002D4 P.AAS.   .BYTE    47, 47, 108, 0, 1, 0, 40, 0, 8, 8, 108, -
00 00 00 6D 0A 0A 00 28 00 00 01 18 2B 2B 00 002E3                          0, 0, 0, 40, 0, 43, 43, 24, 1, 0, 0, 40, -
00 00 6F 1B 1B 00 28 00 00 00 6E 0B 0B 00 28 002F2                          0, 10, 10, 109, 0, 0, 0, 40, 0, 11, 11, -
01 00 28 28 00 28 00 00 00 70 1C 1C 00 28 00 00301                          110, 0, 0, 0, 40, 0, 27, 27, 111, 0, 0, -
                                    00 28 00 00 00310                       0, 40, 0, 28, 28, 112, 0, 0, 0, 40, 0, -
                                                                            40, 40, 0, 1, 0, 0, 40, 0
                                       00000010 00314           .LONG       16
28 00 00 00 8F 08 08 00 28 00 01 00 8F 2F 2F 00318 P.AAT:   .BYTE    47, 47, -113, 0, 1, 0, 40, 0, 8, 8, -113, -
00 00 00 91 0A 0A 00 28 00 00 01 19 2B 2B 00 00327                          0, 0, 0, 40, 0, 43, 43, 25, 1, 0, 0, 40, -
00 00 93 1B 1B 00 28 00 00 00 92 0B 0B 00 28 00336                          0, 10, 10, -111, 0, 0, 0, 40, 0, 11, 11, -
01 03 28 28 00 28 00 00 00 94 1C 1C 00 28 00 00345                          -110, 0, 0, 0, 40, 0, 27, 27, -109, 0, 0, -
                                    00 28 00 00 00354                       0, 40, 0, 28, 28, -108, 0, 0, 0, 40, 0, -
                                                                            40, 40, 3, 1, 0, 0, 40, 0
                                       00000010 00358           .LONG       16
28 00 00 00 7F 08 08 00 28 00 01 00 7F 2F 2F 0035C P.AAU:   .BYTE    47, 47, 127, 0, 1, 0, 40, 0, 8, 8, 127, -
```

```
00  00  00  81  0A  0A  00  28  00  00  01  1A  2B  2B  00  0036B                    0, 0, 0, 40, 0, 43, 43, 26, 1, 0, 0, 40, -
00  00  83  1B  1B  00  28  00  00  00  82  0B  0B  00  28  0037A                    0, 10, 10, -127, 0, 0, 0, 40, 0, 11, 11, -
01  01  28  28  00  28  00  00  00  84  1C  1C  00  28  00  00389                    -126, 0, 0, 0, 40, 0, 27, 27, -125, 0, 0, -
                                          00  28  00  00  00398                    0, 40, 0, 28, 28, -124, 0, 0, 0, 40, 0, -
                                                                                   40, 40, 1, 1, 0, 0, 40, 0
                                              00000010  0039C            .LONG    16
28  00  00  00  87  08  08  00  28  00  01  00  87  2F  2F  003A0  P.AAV: .BYTE    47, 47, -121, 0, 1, 0, 40, 0, 8, 8, -121, -
00  00  00  89  0A  0A  00  28  00  00  01  1B  2B  2B  00  003AF                    0, 0, 0, 40, 0, 43, 43, 27, 1, 0, 0, 40, -
00  00  8B  1B  1B  00  28  00  00  00  8A  0B  0B  00  28  003BE                    0, 10, 10, -119, 0, 0, 0, 40, 0, 11, 11, -
01  04  28  28  00  28  00  00  00  8C  1C  1C  00  28  00  003CD                    -118, 0, 0, 0, 40, 0, 27, 27, -117, 0, 0, -
                                          00  28  00  00  003DC                    0, 40, 0, 28, 28, -116, 0, 0, 0, 40, 0, -
                                                                                   40, 40, 4, 1, 0, 0, 40, 0
                                              00000010  003E0            .LONG    16
28  00  00  00  77  08  08  00  28  00  01  00  77  2F  2F  003E4  P.AAW: .BYTE    47, 47, 119, 0, 1, 0, 40, 0, 8, 8, 119, -
00  00  00  79  0A  0A  00  28  00  00  01  1C  2B  2B  00  003F3                    0, 0, 0, 40, 0, 43, 43, 28, 1, 0, 0, 40, -
00  00  7B  1B  1B  00  28  00  00  00  7A  0B  0B  00  28  00402                    0, 10, 10, 121, 0, 0, 0, 40, 0, 11, 11, -
01  02  28  28  00  28  00  00  00  7C  1C  1C  00  28  00  00411                    122, 0, 0, 0, 40, 0, 27, 27, 123, 0, 0, -
                                          00  28  00  00  00420                    0, 40, 0, 28, 28, 124, 0, 0, 0, 40, 0, -
                                                                                   40, 40, 2, 1, 0, 0, 40, 0
                                              00000002  00424            .LONG    2
                                00  0E  00  00  00  52  0E  0E  00428  P.AAX: .BYTE    14, 14, 82, 0, 0, 0, 14, 0
                                                      00#  00430  ADA_OPINFO_TABLE:
                                                                   .BYTE    0[64]
                  00000000  00000014  00000060  00470            .LONG    96, 20, 0
                                                      01  0047C            .BYTE    1
                                                      00#  0047D            .BYTE    0[3]
                  00000000  00000014  000000BC  00480            .LONG    188, 20, 0
                                                      01  0048C            .BYTE    1
                                                      00#  0048D            .BYTE    0[3]
                  00000000  00000014  0000014C  00490            .LONG    332, 20, 0
                                                      01  0049C            .BYTE    1
                                                      00#  0049D            .BYTE    0[3]
                  00000000  00000014  00000180  004A0            .LONG    384, 20, 0
                                                      01  004AC            .BYTE    1
                                                      00#  004AD            .BYTE    0[3]
                  00000000  00000014  000001B4  004B0            .LONG    436, 20, 0
                                                      01  004BC            .BYTE    1
                                                      00#  004BD            .BYTE    0[3]
                  00000000  00000014  000001E8  004C0            .LONG    488, 20, 0
                                                      01  004CC            .BYTE    1
                                                      00#  004CD            .BYTE    0[3]
                  00000000  00000014  00000234  004D0            .LONG    564, 20, 0
                                                      01  004DC            .BYTE    1
                                                      00#  004DD            .BYTE    0[35]
                  00000000  00000014  00000290  00500            .LONG    656, 20, 0
                                                      01  0050C            .BYTE    1
                                                      00#  0050D            .BYTE    0[3]
                  00000000  00000014  000002D4  00510            .LONG    724, 20, 0
                                                      01  0051C            .BYTE    1
                                                      00#  0051D            .BYTE    0[3]
                  00000000  00000014  0000035C  00520            .LONG    860, 20, 0
                                                      01  0052C            .BYTE    1
                                                      00#  0052D            .BYTE    0[19]
                  00000000  00000014  000003E4  00540            .LONG    996, 20, 0
                                                      01  0054C            .BYTE    1
                                                      00#  0054D            .BYTE    0[19]
                  00000000  00000014  00000318  00560            .LONG    792, 20, 0
```

```
                                    01  0056C          .BYTE   1
                                    00# 0056D          .BYTE   0[19]
                00000000  00000014  000003A0  00580   .LONG   928, 20, 0
                                    01  0058C          .BYTE   1
                                    00# 0058D          .BYTE   0[19]
                00000000  00000014  00000260  005A0   .LONG   608, 20, 0
                                    01  005AC          .BYTE   1
                                    00# 005AD          .BYTE   0[3]
                00000000  00000014  0000026C  005B0   .LONG   620, 20, 0
                                    01  005BC          .BYTE   1
                                    00# 005BD          .BYTE   0[3]
                00000000  00000014  00000278  005C0   .LONG   632, 20, 0
                                    01  005CC          .BYTE   1
                                    00# 005CD          .BYTE   0[3]
                00000000  00000014  00000284  005D0   .LONG   644, 20, 0
                                    01  005DC          .BYTE   1
                                    00# 005DD          .BYTE   0[131]
                00000000  00000014  00000428  00660   .LONG   1064, 20, 0
                                    01  0066C          .BYTE   1
                                    00# 0066D          .BYTE   0[3]
                00000000  00000014  0000021C  00670   .LONG   540, 20, 0
                                    01  0067C          .BYTE   1
                                    00# 0067D          .BYTE   0[3]
                00000000  00000014  00000228  00680   .LONG   552, 20, 0
                                    01  0068C          .BYTE   1
                                    00# 0068D          .BYTE   0[83]
                00000000  00000014  00000118  006E0   .LONG   280, 20, 0
                                    01  006EC          .BYTE   1
                                    00# 006ED          .BYTE   0[99]
                00000000  00000034  00000000  00750   .LONG   0, 52, 0
                                    01  0075C          .BYTE   1
                                    00# 0075D          .BYTE   0[3]
                00000000  00000034  00000000  00760   .LONG   0, 52, 0
                                    01  0076C          .BYTE   1
                                    00# 0076D          .BYTE   0[67]
                00000000  00000000  00000000  007B0   .LONG   0, 0, 0
                                    01  007BC          .BYTE   1
                                        007BD          .BLKB   179
                          00000000  00000000  00870 BASIC_CVTINFO_TABLE:
                                                     .LONG   0, 0
                                    00  00878          .BYTE   0
                                        00879          .BLKB   3
                                  00000005  0087C      .LONG   5
      0000  1C1B  1C0B  1B0A  0B0A  1508  0A08  0807  0706  00880 P.AAY:  .WORD   1798, 2055, 2568, 5384, 2826, 6922, 7179, -
                                                                         7195, 0
                                        00892          .BLKB   2
                                  00000002  00894      .LONG   2
                                0000  0807  0706  00898 P.AAZ:  .WORD   1798, 2055, 0
                                        0089E          .BLKB   2
                                  00000005  008A0      .LONG   5
      0000  061C  1C1B  1B0B  0B0A  0A15  1508  0807  0706  008A4 P.ABA:  .WORD   1798, 2055, 5384, 2581, 2826, 6923, 7195, -
                                                                         1564, 0
                                        008B6          .BLKB   2
                                  00000012  008B8      .LONG   18
07  00  00  00  03  07  07  00  06  00  00  00  01  06  06  008BC P.ABB:  .BYTE   6, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, -
00  00  00  07  0A  0A  00  08  00  00  00  05  08  08  00  008CB                  7, 0, 8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, -
00  00  09  1B  1B  00  0B  00  00  00  08  0B  0B  00  0A  008DA                  0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 0, 11, -
```

```
00 F2 15 15 00 1C 00 00 00 0A 1C 1C 00 1B 00 008E9                      0, 27, 27, 9, 0, 0, 0, 27, 0, 28, 28, 10, -
      00 0E 00 00 00 52 0E 0E 00 15 00 00 008F8                         0, 0, 0, 28, 0, 21, 21, -14, 0, 0, 0, 21, -
                                                                        0, 14, 14, 82, 0, 0, 0, 14, 0
                                  00000010 00904                .LONG   16
07 00 00 00 11 07 07 00 06 00 00 00 0F 06 06 00908 P.ABC:       .BYTE   6, 6, 15, 0, 0, 0, 6, 0, 7, 7, 17, 0, 0, -
0C 00 00 15 0A 0A 00 08 00 00 00 13 08 08 00 00917                      0, 7, 0, 8, 8, 19, 0, 0, 0, 8, 0, 10, 10, -
00 00 17 1B 1B 00 0B 00 00 00 16 0B 0B 00 0A 00926                      21, 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 0, -
00 F3 15 15 00 1C 00 00 00 18 1C 1C 00 1B 00 00935                      11, 0, 27, 27, 23, 0, 0, 0, 27, 0, 28, -
                              00 15 00 00 00944                         28, 24, 0, 0, 0, 28, 0, 21, 21, -13, 0, -
                                                                        0, 0, 21, 0
                                  00000010 00948                .LONG   16
   00 00 00 2D 07 07 00 06 00 00 00 2B 06 06 0094C P.ABD:       .BYTE   6, 6, 43, 0, 0, 0, 6, 0, 7, 7, 45, 0, 0, -
   00 00 31 0A 0A 00 08 00 00 00 2F 08 08 00 00958                      0, 7, 0, 8, 8, 47, 0, 0, 0, 8, 0, 10, 10, -
   00 33 1B 1B 00 0B 00 00 00 32 0B 0B 00 0A 00969                      49, 0, 0, 0, 10, 0, 11, 11, 50, 0, 0, 0, -
   34 15 15 00 1C 00 00 00 34 1C 1C 00 1B 00 00978                      11, 0, 27, 27, 51, 0, 0, 0, 27, 0, 28, -
                              00 15 00 00 00987                         28, 52, 0, 0, 0, 28, 0, 21, 21, -12, 0, -
                                                                        0, 0, 21, 0
                                  00000010 0098B                .LONG   16
   00 00 00 1F 07 07 00 06 00 00 00 1D 06 06 0098F P.ABE:       .BYTE   6, 6, 29, 0, 0, 0, 6, 0, 7, 7, 31, 0, 0, -
   00 00 23 0A 0A 00 08 00 00 00 21 08 08 00 0099E                      0, 7, 0, 8, 8, 33, 0, 0, 0, 8, 0, 10, 10, -
00 00 25 1B 1B 00 0B 00 00 00 24 0B 0B 00 0A 009AD                      35, 0, 0, 0, 10, 0, 11, 11, 36, 0, 0, 0, -
00 F5 15 15 00 1C 00 00 00 26 1C 1C 00 1B 00 009BC                      11, 0, 27, 27, 37, 0, 0, 0, 27, 0, 28, -
                              00 15 00 00 009CB                         28, 38, 0, 0, 0, 28, 0, 21, 21, -11, 0, -
                                                                        0, 0, 21, 0
                                  00000014 009CF                .LONG   20
1A 00 00 01 0C 1A 1A 00 09 00 00 01 0A 09 09 009D3 P.ABF:       .BYTE   9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -
00 00 00 BE 07 07 00 06 00 00 00 BD 06 06 00 009E2                      0, 0, 26, 0, 6, 6, -67, 0, 0, 0, 6, 0, 7, -
00 00 C0 0A 0A 00 08 00 00 00 BF 08 08 00 07 009F1                      7, -66, 0, 0, 0, 7, 0, 8, 8, -65, 0, 0, -
00 C2 1B 1B 00 0B 00 00 00 C1 0B 0B 00 0A 00 00A00                      0, 8, 0, 10, 10, -64, 0, 0, 0, 10, 0, 11, -
F6 15 15 00 1C 00 00 00 C3 1C 1C 00 1B 00 00 00A0F                      11, -63, 0, 0, 0, 11, 0, 27, 27, -62, 0, -
                              00 15 00 00 00 00A1E                      0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, 0, -
                                                                        21, 21, -10, 0, 0, 0, 21, 0
                                  00000014 00A23                .LONG   20
1A 00 00 01 0D 1A 1A 00 09 00 00 01 0B 09 09 00A27 P.ABG:       .BYTE   9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -
00 00 00 B2 07 07 00 06 00 00 00 B1 06 06 00 00A36                      0, 0, 26, 0, 6, 6, -79, 0, 0, 0, 6, 0, 7, -
00 00 B5 0A 0A 00 08 00 00 00 B3 08 08 00 07 00A45                      7, -78, 0, 0, 0, 7, 0, 8, 8, -77, 0, 0, -
00 B7 1B 1B 00 0B 00 00 00 B6 0B 0B 00 0A 00 00A54                      0, 8, 0, 10, 10, -75, 0, 0, 0, 10, 0, 11, -
F7 15 15 00 1C 00 00 00 B8 1C 1C 00 1B 00 00 00A63                      11, -74, 0, 0, 0, 11, 0, 27, 27, -73, 0, -
                              00 15 00 00 00 00A72                      0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, 0, -
                                                                        21, 21, -9, 0, 0, 0, 21, 0
                                  00000018 00A77                .LONG   24
08 00 00 00 41 08 08 00 07 00 00 00 40 07 07 00A7B P.ABH:       .BYTE   7, 7, 64, 0, 0, 0, 7, 0, 8, 8, 65, 0, 0, -
00 00 00 43 0B 08 00 0A 00 00 00 42 0A 08 00 00A8A                      0, 8, 0, 8, 10, 66, 0, 0, 0, 10, 0, 8, -
00 00 45 1C 08 00 1B 00 00 00 44 1B 08 00 0B 00A99                      11, 67, 0, 0, 0, 11, 0, 8, 27, 68, 0, 0, -
00 4A 0B 0A 00 0A 00 00 00 49 0A 0A 00 1C 00 00AA8                      0, 27, 0, 8, 28, 69, 0, 0, 0, 28, 0, 10, -
4C 0B 0B 00 0B 00 00 00 4B 0A 0B 00 0B 00 00 00AB7                      10, 73, 0, 0, 0, 10, 0, 10, 11, 74, 0, 0, -
1C 1C 00 1B 00 00 00 4D 1B 1B 00 0B 00 00 00 00AC6                      0, 11, 0, 11, 10, 75, 0, 0, 0, 11, 0, 11, -
                              00 1C 00 00 00 4E 00AD5                   11, 76, 0, 0, 0, 11, 0, 27, 27, 77, 0, 0, -
                                                                        0, 27, 0, 28, 28, 78, 0, 0, 0, 28, 0
                                  00000014 00ADB                .LONG   20
08 00 00 00 54 0E 0E 00 08 00 00 01 08 3F 3F 00ADF P.ABI:       .BYTE   63, 63, 8, 1, 0, 0, 8, 0, 14, 14, 84, 0, -
00 00 00 60 07 07 00 08 00 00 00 5F 06 06 00 00EE                       0, 0, 8, 0, 6, 6, 95, 0, 0, 0, 8, 0, 7, -
00 00 62 0A 0A 00 08 00 00 00 61 08 08 00 08 00FD                       7, 96, 0, 0, 0, 8, 0, 8, 8, 97, 0, 0, 0, -
00 64 1B 1B 00 08 00 00 00 63 0B 0B 00 08 00 00C                        8, 0, 10, 10, 98, 0, 0, 0, 8, 0, 11, 11, -
F8 15 15 00 08 00 00 00 65 1C 1C 00 08 00 00 01B                        99, 0, 0, 0, 8, 0, 27, 27, 100, 0, 0, 0, -
                              00 08 00 00 00 02A                        8, 0, 28, 28, 101, 0, 0, 0, 8, 0, 21, 21, -
                                                                        -8, 0, 0, 0, 8, 0
```

```
                                                           00000014  00B30          .LONG   20                                                  ;
08  00  00  00  5E  0E  0E  00  08  00  00  01  09  3F  3F  00B34 P.ABJ:  .BYTE   63, 63, 9, 1, 0, 0, 8, 0, 14, 14, 94, 0, -        ;
00  00  00  6B  07  07  00  08  00  00  00  6A  06  06  00  00B43                 0, 0, 8, 0, 6, 6, 106, 0, 0, 0, 8, 0, 7, -        ;
00  00  6D  0A  0A  00  08  00  00  00  6C  08  08  00  08  00B52                 7, 107, 0, 0, 0, 8, 0, 8, 8, 108, 0, 0, -         ;
00  6F  1B  1B  00  08  00  00  00  6E  0B  0B  00  08  00  00B61                 0, 8, 0, 10, 10, 109, 0, 0, 0, 8, 0, 11, -        ;
F9  15  15  00  08  00  00  00  70  1C  1C  00  08  00  00  00B70                 11, 110, 0, 0, 0, 8, 0, 27, 27, 111, 0, -         ;
                                            00  08  00  00  00  00B7F                 0, 0, 8, 0, 28, 28, 112, 0, 0, 0, 8, 0, -         ;
                                                                                  21, 21, -7, 0, 0, 0, 8, 0                          ;
                                                           00000012  00B84          .LONG   18                                                  ;
08  00  00  00  7D  06  06  00  08  00  00  00  58  0E  0E  00B88 P.ABK:  .BYTE   14, 14, 88, 0, 0, 0, 8, 0, 6, 6, 125, 0, -        ;
00  00  00  7F  08  08  00  08  00  00  00  7E  07  07  00  00B97                 0, 0, 8, 0, 7, 7, 126, 0, 0, 0, 8, 0, 8, -        ;
00  00  82  0B  0B  00  08  00  00  00  81  0A  0A  00  08  00BA6                 8, 127, 0, 0, 0, 8, 0, 10, 10, -127, 0, -         ;
00  84  1C  1C  00  08  00  00  00  83  1B  1B  00  08  00  00BB5                 0, 0, 8, 0, 11, 11, -126, 0, 0, 0, 8, 0, -        ;
                00  08  00  00  00  FA  15  15  00  08  00  00  00BC4                 27, 27, -125, 0, 0, 0, 8, 0, 28, 28, -            ;
                                                                                  -124, 0, 0, 0, 8, 0, 21, 21, -6, 0, 0, 0, -       ;
                                                                                  8, 0                                               ;
                                                           00000012  00BD0          .LONG   18                                                  ;
08  00  00  00  75  06  06  00  08  00  00  00  56  0E  0E  00BD4 P.ABL:  .BYTE   14, 14, 86, 0, 0, 0, 8, 0, 6, 6, 117, 0, -        ;
00  00  00  77  08  08  00  08  00  00  00  76  07  07  00  00BE3                 0, 0, 8, 0, 7, 7, 118, 0, 0, 0, 8, 0, 8, -        ;
00  00  7A   3  0B  00  08  00  00  00  79  0A  0A  00  08  00BF2                 8, 119, 0, 0, 0, 8, 0, 10, 10, 121, 0, 0, -       ;
00  7C  1C   C  00  08  00  00  00  7B  1B  1B  00  08  00  00C01                 0, 8, 0, 11, 11, 122, 0, 0, 0, 8, 0, 27, -        ;
                00  08  00  00  00  FB  15  15  00  08  00  00  00C10                 27, 123, 0, 0, 0, 8, 0, 28, 28, 124, 0, -         ;
                                                                                  0, 0, 8, 0, 21, 21, -5, 0, 0, 0, 8, 0              ;
                                                           00000012  00C1C          .LONG   18                                                  ;
08  00  00  00  8D  06  06  00  08  00  00  00  5C  0E  0E  00C20 P.ABM:  .BYTE   14, 14, 92, 0, 0, 0, 8, 0, 6, 6, -115, 0, -       ;
00  00  00  8F  08  08  00  08  00  00  00  8E  07  07  00  00C2F                 0, 0, 8, 0, 7, 7, -114, 0, 0, 0, 8, 0, 8, -       ;
00  00  92  0B  0B  00  08  00  00  00  91  0A  0A  00  08  00C3E                 8, -113, 0, 0, 0, 8, 0, 10, 10, -111, 0, -        ;
00  94  1C  1C  00  08  00  00  00  93  1B  1B  00  08  00  00C4D                 0, 0, 8, 0, 11, 11, -110, 0, 0, 0, 8, 0, -        ;
                00  08  00  00  00  FC  15  15  00  08  00  00  00C5C                 27, 27, -109, 0, 0, 0, 8, 0, 28, 28, -            ;
                                                                                  -108, 0, 0, 0, 8, 0, 21, 21, -4, 0, 0, 0, -       ;
                                                                                  8, 0                                               ;
                                                           00000012  00C68          .LONG   18                                                  ;
08  00  00  00  85  06  06  00  08  00  00  00  5A  0E  0E  00C6C P.ABN:  .BYTE   14, 14, 90, 0, 0, 0, 8, 0, 6, 6, -123, 0, -       ;
00  00  00  87  08  08  00  08  00  00  00  86  07  07  00  00C7B                 0, 0, 8, 0, 7, 7, -122, 0, 0, 0, 8, 0, 8, -       ;
00  00  8A  0B  0B  00  08  00  00  00  89  0A  0A  00  08  00C8A                 8, -121, 0, 0, 0, 8, 0, 10, 10, -119, 0, -        ;
00  8C  1C  1C  00  08  00  00  00  8B  1B  1B  00  08  00  00C99                 0, 0, 8, 0, 11, 11, -118, 0, 0, 0, 8, 0, -        ;
                00  08  00  00  00  FD  15  15  00  08  00  00  00CA8                 27, 27, -117, 0, 0, 0, 8, 0, 28, 28, -            ;
                                                                                  -116, 0, 0, 0, 8, 0, 21, 21, -3, 0, 0, 0, -       ;
                                                                                  8, 0                                               ;
                                                           00000006  00CB4          .LONG   6                                                   ;
07  00  00  00  9C  07  07  00  06  00  00  00  9B  06  06  00CB8 P.ABO:  .BYTE   6, 6, -101, 0, 0, 0, 6, 0, 7, 7, -100, 0, -       ;
                            00  08  00  00  00  9D  08  08  00  00CC7                 0, 0, 7, 0, 8, 8, -99, 0, 0, 0, 8, 0               ;
                                                           00000006  00CD0          .LONG   6                                                   ;
07  00  00  00  96  07  07  00  06  00  00  00  95  06  06  00CD4 P.ABP:  .BYTE   6, 6, -107, 0, 0, 0, 6, 0, 7, 7, -106, 0, -       ;
                            00  08  00  00  00  97  08  08  00  00CE3                 0, 0, 7, 0, 8, 8, -105, 0, 0, 0, 8, 0              ;
                                                           00000006  00CEC          .LONG   6                                                   ;
07  00  00  00  9F  07  07  00  06  00  00  00  9E  06  06  00CF0 P.ABQ:  .BYTE   6, 6, -98, 0, 0, 0, 6, 0, 7, 7, -97, 0, -         ;
                            00  08  00  00  00  A0  08  08  00  00CFF                 0, 0, 7, 0, 8, 8, -96, 0, 0, 0, 8, 0               ;
                                                           00000006  00D08          .LONG   6                                                   ;
07  00  00  00  A2  07  07  00  06  00  00  00  A1  06  06  00D0C P.ABR:  .BYTE   6, 6, -95, 0, 0, 0, 6, 0, 7, 7, -94, 0, -         ;
                            00  08  00  00  00  A3  08  08  00  00D1B                 0, 0, 7, 0, 8, 8, -93, 0, 0, 0, 8, 0               ;
                                                           00000006  00D24          .LONG   6                                                   ;
07  00  00  00  99  07  07  00  06  00  00  00  98  06  06  00D28 P.ABS:  .BYTE   6, 6, -104, 0, 0, 0, 6, 0, 7, 7, -103, 0, -       ;
                            00  08  00  00  00  9A  08  08  00  00D37                 0, 0, 7, 0, 8, 8, -102, 0, 0, 0, 8, 0              ;
                                                           00000006  00D40          .LONG   6                                                   ;
07  00  00  00  E0  07  07  00  06  00  00  00  DF  06  06  00D44 P.ABT:  .BYTE   6, 6, -33, 0, 0, 0, 6, 0, 7, 7, -32, 0, -         ;
                            00  08  00  00  00  E1  08  08  00  00D53                 0, 0, 7, 0, 8, 8, -31, 0, 0, 0, 8, 0               ;
```

```
                                    00# 00D5C BASIC_OPINFO_TABLE:
                                                .BYTE   0[64]
00000000  00000880  000009D4  00D9C            .LONG   2516, 2176, 0
                            01  00DA8          .BYTE   1
                            00# 00DA9          .BYTE   0[3]
00000000  00000880  00000A28  00DAC            .LONG   2600, 2176, 0
                            01  00DB8          .BYTE   1
                            00# 00DB9          .BYTE   0[3]
00000000  00000880  000008BC  00DBC            .LONG   2236, 2176, 0
                            01  00DC8          .BYTE   1
                            00# 00DC9          .BYTE   0[3]
00000000  00000880  00000908  00DCC            .LONG   2312, 2176, 0
                            01  00DD8          .BYTE   1
                            00# 00DD9          .BYTE   0[3]
00000000  00000880  0000094C  00DDC            .LONG   2380, 2176, 0
                            01  00DE8          .BYTE   1
                            00# 00DE9          .BYTE   0[3]
00000000  00000880  00000990  00DEC            .LONG   2448, 2176, 0
                            01  00DF8          .BYTE   1
                            00# 00DF9          .BYTE   0[3]
00000000  00000880  00000A7C  00DFC            .LONG   2684, 2176, 0
                            01  00E08          .BYTE   1
                            00# 00E09          .BYTE   0[35]
00000000  00000880  00000AE0  00E2C            .LONG   2784, 2176, 0
                            01  00E38          .BYTE   1
                            00# 00E39          .BYTE   0[3]
00000000  00000880  00000B34  00E3C            .LONG   2868, 2176, 0
                            01  00E48          .BYTE   1
                            00# 00E49          .BYTE   0[3]
00000000  00000880  00000B88  00E4C            .LONG   2952, 2176, 0
                            01  00E58          .BYTE   1
                            00# 00E59          .BYTE   0[19]
00000000  00000880  00000BD4  00E6C            .LONG   3028, 2176, 0
                            01  00E78          .BYTE   1
                            00# 00E79          .BYTE   0[19]
00000000  00000880  00000C20  00E8C            .LONG   3104, 2176, 0
                            01  00E98          .BYTE   1
                            00# 00E99          .BYTE   0[19]
00000000  00000880  00000C6C  00EAC            .LONG   3180, 2176, 0
                            01  00EB8          .BYTE   1
                            00# 00EB9          .BYTE   0[131]
00000000  00000898  00000CB8  00F3C            .LONG   3256, 2200, 0
                            01  00F48          .BYTE   1
                            00# 00F49          .BYTE   0[3]
00000000  00000898  00000CD4  00F4C            .LONG   3284, 2200, 0
                            01  00F58          .BYTE   1
                            00# 00F59          .BYTE   0[3]
00000000  00000898  00000CF0  00F5C            .LONG   3312, 2200, 0
                            01  00F68          .BYTE   1
                            00# 00F69          .BYTE   0[3]
00000000  00000898  00000D0C  00F6C            .LONG   3340, 2200, 0
                            01  00F78          .BYTE   1
                            00# 00F79          .BYTE   0[3]
00000000  00000898  00000D28  00F7C            .LONG   3368, 2200, 0
                            01  00F88          .BYTE   1
                            00# 00F89          .BYTE   0[243]
00000000  000008A4  00000000  0107C            .LONG   0, 2212, 0
```

```
                                                    01  01088          .BYTE   1
                                                    00# 01089          .BYTE   0[3]
                                00000000  00000898  00000000  0108C    .LONG   0, 2200, 0
                                                    01  01098          .BYTE   1
                                                    00# 01099          .BYTE   0[67]
                                00000000  00000000  00000000  010DC    .LONG   0, 0, 0
                                                    01  010E8          .BYTE   1
                                                    00# 010E9          .BYTE   0[19]
                                00000000  00000898  00000D44  010FC    .LONG   3396, 2200, 0
                                                    01  01108          .BYTE   1
                                                        01109          .BLKB   147
                                                  00000001  0119C      .LONG   1
                                                     083B   011A0 P.ABU: .WORD   2107
                                                        011A2          .BLKB   2
                                          00000000  000011A0  011A4 BLISS_CVTINFO_TABLE:
                                                                       .LONG   4512, 0
                                                    00  011AC          .BYTE   0
                                                        011AD          .BLKB   3
                                                  00000003  011B0      .LONG   3
                           0000  0807  0706  0804  0403  0302  011B4 P.ABV: .WORD   770, 1027, 2052, 1798, 2055, 0
                                                  00000006  011C0      .LONG   6
0601  0129  292A  2A22  2204  0408  0803  0307  0702  0206  011C4 P.ABW: .WORD   518, 1794, 775, 2051, 1032, 8708, 10786, -
                                                    0000  011D8                 10538, 297, 1537, 0
                                                        011DA          .BLKB   2
                                                  00000002  011DC      .LONG   2
                        00  08  00  00  00  05  08  08  011E0 P.ABX:   .BYTE   8, 8, 5, 0, 0, 0, 8, 0
                                                  00000002  011E8      .LONG   2
                        00  08  00  00  00  13  08  08  011EC P.ABY:   .BYTE   8, 8, 19, 0, 0, 0, 8, 0
                                                  00000002  011F4      .LONG   2
                        00  08  00  00  00  2F  08  08  011F8 P.ABZ:   .BYTE   8, 8, 47, 0, 0, 0, 8, 0
                                                  00000002  01200      .LONG   2
                        00  08  00  00  00  21  08  08  01204 P.ACA:   .BYTE   8, 8, 33, 0, 0, 0, 8, 0
                                                  00000002  0120C      .LONG   2
                        00  08  00  00  00  3B  08  08  01210 P.ACB:   .BYTE   8, 8, 59, 0, 0, 0, 8, 0
                                                  00000002  01218      .LONG   2
                        00  08  00  00  00  3D  08  08  0121C P.ACC:   .BYTE   8, 8, 61, 0, 0, 0, 8, 0
                                                  00000002  01224      .LONG   2
                        00  08  00  00  00  61  08  08  01228 P.ACD:   .BYTE   8, 8, 97, 0, 0, 0, 8, 0
                                                  00000002  01230      .LONG   2
                        00  08  00  00  00  6C  08  08  01234 P.ACE:   .BYTE   8, 8, 108, 0, 0, 0, 8, 0
                                                  00000002  0123C      .LONG   2
                        00  08  00  00  00  8F  08  08  01240 P.ACF:   .BYTE   8, 8, -113, 0, 0, 0, 8, 0
                                                  00000002  01248      .LONG   2
                        00  08  00  00  00  90  08  08  0124C P.ACG:   .BYTE   8, 8, -112, 0, 0, 0, 8, 0
                                                  00000002  01254      .LONG   2
                        00  08  00  00  00  7F  08  08  01258 P.ACH:   .BYTE   8, 8, 127, 0, 0, 0, 8, 0
                                                  00000002  01260      .LONG   2
                        00  08  00  00  00  80  08  08  01264 P.ACI:   .BYTE   8, 8, -128, 0, 0, 0, 8, 0
                                                  00000002  0126C      .LONG   2
                        00  08  00  00  00  87  08  08  01270 P.ACJ:   .BYTE   8, 8, -121, 0, 0, 0, 8, 0
                                                  00000002  01278      .LONG   2
                        00  08  00  00  00  88  08  08  0127C P.ACK:   .BYTE   8, 8, -120, 0, 0, 0, 8, 0
                                                  00000002  01284      .LONG   2
                        00  08  00  00  00  77  08  08  01288 P.ACL:   .BYTE   8, 8, 119, 0, 0, 0, 8, 0
                                                  00000002  01290      .LONG   2
                        00  08  00  00  00  78  08  08  01294 P.ACM:   .BYTE   8, 8, 120, 0, 0, 0, 8, 0
                                                  00000002  0129C      .LONG   2
```

```
                                00  08  00  00  00  97  08  08  012A0 P.ACN:   .BYTE   8, 8, -105, 0, 0, 0, 8, 0
                                                    00000002    012A8           .LONG   2
                                00  08  00  00  00  A0  08  08  012AC P.ACO:   .BYTE   8, 8, -96, 0, 0, 0, 8, 0
                                                    00000002    012B4           .LONG   2
                                00  08  00  00  00  A3  08  08  012B8 P.ACP:   .BYTE   8, 8, -93, 0, 0, 0, 8, 0
                                                    00000002    012C0           .LONG   2
                                00  08  00  00  00  9A  08  08  012C4 P.ACQ:   .BYTE   8, 8, -102, 0, 0, 0, 8, 0
                                                    00000014    012CC           .LONG   20
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06  012D0 P.ACR:   .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  00  C1  0B  0B  00  0A  00  00  00  C0  0A  0A  00  012DF                   0, 0, 7, 0, 10, 10, -64, 0, 0, 0, 10, 0, -
00  00  C3  1C  1C  00  1B  00  00  00  C2  1B  00  0B  012EE                   11, 11, -63, 0, 0, 0, 11, 0, 27, 27, -62, -
01  0A  09  09  00  15  00  00  00  F6  15  15  00  1C  00  012FD               0, 0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, -
BF  08  08  00  1A  00  00  01  0C  1A  1A  00  09  00  00  0130C               0, 21, 21, -10, 0, 0, 0, 21, 0, 9, 9, 10, -
                                    00  08  00  00  00  0131B                   1, 0, 0, 9, 0, 26, 26, 12, 1, 0, 0, 26, -
                                                                                0, 8, 8, -65, 0, 0, 0, 8, 0
                                                    00000014    01320           .LONG   20
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06  01324 P.ACS:   .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  00  B6  0B  0B  00  0A  00  00  00  B5  0A  0A  00  01333                   0, 0, 7, 0, 10, 10, -75, 0, 0, 0, 10, 0, -
00  00  B8  1C  1C  00  1B  00  00  00  B7  1B  1B  00  0B  01342                   11, 11, -74, 0, 0, 0, 11, 0, 27, 27, -73, -
01  0B  09  09  00  15  00  00  00  F7  15  15  00  1C  00  01351               0, 0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, -
B3  08  08  00  1A  00  00  01  0D  1A  1A  00  09  00  00  01360               0, 21, 21, -9, 0, 0, 0, 21, 0, 9, 9, 11, -
                                    00  08  00  00  00  0136F                   1, 0, 0, 9, 0, 26, 26, 13, 1, 0, 0, 26, -
                                                                                0, 8, 8, -77, 0, 0, 0, 8, 0
                                                    00000002    01374           .LONG   2
                                00  08  00  00  00  9D  08  08  01378 P.ACT:   .BYTE   8, 8, -99, 0, 0, 0, 8, 0
                                                    00000002    01380           .LONG   2
                                00  08  00  00  00  D1  08  08  01384 P.ACU:   .BYTE   8, 8, -47, 0, 0, 0, 8, 0
                                                    00000014    0138C           .LONG   20
08  00  00  00  CF  07  07  00  08  00  00  00  CF  06  06  01390 P.ACV:   .BYTE   6, 6, -49, 0, 0, 0, 8, 0, 7, 7, -49, 0, -
00  00  00  CF  03  03  00  08  00  00  00  CF  02  02  00  0139F                   0, 0, 8, 0, 2, 2, -49, 0, 0, 0, 8, 0, 3, -
00  00  CF  01  01  00  08  00  00  00  CF  04  04  00  08  013AE                   3, -49, 0, 0, 0, 8, 0, 4, 4, -49, 0, 0, -
00  CF  29  29  00  08  00  00  00  CF  22  22  00  08  00  013BD                   0, 8, 0, 1, 1, -49, 0, 0, 0, 8, 0, 34, -
CF  08  08  00  08  00  00  00  CF  2A  2A  00  08  00  00  013CC                   34, -49, 0, 0, 0, 8, 0, 41, 41, -49, 0, -
                                    00  08  00  00  00  013DB                   0, 0, 8, 0, 42, 42, -49, 0, 0, 0, 8, 0, -
                                                                                8, 8, -49, 0, 0, 0, 8, 0
                                                    00#     013E0 BLISS_OPINFO_TABLE:
                                                                    .BYTE   0[48]
                            00000000    000011B0    00001390    01410           .LONG   5008, 4532, 0
                                                    01      0141C           .BYTE   1
                                                    00#     0141D           .BYTE   0[3]
                            00000000    000011B4    00001200    01420           .LONG   4816, 4532, 0
                                                    00      0142C           .BYTE   0
                                                    00#     0142D           .BYTE   0[3]
                            00000000    000011B4    00001324    01430           .LONG   4900, 4532, 0
                                                    00      0143C           .BYTE   0
                                                    00#     0143D           .BYTE   0[3]
                            00000000    000011B4    000011E0    01440           .LONG   4576, 4532, 0
                                                    00      0144C           .BYTE   0
                                                    00#     0144D           .BYTE   0[3]
                            00000000    000011B4    000011EC    01450           .LONG   4588, 4532, 0
                                                    00      0145C           .BYTE   0
                                                    00#     0145D           .BYTE   0[3]
                            00000000    000011B4    000011F8    01460           .LONG   4600, 4532, 0
                                                    00      0146C           .BYTE   0
                                                    00#     0146D           .BYTE   0[3]
                            00000000    000011B4    00001204    01470           .LONG   4612, 4532, 0
                                                    00      0147C           .BYTE   0
```

```
                                00# 0147D          .BYTE   0[51]
00000000  000011B4  00001228  014B0          .LONG   4648, 4532, 0
                              00  014BC          .BYTE   0
                              00# 014BD          .BYTE   0[3]
00000000  000011B4  00001234  014C0          .LONG   4660, 4532, 0
                              00  014CC          .BYTE   0
                              00# 014CD          .BYTE   0[3]
00000000  000011B4  00001258  014D0          .LONG   4696, 4532, 0
                              00  014DC          .BYTE   0
                              00# 014DD          .BYTE   0[3]
00000000  000011B4  00001264  014E0          .LONG   4708, 4532, 0
                              00  014EC          .BYTE   0
                              00# 014ED          .BYTE   0[3]
00000000  000011B4  00001288  014F0          .LONG   4744, 4532, 0
                              00  014FC          .BYTE   0
                              00# 014FD          .BYTE   0[3]
00000000  000011B4  00001294  01500          .LONG   4756, 4532, 0
                              00  0150C          .BYTE   0
                              00# 0150D          .BYTE   0[3]
00000000  000011B4  00001240  01510          .LONG   4672, 4532, 0
                              00  0151C          .BYTE   0
                              00# 0151D          .BYTE   0[3]
00000000  000011B4  0000124C  01520          .LONG   4684, 4532, 0
                              00  0152C          .BYTE   0
                              00# 0152D          .BYTE   0[3]
00000000  000011B4  00001270  01530          .LONG   4720, 4532, 0
                              00  0153C          .BYTE   0
                              00# 0153D          .BYTE   0[3]
00000000  000011B4  0000127C  01540          .LONG   4732, 4532, 0
                              00  0154C          .BYTE   0
                              00# 0154D          .BYTE   0[115]
00000000  000011B4  00001378  015C0          .LONG   4984, 4532, 0
                              00  015CC          .BYTE   0
                              00# 015CD          .BYTE   0[3]
00000000  000011B4  000012A0  015D0          .LONG   4768, 4532, 0
                              00  015DC          .BYTE   0
                              00# 015DD          .BYTE   0[3]
00000000  000011B4  000012AC  015E0          .LONG   4780, 4532, 0
                              00  015EC          .BYTE   0
                              00# 015ED          .BYTE   0[3]
00000000  000011B4  000012B8  015F0          .LONG   4792, 4532, 0
                              00  015FC          .BYTE   0
                              00# 015FD          .BYTE   0[3]
00000000  000011B4  000012C4  01600          .LONG   4804, 4532, 0
                              00  0160C          .BYTE   0
                              00# 0160D          .BYTE   0[35]
00000000  000011B4  00001210  01630          .LONG   4624, 4532, 0
                              00  0163C          .BYTE   0
                              00# 0163D          .BYTE   0[3]
00000000  000011B4  0000121C  01640          .LONG   4636, 4532, 0
                              00  0164C          .BYTE   0
                              00# 0164D          .BYTE   0[163]
00000000  000011B4  00001384  016F0          .LONG   4996, 4532, 0
                              00  016FC          .BYTE   0
                              00# 016FD          .BYTE   0[3]
00000000  000011C4  00000000  01700          .LONG   0, 4548, 0
                              00  0170C          .BYTE   0
```

```
                                                        00# 0170D          .BYTE   0[3]
                                00000000  000011B4  00000000  01710          .LONG   0, 4532, 0
                                                        00  0171C          .BYTE   0
                                                        00# 0171D          .BYTE   0[67]
                                00000000  00000000  00000000  01760          .LONG   0, 0, 0
                                                        00  0176C          .BYTE   0
                                                            0176D          .BLKB   179
                                          00000000  00000000  01820 C_CVTINFO_TABLE:
                                                                           .LONG   0, 0
                                                        00  01828          .BYTE   0
                                                            01829          .BLKB   3
                                                  00000008  0182C          .LONG   8
0403  0302  0807  0706  082A  0829  0422  0401  082F  060E  01830 P.ACW:     .WORD   1550, 2095, 1025, 1058, 2089, 2090, 1798, -
                          0000  0B0A  0B04  0B08  0408  01844                          2055, 770, 1027, 1032, 2824, 2820, 2826, 0
                                                            0184E          .BLKB   2
                                                  00000008  01850          .LONG   8
312F  2F0B  0B0A  0A08  0804  0403  0307  0702  0206  060E  01854 P.ACX:     .WORD   1550, 518, 1794, 775, 1027, 2052, 2568, -
                          0000  0E2A  2A22  2229  2901  0131  01868                          2826, 12043, 12591, 305, 10497, 8745, -
                                                                                           10786, 3626, 0
                                                  0000000A  01874          .LONG   10
04  00  00  00  06  04  04  00  08  00  00  00  05  08  08  01878 P.ACY:     .BYTE   8, 8, 5, 0, 0, 0, 8, 0, 4, 4, 6, 0, 0, 0, -
00  00  00  DC  31  04  00  0B  00  00  00  08  0B  0B  00  01887                      4, 0, 11, 11, 8, 0, 0, 0, 11, 0, 4, 49, -
                    00  31  00  00  00  DC  04  31  00  31  01896                      -36, 0, 0, 0, 49, 0, 49, 4, -36, 0, 0, 0, -
                                                                                       49, 0
                                                  0000000A  018A0          .LONG   10
04  00  00  00  14  04  04  00  08  00  00  00  13  08  08  018A4 P.ACZ:     .BYTE   8, 8, 19, 0, 0, 0, 8, 0, 4, 4, 20, 0, 0, -
00  00  00  DD  31  04  00  0B  00  00  00  16  0B  0B  00  018B3                      0, 4, 0, 11, 11, 22, 0, 0, 0, 11, 0, 4, -
                    00  08  00  00  00  DE  31  31  00  31  018C2                      49, -35, 0, 0, 0, 49, 0, 49, 49, -34, 0, -
                                                                                       0, 0, 8, 0
                                                  00000006  018CC          .LONG   6
04  00  00  00  30  04  04  00  08  00  00  00  2F  08  08  018D0 P.ADA:     .BYTE   8, 8, 47, 0, 0, 0, 8, 0, 4, 4, 48, 0, 0, -
                          00  0B  00  00  00  32  0B  0B  00  018DF                      0, 4, 0, 11, 11, 50, 0, 0, 0, 11, 0
                                                  00000006  018E8          .LONG   6
04  00  00  00  22  04  04  00  08  00  00  00  21  08  08  018EC P.ADB:     .BYTE   8, 8, 33, 0, 0, 0, 8, 0, 4, 4, 34, 0, 0, -
                          00  0B  00  00  00  24  0B  0B  00  018FB                      0, 4, 0, 11, 11, 36, 0, 0, 0, 11, 0
                                                  00000016  01904          .LONG   22
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06  01908 P.ADC:     .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  00  B7  1B  1B  00  0A  00  00  00  B5  0A  0A  00  01917                      0, 0, 7, 0, 10, 10, -75, 0, 0, 0, 10, 0, -
00  00  F7  15  15  00  1C  00  00  00  B8  1C  1C  00  1B  01926                      27, 27, -73, 0, 0, 0, 27, 0, 28, 28, -72, -
01  0D  1A  1A  00  09  00  00  01  0B  09  09  00  15  00  01935                      0, 0, 0, 28, 0, 21, 21, -9, 0, 0, 0, 21, -
B4  04  04  00  08  00  00  00  00  B3  08  08  00  1A  00  00  01944                      0, 9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, -
            00  0B  00  00  00  B6  0B  0B  00  04  00  00  00  01953                      1, 0, 0, 26, 0, 8, 8, -77, 0, 0, 0, 8, 0, -
                                                                                       4, 4, -76, 0, 0, 0, 4, 0, 11, 11, -74, 0, -
                                                                                       0, 0, 11, 0
                                                  00000014  01960          .LONG   20
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06  01964 P.ADD:     .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  00  C0  0A  0A  00  08  00  00  00  BF  08  08  00  01973                      0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 10, -
00  00  C2  1B  1B  00  0B  00  00  00  C1  0B  0B  00  0A  01982                      10, -64, 0, 0, 0, 10, 0, 11, 11, -63, 0, -
00  F6  15  15  00  1C  00  00  00  C3  1C  1C  00  1B  00  01991                      0, 0, 11, 0, 27, 27, -62, 0, 0, 0, 27, 0, -
0C  1A  1A  00  09  00  00  01  0A  09  09  00  15  00  00  019A0                      28, 28, -61, 0, 0, 0, 28, 0, 21, 21, -10, -
                                            00  1A  00  00  01  019AF                      0, 0, 0, 21, 0, 9, 9, 10, 1, 0, 0, 9, 0, -
                                                                                       26, 26, 12, 1, 0, 0, 26, 0
                                                  00000004  019B4          .LONG   4
04  00  00  00  3C  04  04  00  08  00  00  00  3B  08  08  019B8 P.ADE:     .BYTE   8, 8, 59, 0, 0, 0, 8, 0, 4, 4, 60, 0, 0, -
                                                        00  019C7                      0, 4, 0
                                                  0000000C  019C8          .LONG   12
08  00  00  00  61  04  31  00  08  00  00  00  61  31  04  019CC P.ADF:     .BYTE   4, 49, 97, 0, 0, 0, 8, 0, 49, 4, 97, 0, -
```

```
00 00 00 61 08 08 00 08 00 00 00 61 31 31 00 019DB                      0, 0, 8, 0, 49, 49, 97, 0, 0, 0, 8, 0, 8, -
00 00 63 0B 0B 00 08 00 00 00 61 04 04 00 08 019EA                      8, 97, 0, 0, 0, 8, 0, 4, 4, 97, 0, 0, 0, -
                                   00 08 00 019F9                       8, 0, 11, 11, 99, 0, 0, 0, 8, 0
                                00000000C 019FC            .LONG        12
08 00 00 00 6C 04 31 00 08 00 00 00 6C 31 04 01A00 P.ADG:  .BYTE        4, 49, 108, 0, 0, 0, 8, 0, 49, 4, 108, 0, -
00 00 00 6C 08 08 00 08 00 00 00 6C 31 31 00 01A0F                      0, 0, 8, 0, 49, 49, 108, 0, 0, 0, 8, 0, -
00 00 6E 0B 0B 00 08 00 00 00 6C 04 04 00 08 01A1E                      8, 8, 108, 0, 0, 0, 8, 0, 4, 4, 108, 0, -
                                   00 08 00 01A2D                       0, 0, 8, 0, 11, 11, 110, 0, 0, 0, 8, 0
                                00000010 01A30            .LONG        16
08 00 00 00 90 31 04 00 08 00 00 00 8F 31 08 01A34 P.ADH:  .BYTE        8, 49, -113, 0, 0, 0, 8, 0, 4, 49, -112, -
00 00 00 90 04 31 00 08 00 00 00 8F 08 31 00 01A43                      0, 0, 0, 8, 0, 49, 8, -113, 0, 0, 0, 8, -
00 00 8F 08 08 00 08 00 00 00 8F 31 31 00 08 01A52                      0, 49, 4, -112, 0, 0, 0, 8, 0, 49, 49, -
00 92 0B 0B 00 08 00 00 00 90 04 04 00 08 00 01A61                      -113, 0, 0, 0, 8, 0, 8, 8, -113, 0, 0, 0, -
                                   00 08 00 00 01A70                    8, 0, 4, 4, -112, 0, 0, 0, 8, 0, 11, 11, -
                                                                        -110, 0, 0, 0, 8, 0
                                00000010 01A74            .LONG        16
08 00 00 00 80 31 04 00 08 00 00 00 7F 31 08 01A78 P.ADI:  .BYTE        8, 49, 127, 0, 0, 0, 8, 0, 4, 49, -128, -
00 00 00 80 04 31 00 08 00 00 00 7F 08 31 00 01A87                      0, 0, 0, 8, 0, 49, 8, 127, 0, 0, 0, 8, 0, -
00 00 7F 08 08 00 08 00 00 00 7F 31 31 00 08 01A96                      49, 4, -128, 0, 0, 0, 8, 0, 49, 49, 127, -
00 82 0B 0B 00 08 00 00 00 80 04 04 00 08 00 01AA5                      0, 0, 0, 8, 0, 8, 8, 127, 0, 0, 0, 8, 0, -
                                   00 08 00 00 01AB4                    4, 4, -128, 0, 0, 0, 8, 0, 11, 11, -126, -
                                                                        0, 0, 0, 8, 0
                                00000010 01AB8            .LONG        16
08 00 00 00 88 31 04 00 08 00 00 00 87 31 08 01ABC P.ADJ:  .BYTE        8, 49, -121, 0, 0, 0, 8, 0, 4, 49, -120, -
00 00 00 88 04 31 00 08 00 00 00 87 08 31 00 01ACB                      0, 0, 0, 8, 0, 49, 8, -121, 0, 0, 0, 8, -
00 00 87 08 08 00 08 00 00 00 87 31 31 00 08 01ADA                      0, 49, 4, -120, 0, 0, 0, 8, 0, 49, 49, -
00 8A 0B 0B 00 08 00 00 00 88 04 04 00 08 00 01AE9                      -121, 0, 0, 0, 8, 0, 8, 8, -121, 0, 0, 0, -
                                   00 08 00 00 01AF8                    8, 0, 4, 4, -120, 0, 0, 0, 8, 0, 11, 11, -
                                                                        -118, 0, 0, 0, 8, 0
                                00000010 01AFC            .LONG        16
08 00 00 00 78 31 04 00 08 00 00 00 77 31 08 01B00 P.ADK:  .BYTE        8, 49, 119, 0, 0, 0, 8, 0, 4, 49, 120, 0, -
00 00 00 78 04 31 00 08 00 00 00 77 08 31 00 01B0F                      0, 0, 8, 0, 49, 8, 119, 0, 0, 0, 8, 0, -
00 00 77 08 08 00 08 00 00 00 77 31 31 00 08 01B1E                      49, 4, 120, 0, 0, 0, 8, 0, 49, 49, 119, -
00 7A 0B 0B 00 08 00 00 00 78 04 04 00 08 00 01B2D                      0, 0, 0, 8, 0, 8, 8, 119, 0, 0, 0, 8, 0, -
                                   00 08 00 00 01B3C                    4, 4, 120, 0, 0, 0, 8, 0, 11, 11, 122, 0, -
                                                                        0, 0, 8, 0
                                00000004 01B40            .LONG        4
04 00 00 00 97 04 04 00 08 00 00 00 97 08 08 01B44 P.ADL:  .BYTE        8, 8, -105, 0, 0, 0, 8, 0, 4, 4, -105, 0, -
                                      00 01B53                          0, 0, 4, 0
                                00000004 01B54            .LONG        4
04 00 00 00 A0 04 04 00 08 00 00 00 A0 08 08 01B58 P.ADM:  .BYTE        8, 8, -96, 0, 0, 0, 8, 0, 4, 4, -96, 0, -
                                      00 01B67                          0, 0, 4, 0
                                00000004 01B68            .LONG        4
04 00 00 00 A3 04 04 00 08 00 00 00 A3 08 08 01B6C P.ADN:  .BYTE        8, 8, -93, 0, 0, 0, 8, 0, 4, 4, -93, 0, -
                                      00 01B7B                          0, 0, 4, 0
                                00000004 01B7C            .LONG        4
04 00 00 00 9D 04 04 00 08 00 00 00 9D 08 08 01B80 P.ADO:  .BYTE        8, 8, -99, 0, 0, 0, 8, 0, 4, 4, -99, 0, -
                                      00 01B8F                          C, 0, 4, 0
                                0000000C 01B90            .LONG        12
08 00 00 00 A6 31 04 00 08 00 00 00 A6 31 31 01B94 P.ADP:  .BYTE        49, 49, -90, 0, 0, 0, 8, 0, 4, 49, -90, -
00 00 00 A6 08 08 00 08 00 00 00 A6 04 31 00 01BA3                      0, 0, 0, 8, 0, 49, 4, -90, 0, 0, 0, 8, 0, -
00 00 A7 0B 0B 00 08 00 00 00 A6 04 04 00 08 01BB2                      8, 8, -90, 0, 0, 0, 8, 0, 4, 4, -90, 0, -
                                   00 08 00 01BC1                       0, 0, 8, 0, 11, 11, -89, 0, 0, 0, 8, 0
                                0000000C 01BC4            .LONG        12
08 00 00 00 AE 31 04 00 08 00 00 00 AE 31 31 01BC8 P.ADQ:  .BYTE        49, 49, -82, 0, 0, 0, 8, 0, 4, 49, -82, -
00 00 00 AE 08 08 00 08 00 00 00 AE 04 31 00 01BD7                      0, 0, 0, 8, 0, 49, 4, -82, 0, 0, 0, 8, 0, -
00 00 AF 0B 0B 00 08 00 00 00 AE 04 04 00 08 01BE6                      8, 8, -82, 0, 0, 0, 8, 0, 4, 4, -82, 0, -
```

```
                                                        00  08  00    01BF5                       0, 0, 8, 0, 11, 11, -81, 0, 0, 0, 8, 0
                                                          00000008    01BF8              .LONG    8
08  00  00  00  AA  08  08  00  08  00  00  00  AA  31  31  01BFC P.ADR:  .BYTE    49, 49, -86, 0, 0, 0, 8, 0, 8, 8, -86, 0, -
00  00  00  AB  0B  0B  00  08  00  00  00  AA  04  04  00  01C0B                       0, 0, 8, 0, 4, 4, -86, 0, 0, 0, 8, 0, 11, -
                                                        00  08  01C1A                       11, -85, 0, 0, 0, 8, 0
                                                          00000004    01C1C              .LONG    4
04  00  00  00  3D  04  04  00  08  00  00  00  3D  08  08  01C20 P.ADS:  .BYTE    8, 8, 61, 0, 0, 0, 8, 0, 4, 4, 61, 0, 0, -
                                                            00  01C2F                       0, 4, 0
                                                          00000004    01C30              .LONG    4
04  00  00  00  3F  04  04  00  08  00  00  00  3E  08  08  01C34 P.ADT:  .BYTE    8, 8, 62, 0, 0, 0, 8, 0, 4, 4, 63, 0, 0, -
                                                            00  01C43                       0, 4, 0
                                                          00000008    01C44              .LONG    8
04  00  00  00  E3  04  04  00  08  00  00  00  E2  08  08  01C48 P.ADU:  .BYTE    8, 8, -30, 0, 0, 0, 8, 0, 4, 4, -29, 0, -
00  00  00  E5  31  31  00  0B  00  00  00  E4  0B  0B  00  01C57                       0, 0, 4, 0, 11, 11, -28, 0, 0, 0, 11, 0, -
                                                        00  31  01C66                       49, 49, -27, 0, 0, 0, 49, 0
                                                          00000008    01C68              .LONG    8
04  00  00  00  E7  04  04  00  08  00  00  00  E6  08  08  01C6C P.ADV:  .BYTE    8, 8, -26, 0, 0, 0, 8, 0, 4, 4, -25, 0, -
00  00  00  E9  31  31  00  0B  00  00  00  E8  0B  0B  00  01C7B                       0, 0, 4, 0, 11, 11, -24, 0, 0, 0, 11, 0, -
                                                        00  31  01C8A                       49, 49, -23, 0, 0, 0, 49, 0
                                                          00000008    01C8C              .LONG    8
04  00  00  00  EB  04  04  00  08  00  00  00  EA  08  08  01C90 P.ADW:  .BYTE    8, 8, -22, 0, 0, 0, 8, 0, 4, 4, -21, 0, -
00  00  00  ED  31  31  00  0B  00  00  00  EC  0B  0B  00  01C9F                       0, 0, 4, 0, 11, 11, -20, 0, 0, 0, 11, 0, -
                                                        00  31  01CAE                       49, 49, -19, 0, 0, 0, 49, 0
                                                          00000008    01CB0              .LONG    8
04  00  00  00  EF  04  04  00  08  00  00  00  EE  08  08  01CB4 P.ADX:  .BYTE    8, 8, -18, 0, 0, 0, 8, 0, 4, 4, -17, 0, -
00  00  00  F1  31  31  00  0B  00  00  00  F0  0B  0B  00  01CC3                       0, 0, 4, 0, 11, 11, -16, 0, 0, 0, 11, 0, -
                                                        00  31  01CD2                       49, 49, -15, 0, 0, 0, 49, 0
                                                          00000002    01CD4              .LONG    2
                                00  31  00  00  00  DA  08  08  01CD8 P.ADY:  .BYTE    8, 8, -38, 0, 0, 0, 49, 0
                                                          00000002    01CE0              .LONG    2
                                00  08  00  00  00  DB  08  08  01CE4 P.ADZ:  .BYTE    8, 8, -37, 0, 0, 0, 8, 0
                                                          00000002    01CEC              .LONG    2
00  00  00  00  00  D0  31  31  01CF0 P.AEA:  .BYTE    49, 49, -48, 0, 0, 0, 0, 0
                                                            00# 01CF8 C_OPINFO_TABLE:
                                                                       .BYTE    0[48]
  00000000  00001830  00001CF0  01D28                      .LONG    7408, 6192, 0
                                  01  01D34                .BYTE    1
                                  00# 01D35                .BYTE    0[3]
  00000000  00001830  00001964  01D38                      .LONG    6500, 6192, 0
                                  01  01D44                .BYTE    1
                                  00# 01D45                .BYTE    0[3]
  00000000  00001830  00001908  01D48                      .LONG    6408, 6192, 0
                                  01  01D54                .BYTE    1
                                  00# 01D55                .BYTE    0[3]
  00000000  00001830  00001878  01D58                      .LONG    6264, 6192, 0
                                  01  01D64                .BYTE    1
                                  00# 01D65                .BYTE    0[3]
  00000000  00001830  000018A4  01D68                      .LONG    6308, 6192, 0
                                  01  01D74                .BYTE    1
                                  00# 01D75                .BYTE    0[3]
  00000000  00001830  000018D0  01D78                      .LONG    6352, 6192, 0
                                  01  01D84                .BYTE    1
                                  00# 01D85                .BYTE    0[3]
  00000000  00001830  000018EC  01D88                      .LONG    6380, 6192, 0
                                  01  01D94                .BYTE    1
                                  00# 01D95                .BYTE    0[51]
  00000000  00001830  000019CC  01DC8                      .LONG    6604, 6192, 0
```

```
                                 01  01DD4          .BYTE   1
                                 00# 01DD5          .BYTE   0[3]
00000000  00001830  00001A00       01DD8          .LONG   6656, 6192, 0
                                 01  01DE4          .BYTE   1
                                 00# 01DE5          .BYTE   0[3]
00000000  00001830  00001A78       01DE8          .LONG   6776, 6192, 0
                                 01  01DF4          .BYTE   1
                                 00# 01DF5          .BYTE   0[19]
00000000  00001830  00001B00       01E08          .LONG   6912, 6192, 0
                                 01  01E14          .BYTE   1
                                 00# 01E15          .BYTE   0[19]
00000000  00001830  00001A34       01E28          .LONG   6708, 6192, 0
                                 01  01E34          .BYTE   1
                                 00# 01E35          .BYTE   0[19]
00000000  00001830  00001ABC       01E48          .LONG   6844, 6192, 0
                                 01  01E54          .BYTE   1
                                 00# 01E55          .BYTE   0[19]
00000000  00001830  00001BFC       01E68          .LONG   7164, 6192, 0
                                 01  01E74          .BYTE   1
                                 00# 01E75          .BYTE   0[67]
00000000  00001830  00001B94       01EB8          .LONG   7060, 6192, 0
                                 01  01EC4          .BYTE   1
                                 00# 01EC5          .BYTE   0[3]
00000000  00001830  00001BC8       01EC8          .LONG   7112, 6192, 0
                                 01  01ED4          .BYTE   1
                                 00# 01ED5          .BYTE   0[3]
00000000  00001830  00001B80       01ED8          .LONG   7040, 6192, 0
                                 01  01EE4          .BYTE   1
                                 00# 01EE5          .BYTE   0[3]
00000000  00001830  00001B44       01EE8          .LONG   6980, 6192, 0
                                 01  01EF4          .BYTE   1
                                 00# 01EF5          .BYTE   0[3]
00000000  00001830  00001B58       01EF8          .LONG   7000, 6192, 0
                                 01  01F04          .BYTE   1
                                 00# 01F05          .BYTE   0[3]
00000000  00001830  00001B6C       01F08          .LONG   7020, 6192, 0
                                 01  01F14          .BYTE   1
                                 00# 01F15          .BYTE   0[51]
00000000  00001830  00001988       01F48          .LONG   6584, 6192, 0
                                 01  01F54          .BYTE   1
                                 00# 01F55          .BYTE   0[3]
00000000  00001830  00001C20       01F58          .LONG   7200, 6192, 0
                                 01  01F64          .BYTE   1
                                 00# 01F65          .BYTE   0[3]
00000000  00001830  00001C34       01F68          .LONG   7220, 6192, 0
                                 01  01F74          .BYTE   1
                                 00# 01F75          .BYTE   0[3]
00000000  00001830  00001C08       01F78          .LONG   7384, 6192, 0
                                 00  01F84          .BYTE   0
                                 00# 01F85          .BYTE   0[3]
00000000  00001830  00001CE4       01F88          .LONG   7396, 6192, 0
                                 00  01F94          .BYTE   0
                                 00# 01F95          .BYTE   0[131]
00000000  00001854  00000000       02018          .LONG   0, 6228, 0
                                 01  02024          .BYTE   1
                                 00# 02025          .BYTE   0[3]
00000000  00001830  00000000       02028          .LONG   0, 6192, 0
```

```
                                                    01  02034          .BYTE   1
                                                    00# 02035          .BYTE   0[67]
                                00000000  00000000  00000000  02078          .LONG   0, 0, 0
                                                    01  02084          .BYTE   1
                                                    00# 02085          .BYTE   0[35]
                                00000000  00001830  00001C48  020A8          .LONG   7240, 6192, 0
                                                    01  020B4          .BYTE   1
                                                    00# 020B5          .BYTE   0[3]
                                00000000  00001830  00001C6C  020B8          .LONG   7276, 6192, 0
                                                    01  020C4          .BYTE   1
                                                    00# 020C5          .BYTE   0[3]
                                00000000  0000183C  00001C90  020C8          .LONG   7312, 6192, 0
                                                    01  020D4          .BYTE   1
                                                    00# 020D5          .BYTE   0[3]
                                00000000  00001830  00001CB4  020D8          .LONG   7348, 6192, 0
                                                    01  020E4          .BYTE   1
                                                        020E5          .BLKB   83
                                                  00000002  02138          .LONG   2
                                00000000  0002FF30  0213C  P.AEB:   .LONG   196400, 0
                                0000213C  00000000  02144  COBOL_CVTINFO_TABLE:
                                                                     .LONG   0, 8508
                                                    00  0214C          .BYTE   0
                                                        0214D          .BLKB   3
                                                  00000007  02150          .LONG   7
1512  1511  1510  150F  1509  1505  1508  1504  1507  1503  02154  P.AEC:   .WORD   5379, 5383, 5380, 5384, 5381, 5385, 5391, -
                          0000  150B  0B0A  1513  02168                           5392, 5393, 5394, 5395, 2826, 5387, 0
                                                  00000008  02170          .LONG   8
1211  1110  100F  0F09  0905  0508  0804  0407  0703  0330  02174  P.AED:   .WORD   816, 1795, 1031, 2052, 1288, 2309, 3849, -
                          0000  300B  0B0A  0A15  1513  1312  02188                           4111, 4368, 4625, 4882, 5395, 2581, 2826, -
                                                                                            12299, 0
                                                  00000006  02194          .LONG   6
0A  00  00  00  07  0A  0A  00  15  00  00  00  F2  15  15  02198  P.AEE:   .BYTE   21, 21, -14, 0, 0, 0, 21, 0, 10, 10, 7, -
                          00  0B  00  00  00  08  0B  0B  00  021A7                   0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 0, 11, 0
                                                  00000006  021B0          .LONG   6
0A  00  00  00  15  0A  0A  00  15  00  00  00  F3  15  15  021B4  P.AEF:   .BYTE   21, 21, -13, 0, 0, 0, 21, 0, 10, 10, 21, -
                          00  0B  00  00  00  16  0B  0B  00  021C3                   0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 0, 11, 0
                                                  00000006  021CC          .LONG   6
0A  00  00  00  31  0A  0A  00  15  00  00  00  F4  15  15  021D0  P.AEG:   .BYTE   21, 21, -12, 0, 0, 0, 21, 0, 10, 10, 49, -
                          00  0B  00  00  00  32  0B  0B  00  021DF                   0, 0, 0, 10, 0, 11, 11, 50, 0, 0, 0, 11, 0
                                                  00000006  021E8          .LONG   6
0A  00  00  00  23  0A  0A  00  15  00  00  00  F5  15  15  021EC  P.AEH:   .BYTE   21, 21, -11, 0, 0, 0, 21, 0, 10, 10, 35, -
                          00  0B  00  00  00  24  0B  0B  00  021FB                   0, 0, 0, 10, 0, 11, 11, 36, 0, 0, 0, 11, 0
                                                  00000014  02204          .LONG   20
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06  02208  P.AEI:   .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  00  C2  1B  1B  00  08  00  00  00  BF  08  08  00  02217                   0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 27, -
00  01  0A  09  09  00  1C  00  00  00  C3  1C  1C  00  1B  02226                   27, -62, 0, 0, 0, 27, 0, 28, 28, -61, 0, -
00  F6  15  15  00  1A  00  00  01  0C  1A  1A  00  09  00  02235                   0, 0, 28, 0, 9, 9, 10, 1, 0, 0, 9, 0, 26, -
C1  0B  0B  00  0A  00  00  00  C0  0A  0A  00  15  00  00  02244                   26, 12, 1, 0, 0, 26, 0, 21, 21, -10, 0, -
                                    00  0B  00  00  00  02253                       0, 0, 21, 0, 10, 10, -64, 0, 0, 0, 10, 0, -
                                                                                    11, 11, -63, 0, 0, 0, 11, 0
                                                  00000014  02258          .LONG   20
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06  0225C  P.AEJ:   .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  00  B7  1B  1B  00  08  00  00  00  B3  08  08  00  0226B                   0, 0, 7, 0, 8, 8, -77, 0, 0, 0, 8, 0, 27, -
00  01  0B  09  09  00  1C  00  00  00  B8  1C  1C  00  1B  0227A                   27, -73, 0, 0, 0, 27, 0, 28, 28, -72, 0, -
00  F7  15  15  00  1A  00  00  01  0D  1A  1A  00  09  00  02289                   0, 0, 28, 0, 9, 9, 11, 1, 0, 0, 9, 0, 26, -
B6  0B  0B  00  0A  00  00  00  B5  0A  0A  00  15  00  00  02298                   26, 13, 1, 0, 0, 26, 0, 21, 21, -9, 0, 0, -
                                    00  0B  00  00  00  022A7                       0, 21, 0, 10, 10, -75, 0, 0, 0, 10, 0, -
```

```
                                                                              11, 11, -74, 0, 0, 0, 11, 0
                                               00000008  022AC          .LONG  8
28 00 00 00 F8 15 15 00 28 00 00 00 54  0E  0E  022B0 P.AEK:   .BYTE  14, 14, 84, 0, 0, 0, 40, 0, 21, 21, -8, -
00 00 00 63 0B 0B 00 28 00 00 00 62 0A  0A  00  022BF                 0, 0, 0, 40, 0, 10, 10, 98, 0, 0, 0, 40, -
                                        00  28  022CE                 0, 11, 11, 99, 0, 0, 0, 40, 0
                                               00000008  022D0          .LONG  8
28 00 00 00 F9 15 15 00 28 00 00 00 5E  0E  0E  022D4 P.AEL:   .BYTE  14, 14, 94, 0, 0, 0, 40, 0, 21, 21, -7, -
00 00 00 6E 0B 0B 00 28 00 00 00 6D 0A  0A  00  022E3                 0, 0, 0, 40, 0, 10, 10, 109, 0, 0, 0, 40, -
                                        00  28  022F2                 0, 11, 11, 110, 0, 0, 0, 40, 0
                                               00000008  022F4          .LONG  8
28 00 00 00 FA 15 15 00 28 00 00 00 58  0E  0E  022F8 P.AEM:   .BYTE  14, 14, 88, 0, 0, 0, 40, 0, 21, 21, -6, -
00 00 00 82 0B 0B 00 28 00 00 00 81 0A  0A  00  02307                 0, 0, 0, 40, 0, 10, 10, -127, 0, 0, 0, -
                                        00  28  02316                 40, 0, 11, 11, -126, 0, 0, 0, 40, 0
                                               00000008  02318          .LONG  8
28 00 00 00 FB 15 15 00 28 00 00 00 56  0E  0E  0231C P.AEN:   .BYTE  14, 14, 86, 0, 0, 0, 40, 0, 21, 21, -5, -
00 00 00 7A 0B 0B 00 28 00 00 00 79 0A  0A  00  0232B                 0, 0, 0, 40, 0, 10, 10, 121, 0, 0, 0, 40, -
                                        00  28  0233A                 0, 11, 11, 122, 0, 0, 0, 40, 0
                                               00000008  0233C          .LONG  8
28 00 00 00 FC 15 15 00 28 00 00 00 5C  0E  0E  02340 P.AEO:   .BYTE  14, 14, 92, 0, 0, 0, 40, 0, 21, 21, -4, -
00 00 00 92 0B 0B 00 28 00 00 00 91 0A  0A  00  0234F                 0, 0, 0, 40, 0, 10, 10, -111, 0, 0, 0, -
                                        00  28  0235E                 40, 0, 11, 11, -110, 0, 0, 0, 40, 0
                                               00000008  02360          .LONG  8
28 00 00 00 FD 15 15 00 28 00 00 00 5A  0E  0E  02364 P.AEP:   .BYTE  14, 14, 90, 0, 0, 0, 40, 0, 21, 21, -3, -
00 00 00 8A 0B 0B 00 28 00 00 00 89 0A  0A  00  02373                 0, 0, 0, 40, 0, 10, 10, -119, 0, 0, 0, -
                                        00  28  02382                 40, 0, 11, 11, -118, 0, 0, 0, 40, 0
                                               00000002  02384          .LONG  2
                        00 28 00 00 00 AA  28  28  02388 P.AEQ:   .BYTE  40, 40, -86, 0, 0, 0, 40, 0
                                               00000002  02390          .LONG  2
                        00 28 00 00 00 A6  28  28  02394 P.AER:   .BYTE  40, 40, -90, 0, 0, 0, 40, 0
                                               00000002  0239C          .LONG  2
                        00 28 00 00 00 AE  28  28  023A0 P.AES:   .BYTE  40, 40, -82, 0, 0, 0, 40, 0
                                                   00#  023A8 COBOL_OPINFO_TABLE:
                                                                .BYTE  0[64]
                     00000000  00002154  00002208  023E8        .LONG  8712, 8532, 0
                                               01  023F4        .BYTE  1
                                              00#  023F5        .BYTE  0[3]
                     00000000  00002154  0000225C  023F8        .LONG  8796, 8532, 0
                                               01  02404        .BYTE  1
                                              00#  02405        .BYTE  0[3]
                     00000000  00002154  00002198  02408        .LONG  8600, 8532, 0
                                               01  02414        .BYTE  1
                                              00#  02415        .BYTE  0[3]
                     00000000  00002154  000021B4  02418        .LONG  8628, 8532, 0
                                               01  02424        .BYTE  1
                                              00#  02425        .BYTE  0[3]
                     00000000  00002154  000021D0  02428        .LONG  8656, 8532, 0
                                               01  02434        .BYTE  1
                                              00#  02435        .BYTE  0[3]
                     00000000  00002154  000021EC  02438        .LONG  8684, 8532, 0
                                               01  02444        .BYTE  1
                                              00#  02445        .BYTE  0[51]
                     00000000  00002154  000022B0  02478        .LONG  8880, 8532, 0
                                               01  02484        .BYTE  1
                                              00#  02485        .BYTE  0[3]
                     00000000  00002154  000022D4  02488        .LONG  8916, 8532, 0
                                               01  02494        .BYTE  1
                                              00#  02495        .BYTE  0[3]
```

```
00000000  00002154  000022F8  02498          .LONG   8952, 8532, 0                                                  ;
                        01  024A4          .BYTE   1                                                              ;
                        00# 024A5          .BYTE   0[19]                                                          ;
00000000  00002154  0000231C  024B8          .LONG   8988, 8532, 0                                                  ;
                        01  024C4          .BYTE   1                                                              ;
                        00# 024C5          .BYTE   0[19]                                                          ;
00000000  00002154  00002340  024D8          .LONG   9024, 8532, 0                                                  ;
                        01  024E4          .BYTE   1                                                              ;
                        00# 024E5          .BYTE   0[19]                                                          ;
00000000  00002154  00002364  024F8          .LONG   9060, 8532, 0                                                  ;
                        01  02504          .BYTE   1                                                              ;
                        00# 02505          .BYTE   0[19]                                                          ;
00000000  00002154  00002388  02518          .LONG   9096, 8532, 0                                                  ;
                        01  02524          .BYTE   1                                                              ;
                        00# 02525          .BYTE   0[3]                                                           ;
00000000  00002154  00002394  02528          .LONG   9108, 8532, 0                                                  ;
                        01  02534          .BYTE   1                                                              ;
                        00# 02535          .BYTE   0[3]                                                           ;
00000000  00002154  000023A0  02538          .LONG   9120, 8532, 0                                                  ;
                        01  02544          .BYTE   1                                                              ;
                        00# 02545          .BYTE   0[387]                                                         ;
00000000  00002174  00000000  026C8          .LONG   0, 8564, 0                                                     ;
                        01  026D4          .BYTE   1                                                              ;
                        00# 026D5          .BYTE   0[3]                                                           ;
00000000  00002174  00000000  026D8          .LONG   0, 8564, 0                                                     ;
                        01  026E4          .BYTE   1                                                              ;
                        00# 026E5          .BYTE   0[67]                                                          ;
00000000  00000000  00000000  02728          .LONG   0, 0, 0                                                        ;
                        01  02734          .BYTE   1                                                              ;
                            02735          .BLKB   179                                                            ;
                  00000002  027E8          .LONG   2                                                              ;
                0000  0804  0703  0602  027EC P.AET:   .WORD   1538, 1795, 2052, 0                                   ;
                00000000  000027EC  027F4 FORTRAN_CVTINFO_TABLE:                                                    ;
                                       .LONG   10220, 0                                                         ;
                                    00 027FC          .BYTE   0                                                    ;
                                       027FD          .BLKB   3                                                    ;
                                  00000008 02800          .LONG   8                                              ;
1D1B  1C1B  0D0B  1C0B  0C0A  1B0A  0B0A  0A08  0807  0706  02804 P.AEU:   .WORD   1798, 2055, 2568, 2826, 6922, 3082, 7179, -  ;
                      0000  1D0C  0D0C  1D1C  0D1C  02818                3339, 7195, 7451, 3356, 7452, 3340, 7436, -  ;
                                                                        0                                           ;
                                       02822          .BLKB   2                                                    ;
                                  00000004 02824          .LONG   4                                              ;
            0000  1C1B  1C0B  1B0A  0B0A  0A08  0807  0706  02828 P.AEV:   .WORD   1798, 2055, 2568, 2826, 6922, 7179, 7195, -  ;
                                                                        0                                           ;
                                  00000002 02838          .LONG   2                                              ;
                            0000  0807  0706  0283C P.AEW:   .WORD   1798, 2055, 0                                 ;
                                       02842          .BLKB   2                                                    ;
                                  00000006 02844          .LONG   6                                              ;
061D  1D0D  0D0C  0C1C  1C1B  1B0B  0B0A  0A08  0807  0706  02848 P.AEX:   .WORD   1798, 2055, 2568, 2826, 6923, 7195, 3100, -  ;
                                  0000  C285C                       3340, 7437, 1565, 0                           ;
                                       0285E          .BLKB   2                                                    ;
                                  00000003 02860          .LONG   3                                              ;
                      0000  1D0D  0D1B  1D0B  1B0B  02864 P.AEY:   .WORD   6923, 7435, 3355, 7437, 0              ;
                                       0286E          .BLKB   2                                                    ;
                                  00000014 02870          .LONG   20                                             ;
07  00  00  00  03  07  07  00  06  00  00  00  01  06  06  02874 P.AEZ:   .BYTE   6, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, -  ;
00  00  00  07  0A  0A  00  08  00  00  00  05  08  08  00  02883                7, 0, 8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, -  ;
```

```
00  00  09  1B  1B  00  0B  00  00  00  08  0B  0B  00  0A  02892                    0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 0, 11, -
00  0B  0C  0C  00  1C  00  00  00  0A  1C  1C  00  1B  00  028A1                    0, 27, 27, 9, 0, 0, 0, 27, 0, 28, 28, 10, -
0D  1D  1D  00  0D  00  00  00  0C  0D  0D  00  0C  00  00  028B0                    0, 0, 0, 28, 0, 12, 12, 11, 0, 0, 0, 12, -
                                        00  1D  00  00  00  028BF                    0, 13, 13, 12, 0, 0, 0, 13, 0, 29, 29, -
                                                                                     13, 0, 0, 0, 29, 0
                                                00000014  028C4          .LONG      20
07  00  00  00  11  07  07  00  06  00  00  00  0F  06  06  028C8  P.AFA:  .BYTE     6, 6, 15, 0, 0, 0, 6, 0, 7, 7, 17, 0, 0, -
00  00  00  15  0A  0A  00  08  00  00  00  13  08  08  00  028D7                    0, 7, 0, 8, 8, 19, 0, 0, 0, 8, 0, 10, 10, -
00  00  17  1B  1B  00  0B  00  00  00  16  0B  0B  00  0A  028E6                    21, 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 0, -
00  19  0C  0C  00  1C  00  00  00  18  1C  1C  00  1B  00  028F5                    11, 0, 27, 27, 23, 0, 0, 0, 27, 0, 28, -
1B  1D  1D  00  0D  00  00  00  1A  0D  0D  00  0C  00  00  02904                    28, 24, 0, 0, 0, 28, 0, 12, 12, 25, 0, 0, -
                                        00  1D  00  00  00  02913                    0, 12, 0, 13, 13, 26, 0, 0, 0, 13, 0, 29, -
                                                                                     29, 27, 0, 0, 0, 29, 0
                                                00000014  02918          .LONG      20
07  00  00  00  2D  07  07  00  06  00  00  00  2B  06  06  0291C  P.AFB:  .BYTE     6, 6, 43, 0, 0, 0, 6, 0, 7, 7, 45, 0, 0, -
00  00  00  31  0A  0A  00  08  00  00  00  2F  08  08  00  0292B                    0, 7, 0, 8, 8, 47, 0, 0, 0, 8, 0, 10, 10, -
00  00  33  1B  1B  00  0B  00  00  00  32  0B  0B  00  0A  0293A                    49, 0, 0, 0, 10, 0, 11, 11, 50, 0, 0, 0, -
00  35  0C  0C  00  1C  00  00  00  34  1C  1C  00  1B  00  02949                    11, 0, 27, 27, 51, 0, 0, 0, 27, 0, 28, -
37  1D  1D  00  0D  00  00  00  36  0D  0D  00  0C  00  00  02958                    28, 52, 0, 0, 0, 28, 0, 12, 12, 53, 0, 0, -
                                        00  1D  00  00  00  02967                    0, 12, 0, 13, 13, 54, 0, 0, 0, 13, 0, 29, -
                                                                                     29, 55, 0, 0, 0, 29, 0
                                                00000014  0296C          .LONG      20
07  00  00  00  1F  07  07  00  06  00  00  00  1D  06  06  02970  P.AFC:  .BYTE     6, 6, 29, 0, 0, 0, 6, 0, 7, 7, 31, 0, 0, -
00  00  00  23  0A  0A  00  08  00  00  00  21  08  08  00  0297F                    0, 7, 0, 8, 8, 33, 0, 0, 0, 8, 0, 10, 10, -
00  00  25  1B  1B  00  0B  00  00  00  24  0B  0B  00  0A  0298E                    35, 0, 0, 0, 10, 0, 11, 11, 36, 0, 0, 0, -
00  27  0C  0C  00  1C  00  00  00  26  1C  1C  00  1B  00  0299D                    11, 0, 27, 27, 37, 0, 0, 0, 27, 0, 28, -
29  1D  1D  00  0D  00  00  00  28  0D  0D  00  0C  00  00  029AC                    28, 38, 0, 0, 0, 28, 0, 12, 12, 39, 0, 0, -
                                        00  1D  00  00  00  029BB                    0, 12, 0, 13, 13, 40, 0, 0, 0, 13, 0, 29, -
                                                                                     29, 41, 0, 0, 0, 29, 0
                                                0000001A  029C0          .LONG      26
09  00  00  01  0A  09  09  00  15  00  00  00  F6  15  15  029C4  P.AFD:  .BYTE     21, 21, -10, 0, 0, 0, 21, 0, 9, 9, 10, 1, -
00  00  00  BD  06  06  00  1A  00  00  01  0C  1A  1A  00  029D3                    0, 0, 9, 0, 26, 26, 12, 1, 0, 0, 26, 0, -
00  00  BF  08  08  00  07  00  00  00  BE  07  07  00  06  029E2                    6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  C1  0B  0B  00  0A  00  00  00  C0  0A  0A  00  08  00  029F1                    0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 10, -
C3  1C  1C  00  1B  00  00  00  C2  1B  1B  00  0B  00  00  02A00                    10, -64, 0, 0, 0, 10, 0, 11, 11, -63, 0, -
0D  0D  00  0C  00  00  00  C4  0C  0C  00  1C  00  00  00  02A0F                    0, 0, 11, 0, 27, 27, -62, 0, 0, 0, 27, 0, -
    00  1D  00  00  00  C6  1D  1D  00  0D  00  00  00  C5  02A1E                    28, 28, -61, 0, 0, 0, 28, 0, 12, 12, -60, -
                                                                                     0, 0, 0, 12, 0, 13, 13, -59, 0, 0, 0, 13, -
                                                                                     0, 29, 29, -58, 0, 0, 0, 29, 0
                                                0000001A  02A2C          .LONG      26
09  00  00  01  0B  09  09  00  15  00  00  00  F7  15  15  02A30  P.AFE:  .BYTE     21, 21, -9, 0, 0, 0, 21, 0, 9, 9, 11, 1, -
00  00  00  B1  06  06  00  1A  00  00  01  0D  1A  1A  00  02A3F                    0, 0, 9, 0, 26, 26, 13, 1, 0, 0, 26, 0, -
00  00  B3  08  08  00  07  00  00  00  B2  07  07  00  06  02A4E                    6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  B6  0B  0B  00  0A  00  00  00  B5  0A  0A  00  08  00  02A5D                    0, 0, 7, 0, 8, 8, -77, 0, 0, 0, 8, 0, 10, -
B8  1C  1C  00  1B  00  00  00  B7  1B  1B  00  0B  00  00  02A6C                    10, -75, 0, 0, 0, 10, 0, 11, 11, -74, 0, -
0D  0D  00  0C  00  00  00  B9  0C  0C  00  1C  00  00  00  02A7B                    0, 0, 11, 0, 27, 27, -73, 0, 0, 0, 27, 0, -
    00  1D  00  00  00  BB  1D  1D  00  0D  00  00  00  BA  02A8A                    28, 28, -72, 0, 0, 0, 28, 0, 12, 12, -71, -
                                                                                     0, 0, 0, 12, 0, 13, 13, -70, 0, 0, 0, 13, -
                                                                                     0, 29, 29, -69, 0, 0, 0, 29, 0
                                                00000024  02A98          .LONG      36
08  00  00  00  41  08  08  00  07  00  00  00  40  07  07  02A9C  P.AFF:  .BYTE     7, 7, 64, 0, 0, 0, 7, 0, 8, 8, 65, 0, 0, -
00  00  00  43  0B  08  00  0A  00  00  00  42  0A  08  00  02AAB                    0, 8, 0, 8, 10, 66, 0, 0, 0, 10, 0, 8, -
00  00  45  1C  08  00  1B  00  00  00  44  1B  08  00  0B  02ABA                    11, 67, 0, 0, 0, 11, 0, 8, 27, 68, 0, 0, -
00  47  0D  08  00  0C  00  00  00  46  0C  08  00  1C  00  02AC9                    0, 27, 0, 8, 28, 69, 0, 0, 0, 28, 0, 8, -
49  0A  0A  00  1D  00  00  00  48  1D  08  00  0D  00  00  02AD8                    12, 70, 0, 0, 0, 12, 0, 8, 13, 71, 0, 0, -
0A  0B  00  0B  00  00  00  4A  0B  0A  00  0A  00  00  00  02AE7                    0, 13, 0, 8, 29, 72, 0, 0, 0, 29, 0, 10, -
1B  00  0B  00  00  00  4C  0B  0B  00  0B  00  00  00  4B  02AF6                    10, 73, 0, 0, 0, 10, 0, 10, 11, 74, 0, 0, -
```

```
00  1C  00  00  00  4E  1C  1C  00  1B  00  00  00  4D  1B  02B05                                0, 11, 0, 11, 10, 75, 0, 0, 0, 11, 0, 11, -
0D  00  00  00  50  0D  0D  00  0C  00  00  00  4F  0C  0C  02B14                                11, 76, 0, 0, 0, 11, 0, 27, 27, 77, 0, 0, -
                        00  1D  00  00  00  51  1D  1D  00  02B23                                0, 27, 0, 28, 28, 78, 0, 0, 0, 28, 0, 12, -
                                                                                                12, 79, 0, 0, 0, 12, 0, 13, 13, 80, 0, 0, -
                                                                                                0, 13, 0, 29, 29, 81, 0, 0, 0, 29, 0
                                                        00000002  02B2C            .LONG      2
                            00  0E  00  00  00  52  0E  0E  02B30  P.AFG:  .BYTE    14, 14, 82, 0, 0, 0, 14, 0
                                                        00000016  02B38            .LONG      22
08  00  00  00  5F  06  06  00  08  00  00  00  54  0E  0E  02B3C  P.AFH:  .BYTE    14, 14, 84, 0, 0, 0, 8, 0, 6, 6, 95, 0, -
00  00  00  61  08  08  00  08  00  00  00  60  07  07  00  02B4B                                0, 0, 8, 0, 7, 7, 96, 0, 0, 0, 8, 0, 8, -
00  00  63  0B  0B  00  08  00  00  00  62  0A  0A  00  08  02B5A                                8, 97, 0, 0, 0, 8, 0, 10, 10, 98, 0, 0, -
00  65  1C  1C  00  08  00  00  00  64  1B  1B  00  08  00  02B69                                0, 8, 0, 11, 11, 99, 0, 0, 0, 8, 0, 27, -
67  0D  0D  00  08  00  00  00  66  0C  0C  00  08  00  00  02B78                                27, 100, 0, 0, 0, 8, 0, 28, 28, 101, 0, -
        00  08  00  00  00  68  1D  1D  00  08  00  00  00  02B87                                0, 0, 8, 0, 12, 12, 102, 0, 0, 0, 8, 0, -
                                                                                                13, 13, 103, 0, 0, 0, 8, 0, 29, 29, 104, -
                                                                                                0, 0, 0, 8, 0
                                                        00000016  02B94            .LONG      22
08  00  00  00  6A  06  06  00  08  00  00  00  5E  0E  0E  02B98  P.AFI:  .BYTE    14, 14, 94, 0, 0, 0, 8, 0, 6, 6, 106, 0, -
00  00  00  6C  08  08  00  08  00  00  00  6B  07  07  00  02BA7                                0, 0, 8, 0, 7, 7, 107, 0, 0, 0, 8, 0, 8, -
00  00  6E  0B  0B  00  08  00  00  00  6D  0A  0A  00  08  02BB6                                8, 108, 0, 0, 0, 8, 0, 10, 10, 109, 0, 0, -
00  70  1C  1C  00  08  00  00  00  6F  1B  1B  00  08  00  02BC5                                0, 8, 0, 11, 11, 110, 0, 0, 0, 8, 0, 27, -
72  0D  0D  00  08  00  00  00  71  0C  0C  00  08  00  00  02BD4                                27, 111, 0, 0, 0, 8, 0, 28, 28, 112, 0, -
        00  08  00  00  00  73  1D  1D  00  08  00  00  00  02BE3                                0, 0, 8, 0, 12, 12, 113, 0, 0, 0, 8, 0, -
                                                                                                13, 13, 114, 0, 0, 0, 8, 0, 29, 29, 115, -
                                                                                                0, 0, 0, 8, 0
                                                        00000010  02BF0            .LONG      16
08  00  00  00  7D  06  06  00  08  00  00  00  58  0E  0E  02BF4  P.AFJ:  .BYTE    14, 14, 88, 0, 0, 0, 8, 0, 6, 6, 125, 0, -
00  00  00  7F  08  08  00  08  00  00  00  7E  07  07  00  02C03                                0, 0, 8, 0, 7, 7, 126, 0, 0, 0, 8, 0, 8, -
00  00  82  0B  0B  00  08  00  00  00  81  0A  0A  00  08  02C12                                8, 127, 0, 0, 0, 8, 0, 10, 10, -127, 0, -
00  84  1C  1C  00  08  00  00  00  83  1B  1B  00  08  00  02C21                                0, 0, 8, 0, 11, 11, -126, 0, 0, 0, 8, 0, -
                                            00  08  00  00  02C30                                27, 27, -125, 0, 0, 0, 8, 0, 28, 28, -
                                                                                                -124, 0, 0, 0, 8, 0
                                                        00000010  02C34            .LONG      16
08  00  00  00  75  06  06  00  08  00  00  00  56  0E  0E  02C38  P.AFK:  .BYTE    14, 14, 86, 0, 0, 0, 8, 0, 6, 6, 117, 0, -
00  00  00  77  08  08  00  08  00  00  00  76  07  07  00  02C47                                0, 0, 8, 0, 7, 7, 118, 0, 0, 0, 8, 0, 8, -
00  00  7A  0B  0B  00  08  00  00  00  79  0A  0A  00  08  02C56                                8, 119, 0, 0, 0, 8, 0, 10, 10, 121, 0, 0, -
00  7C  1C  1C  00  08  00  00  00  7B  1B  1B  00  08  00  02C65                                0, 8, 0, 11, 11, 122, 0, 0, 0, 8, 0, 27, -
                                            00  08  00  00  02C74                                27, 123, 0, 0, 0, 8, 0, 28, 28, 124, 0, -
                                                                                                0, 0, 8, 0
                                                        00000010  02C78            .LONG      16
08  00  00  00  8D  06  06  00  08  00  00  00  5C  0E  0E  02C7C  P.AFL:  .BYTE    14, 14, 92, 0, 0, 0, 8, 0, 6, 6, -115, 0, -
00  00  00  8F  08  08  00  08  00  00  00  8E  07  07  00  02C8B                                0, 0, 8, 0, 7, 7, -114, 0, 0, 0, 8, 0, 8, -
00  00  92  0B  0B  00  08  00  00  00  91  0A  0A  00  08  02C9A                                8, -113, 0, 0, 0, 8, 0, 10, 10, -111, 0, -
00  94  1C  1C  00  08  00  00  00  93  1B  1B  00  08  00  02CA9                                0, 0, 8, 0, 11, 11, -110, 0, 0, 0, 8, 0, -
                                            00  08  00  00  02CB8                                27, 27, -109, 0, 0, 0, 8, 0, 28, 28, -
                                                                                                -108, 0, 0, 0, 8, 0
                                                        00000010  02CBC            .LONG      16
08  00  00  00  85  06  06  00  08  00  00  00  5A  0E  0E  02CC0  P.AFM:  .BYTE    14, 14, 90, 0, 0, 0, 8, 0, 6, 6, -123, 0, -
00  00  00  87  08  08  00  08  00  00  00  86  07  07  00  02CCF                                0, 0, 8, 0, 7, 7, -122, 0, 0, 0, 8, 0, 8, -
00  00  8A  0B  0B  00  08  00  00  00  89  0A  0A  00  08  02CDE                                8, -121, 0, 0, 0, 8, 0, 10, 10, -119, 0, -
00  8C  1C  1C  00  08  00  00  00  8B  1B  1B  00  08  00  02CED                                0, 0, 8, 0, 11, 11, -118, 0, 0, 0, 8, 0, -
                                            00  08  00  00  02CFC                                27, 27, -117, 0, 0, 0, 8, 0, 28, 28, -
                                                                                                -116, 0, 0, 0, 8, 0
                                                        00000006  02D00            .LONG      6
07  00  00  00  9C  07  07  00  06  00  00  00  9B  06  06  02D04  P.AFN:  .BYTE    6, 6, -101, 0, 0, 0, 6, 0, 7, 7, -100, 0, -
                        00  08  00  00  00  9D  08  08  00  02D13                                0, 0, 7, 0, 8, 8, -99, 0, 0, 0, 8, 0
                                                        00000006  02D1C            .LONG      6
```

```
07  00  00  00  96  07  07  00  06  00  00  00  95  06  06  02D20 P.AFO:  .BYTE   6, 6, -107, 0, 0, 0, 6, 0, 7, 7, -106, 0, -
                        00  08  00  00  00  97  08  08  00  02D2F                  0, 0, 7, 0, 8, 8, -105, 0, 0, 0, 8, 0
                                                00000006  02D38          .LONG   6
07  00  00  00  9F  07  07  00  06  00  00  00  9E  06  06  02D3C P.AFP:  .BYTE   6, 6, -98, 0, 0, 0, 6, 0, 7, 7, -97, 0, -
                        00  08  00  00  00  A0  08  08  00  02D4B                  0, 0, 7, 0, 8, 8, -96, 0, 0, 0, 8, 0
                                                00000006  02D54          .LONG   6
07  00  00  00  A2  07  07  00  06  00  00  00  A1  06  06  02D58 P.AFQ:  .BYTE   6, 6, -95, 0, 0, 0, 6, 0, 7, 7, -94, 0, -
                        00  08  00  00  00  A3  08  08  00  02D67                  0, 0, 7, 0, 8, 8, -93, 0, 0, 0, 8, 0
                                                00000006  02D70          .LONG   6
07  00  00  00  99  07  07  00  06  00  00  00  98  06  06  02D74 P.AFR:  .BYTE   6, 6, -104, 0, 0, 0, 6, 0, 7, 7, -103, 0, -
                        00  08  00  00  00  9A  08  08  00  02D83                  0, 0, 7, 0, 8, 8, -102, 0, 0, 0, 8, 0
                                                        00# 02D8C FORTRAN_OPINFO_TABLE:
                                                                   .BYTE   0[64]
                                00002864  00002804  00002904  02DCC          .LONG   10692, 10244, 10340
                                                        01  02DD8          .BYTE   1
                                                        00# 02DD9          .BYTE   0[3]
                                00002864  00002804  00002A30  02DDC          .LONG   10800, 10244, 10340
                                                        01  02DE8          .BYTE   1
                                                        00# 02DE9          .BYTE   0[3]
                                00002864  00002804  00002874  02DEC          .LONG   10356, 10244, 10340
                                                        01  02DF8          .BYTE   1
                                                        00# 02DF9          .BYTE   0[3]
                                00002864  00002804  000028C8  02DFC          .LONG   10440, 10244, 10340
                                                        01  02E08          .BYTE   1
                                                        00# 02E09          .BYTE   0[3]
                                00002864  00002804  0000291C  02E0C          .LONG   10524, 10244, 10340
                                                        01  02E18          .BYTE   1
                                                        00# 02E19          .BYTE   0[3]
                                00002864  00002804  00002970  02E1C          .LONG   10608, 10244, 10340
                                                        01  02E28          .BYTE   1
                                                        00# 02E29          .BYTE   0[3]
                                00002864  00002804  00002A9C  02E2C          .LONG   10908, 10244, 10340
                                                        01  02E38          .BYTE   1
                                                        00# 02E39          .BYTE   0[35]
                                00002864  00002804  00002B3C  02E5C          .LONG   11068, 10244, 10340
                                                        01  02E68          .BYTE   1
                                                        00# 02E69          .BYTE   0[3]
                                00002864  00002804  00002B98  02E6C          .LONG   11160, 10244, 10340
                                                        01  02E78          .BYTE   1
                                                        00# 02E79          .BYTE   0[3]
                                00002864  00002828  00002BF4  02E7C          .LONG   11252, 10280, 10340
                                                        01  02E88          .BYTE   1
                                                        00# 02E89          .BYTE   0[19]
                                00002864  00002828  00002C38  02E9C          .LONG   11320, 10280, 10340
                                                        01  02EA8          .BYTE   1
                                                        00# 02EA9          .BYTE   0[19]
                                00002864  00002828  00002C7C  02EBC          .LONG   11388, 10280, 10340
                                                        01  02EC8          .BYTE   1
                                                        00# 02EC9          .BYTE   0[19]
                                00002864  00002828  00002CC0  02EDC          .LONG   11456, 10280, 10340
                                                        01  02EE8          .BYTE   1
                                                        00# 02EE9          .BYTE   0[19]
                                00000000  0000283C  00002D04  02EFC          .LONG   11524, 10300, 0
                                                        01  02F08          .BYTE   1
                                                        00# 02F09          .BYTE   0[3]
                                00000000  0000283C  00002D20  02F0C          .LONG   11552, 10300, 0
                                                        01  02F18          .BYTE   1
```

```
                                00#  02F19                .BYTE   0[3]
00000000  0000283C  00002D3C      02F1C                .LONG   11580, 10300, 0
                                01   02F28                .BYTE   1
                                00#  02F29                .BYTE   0[3]
00000000  0000283C  00002D58      02F2C                .LONG   11608, 10300, 0
                                01   02F38                .BYTE   1
                                00#  02F39                .BYTE   0[3]
00000000  0000283C  00002D74      02F3C                .LONG   11636, 10300, 0
                                01   02F48                .BYTE   1
                                00#  02F49                .BYTE   0[115]
00000000  00000000  00002B30      02FBC                .LONG   11056, 0, 0
                                01   02FC8                .BYTE   1
                                00#  02FC9                .BYTE   0[227]
00002864  00002848  00000000      030AC                .LONG   0, 10312, 10340
                                01   030B8                .BYTE   1
                                00#  030B9                .BYTE   0[3]
00000000  0000283C  00000000      030BC                .LONG   0, 10300, 0
                                01   030C8                .BYTE   1
                                00#  030C9                .BYTE   0[67]
00000000  00000000  00000000      0310C                .LONG   0, 0, 0
                                01   03118                .BYTE   1
                                     03119                .BLKB   179
                          00000001   031CC                .LONG   1
                      0000  0816     031D0 P.AFS:         .WORD   2070, 0
                  00000000  000031D0 031D4 MACRO_CVTINFO_TABLE:
                                                         .LONG   12752, 0
                                00   031DC                .BYTE   0
                                     031DD                .BLKB   3
                          00000003   031E0                .LONG   3
                0000  0807  0706  0804  0403  0302  031E4 P.AFT:   .WORD   770, 1027, 2052, 1798, 2055, 0
                          00000007   031F0                .LONG   7
1601  0129  292A  2A22  2204  0408  0803  0307  0702  0206  031F4 P.AFU:   .WORD   518, 1794, 775, 2051, 1032, 8708, 10786, -
                                0000  0617  1716  03208                        10538, 297, 5633, 5910, 1559, 0
                                     0320E                .BLKB   2
                          00000002   03210                .LONG   2
                00  08  00  00  00  05  08  08  03214 P.AFV:   .BYTE   8, 8, 5, 0, 0, 0, 8, 0
                          00000002   0321C                .LONG   2
                00  08  00  00  00  13  08  08  03220 P.AFW:   .BYTE   8, 8, 19, 0, 0, 0, 8, 0
                          00000002   03228                .LONG   2
                00  08  00  00  00  2F  08  08  0322C P.AFX:   .BYTE   8, 8, 47, 0, 0, 0, 8, 0
                          00000002   03234                .LONG   2
                00  08  00  00  00  21  08  08  03238 P.AFY:   .BYTE   8, 8, 33, 0, 0, 0, 8, 0
                          00000002   03240                .LONG   2
                00  08  00  00  00  3D  08  08  03244 P.AFZ:   .BYTE   8, 8, 61, 0, 0, 0, 8, 0
                          00000002   0324C                .LONG   2
                00  08  00  00  00  39  08  08  03250 P.AGA:   .BYTE   8, 8, 57, 0, 0, 0, 8, 0
                          00000014   03258                .LONG   20
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06  0325C P.AGB:   .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  00  C1  0B  0B  00  0A  00  00  00  C0  0A  0A  00  0326B                  0, 0, 7, 0, 10, 10, -64, 0, 0, 0, 10, 0, -
00  00  C3  1C  1C  00  1B  00  00  00  C2  1B  1B  00  0B  0327A                  11, 11, -63, 0, 0, 0, 11, 0, 27, 27, -62, -
01  0A  09  09  00  15  00  00  00  F6  15  15  00  1C  00  03289                  0, 0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, -
BF  08  08  00  1A  00  00  01  0C  1A  1A  00  09  00  00  03298                  0, 21, 21, -10, 0, 0, 0, 21, 0, 9, 9, 10, -
                                0000  0807  0706  0804  0403  0302  031E4 P.AFT:   .WORD   770, 1027, 2052, 1798, 2055, 0
                                     0320E                .BLKB   2
                          00000002   03210                .LONG   2
                00  08  00  00  00  05  08  08  03214 P.AFV:   .BYTE   8, 8, 5, 0, 0, 0, 8, 0
                          00000002   0321C                .LONG   2
                00  08  00  00  00  13  08  08  03220 P.AFW:   .BYTE   8, 8, 19, 0, 0, 0, 8, 0
                          00000002   03228                .LONG   2
                00  08  00  00  00  2F  08  08  0322C P.AFX:   .BYTE   8, 8, 47, 0, 0, 0, 8, 0
                          00000002   03234                .LONG   2
                00  08  00  00  00  21  08  08  03238 P.AFY:   .BYTE   8, 8, 33, 0, 0, 0, 8, 0
                          00000002   03240                .LONG   2
                00  08  00  00  00  3D  08  08  03244 P.AFZ:   .BYTE   8, 8, 61, 0, 0, 0, 8, 0
                          00000002   0324C                .LONG   2
                00  08  00  00  00  39  08  08  03250 P.AGA:   .BYTE   8, 8, 57, 0, 0, 0, 8, 0
                          00000014   03258                .LONG   20
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06  032B0 P.AGC:   .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  00  B6  0B  0B  00  0A  00  00  00  B5  0A  0A  00  032BF                  0, 0, 7, 0, 10, 10, -75, 0, 0, 0, 10, 0, -
```

```
00 00 B8 1C 1C 00 1B 00 00 00 B7 1B 1B 00 0B 032CE                      11, 11, -74, 0, 0, 0, 11, 0, 27, 27, -73, -
01 0B 09 09 00 15 00 00 00 F7 15 15 00 1C 00 032DD                      0, 0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, -
B3 08 08 00 1A 00 00 01 0D 1A 1A 00 09 00 00 032EC                      0, 21, 21, -9, 0, 0, 0, 21, 0, 9, 9, 11, -
                              00 08 00 00 00 032FB                      1, 0, 0, 9, 0, 26, 26, 13, 1, 0, 0, 26, -
                                                                        0, 8, 8, -77, 0, 0, 0, 8, 0
                                 00000002 03300                 .LONG   2
                        00 08 00 00 00 61 08 08 03304 P.AGD:    .BYTE   8, 8, 97, 0, 0, 0, 8, 0
                                 00000002 0330C                 .LONG   2
                        00 08 00 00 00 6C 08 08 03310 P.AGE:    .BYTE   8, 8, 108, 0, 0, 0, 8, 0
                                 00000002 03318                 .LONG   2
                        00 08 00 00 00 8F 08 08 0331C P.AGF:    .BYTE   8, 8, -113, 0, 0, 0, 8, 0
                                 00000002 03324                 .LONG   2
                        00 08 00 00 00 7F 08 08 03328 P.AGG:    .BYTE   8, 8, 127, 0, 0, 0, 8, 0
                                 00000002 03330                 .LONG   2
                        00 08 00 00 00 87 08 08 03334 P.AGH:    .BYTE   8, 8, -121, 0, 0, 0, 8, 0
                                 00000002 0333C                 .LONG   2
                        00 08 00 00 00 77 08 08 03340 P.AGI:    .BYTE   8, 8, 119, 0, 0, 0, 8, 0
                                 00000002 03348                 .LONG   2
                        00 08 00 00 00 90 08 08 0334C P.AGJ:    .BYTE   8, 8, -112, 0, 0, 0, 8, 0
                                 00000002 03354                 .LONG   2
                        00 08 00 00 00 80 08 08 03358 P.AGK:    .BYTE   8, 8, -128, 0, 0, 0, 8, 0
                                 00000002 03360                 .LONG   2
                        00 08 00 00 00 88 08 08 03364 P.AGL:    .BYTE   8, 8, -120, 0, 0, 0, 8, 0
                                 00000002 0336C                 .LONG   2
                        00 08 00 00 00 78 08 08 03370 P.AGM:    .BYTE   8, 8, 120, 0, 0, 0, 8, 0
                                 00000002 03378                 .LONG   2
                        00 08 00 00 00 97 08 08 0337C P.AGN:    .BYTE   8, 8, -105, 0, 0, 0, 8, 0
                                 00000002 03384                 .LONG   2
                        00 08 00 00 00 A0 08 08 03388 P.AGO:    .BYTE   8, 8, -96, 0, 0, 0, 8, 0
                                 00000002 03390                 .LONG   2
                        00 08 00 00 00 A3 08 08 03394 P.AGP:    .BYTE   8, 8, -93, 0, 0, 0, 8, 0
                                 00000002 0339C                 .LONG   2
                        00 08 00 00 00 9A 08 08 033A0 P.AGQ:    .BYTE   8, 8, -102, 0, 0, 0, 8, 0
                                 00000002 033A8                 .LONG   2
                        00 08 00 00 00 9D 08 08 033AC P.AGR:    .BYTE   8, 8, -99, 0, 0, 0, 8, 0
                                 00000002 033B4                 .LONG   2
                        00 08 00 00 00 D1 08 08 033B8 P.AGS:    .BYTE   8, 8, -47, 0, 0, 0, 8, 0
                                 00000014 033C0                 .LONG   20
08 00 00 00 CF 07 07 00 08 00 00 00 CF 06 06 033C4 P.AGT:       .BYTE   6, 6, -49, 0, 0, 0, 8, 0, 7, 7, -49, 0, -
00 00 00 CF 03 03 00 08 00 00 00 CF 02 02 00 033D3                      0, 0, 8, 0, 2, 2, -49, 0, 0, 0, 8, 0, 3, -
00 00 CF 01 01 00 08 00 00 00 CF 04 04 00 08 033E2                      3, -49, 0, 0, 0, 8, 0, 4, 4, -49, 0, 0, -
00 CF 29 29 00 08 00 00 00 CF 22 22 00 08 00 033F1                      0, 8, 0, 1, 1, -49, 0, 0, 0, 8, 0, 34, -
CF 08 08 00 08 00 00 00 CF 2A 2A 00 08 00 00 03400                      34, -49, 0, 0, 0, 8, 0, 41, 41, -49, 0, -
                              00 08 00 00 00 0340F                      0, 0, 8, 0, 42, 42, -49, 0, 0, 0, 8, 0, -
                                                                        8, 8, -49, 0, 0, 0, 8, 0
                                    00# 03414 MACRO_OPINFO_TABLE:
                                                       .BYTE    0[48]
                  00000000 000031E4 000033C4  03444    .LONG    13252, 12772, 0
                                    01  03450    .BYTE    1
                                    00# 03451    .BYTE    0[3]
                  00000000 000031E4 0000325C  03454    .LONG    12892, 12772, 0
                                    00  03460    .BYTE    0
                                    00# 03461    .BYTE    0[3]
                  00000000 000031E4 000032B0  03464    .LONG    12976, 12772, 0
                                    00  03470    .BYTE    0
                                    00# 03471    .BYTE    0[3]
                  00000000 000031E4 00003214  03474    .LONG    12820, 12772, 0
```

```
                                    00  03480          .BYTE   0
                                    00# 03481          .BYTE   0[3]
00000000  000031E4  00003220  03484          .LONG   12832, 12772, 0
                                    00  03490          .BYTE   0
                                    00# 03491          .BYTE   0[3]
00000000  000031E4  0000322C  03494          .LONG   12844, 12772, 0
                                    00  034A0          .BYTE   0
                                    00# 034A1          .BYTE   0[3]
00000000  000031E4  00003238  034A4          .LONG   12856, 12772, 0
                                    00  034B0          .BYTE   0
                                    00# 034B1          .BYTE   0[51]
00000000  000031E4  00003304  034E4          .LONG   13060, 12772, 0
                                    00  034F0          .BYTE   0
                                    00# 034F1          .BYTE   0[3]
00000000  000031E4  00003310  034F4          .LONG   13072, 12772, 0
                                    00  03500          .BYTE   0
                                    00# 03501          .BYTE   0[3]
00000000  000031E4  00003328  03504          .LONG   13096, 12772, 0
                                    00  03510          .BYTE   0
                                    00# 03511          .BYTE   0[3]
00000000  000031E4  00003358  03514          .LONG   13144, 12772, 0
                                    00  03520          .BYTE   0
                                    00# 03521          .BYTE   0[3]
00000000  000031E4  00003340  03524          .LONG   13120, 12772, 0
                                    00  03530          .BYTE   0
                                    00# 03531          .BYTE   0[3]
00000000  000031E4  00003370  03534          .LONG   13168, 12772, 0
                                    00  03540          .BYTE   0
                                    00# 03541          .BYTE   0[3]
00000000  000031E4  0000331C  03544          .LONG   13084, 12772, 0
                                    00  03550          .BYTE   0
                                    00# 03551          .BYTE   0[3]
00000000  000031E4  0000334C  03554          .LONG   13132, 12772, 0
                                    00  03560          .BYTE   0
                                    00# 03561          .BYTE   0[3]
00000000  000031E4  00003334  03564          .LONG   13108, 12772, 0
                                    00  03570          .BYTE   0
                                    00# 03571          .BYTE   0[3]
00000000  000031E4  00003364  03574          .LONG   13156, 12772, 0
                                    00  03580          .BYTE   0
                                    00# 03581          .BYTE   0[115]
00000000  000031E4  000033AC  035F4          .LONG   13228, 12772, 0
                                    00  03600          .BYTE   0
                                    00# 03601          .BYTE   0[3]
00000000  000031E4  0000337C  03604          .LONG   13180, 12772, 0
                                    00  03610          .BYTE   0
                                    00# 03611          .BYTE   0[3]
00000000  000031E4  00003388  03614          .LONG   13192, 12772, 0
                                    00  03620          .BYTE   0
                                    00# 03621          .BYTE   0[3]
00000000  000031E4  00003394  03624          .LONG   13204, 12772, 0
                                    00  03630          .BYTE   0
                                    00# 03631          .BYTE   0[3]
00000000  000031E4  000033A0  03634          .LONG   13216, 12772, 0
                                    00  03640          .BYTE   0
                                    00# 03641          .BYTE   0[35]
00000000  000031E4  00003250  03664          .LONG   12880, 12772, 0
```

```
                                                    00  03670          .BYTE   0
                                                    00# 03671          .BYTE   0[3]
                                00000000  000031E4  00003244  03674          .LONG   12868, 12772, 0
                                                    00  03680          .BYTE   0
                                                    00# 03681          .BYTE   0[163]
                                00000000  000031E4  00003388  03724          .LONG   13240, 12772, 0
                                                    00  03730          .BYTE   0
                                                    00# 03731          .BYTE   0[3]
                                00000000  000031F4  00000000  03734          .LONG   0, 12788, 0
                                                    00  03740          .BYTE   0
                                                    00# 03741          .BYTE   0[3]
                                00000000  000031E4  00000000  03744          .LONG   0, 12772, 0
                                                    00  03750          .BYTE   0
                                                    00# 03751          .BYTE   0[67]
                                00000000  00000000  00000000  03794          .LONG   0, 0, 0
                                                    00  037A0          .BYTE   0
                                                        037A1          .BLKB   179
                                                00000001  03854          .LONG   1
                                              289E  0431  03858 P.AGU:   .WORD   1073, 10398
                                          00000000  00003858  0385C PASCAL_CVTINFO_TABLE:
                                                                         .LONG   14424, 0
                                                    01  03864          .BYTE   1
                                                        03865          .BLKB   3
                                                00000005  03868          .LONG   5
0000  1C1B  1C0B  1B0A  0B0A  0A04  0408  0807  0706  250E  0386C P.AGV:   .WORD   9486, 1798, 2055, 1032, 2564, 2826, 6922, -
                                                                               7179, 7195, 0
                                                00000002  03880          .LONG   2
                                      0000  0408  0807  0706  03884 P.AGW:   .WORD   1798, 2055, 1032, 0
                                                00000008  0388C          .LONG   8
250E  1C1B  1C0B  1B0A  0B0A  0A04  3304  2F04  3404  0408  03890 P.AGX:   .WORD   1032, 13316, 12036, 13060, 2564, 2826, -
                          3428  3328  342F  332F  340E  330E  038A4                6922, 7179, 7195, 9486, 13070, 13326, -
                                                                               13103, 13359, 13096, 13352
                                                00000001  038B0          .LONG   1
                                              0000  1B0B  038B4 P.AGY:   .WORD   6923, 0
                                                00000010  038B8          .LONG   16
33  00  02  00  D9  33  33  00  25  00  00  00  52  25  25  038BC P.AGZ:   .BYTE   37, 37, 82, 0, 0, 0, 37, 0, 51, 51, -39, -
00  00  00  06  04  04  00  08  00  00  00  05  08  08  00  038CB                0, 2, 0, 51, 0, 8, 8, 5, 0, 0, 0, 8, 0, -
00  00  08  0B  0B  00  0A  00  00  00  07  0A  0A  00  04  038DA                4, 4, 6, 0, 0, 0, 4, 0, 10, 10, 7, 0, 0, -
00  0A  1C  1C  00  1B  00  00  00  09  1B  1B  00  0B  00  038E9                0, 10, 0, 11, 11, 8, 0, 0, 0, 11, 0, 27, -
                                                    00  1C  00  00  038F8          27, 9, 0, 0, 0, 27, 0, 28, 28, 10, 0, 0, -
                                                                               0, 28, 0
                                                0000000E  038FC          .LONG   14
08  00  00  00  13  08  08  00  33  00  02  00  D2  33  33  03900 P.AHA:   .BYTE   51, 51, -46, 0, 2, 0, 51, 0, 8, 8, 19, 0, -
00  00  00  15  0A  0A  00  04  00  00  00  14  04  04  00  0390F                0, 0, 8, 0, 4, 4, 20, 0, 0, 0, 4, 0, 10, -
00  00  17  1B  1B  00  0B  00  00  00  16  0B  0B  00  0A  0391E                10, 21, 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, -
                    00  1C  00  00  00  18  1C  1C  00  1B  00  0392D          0, 11, 0, 27, 27, 23, 0, 0, 0, 27, 0, 28, -
                                                                               28, 24, 0, 0, 0, 28, 0
                                                0000000E  03938          .LONG   14
08  00  00  00  2F  08  08  00  33  00  02  00  D6  33  33  0393C P.AHB:   .BYTE   51, 51, -42, 0, 2, 0, 51, 0, 8, 8, 47, 0, -
00  00  00  31  0A  0A  00  04  00  00  00  30  04  04  00  0394B                0, 0, 8, 0, 4, 4, 48, 0, 0, 0, 4, 0, 10, -
00  00  33  1B  1B  00  0B  00  00  00  32  0B  0B  00  0A  0395A                10, 49, 0, 0, 0, 10, 0, 11, 11, 50, 0, 0, -
                    00  1C  00  00  00  34  1C  1C  00  1B  00  03969          0, 11, 0, 27, 27, 51, 0, 0, 0, 27, 0, 28, -
                                                                               28, 52, 0, 0, 0, 28, 0
                                                0000000C  03974          .LONG   12
04  00  00  00  41  04  04  00  08  00  00  00  41  08  08  03978 P.AHC:   .BYTE   8, 8, 65, 0, 0, 0, 8, 0, 4, 4, 65, 0, 0, -
00  00  00  4C  0B  0B  00  0A  00  00  00  49  0A  0A  00  03987                0, 4, 0, 10, 10, 73, 0, 0, 0, 10, 0, 11, -
00  00  4E  1C  1C  00  1B  00  00  00  4D  1B  1B  00  0B  03996                11, 76, 0, 0, 0, 11, 0, 27, 27, 77, 0, 0, -
```

```
                                                  00  1C  00   039A5                    0, 27, 0, 28, 28, 78, 0, 0, 0, 28, 0
                                                00000008   039A8            .LONG   8
0B  00  00  00  24  0B  0B  00  0A  00  00  00  23  0A  0A   039AC P.AHD:    .BYTE   10, 10, 35, 0, 0, 0, 10, 0, 11, 11, 36, -
00  00  00  26  1C  1C  00  1B  00  00  00  25  1B  1B  00   039BB                    0, 0, 0, 11, 0, 27, 27, 37, 0, 0, 0, 27, -
                                                    00  1C   039CA                    0, 28, 28, 38, 0, 0, 0, 28, 0
                                                00000016   039CC            .LONG   22
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06   039D0 P.AHE:    .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  01  0A  09  09  00  15  00  00  00  F6  15  15  00   039DF                    0, 0, 7, 0, 21, 21, -10, 0, 0, 0, 21, 0, -
00  00  BF  08  08  00  1A  00  00  01  0C  1A  1A  00  09   039EE                    9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -
00  C0  0A  0A  00  04  00  00  00  BF  04  04  00  08  00   039FD                    0, 0, 26, 0, 8, 8, -65, 0, 0, 0, 8, 0, 4, -
C2  1B  1B  00  0B  00  00  00  C1  0B  0B  00  0A  00  00   03A0C                    4, -65, 0, 0, 0, 4, 0, 10, 10, -64, 0, 0, -
        00  1C  00  00  00  C3  1C  1C  00  1B  00  00  00   03A1B                    0, 10, 0, 11, 11, -63, 0, 0, 0, 11, 0, -
                                                                                     27, 27, -62, 0, 0, 0, 27, 0, 28, 28, -61, -
                                                                                     0, 0, 0, 28, 0
                                                00000016   03A28            .LONG   22
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06   03A2C P.AHF:    .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  01  0B  09  09  00  15  00  00  00  F7  15  15  00   03A3B                    0, 0, 7, 0, 21, 21, -9, 0, 0, 0, 21, 0, -
00  00  B3  08  08  00  1A  00  00  01  0D  1A  1A  00  09   03A4A                    9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -
00  B5  0A  0A  00  04  00  00  00  B3  04  04  00  08  00   03A59                    0, 0, 26, 0, 8, 8, -77, 0, 0, 0, 8, 0, 4, -
B7  1B  1B  00  0B  00  00  00  B6  0B  0B  00  0A  00  00   03A68                    4, -77, 0, 0, 0, 4, 0, 10, 10, -75, 0, 0, -
        00  1C  00  00  00  B8  1C  1C  00  1B  00  00  00   03A77                    0, 10, 0, 11, 11, -74, 0, 0, 0, 11, 0, -
                                                                                     27, 27, -73, 0, 0, 0, 27, 0, 28, 28, -72, -
                                                                                     0, 0, 0, 28, 0
                                                00000004   03A84            .LONG   4
04  00  00  00  22  04  04  00  08  00  00  00  21  08  08   03A88 P.AHG:    .BYTE   8, 8, 33, 0, 0, 0, 8, 0, 4, 4, 34, 0, 0, -
                                                        00   03A97                    0, 4, 0
                                                00000004   03A98            .LONG   4
04  00  00  00  3A  04  04  00  08  00  00  00  39  08  08   03A9C P.AHH:    .BYTE   8, 8, 57, 0, 0, 0, 8, 0, 4, 4, 58, 0, 0, -
                                                        00   03AAB                    0, 4, 0
                                                00000004   03AAC            .LONG   4
04  00  00  00  3C  04  04  00  08  00  00  00  3B  08  08   03AB0 P.AHI:    .BYTE   8, 8, 59, 0, 0, 0, 8, 0, 4, 4, 60, 0, 0, -
                                                        00   03ABF                    0, 4, 0
                                                00000014   03AC0            .LONG   20
28  00  00  00  61  28  28  00  28  00  02  00  D3  33  33   03AC4 P.AHJ:    .BYTE   51, 51, -45, 0, 2, 0, 40, 0, 40, 40, 97, -
00  00  00  53  25  25  00  28  00  01  00  61  2F  2F  00   03AD3                    0, 0, 0, 40, 0, 47, 47, 97, 0, 1, 0, 40, -
00  00  61  04  04  00  28  00  00  00  61  08  08  00  28   03AE2                    0, 37, 37, 83, 0, 0, 0, 40, 0, 8, 8, 97, -
00  63  0B  0B  00  28  00  00  00  62  0A  0A  00  28  00   03AF1                    0, 0, 0, 40, 0, 4, 4, 97, 0, 0, 0, 40, 0, -
65  1C  1C  00  28  00  00  00  64  1B  1B  00  28  00  00   03B00                    10, 10, 98, 0, 0, 0, 40, 0, 11, 11, 99, -
                                                    00  28   03B0F                    0, 0, 0, 40, 0, 27, 27, 100, 0, 0, 0, 40, -
                                                                                     0, 28, 28, 101, 0, 0, 0, 40, 0
                                                00000014   03B14            .LONG   20
28  00  00  00  6C  28  28  00  28  00  02  00  D8  33  33   03B18 P.AHK:    .BYTE   51, 51, -40, 0, 2, 0, 40, 0, 40, 40, 108, -
00  00  00  5D  25  25  00  28  00  01  00  6C  2F  2F  00   03B27                    0, 0, 0, 40, 0, 47, 47, 108, 0, 1, 0, 40, -
00  00  6C  04  04  00  28  00  00  00  6C  08  08  00  28   03B36                    0, 37, 37, 93, 0, 0, 0, 40, 0, 8, 8, 108, -
00  6E  0B  0B  00  28  00  00  00  6D  0A  0A  00  28  00   03B45                    0, 0, 0, 40, 0, 4, 4, 108, 0, 0, 0, 40, -
70  1C  1C  00  28  00  00  00  6F  1B  1B  00  28  00  00   03B54                    0, 10, 10, 109, 0, 0, 0, 40, 0, 11, 11, -
                                                    00  28   03B63                    110, 0, 0, 0, 40, 0, 27, 27, 111, 0, 0, -
                                                                                     0, 40, 0, 28, 28, 112, 0, 0, 0, 40, 0
                                                00000014   03B68            .LONG   20
28  00  00  00  77  28  28  00  28  00  02  00  D4  33  33   03B6C P.AHL:    .BYTE   51, 51, -44, 0, 2, 0, 40, 0, 40, 40, 119, -
00  00  00  55  25  25  00  28  00  01  00  77  2F  2F  00   03B7B                    0, 0, 0, 40, 0, 47, 47, 119, 0, 1, 0, 40, -
00  00  78  04  04  00  28  00  00  00  77  08  08  00  28   03B8A                    0, 37, 37, 85, 0, 0, 0, 40, 0, 8, 8, 119, -
00  7A  0B  0B  00  28  00  00  00  79  0A  0A  00  28  00   03B99                    0, 0, 0, 40, 0, 4, 4, 120, 0, 0, 0, 40, -
7C  1C  1C  00  28  00  00  00  7B  1B  1B  00  28  00  00   03BA8                    0, 10, 10, 121, 0, 0, 0, 40, 0, 11, 11, -
                                                    00  28   03BB7                    122, 0, 0, 0, 40, 0, 27, 27, 123, 0, 0, -
                                                                                     0, 40, 0, 28, 28, 124, 0, 0, 0, 40, 0
                                                00000012   03BBC            .LONG   18
```

```
28 00 01 00 7F 2F 2F 00 28 00 00 00 7F 28 28 03BC0 P.AHM:  .BYTE   40, 40, 127, 0, 0, 0, 40, 0, 47, 47, 127, -
00 00 00 7F 08 08 00 28 00 00 00 57 25 25 00 03BCF                0, 1, 0, 40, 0, 37, 37, 87, 0, 0, 0, 40, -
00 00 81 0A 0A 00 28 00 00 00 80 04 04 00 28 03BDE                0, 8, 8, 127, 0, 0, 0, 40, 0, 4, 4, -128, -
00 83 1B 1B 00 28 00 00 00 82 0B 0B 00 28 00 03BED                0, 0, 0, 40, 0, 10, 10, -127, 0, 0, 0, -
         00 28 00 00 00 84 1C 1C 00 28 00 00 03BFC                40, 0, 11, 11, -126, 0, 0, 0, 40, 0, 27, -
                                                                  27, -125, 0, 0, 0, 40, 0, 28, 28, -124, -
                                                                  0, 0, 0, 40, 0
                                    00000012 03C08        .LONG   18
28 00 01 00 8F 2F 2F 00 28 00 00 00 8F 28 28 03C0C P.AHN:  .BYTE   40, 40, -113, 0, 0, 0, 40, 0, 47, 47, -
00 00 00 8F 08 08 00 28 00 00 00 5B 25 25 00 03C1B                -113, 0, 1, 0, 40, 0, 37, 37, 91, 0, 0, -
00 00 91 0A 0A 00 28 00 00 00 90 04 04 00 28 03C2A                0, 40, 0, 8, 8, -113, 0, 0, 0, 40, 0, 4, -
00 93 1B 1B 00 28 00 00 00 92 0B 0B 00 28 00 03C39                4, -112, 0, 0, 0, 40, 0, 10, 10, -111, 0, -
         00 28 00 00 00 94 1C 1C 00 28 00 00 03C48                0, 0, 40, 0, 11, 11, -110, 0, 0, 0, 40, -
                                                                  0, 27, 27, -109, 0, 0, 0, 40, 0, 28, 28, -
                                                                  -108, 0, 0, 0, 40, 0
                                    00000014 03C54        .LONG   20
28 00 00 00 87 28 28 00 28 00 02 00 D7 33 33 03C58 P.AHO:  .BYTE   51, 51, -41, 0, 2, 0, 40, 0, 40, 40, -
00 00 00 59 25 25 00 28 00 01 00 87 2F 2F 00 03C67                -121, 0, 0, 0, 40, 0, 47, 47, -121, 0, 1, -
00 00 88 04 04 00 28 00 00 00 87 08 08 00 28 03C76                0, 40, 0, 37, 37, 89, 0, 0, 0, 40, 0, 8, -
00 8A 0B 0B 00 28 00 00 00 89 0A 0A 00 28 00 03C85                8, -121, 0, 0, 0, 40, 0, 4, 4, -120, 0, -
8C 1C 1C 00 28 00 00 00 88 1B 1B 00 28 00 03C94                   0, 0, 40, 0, 10, 10, -119, 0, 0, 0, 40, -
                                    00 28 00 00 00 03CA3          0, 11, 11, -118, 0, 0, 0, 40, 0, 27, 27, -
                                                                  -117, 0, 0, 0, 40, 0, 28, 28, -116, 0, 0, -
                                                                  0, 40, 0
                                    00000002 03CA8        .LONG   2
                        00 28 00 00 00 AA 28 28 03CAC P.AHP:  .BYTE   40, 40, -86, 0, 0, 0, 40, 0
                                    00000002 03CB4        .LONG   2
                        00 28 00 00 00 A6 28 28 03CB8 P.AHQ:  .BYTE   40, 40, -90, 0, 0, 0, 40, 0
                                    00000002 03CC0        .LONG   2
                        00 28 00 00 00 AE 28 28 03CC4 P.AHR:  .BYTE   40, 40, -82, 0, 0, 0, 40, 0
                                    0000000A 03CCC        .LONG   10
28 00 02 00 D5 04 33 00 28 00 02 00 D5 08 33 03CD0 P.AHS:  .BYTE   51, 8, -43, 0, 2, 0, 40, 0, 51, 4, -43, -
00 02 00 D5 28 33 00 28 00 02 00 D5 0E 33 00 03CDF                0, 2, 0, 40, 0, 51, 14, -43, 0, 2, 0, 40, -
         00 28 00 02 00 D5 2F 33 00 28 03CEE                      0, 51, 40, -43, 0, 2, 0, 40, 0, 51, 47, -
                                                                  -43, 0, 2, 0, 40, 0
                                    00000002 03CF8        .LONG   2
                        00 2F 00 01 01 0E 2F 2F 03CFC P.AHT:  .BYTE   47, 47, 14, 1, 1, 0, 47, 0
                                    00000002 03D04        .LONG   2
                        00 2F 00 01 01 0F 2F 2F 03D08 P.AHU:  .BYTE   47, 47, 15, 1, 1, 0, 47, 0
                                             00# 03D10 PASCAL_OPINFO_TABLE:
                                                       .BYTE   0[64]
                  00003884 0000386C 00039D0  03D50           .LONG   14800, 14444, 14516
                                             01  03D5C           .BYTE   1
                                             00# 03D5D           .BYTE   0[3]
                  00003884 0000386C 00003A2C 03D60           .LONG   14892, 14444, 14516
                                             01  03D6C           .BYTE   1
                                             00# 03D6D           .BYTE   0[3]
                  00003884 0000386C 000038BC 03D70           .LONG   14524, 14444, 14516
                                             01  03D7C           .BYTE   1
                                             00# 03D7D           .BYTE   0[3]
                  00003884 0000386C 00003900 03D80           .LONG   14592, 14444, 14516
                                             01  03D8C           .BYTE   1
                                             00# 03D8D           .BYTE   0[3]
                  00003884 0000386C 0000393C 03D90           .LONG   14652, 14444, 14516
                                             01  03D9C           .BYTE   1
                                             00# 03D9D           .BYTE   0[3]
                  00003884 0000386C 000039AC 03DA0           .LONG   14764, 14444, 14516
```

```
                                    01  03DAC          .BYTE   1
                                    00# 03DAD          .BYTE   0[3]
00003884  0000386C  00003978  03DB0  .LONG   14712, 14444, 14516
                                    01  03DBC          .BYTE   1
                                    00# 03DBD          .BYTE   0[35]
00003884  0000386C  00003AC4  03DE0  .LONG   15044, 14444, 14516
                                    01  03DEC          .BYTE   1
                                    00# 03DED          .BYTE   0[3]
00003884  0000386C  00003B18  03DF0  .LONG   15128, 14444, 14516
                                    01  03DFC          .BYTE   1
                                    00# 03DFD          .BYTE   0[3]
00003884  0000386C  00003BC0  03E00  .LONG   15296, 14444, 14516
                                    01  03E0C          .BYTE   1
                                    00# 03E0D          .BYTE   0[19]
00003884  0000386C  00003B6C  03E20  .LONG   15212, 14444, 14516
                                    01  03E2C          .BYTE   1
                                    00# 03E2D          .BYTE   0[19]
00003884  0000386C  00003C0C  03E40  .LONG   15372, 14444, 14516
                                    01  03E4C          .BYTE   1
                                    00# 03E4D          .BYTE   0[19]
00003884  0000386C  00003C58  03E60  .LONG   15448, 14444, 14516
                                    01  03E6C          .BYTE   1
                                    00# 03E6D          .BYTE   0[19]
00000000  00000000  00003CAC  03E80  .LONG   15532, 0, 0
                                    01  03E8C          .BYTE   1
                                    00# 03E8D          .BYTE   0[3]
00000000  00000000  00003CB8  03E90  .LONG   15544, 0, 0
                                    01  03E9C          .BYTE   1
                                    00# 03E9D          .BYTE   0[3]
00000000  00000000  00003CC4  03EA0  .LONG   15556, 0, 0
                                    01  03EAC          .BYTE   1
                                    00# 03EAD          .BYTE   0[163]
00000000  00003884  00003A9C  03F50  .LONG   15004, 14468, 0
                                    01  03F5C          .BYTE   1
                                    00# 03F5D          .BYTE   0[3]
00003884  00003884  00003AB0  03F60  .LONG   15024, 14468, 14516
                                    01  03F6C          .BYTE   1
                                    00# 03F6D          .BYTE   0[67]
00000000  00000000  00003CD0  03FB0  .LONG   15568, 0, 0
                                    01  03FBC          .BYTE   1
                                    00# 03FBD          .BYTE   0[35]
00000000  00000000  00003CFC  03FE0  .LONG   15612, 0, 0
                                    01  03FEC          .BYTE   1
                                    00# 03FED          .BYTE   0[3]
00000000  00000000  00003D08  03FF0  .LONG   15624, 0, 0
                                    01  03FFC          .BYTE   1
                                    00# 03FFD          .BYTE   0[19]
00000000  00003884  00003A88  04010  .LONG   14984, 14468, 0
                                    01  0401C          .BYTE   1
                                    00# 0401D          .BYTE   0[19]
00003884  00003890  00000000  04030  .LONG   0, 14480, 14516
                                    01  0403C          .BYTE   1
                                    00# 0403D          .BYTE   0[3]
00003884  00003890  00000000  04040  .LONG   0, 14480, 14516
                                    01  0404C          .BYTE   1
                                    00# 0404D          .BYTE   0[67]
00000000  00000000  00000000  04090  .LONG   0, 0, 0
```

```
                                                      01  0409C             .BYTE   1                                       ;
                                                          0409D             .BLKB   179
                                                00000002  04150             .LONG   2                                       ;
                                      0000  0128  083B  04154 P.AHV:      .WORD   2107, 296, 0                            ;
                                                          0415A             .BLKB   2
                                                00000001  0415C             .LONG   1                                       ;
                                            0000  1B0B  04160 P.AHW:      .WORD   6923, 0                                 ;
                                                0000000B  04164             .LONG   11                                      ;
0001FF0E  00010EFF  0001FF22  000122FF  0001FF01  000101FF  04168 P.AHX:      .LONG   66047, 130817, 74495, 130850, 69375, -  ;
          00000000  000130FF  0001FF30  0001FF25  000125FF  04180                     130830, 75263, 130853, 130864, 78079, 0  ;
                                        00004168  00004154  04194 PLI_CVTINFO_TABLE:
                                                                            .LONG   16724, 16744                            ;
                                                      00  0419C             .BYTE   0                                       ;
                                                          0419D             .BLKB   3
                                                00000008  041A0             .LONG   8                                       ;
2B08  0A08  0815  1507  0706  1530  1522  1501  1525  150E  041A4 P.AHY:      .WORD   5390, 5413, 5377, 5410, 5424, 1798, 5383, - ;
                        0000  1C1B  1C0B  1C2B  1B0A  0B0A  041B8                     2069, 2568, 11016, 2826, 6922, 7211, -  ;
                                                                                    7179, 7195, 0                           ;
                                                0000000B  041C4             .LONG   11                                      ;
2501  0E01  1501  0122  2522  0E22  1522  150E  0E25  1525  041C8 P.AHZ:      .WORD   5413, 3621, 5390, 5410, 3618, 9506, 290, - ;
1C0B  1C2B  1B0A  0B0A  2B08  0A08  0815  1507  0706  1530  041DC                     5377, 3585, 9473, 5424, 1798, 5383, 2069, - ;
                                                0000  1C1B  041F0                     2568, 11016, 2826, 6922, 7211, 7179, -  ;
                                                                                    7195, 0                                 ;
                                                00000007  041F4             .LONG   7                                       ;
010E  0130  011C  011B  010B  010A  0115  0108  0107  0106  041F8 P.AIA:      .WORD   262, 263, 264, 277, 266, 267, 283, 284, - ;
                                          0000  0122  0125  0420C                     304, 270, 293, 290, 0                   ;
                                                          04212             .BLKB   2
                                                00000007  04214             .LONG   7                                       ;
0E25  0E30  0E1C  0E1B  0E0B  0E0A  0E15  0E08  0E07  0E06  04218 P.AIB:      .WORD   3590, 3591, 3592, 3605, 3594, 3595, 3611, - ;
                                          0000  0E01  0122  0422C                     3612, 3632, 3621, 290, 3585, 0          ;
                                                          04232             .BLKB   2
                                                00000008  04234             .LONG   8                                       ;
2B08  0A08  0815  1507  0706  1530  0822  0801  1525  150E  04238 P.AIC:      .WORD   5390, 5413, 2049, 2082, 5424, 1798, 5383, - ;
                        0000  1C1B  1C0B  1C2B  1B0A  0B0A  0424C                     2069, 2568, 11016, 2826, 6922, 7211, -  ;
                                                                                    7179, 7195, 0                           ;
                                                00000008  04258             .LONG   8                                       ;
2201  011C  1C1B  1B0B  0B0A  0A2B  2B15  1508  0807  0706  0425C P.AID:      .WORD   1798, 2055, 5384, 11029, 2603, 2826, -  ;
                              0000  0630  3025  250E  0E22  04270                     6923, 7195, 284, 8705, 3618, 9486, 12325, - ;
                                                                                    1584, 0                                 ;
                                                          0427A             .BLKB   2
                                                00000012  0427C             .LONG   18                                      ;
07  00  00  00  03  07  07  00  06  00  00  00  01  06  06  04280 P.AIE:      .BYTE   6, 6, 1, 0, 0, 0, 6, 0, 7, 7, 3, 0, 0, 0, - ;
00  00  00  07  0A  0A  00  08  00  00  00  05  08  08  00  0428F                     7, 0, 8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, - ;
00  00  09  1B  1B  00  0B  00  00  00  08  0B  0B  00  0A  0429E                     0, 0, 0, 10, 0, 11, 11, 8, 0, 0, 0, 11, - ;
00  F2  15  15  00  1C  00  00  00  0A  1C  1C  00  1B  00  042AD                     0, 27, 27, 9, 0, 0, 0, 27, 0, 28, 28, 10, - ;
            00  2B  00  00  01  13  2B  2B  00  15  00  00  042BC                     0, 0, 0, 28, 0, 21, 21, -14, 0, 0, 0, 21, - ;
                                                                                    0, 43, 43, 19, 1, 0, 0, 43, 0           ;
                                                00000012  042C8             .LONG   18                                      ;
07  00  00  00  11  07  07  00  06  00  00  00  0F  06  06  042CC P.AIF:      .BYTE   6, 6, 15, 0, 0, 0, 6, 0, 7, 7, 17, 0, 0, - ;
00  00  00  15  0A  0A  00  08  00  00  00  13  08  08  00  042DB                     0, 7, 0, 8, 8, 19, 0, 0, 0, 8, 0, 10, 10, - ;
00  00  17  1B  1B  00  0B  00  00  00  16  0B  0B  00  0A  042EA                     21, 0, 0, 0, 10, 0, 11, 11, 22, 0, 0, 0, - ;
00  F3  15  15  00  1C  00  00  00  18  1C  1C  00  1B  00  042F9                     11, 0, 27, 27, 23, 0, 0, 0, 27, 0, 28, - ;
            00  2B  00  00  01  14  2B  2B  00  15  00  00  04308                     28, 24, 0, 0, 0, 28, 0, 21, 21, -13, 0, - ;
                                                                                    0, 0, 21, 0, 43, 43, 20, 1, 0, 0, 43, 0 ;
                                                00000012  04314             .LONG   18                                      ;
07  00  00  00  2D  07  07  00  06  00  00  00  2B  06  06  04318 P.AIG:      .BYTE   6, 6, 43, 0, 0, 0, 6, 0, 7, 7, 45, 0, 0, - ;
00  00  00  31  0A  0A  00  08  00  00  00  2F  08  08  00  04327                     0, 7, 0, 8, 8, 47, 0, 0, 0, 8, 0, 10, 10, - ;
```

```
00 00 33 1B 1B 00 0B 00 00 00 32 0B 0B 00 0A 04336                                  49, 0, 0, 0, 10, 0, 11, 11, 50, 0, 0, 0, -
00 F4 15 15 00 1C 00 00 00 34 1C 1C 00 1B 00 04345                                  11, 0, 27, 27, 51, 0, 0, 0, 27, 0, 28, -
         00 2B 00 00 01 15 2B 2B 00 15 00 00 04354                                  28, 52, 0, 0, 0, 28, 0, 21, 21, -12, 0, -
                                                                                    0, 0, 21, 0, 43, 43, 21, 1, 0, 0, 43, 0
                                    00000012 04360                  .LONG           18
07 00 00 00 1F 07 07 00 06 00 00 00 1D 06 06 04364 P.AIH:   .BYTE   6, 6, 29, 0, 0, 0, 6, 0, 7, 7, 31, 0, 0, -
00 00 00 23 0A 0A 00 08 00 00 00 21 08 08 00 04373                                  0, 7, 0, 8, 8, 33, 0, 0, 0, 8, 0, 10, 10, -
00 00 25 1B 1B 00 0B 00 00 00 24 0B 0B 00 0A 04382                                  35, 0, 0, 0, 10, 0, 11, 11, 36, 0, 0, 0, -
00 F5 15 15 00 1C 00 00 00 26 1C 1C 00 1B 00 04391                                  11, 0, 27, 27, 37, 0, 0, 0, 27, 0, 28, -
         00 2B 00 00 01 16 2B 2B 00 15 00 00 043A0                                  28, 38, 0, 0, 0, 28, 0, 21, 21, -11, 0, -
                                                                                    0, 0, 21, 0, 43, 43, 22, 1, 0, 0, 43, 0
                                    00000016 043AC                  .LONG           22
1A 00 00 01 0C 1A 1A 00 09 00 00 01 0A 09 09 043B0 P.AII:   .BYTE   9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -
00 00 00 BE 07 07 00 06 00 00 00 BD 06 06 00 043BF                                  0, 0, 26, 0, 6, 6, -67, 0, 0, 0, 6, 0, 7, -
00 00 C0 0A 0A 00 08 00 00 00 BF 08 08 00 07 043CE                                  7, -66, 0, 0, 0, 7, 0, 8, 8, -65, 0, 0, -
00 C2 1B 1B 00 0B 00 00 00 C1 0B 0B 00 0A 00 043DD                                  0, 8, 0, 10, 10, -64, 0, 0, 0, 10, 0, 11, -
F6 15 15 00 1C 00 00 00 C3 1C 1C 00 1B 00 00 043EC                                  11, -63, 0, 0, 0, 11, 0, 27, 27, -62, 0, -
         00 2B 00 00 01 10 2B 2B 00 15 00 00 00 043FB                               0, 0, 27, 0, 28, 28, -61, 0, 0, 0, 28, 0, -
                                                                                    21, 21, -10, 0, 0, 0, 21, 0, 43, 43, 16, -
                                                                                    1, 0, 0, 43, 0
                                    00000016 04408                  .LONG           22
1A 00 00 01 0D 1A 1A 00 09 00 00 01 0B 09 09 0440C P.AIJ:   .BYTE   9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -
00 00 00 B2 07 07 00 06 00 00 00 B1 06 06 00 0441B                                  0, 0, 26, 0, 6, 6, -79, 0, 0, 0, 6, 0, 7, -
00 00 B5 0A 0A 00 08 00 00 00 B3 08 08 00 07 0442A                                  7, -78, 0, 0, 0, 7, 0, 8, 8, -77, 0, 0, -
00 B7 1B 1B 00 0B 00 00 00 B6 0B 0B 00 0A 00 04439                                  0, 8, 0, 10, 10, -75, 0, 0, 0, 10, 0, 11, -
F7 15 15 00 1C 00 00 00 B8 1C 1C 00 1B 00 00 04448                                  11, -74, 0, 0, 0, 11, 0, 27, 27, -73, 0, -
         00 2B 00 00 01 11 2B 2B 00 15 00 00 00 04457                               0, 0, 27, 0, 28, 28, -72, 0, 0, 0, 28, 0, -
                                                                                    21, 21, -9, 0, 0, 0, 21, 0, 43, 43, 17, -
                                                                                    1, 0, 0, 43, 0
                                    00000014 04464                  .LONG           20
0B 00 00 00 43 0B 08 00 0A 00 00 00 42 0A 08 04468 P.AIK:   .BYTE   8, 10, 66, 0, 0, 0, 10, 0, 8, 11, 67, 0, -
00 00 00 45 1C 08 00 1B 00 00 00 44 1B 08 00 04477                                  0, 0, 11, 0, 8, 27, 68, 0, 0, 0, 27, 0, -
00 00 4A 0B 0A 00 0A 00 00 00 49 0A 0A 00 1C 04486                                  8, 28, 69, 0, 0, 0, 28, 0, 10, 10, 73, 0, -
00 4C 0B 0B 00 0B 00 00 00 4B 0A 0B 00 0B 00 04495                                  0, 0, 10, 0, 10, 11, 74, 0, 0, 0, 11, 0, -
4E 1C 1C 00 1B 00 00 00 4D 1B 1B 00 0B 00 00 044A4                                  11, 10, 75, 0, 0, 0, 11, 0, 11, 11, 76, -
                                 00 1C 00 00 00 044B3                               0, 0, 0, 11, 0, 27, 27, 77, 0, 0, 0, 27, -
                                                                                    0, 28, 28, 78, 0, 0, 0, 28, 0
                                    00000004 044B8                  .LONG           4
01 00 00 00 FE 01 01 00 0E 00 00 00 52 0E 0E 044BC P.AIL:   .BYTE   14, 14, 82, 0, 0, 0, 14, 0, 1, 1, -2, 0, -
                                                 00 044CB                           0, 0, 1, 0
                                    0000001A 044CC                  .LONG           26
28 00 00 00 60 07 07 00 28 00 00 00 5F 06 06 044D0 P.AIM:   .BYTE   6, 6, 95, 0, 0, 0, 40, 0, 7, 7, 96, 0, 0, -
00 00 00 62 0A 0A 00 28 00 00 00 61 08 08 00 044DF                                  0, 40, 0, 8, 8, 97, 0, 0, 0, 40, 0, 10, -
00 00 64 1B 1B 00 28 00 00 00 63 0B 0B 00 28 044EE                                  10, 98, 0, 0, 0, 40, 0, 11, 11, 99, 0, 0, -
00 F8 15 15 00 28 00 00 00 65 1C 1C 00 28 00 044FD                                  0, 40, 0, 27, 27, 100, 0, 0, 0, 40, 0, -
54 0E 0E 00 28 00 00 01 17 2B 2B 00 28 00 00 0450C                                  28, 28, 101, 0, 0, 0, 40, 0, 21, 21, -8, -
01 01 00 28 00 00 00 53 25 25 00 28 00 00 00 0451B                                  0, 0, 0, 40, 0, 43, 43, 23, 1, 0, 0, 40, -
   00 28 00 00 00 FF 22 22 00 28 00 00 00 FF 0452A                                  0, 14, 14, 84, 0, 0, 0, 40, 0, 37, 37, -
                                                                                    83, 0, 0, 0, 40, 0, 1, 1, -1, 0, 0, 0, -
                                                                                    40, 0, 34, 34, -1, 0, 0, 0, 40, 0
                                    0000001A 04538                  .LONG           26
28 00 00 00 6B 07 07 00 28 00 00 00 6A 06 06 0453C P.AIN:   .BYTE   6, 6, 106, 0, 0, 0, 40, 0, 7, 7, 107, 0, -
00 00 00 6D 0A 0A 00 28 00 00 00 6C 08 08 00 0454B                                  0, 0, 40, 0, 8, 8, 108, 0, 0, 0, 40, 0, -
00 00 6F 1B 1B 00 28 00 00 00 6E 0B 0B 00 28 0455A                                  10, 10, 109, 0, 0, 0, 40, 0, 11, 11, 110, -
00 F9 15 15 00 28 00 00 00 70 1C 1C 00 28 00 04569                                  0, 0, 0, 40, 0, 27, 27, 111, 0, 0, 0, 40, -
5E 0E 0E 00 28 00 00 01 18 2B 2B 00 28 00 00 04578                                  0, 28, 28, 112, 0, 0, 0, 40, 0, 21, 21, -
01 01 00 28 00 00 00 5D 25 25 00 28 00 00 00 04587                                  -7, 0, 0, 0, 40, 0, 43, 43, 24, 1, 0, 0, -
```

```
     00  28  00  00  01  00  22  22  00  28  00  00  01  00  04596                    40, 0, 14, 14, 94, 0, 0, 0, 40, 0, 37, -
                                                                                    37, 93, 0, 0, 0, 40, 0, 1, 1, 0, 1, 0, 0, -
                                                                                    40, 0, 34, 34, 0, 1, 0, 0, 40, 0
                                                  0000001A  045A4          .LONG    26
28  00  00  00  7E  07  07  00  28  00  00  00  7D  06  06  045A8 P.AIO:   .BYTE    6, 6, 125, 0, 0, 0, 40, 0, 7, 7, 126, 0, -
00  00  00  81  0A  0A  00  28  00  00  00  7F  08  08  00  045B7                   0, 0, 40, 0, 8, 8, 127, 0, 0, 0, 40, 0, -
00  00  83  1B  1B  00  28  00  00  00  82  0B  0B  00  28  045C6                   10, 10, -127, 0, 0, 0, 40, 0, 11, 11, -
00  FA  15  15  00  28  00  00  00  84  1C  1C  00  28  00  045D5                   -126, 0, 0, 0, 40, 0, 27, 27, -125, 0, 0, -
58  0E  0E  00  28  00  00  01  1A  2B  2B  00  28  00  00  045E4                   0, 40, 0, 28, 28, -124, 0, 0, 0, 40, 0, -
22  22  00  28  00  00  00  57  25  25  00  28  00  00  00  045F3                   21, 21, -6, 0, 0, 0, 40, 0, 43, 43, 26, -
    00  28  00  00  01  01  01  01  00  28  00  00  01  01  04602                   1, 0, 0, 40, 0, 14, 14, 88, 0, 0, 0, 40, -
                                                                                    0, 37, 37, 87, 0, 0, 0, 40, 0, 34, 34, 1, -
                                                                                    1, 0, 0, 40, 0, 1, 1, 1, 1, 0, 0, 40, 0
                                                  0000001A  04610          .LONG    26
28  00  00  00  76  07  07  00  28  00  00  00  75  06  06  04614 P.AIP:   .BYTE    6, 6, 117, 0, 0, 0, 40, 0, 7, 7, 118, 0, -
00  00  00  79  0A  0A  00  28  00  00  00  77  08  08  00  04623                   0, 0, 40, 0, 8, 8, 119, 0, 0, 0, 40, 0, -
00  00  7B  1B  1B  00  28  00  00  00  7A  0B  0B  00  28  04632                   10, 10, 121, 0, 0, 0, 40, 0, 11, 11, 122, -
00  FB  15  15  00  28  00  00  00  7C  1C  1C  00  28  00  04641                   0, 0, 0, 40, 0, 27, 27, 123, 0, 0, 0, 40, -
56  0E  0E  00  28  00  00  01  1C  2B  2B  00  28  00  00  04650                   0, 28, 28, 124, 0, 0, 0, 40, 0, 21, 21, -
22  22  00  28  00  00  00  55  25  25  00  28  00  00  00  0465F                   -5, 0, 0, 0, 40, 0, 43, 43, 28, 1, 0, 0, -
    00  28  00  00  01  02  01  01  00  28  00  00  01  02  0466E                   40, 0, 14, 14, 86, 0, 0, 0, 40, 0, 37, -
                                                                                    37, 85, 0, 0, 0, 40, 0, 34, 34, 2, 1, 0, -
                                                                                    0, 40, 0, 1, 1, 2, 1, 0, 0, 40, 0
                                                  0000001A  0467C          .LONG    26
28  00  00  00  8E  07  07  00  28  00  00  00  8D  06  06  04680 P.AIQ:   .BYTE    6, 6, -115, 0, 0, 0, 40, 0, 7, 7, -114, -
00  00  00  91  0A  0A  00  28  00  00  00  8F  08  08  00  0468F                   0, 0, 0, 40, 0, 8, 8, -113, 0, 0, 0, 40, -
00  00  93  1B  1B  00  28  00  00  00  92  0B  0B  00  28  0469E                   0, 10, 10, -111, 0, 0, 0, 40, 0, 11, 11, -
00  FC  15  15  00  28  00  00  00  94  1C  1C  00  28  00  046AD                   -110, 0, 0, 0, 40, 0, 27, 27, -109, 0, 0, -
5C  0E  0E  00  28  00  00  01  19  2B  2B  00  28  00  00  046BC                   0, 40, 0, 28, 28, -108, 0, 0, 0, 40, 0, -
22  22  00  28  00  00  00  5B  25  25  00  28  00  00  00  046CB                   21, 21, -4, 0, 0, 0, 40, 0, 43, 43, 25, -
    00  28  00  00  01  03  01  01  00  28  00  00  01  03  046DA                   1, 0, 0, 40, 0, 14, 14, 92, 0, 0, 0, 40, -
                                                                                    0, 37, 37, 91, 0, 0, 0, 40, 0, 34, 34, 3, -
                                                                                    1, 0, 0, 40, 0, 1, 1, 3, 1, 0, 0, 40, 0
                                                  0000001A  046E8          .LONG    26
28  00  00  00  86  07  07  00  28  00  00  00  85  06  06  046EC P.AIR:   .BYTE    6, 6, -123, 0, 0, 0, 40, 0, 7, 7, -122, -
00  00  00  89  0A  0A  00  28  00  00  00  87  08  08  00  046FB                   0, 0, 0, 40, 0, 8, 8, -121, 0, 0, 0, 40, -
00  00  8B  1B  1B  00  28  00  00  00  8A  0B  0B  00  28  0470A                   0, 10, 10, -119, 0, 0, 0, 40, 0, 11, 11, -
00  FD  15  15  00  28  00  00  00  8C  1C  1C  00  28  00  04719                   -118, 0, 0, 0, 40, 0, 27, 27, -117, 0, 0, -
5A  0E  0E  00  28  00  00  01  1B  2B  2B  00  28  00  00  04728                   0, 40, 0, 28, 28, -116, 0, 0, 0, 40, 0, -
22  22  00  28  00  00  00  59  25  25  00  28  00  00  00  04737                   21, 21, -3, 0, 0, 0, 40, 0, 43, 43, 27, -
    00  28  00  00  01  04  01  01  00  28  00  00  01  04  04746                   1, 0, 0, 40, 0, 14, 14, 90, 0, 0, 0, 40, -
                                                                                    0, 37, 37, 89, 0, 0, 0, 40, 0, 34, 34, 4, -
                                                                                    1, 0, 0, 40, 0, 1, 1, 4, 1, 0, 0, 40, 0
                                                  00000002  04754          .LONG    2
                                00  01  00  00  01  05  01  01  04758 P.AIS:   .BYTE    1, 1, 5, 1, 0, 0, 1, 0
                                                  00000002  04760          .LONG    2
                                00  01  00  00  01  06  01  01  04764 P.AIT:   .BYTE    1, 1, 6, 1, 0, 0, 1, 0
                                                  00000002  0476C          .LONG    2
                                00  01  00  00  01  07  01  01  04770 P.AIU:   .BYTE    1, 1, 7, 1, 0, 0, 1, 0
                                                  00#  04778 PLI_OPINFO_TABLE:
                                                                           .BYTE    0[64]
                              00004160  00004238  000043B0  047B8          .LONG    17328, 16952, 16736
                                                  01  047C4               .BYTE    1
                                                  00#  047C5              .BYTE    0[3]
                              00004160  00004238  0000440C  047C8          .LONG    17420, 16952, 16736
                                                  01  047D4               .BYTE    1
                                                  00#  047D5              .BYTE    0[3]
```

```
00004160  000041A4  00004280  047D8          .LONG   17024, 16804, 16736
                         01  047E4          .BYTE   1
                        00#  047E5          .BYTE   0[3]
00004160  000041A4  000042CC  047E8          .LONG   17100, 16804, 16736
                         01  047F4          .BYTE   1
                        00#  047F5          .BYTE   0[3]
00004160  000041A4  00004318  047F8          .LONG   17176, 16804, 16736
                         01  04804          .BYTE   1
                        00#  04805          .BYTE   0[3]
00004160  000041A4  00004364  04808          .LONG   17252, 16804, 16736
                         01  04814          .BYTE   1
                        00#  04815          .BYTE   0[3]
00004160  000041A4  00004468  04818          .LONG   17512, 16804, 16736
                         01  04824          .BYTE   1
                        00#  04825          .BYTE   0[35]
00004160  000041C8  000044D0  04848          .LONG   17616, 16840, 16736
                         01  04854          .BYTE   1
                        00#  04855          .BYTE   0[3]
00004160  000041C8  0000453C  04858          .LONG   17724, 16840, 16736
                         01  04864          .BYTE   1
                        00#  04865          .BYTE   0[3]
00004160  000041C8  000045A8  04868          .LONG   17832, 16840, 16736
                         01  04874          .BYTE   1
                        00#  04875          .BYTE   0[19]
00004160  000041C8  00004614  04888          .LONG   17940, 16840, 16736
                         01  04894          .BYTE   1
                        00#  04895          .BYTE   0[19]
00004160  000041C8  00004680  048A8          .LONG   18048, 16840, 16736
                         01  048B4          .BYTE   1
                        00#  048B5          .BYTE   0[19]
00004160  000041C8  000046EC  048C8          .LONG   18156, 16840, 16736
                         01  048D4          .BYTE   1
                        00#  048D5          .BYTE   0[131]
00004160  000041F8  00004758  04958          .LONG   18264, 16888, 16736
                         01  04964          .BYTE   1
                        00#  04965          .BYTE   0[3]
00004160  000041F8  00004764  04968          .LONG   18276, 16888, 16736
                         01  04974          .BYTE   1
                        00#  04975          .BYTE   0[3]
00004160  000041F8  00004770  04978          .LONG   18288, 16888, 16736
                         01  04984          .BYTE   1
                        00#  04985          .BYTE   0[35]
00004160  00004218  000044BC  049A8          .LONG   17596, 16920, 16736
                         01  049B4          .BYTE   1
                        00#  049B5          .BYTE   0[227]
00004160  0000425C  00000000  04A98          .LONG   0, 16988, 16736
                         01  04AA4          .BYTE   1
                        00#  04AA5          .BYTE   0[3]
00000000  0000425C  00000000  04AA8          .LONG   0, 16988, 0
                         01  04AB4          .BYTE   1
                        00#  04AB5          .BYTE   0[67]
00000000  00000000  00000000  04AF8          .LONG   0, 0, 0
                         01  04B04          .BYTE   1
                             04B05          .BLKB   179
                  00000000  00000000  04BB8 RPG_CVTINFO_TABLE:
                                             .LONG   0, 0
                         00  04BC0          .BYTE   0
```

```
                                                04BC1             .BLKB    3
                                      00000002  04BC4             .LONG    2                                          ;
                        0000   1513  1508 1507  04BC8 P.AIV:      .WORD    5383, 5384, 5395, 0                        ;
                                      00000003  04BD0             .LONG    3                                          ;
                  0000  0715   1513  1308 0807  04BD4 P.AIW:      .WORD    2055, 4872, 5395, 1813, 0                  ;
                                                04BDE             .BLKB    2
                                      00000002  04BE0             .LONG    2                                          ;
                  00  15  00  00  00  F2  15 15 04BE4 P.AIX:      .BYTE    21, 21, -14, 0, 0, 0, 21, 0                ;
                                      00000002  04BEC             .LONG    2                                          ;
                  00  15  00  00  00  F3  15 15 04BF0 P.AIY:      .BYTE    21, 21, -13, 0, 0, 0, 21, 0                ;
                                      00000002  04BF8             .LONG    2                                          ;
                  00  15  00  00  00  F4  15 15 04BFC P.AIZ:      .BYTE    21, 21, -12, 0, 0, 0, 21, 0                ;
                                      00000002  04C04             .LONG    2                                          ;
                  00  15  00  00  00  F5  15 15 04C08 P.AJA:      .BYTE    21, 21, -11, 0, 0, 0, 21, 0                ;
                                      00000014  04C10             .LONG    20                                         ;
07 00 00 00 BE 07 07 00 06 00 00 00 BD 06 06    04C14 P.AJB:      .BYTE    6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -  ;
00 00 00 C0 0A 0A 00 08 00 00 00 BF 08 08 00    04C23                      0, 0, 7, 0, 8, 8, -65, 0, 0, 0, 8, 0, 10, - ;
00 00 C2 1B 1B 00 0B 00 00 00 C1 0B 0B 00 0A    04C32                      10, -64, 0, 0, 0, 10, 0, 11, 11, -63, 0, - ;
01 0A 09 09 00 1C 00 00 00 C3 1C 1C 00 1B 00    04C41                      0, 0, 11, 0, 27, 27, -62, 0, 0, 0, 27, 0, - ;
F6 15 15 00 1A 00 00 01 0C 1A 1A 00 09 00 00    04C50                      28, 28, -61, 0, 0, 0, 28, 0, 9, 9, 10, 1, - ;
                              00 15 00 00 00    04C5F                      0, 0, 9, 0, 26, 26, 12, 1, 0, 0, 26, 0, -  ;
                                                                           21, 21, -10, 0, 0, 0, 21, 0                ;
                                      00000014  04C64             .LONG    20                                         ;
07 00 00 00 B2 07 07 00 06 00 00 00 B1 06 06    04C68 P.AJC:      .BYTE    6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -  ;
00 00 00 B5 0A 0A 00 08 00 00 00 B3 08 08 00    04C77                      0, 0, 7, 0, 8, 8, -77, 0, 0, 0, 8, 0, 10, - ;
00 00 B7 1B 1B 00 0B 00 00 00 B6 0B 0B 00 0A    04C86                      10, -75, 0, 0, 0, 10, 0, 11, 11, -74, 0, - ;
01 0B 09 09 00 1C 00 00 00 B8 1C 1C 00 1B 00    04C95                      0, 0, 11, 0, 27, 27, -73, 0, 0, 0, 27, 0, - ;
F7 15 15 00 1A 00 00 01 0D 1A 1A 00 09 00 00    04CA4                      28, 28, -72, 0, 0, 0, 28, 0, 9, 9, 11, 1, - ;
                              00 15 00 00 00    04CB3                      0, 0, 9, 0, 26, 26, 13, 1, 0, 0, 26, 0, -  ;
                                                                           21, 21, -9, 0, 0, 0, 21, 0                 ;
                                      00000004  04CB8             .LONG    4                                          ;
28 00 00 00 F8 15 15 00 28 00 00 00 54 0E 0E    04CBC P.AJD:      .BYTE    14, 14, 84, 0, 0, 0, 40, 0, 21, 21, -8, -  ;
                                            00  04CCB                      0, 0, 0, 40, 0                             ;
                                      00000004  04CCC             .LONG    4                                          ;
28 00 00 00 F9 15 15 00 28 00 00 00 5E 0E 0E    04CD0 P.AJE:      .BYTE    14, 14, 94, 0, 0, 0, 40, 0, 21, 21, -7, -  ;
                                            00  04CDF                      0, 0, 0, 40, 0                             ;
                                      00000004  04CE0             .LONG    4                                          ;
28 00 00 00 FA 15 15 00 28 00 00 00 58 0E 0E    04CE4 P.AJF:      .BYTE    14, 14, 88, 0, 0, 0, 40, 0, 21, 21, -6, -  ;
                                            00  04CF3                      0, 0, 0, 40, 0                             ;
                                      00000004  04CF4             .LONG    4                                          ;
28 00 00 00 FB 15 15 00 28 00 00 00 56 0E 0E    04CF8 P.AJG:      .BYTE    14, 14, 86, 0, 0, 0, 40, 0, 21, 21, -5, -  ;
                                            00  04D07                      0, 0, 0, 40, 0                             ;
                                      00000004  04D08             .LONG    4                                          ;
28 00 00 00 FC 15 15 00 28 00 00 00 5C 0E 0E    04D0C P.AJH:      .BYTE    14, 14, 92, 0, 0, 0, 40, 0, 21, 21, -4, -  ;
                                            00  04D1B                      0, 0, 0, 40, 0                             ;
                                      00000004  04D1C             .LONG    4                                          ;
28 00 00 00 FD 15 15 00 28 00 00 00 5A 0E 0E    04D20 P.AJI:      .BYTE    14, 14, 90, 0, 0, 0, 40, 0, 21, 21, -3, -  ;
                                            00  04D2F                      0, 0, 0, 40, 0                             ;
                                      00000002  04D30             .LONG    2                                          ;
                  00  28  00  00  00  AA  28 28 04D34 P.AJJ:      .BYTE    40, 40, -86, 0, 0, 0, 40, 0                ;
                                      00000002  04D3C             .LONG    2                                          ;
                  00  28  00  00  00  A6  28 28 04D40 P.AJK:      .BYTE    40, 40, -90, 0, 0, 0, 40, 0                ;
                                      00000002  04D48             .LONG    2                                          ;
                  00  28  00  00  00  AE  28 28 04D4C P.AJL:      .BYTE    40, 40, -82, 0, 0, 0, 40, 0                ;
                                           00#  04D54 RPG_OPINFO_TABLE:
                                                                  .BYTE    0[64]                                      ;
                  00000000  00004BC8  00004C14  04D94             .LONG    19476, 19400, 0                            ;
```

```
                                    01  04DA0          .BYTE   1
                                    00# 04DA1          .BYTE   0[3]
00000000  00004BC8  00004C68  04DA4          .LONG   19560, 19400, 0
                                    01  04DB0          .BYTE   1
                                    00# 04DB1          .BYTE   0[3]
00000000  00004BC8  00004BE4  04DB4          .LONG   19428, 19400, 0
                                    01  04DC0          .BYTE   1
                                    00# 04DC1          .BYTE   0[3]
00000000  00004BC8  00004BF0  04DC4          .LONG   19440, 19400, 0
                                    01  04DD0          .BYTE   1
                                    00# 04DD1          .BYTE   0[3]
00000000  00004BC8  00004BFC  04DD4          .LONG   19452, 19400, 0
                                    01  04DE0          .BYTE   1
                                    00# 04DE1          .BYTE   0[3]
00000000  00004BC8  00004C08  04DE4          .LONG   19464, 19400, 0
                                    01  04DF0          .BYTE   1
                                    00# 04DF1          .BYTE   0[51]
00000000  00004BC8  00004CBC  04E24          .LONG   19644, 19400, 0
                                    01  04E30          .BYTE   1
                                    00# 04E31          .BYTE   0[3]
00000000  00004BC8  00004CD0  04E34          .LONG   19664, 19400, 0
                                    01  04E40          .BYTE   1
                                    00# 04E41          .BYTE   0[3]
00000000  00004BC8  00004CE4  04E44          .LONG   19684, 19400, 0
                                    01  04E50          .BYTE   1
                                    00# 04E51          .BYTE   0[19]
00000000  00004BC8  00004CF8  04E64          .LONG   19704, 19400, 0
                                    01  04E70          .BYTE   1
                                    00# 04E71          .BYTE   0[19]
00000000  00004BC8  00004D0C  04E84          .LONG   19724, 19400, 0
                                    01  04E90          .BYTE   1
                                    00# 04E91          .BYTE   0[19]
00000000  00004BC8  00004D20  04EA4          .LONG   19744, 19400, 0
                                    01  04EB0          .BYTE   1
                                    00# 04EB1          .BYTE   0[19]
00000000  00004BC8  00004D34  04EC4          .LONG   19764, 19400, 0
                                    01  04ED0          .BYTE   1
                                    00# 04ED1          .BYTE   0[3]
00000000  00004BC8  00004D40  04ED4          .LONG   19776, 19400, 0
                                    01  04EE0          .BYTE   1
                                    00# 04EE1          .BYTE   0[3]
00000000  00004BC8  00004D4C  04EE4          .LONG   19788, 19400, 0
                                    01  04EF0          .BYTE   1
                                    00# 04EF1          .BYTE   0[387]
00000000  00004BD4  00000000  05074          .LONG   0, 19412, 0
                                    01  05080          .BYTE   1
                                    00# 05081          .BYTE   0[3]
00000000  00004BD4  00000000  05084          .LONG   0, 19412, 0
                                    01  05090          .BYTE   1
                                    00# 05091          .BYTE   0[67]
00000000  00000000  00000000  050D4          .LONG   0, 0, 0
                                    01  050E0          .BYTE   1
                                        050E1          .BLKB   179
                  00000000  00000000  05194 UNKNOWN_CVTINFO_TABLE:
                                                       .LONG   0, 0
                                    00  0519C          .BYTE   0
                                        0519D          .BLKB   3
```

```
                                                  0000000A 051A0                 .LONG   10
0A08  0807  0706  0804  0403  0302  082F  0831  083B  0828 051A4 P.AJM:  .WORD   2088, 2107, 2097, 2095, 770, 1027, 2052, -
0000  1D0C  0D0C  1D1B  1C1B  0D0B  1C0B  0C0A  1B0A  0B0A 051B8                   1798, 2055, 2568, 2826, 6922, 3082, 7179, -
                                                                                3339, 7195, 7451, 3340, 7436, 0
                                                  0000000A 051CC                 .LONG   10
1C1B  1C0B  1B0A  0B0A  0A08  0804  0407  0703  0306  0602 051D0 P.AJN:  .WORD   1538, 774, 1795, 1031, 2052, 2568, 2826, -
0000  0228  283B  3B31  312F  2F1D  2F0D  1D0C  0D0C  0C1C 051E4                   6922, 7179, 7195, 3100, 3340, 7436, -
                                                                                12045, 12061, 12591, 15153, 10299, 552, 0
                                                  00000003 051F8                 .LONG   3
                                  0000  1D0D  0D1B  1D0B  1B0B 051FC P.AJO:  .WORD   6923, 7435, 3355, 7437, 0
                                                           05206                 .BLKB   2
                                                  00000010 05208                 .LONG   16
0A  00  00  00  07  0A  0A  00  08  00  00  00  05  08  08 0520C P.AJP:  .BYTE   8, 8, 5, 0, 0, 0, 8, 0, 10, 10, 7, 0, 0, -
00  00  00  09  1B  1B  00  0B  00  00  00  08  0B  0B  00 0521B                   0, 10, 0, 11, 11, 8, 0, 0, 0, 11, 0, 27, -
00  00  0B  0C  0C  00  1C  00  00  00  0A  1C  1C  00  1B 0522A                   27, 9, 0, 0, 0, 27, 0, 28, 28, 10, 0, 0, -
00  0D  1D  1D  00  0D  00  00  00  0C  0D  0D  00  0C  00 05239                   0, 28, 0, 12, 12, 11, 0, 0, 0, 12, 0, 13, -
                                                  00  1D  00  00 05248           13, 12, 0, 0, 0, 13, 0, 29, 29, 13, 0, 0, -
                                                                                0, 29, 0
                                                  00000010 0524C                 .LONG   16
0A  00  00  00  15  0A  0A  00  08  00  00  00  13  08  08 05250 P.AJQ:  .BYTE   8, 8, 19, 0, 0, 0, 8, 0, 10, 10, 21, 0, -
00  00  00  17  1B  1B  00  0B  00  00  00  16  0B  0B  00 0525F                   0, 0, 10, 0, 11, 11, 22, 0, 0, 0, 11, 0, -
00  00  19  0C  0C  00  1C  00  00  00  18  1C  1C  00  1B 0526E                   27, 27, 23, 0, 0, 0, 27, 0, 28, 28, 24, -
00  1B  1D  1D  00  0D  00  00  00  1A  0D  0D  00  0C  00 0527D                   0, 0, 0, 28, 0, 12, 12, 25, 0, 0, 0, 12, -
                                                  00  1D  00  00 0528C           0, 13, 13, 26, 0, 0, 0, 13, 0, 29, 29, -
                                                                                27, 0, 0, 0, 29, 0
                                                  00000010 05290                 .LONG   16
0A  00  00  00  31  0A  0A  00  08  00  00  00  2F  08  08 05294 P.AJR:  .BYTE   8, 8, 47, 0, 0, 0, 8, 0, 10, 10, 49, 0, -
00  00  00  33  1B  1B  00  0B  00  00  00  32  0B  0B  00 052A3                   0, 0, 10, 0, 11, 11, 50, 0, 0, 0, 11, 0, -
00  00  35  0C  0C  00  1C  00  00  00  34  1C  1C  00  1B 052B2                   27, 27, 51, 0, 0, 0, 27, 0, 28, 28, 52, -
00  37  1D  1D  00  0D  00  00  00  36  0D  0D  00  0C  00 052C1                   0, 0, 0, 28, 0, 12, 12, 53, 0, 0, 0, 12, -
                                                  00  1D  00  00 052D0           0, 13, 13, 54, 0, 0, 0, 13, 0, 29, 29, -
                                                                                55, 0, 0, 0, 29, 0
                                                  00000010 052D4                 .LONG   16
0A  00  00  00  23  0A  0A  00  08  00  00  00  21  08  08 052D8 P.AJS:  .BYTE   8, 8, 33, 0, 0, 0, 8, 0, 10, 10, 35, 0, -
00  00  00  25  1B  1B  00  0B  00  00  00  24  0B  0B  00 052E7                   0, 0, 10, 0, 11, 11, 36, 0, 0, 0, 11, 0, -
00  00  27  0C  0C  00  1C  00  00  00  26  1C  1C  00  1B 052F6                   27, 27, 37, 0, 0, 0, 27, 0, 28, 28, 38, -
00  29  1D  1D  00  0D  00  00  00  28  0D  0D  00  0C  00 05305                   0, 0, 0, 28, 0, 12, 12, 39, 0, 0, 0, 12, -
                                                  00  1D  00  00 05314           0, 13, 13, 40, 0, 0, 0, 13, 0, 29, 29, -
                                                                                41, 0, 0, 0, 29, 0
                                                  0000001A 05318                 .LONG   26
07  00  00  00  BE  07  07  00  06  00  00  00  BD  06  06 0531C P.AJT:  .BYTE   6, 6, -67, 0, 0, 0, 6, 0, 7, 7, -66, 0, -
00  00  01  0A  09  09  00  15  00  00  00  F6  15  15  00 0532B                   0, 0, 7, 0, 21, 21, -10, 0, 0, 0, 21, 0, -
00  00  BF  08  08  00  1A  00  00  01  0C  1A  1A  00  09 0533A                   9, 9, 10, 1, 0, 0, 9, 0, 26, 26, 12, 1, -
00  C1  0B  0B  00  0A  00  00  00  C0  0A  0A  00  08  00 05349                   0, 0, 26, 0, 8, 8, -65, 0, 0, 0, 8, 0, -
C3  1C  1C  00  1B  00  00  00  C2  1B  1B  00  0B  00  00 05358                   10, 10, -64, 0, 0, 0, 10, 0, 11, 11, -63, -
0D  0D  00  0C  00  00  00  C4  0C  0C  00  1C  00  00  00 05367                   0, 0, 0, 11, 0, 27, 27, -62, 0, 0, 0, 27, -
    00  1D  00  00  00  C6  1D  1D  00  0D  00  00  00  C5 05376                   0, 28, 28, -61, 0, 0, 0, 28, 0, 12, 12, -
                                                                                -60, 0, 0, 0, 12, 0, 13, 13, -59, 0, 0, -
                                                                                0, 13, 0, 29, 29, -58, 0, 0, 0, 29, 0
                                                  0000001A 05384                 .LONG   26
07  00  00  00  B2  07  07  00  06  00  00  00  B1  06  06 05388 P.AJU:  .BYTE   6, 6, -79, 0, 0, 0, 6, 0, 7, 7, -78, 0, -
00  00  01  0B  09  09  00  15  00  00  00  F7  15  15  00 05397                   0, 0, 7, 0, 21, 21, -9, 0, 0, 0, 21, 0, -
00  00  B3  08  08  00  1A  00  00  01  0D  1A  1A  00  09 053A6                   9, 9, 11, 1, 0, 0, 9, 0, 26, 26, 13, 1, -
00  B6  0B  0B  00  0A  00  00  00  B5  0A  0A  00  08  00 053B5                   0, 0, 26, 0, 8, 8, -77, 0, 0, 0, 8, 0, -
B8  1C  1C  00  1B  00  00  00  B7  1B  1B  00  0B  00  00 053C4                   10, 10, -75, 0, 0, 0, 10, 0, 11, 11, -74, -
0D  0D  00  0C  00  00  00  B9  0C  0C  00  1C  00  00  00 053D3                   0, 0, 0, 11, 0, 27, 27, -73, 0, 0, 0, 27, -
    00  1D  00  00  00  BB  1D  1D  00  0D  00  00  00  BA 053E2                   0, 28, 28, -72, 0, 0, 0, 28, 0, 12, 12, -
```

```
                                                                                  -71, 0, 0, 0, 12, 0, 13, 13, -70, 0, 0, -  ;
                                                                                  0, 13, 0, 29, 29, -69, 0, 0, 0, 29, 0
                                                   00000024  053F0          .LONG 36
08 00 00 00 41 08 08 00 07 00 00 00 40 07 07  053F4 P.AJV:   .BYTE 7, 7, 64, 0, 0, 0, 7, 0, 8, 8, 65, 0, 0, -
00 00 00 43 0B 08 00 0A 00 00 00 42 0A 08 00  05403                 0, 8, 0, 8, 10, 66, 0, 0, 0, 10, 0, 8, -
00 00 45 1C 08 00 1B 00 00 00 44 1B 08 00 0B  05412                 11, 67, 0, 0, 0, 11, 0, 8, 27, 68, 0, 0, -
00 47 0D 08 00 0C 00 00 00 46 0C 08 00 1C 00  05421                 0, 27, 0, 8, 28, 69, 0, 0, 0, 28, 0, 8, -
49 0A 0A 00 1D 00 00 00 48 1D 08 00 0D 00 00  05430                 12, 70, 0, 0, 0, 12, 0, 8, 13, 71, 0, 0, -
0A 0B 00 0B 00 00 00 4A 0B 0A 00 0A 00 00 00  0543F                 0, 13, 0, 8, 29, 72, 0, 0, 0, 29, 0, 10, -
1B 00 0B 00 00 00 4C 0B 0B 00 0B 00 00 00 4B  0544E                 10, 73, 0, 0, 0, 10, 0, 10, 11, 74, 0, 0, -
00 1C 00 00 00 4E 1C 1C 00 1B 00 00 00 4D 1B  0545D                 0, 11, 0, 11, 10, 75, 0, 0, 0, 11, 0, 11, -
0D 00 00 00 50 0D 0D 00 0C 00 00 00 4F 0C 0C  0546C                 11, 76, 0, 0, 0, 11, 0, 27, 27, 77, 0, 0, -
                  00 1D 00 00 00 51 1D 1D 00  0547B                 0, 27, 0, 28, 28, 78, 0, 0, 0, 28, 0, 12, -
                                                                    12, 79, 0, 0, 0, 12, 0, 13, 13, 80, 0, 0, -
                                                                    0, 13, 0, 29, 29, 81, 0, 0, 0, 29, 0
                                                   00000012  05484          .LONG 18
28 00 00 00 61 08 08 00 28 00 00 00 54 0E 0E  05488 P.AJW:   .BYTE 14, 14, 84, 0, 0, 0, 40, 0, 8, 8, 97, 0, -
00 00 00 63 0B 0B 00 28 00 00 00 62 0A 0A 00  05497                 0, 0, 40, 0, 10, 10, 98, 0, 0, 0, 40, 0, -
00 00 65 1C 1C 00 28 00 00 00 64 1B 1B 00 28  054A6                 11, 11, 99, 0, 0, 0, 40, 0, 27, 27, 100, -
00 67 0D 0D 00 28 00 00 00 66 0C 0C 00 28 00  054B5                 0, 0, 0, 40, 0, 28, 28, 101, 0, 0, 0, 40, -
            00 28 00 00 00 68 1D 1D 00 28 00 00  054C4                 0, 12, 12, 102, 0, 0, 0, 40, 0, 13, 13, -
                                                                    103, 0, 0, 0, 40, 0, 29, 29, 104, 0, 0, -
                                                                    0, 40, 0
                                                   00000012  054D0          .LONG 18
28 00 00 00 6C 08 08 00 28 00 00 00 5E 0E 0E  054D4 P.AJX:   .BYTE 14, 14, 94, 0, 0, 0, 40, 0, 8, 8, 108, 0, -
00 00 00 6E 0B 0B 00 28 00 00 00 6D 0A 0A 00  054E3                 0, 0, 40, 0, 10, 10, 109, 0, 0, 0, 40, 0, -
00 00 70 1C 1C 00 28 00 00 00 6F 1B 1B 00 28  054F2                 11, 11, 110, 0, 0, 0, 40, 0, 27, 27, 111, -
00 72 0D 0D 00 28 00 00 00 71 0C 0C 00 28 00  05501                 0, 0, 0, 40, 0, 28, 28, 112, 0, 0, 0, 40, -
            00 28 00 00 00 73 1D 1D 00 28 00 00  05510                 0, 12, 12, 113, 0, 0, 0, 40, 0, 13, 13, -
                                                                    114, 0, 0, 0, 40, 0, 29, 29, 115, 0, 0, -
                                                                    0, 40, 0
                                                   0000000C  0551C          .LONG 12
28 00 00 00 8F 08 08 00 28 00 00 00 5C 0E 0E  05520 P.AJY:   .BYTE 14, 14, 92, 0, 0, 0, 40, 0, 8, 8, -113, -
00 00 00 92 0B 0B 00 28 00 00 00 91 0A 0A 00  0552F                 0, 0, 0, 40, 0, 10, 10, -111, 0, 0, 0, -
00 00 94 1C 1C 00 28 00 00 00 93 1B 1B 00 28  0553E                 40, 0, 11, 11, -110, 0, 0, 0, 40, 0, 27, -
                                    00 28 00  0554D                 27, -109, 0, 0, 0, 40, 0, 28, 28, -108, -
                                                                    0, 0, 0, 40, 0
                                                   0000000C  05550          .LONG 12
28 00 00 00 7F 08 08 00 28 00 00 00 58 0E 0E  05554 P.AJZ:   .BYTE 14, 14, 88, 0, 0, 0, 40, 0, 8, 8, 127, 0, -
00 00 00 82 0B 0B 00 28 00 00 00 81 0A 0A 00  05563                 0, 0, 40, 0, 10, 10, -127, 0, 0, 0, 40, -
00 00 84 1C 1C 00 28 00 00 00 83 1B 1B 00 28  05572                 0, 11, 11, -126, 0, 0, 0, 40, 0, 27, 27, -
                                    00 28 00  05581                 -125, 0, 0, 0, 40, 0, 28, 28, -124, 0, 0, -
                                                                    0, 40, 0
                                                   0000000C  05584          .LONG 12
28 00 00 00 87 08 08 00 28 00 00 00 5A 0E 0E  05588 P.AKA:   .BYTE 14, 14, 90, 0, 0, 0, 40, 0, 8, 8, -121, -
00 00 00 8A 0B 0B 00 28 00 00 00 89 0A 0A 00  05597                 0, 0, 0, 40, 0, 10, 10, -119, 0, 0, 0, -
00 00 8C 1C 1C 00 28 00 00 00 8B 1B 1B 00 28  055A6                 40, 0, 11, 11, -118, 0, 0, 0, 40, 0, 27, -
                                    00 28 00  055B5                 27, -117, 0, 0, 0, 40, 0, 28, 28, -116, -
                                                                    0, 0, 0, 40, 0
                                                   0000000C  055B8          .LONG 12
28 00 00 00 77 08 08 00 28 00 00 00 56 0E 0E  055BC P.AKB:   .BYTE 14, 14, 86, 0, 0, 0, 40, 0, 8, 8, 119, 0, -
00 00 00 7A 0B 0B 00 28 00 00 00 79 0A 0A 00  055CB                 0, 0, 40, 0, 10, 10, 121, 0, 0, 0, 40, 0, -
00 00 7C 1C 1C 00 28 00 00 00 7B 1B 1B 00 28  055DA                 11, 11, 122, 0, 0, 0, 40, 0, 27, 27, 123, -
                                    00 28 00  055E9                 0, 0, 0, 40, 0, 28, 28, 124, 0, 0, 0, 40, -
                                                                    0
                                                   00000002  055EC          .LONG 2
                  00 08 00 00 00 97 08 08  055F0 P.AKC:   .BYTE 8, 8, -105, 0, 0, 0, 8, 0
```

```
                                00000002  055F8                   .LONG   2
00  08  00  00  00  A0  08  08  055FC P.AKD:   .BYTE   8, 8, -96, 0, 0, 0, 8, 0
                                00000002  05604                   .LONG   2
00  08  00  00  00  A3  08  08  05608 P.AKE:   .BYTE   8, 8, -93, 0, 0, 0, 8, 0
                                00000002  05610                   .LONG   2
00  08  00  00  00  9A  08  08  05614 P.AKF:   .BYTE   8, 8, -102, 0, 0, 0, 8, 0
                                00000002  0561C                   .LONG   2
00  08  00  00  00  9D  08  08  05620 P.AKG:   .BYTE   8, 8, -99, 0, 0, 0, 8, 0
                                00000002  05628                   .LONG   2
00  0E  00  00  00  52  0E  0E  0562C P.AKH:   .BYTE   14, 14, 82, 0, 0, 0, 14, 0
                                    00# 05634 UNKNOWN_OPINFO_TABLE:
                                                                   .BYTE   0[64]
  000051FC  000051A4  0000531C  05674                   .LONG   21276, 20900, 20988
                                    01  05680                   .BYTE   1
                                    00# 05681                   .BYTE   0[3]
  000051FC  000051A4  00005388  05684                   .LONG   21384, 20900, 20988
                                    01  05690                   .BYTE   1
                                    00# 05691                   .BYTE   0[3]
  000051FC  000051A4  0000520C  05694                   .LONG   21004, 20900, 20988
                                    01  056A0                   .BYTE   1
                                    00# 056A1                   .BYTE   0[3]
  000051FC  000051A4  00005250  056A4                   .LONG   21072, 20900, 20988
                                    01  056B0                   .BYTE   1
                                    00# 056B1                   .BYTE   0[3]
  000051FC  000051A4  00005294  056B4                   .LONG   21140, 20900, 20988
                                    01  056C0                   .BYTE   1
                                    00# 056C1                   .BYTE   0[3]
  000051FC  000051A4  000052D8  056C4                   .LONG   21208, 20900, 20988
                                    01  056D0                   .BYTE   1
                                    00# 056D1                   .BYTE   0[3]
  000051FC  000051A4  000053F4  056D4                   .LONG   21492, 20900, 20988
                                    01  056E0                   .BYTE   1
                                    00# 056E1                   .BYTE   0[35]
  000051FC  000051A4  00005488  05704                   .LONG   21640, 20900, 20988
                                    01  05710                   .BYTE   1
                                    00# 05711                   .BYTE   0[3]
  000051FC  000051A4  000054D4  05714                   .LONG   21716, 20900, 20988
                                    01  05720                   .BYTE   1
                                    00# 05721                   .BYTE   0[3]
  000051FC  000051A4  00005554  05724                   .LONG   21844, 20900, 20988
                                    01  05730                   .BYTE   1
                                    00# 05731                   .BYTE   0[19]
  000051FC  000051A4  000055BC  05744                   .LONG   21948, 20900, 20988
                                    01  05750                   .BYTE   1
                                    00# 05751                   .BYTE   0[19]
  000051FC  000051A4  00005520  05764                   .LONG   21792, 20900, 20988
                                    01  05770                   .BYTE   1
                                    00# 05771                   .BYTE   0[19]
  000051FC  000051A4  00005588  05784                   .LONG   21896, 20900, 20988
                                    01  05790                   .BYTE   1
                                    00# 05791                   .BYTE   0[19]
  000051FC  000051A4  00005620  057A4                   .LONG   22048, 20900, 20988
                                    01  057B0                   .BYTE   1
                                    00# 057B1                   .BYTE   0[3]
  000051FC  000051A4  000055F0  057B4                   .LONG   22000, 20900, 20988
                                    01  057C0                   .BYTE   1
                                    00# 057C1                   .BYTE   0[3]
```

```
000051FC  000051A4  000055FC  057C4          .LONG   22012, 20900, 20988
                        01  057D0          .BYTE   1
                        00# 057D1          .BYTE   0[3]
000051FC  000051A4  00005608  057D4          .LONG   22024, 20900, 20988
                        01  057E0          .BYTE   1
                        00# 057E1          .BYTE   0[3]
000051FC  000051A4  00005614  057E4          .LONG   22036, 20900, 20988
                        01  057F0          .BYTE   1
                        00# 057F1          .BYTE   0[115]
000051FC  000051A4  0000562C  05864          .LONG   22060, 20900, 20988
                        01  05870          .BYTE   1
                        00# 05871          .BYTE   0[227]
000051FC  000051D0  00000000  05954          .LONG   0, 20944, 20988
                        01  05960          .BYTE   1
                        00# 05961          .BYTE   0[3]
000051FC  000051A4  00000000  05964          .LONG   0, 20900, 20988
                        01  05970          .BYTE   1
                        00# 05971          .BYTE   0[67]
00000000  00000000  00000000  059B4          .LONG   0, 0, 0
                        01  059C0          .BYTE   1
                            059C1          .BLKB   179

                                           .PSECT  DBG$OWN,NOEXE,  PIC,2

                    00000000  00000 BLISS_BITSELECTION_FLAG1:
                                           .LONG   0
                    00000000  00004 BLISS_BITSELECTION_FLAG2:
                                           .LONG   0
                              00008 BLISS_INDIRECTION_FLAG:
                                           .BLKB   4
                              0000C CVT_ROUND_FLAG:
                                           .BLKB   4
                              00010 CVT_TBL:.BLKB   4
                              00014 CVT_TBL_SIZE:
                                           .BLKB   4
                              00018 CVTINFO_TABLE:
                                           .BLKB   4
                              0001C MAP_TBL:.BLKB   4
                              00020 MAP_TBL_SIZE:
                                           .BLKB   4
                              00024 MAX_DEPTH:
                                           .BLKB   4
                              00028 OPINFO_TABLE:
                                           .BLKB   4
                              0002C SAVE_RESULT_DESC:
                                           .BLKB   12

                                           .PSECT  DBG$GLOBAL,NOEXE,  PIC,2

                              00000 DBG$GL_OPCODE_NAME::
                                           .BLKB   4

                                     TABLEBASE=            P.AAA
                                     ADA_HIER_TABLE=       P.AAB
                                     ADA_DEPOSIT_TABLE=    P.AAC
                                     ADA_UNARY_PLUS_TABLE=
                                                           P.AAD
```

```
ADA_UNARY_MINUS_TABLE=
                        P.AAE
ADA_ABSOLUTE_TABLE= P.AAF
ADA_ADD_TABLE=      P.AAG
ADA_SUBTRACT_TABLE= P.AAH
ADA_MULTIPLY_TABLE= P.AAI
ADA_DIVIDE_TABLE=   P.AAJ
ADA_MODULUS_TABLE=  P.AAK
ADA_REMAINDER_TABLE=P.AAL
ADA_POWER_OF_TABLE= P.AAM
ADA_NOT_TABLE=      P.AAN
ADA_AND_TABLE=      P.AAO
ADA_OR_TABLE=       P.AAP
ADA_XOR_TABLE=      P.AAQ
ADA_EQUAL_TABLE=    P.AAR
ADA_NOT_EQUAL_TABLE=P.AAS
ADA_LSS_THAN_TABLE= P.AAT
ADA_GTR_THAN_TABLE= P.AAU
ADA_LSS_EQUAL_TABLE=P.AAV
ADA_GTR_EQUAL_TABLE=P.AAW
ADA_CONCATENATE_TABLE=
                        P.AAX
BASIC_HIER1_TABLE=  P.AAY
BASIC_HIER2_TABLE=  P.AAZ
BASIC_HIERD_TABLE=  P.ABA
BASIC_ADD_TABLE=    P.ABB
BASIC_SUB_TABLE=    P.ABC
BASIC_MUL_TABLE=    P.ABD
BASIC_DIV_TABLE=    P.ABE
BASIC_UNARY_PLUS_TABLE=
                        P.ABF
BASIC_UNARY_MINUS_TABLE=
                        P.ABG
BASIC_POWER_TABLE=  P.ABH
BASIC_EQL_TABLE=    P.ABI
BASIC_NEQ_TABLE=    P.ABJ
BASIC_GTR_TABLE=    P.ABK
BASIC_GEQ_TABLE=    P.ABL
BASIC_LSS_TABLE=    P.ABM
BASIC_LEQ_TABLE=    P.ABN
BASIC_BIT_NOT_TABLE=P.ABO
BASIC_BIT_AND_TABLE=P.ABP
BASIC_BIT_OR_TABLE= P.ABQ
BASIC_BIT_XOR_TABLE=P.ABR
BASIC_BIT_EQV_TABLE=P.ABS
BASIC_BIT_IMP_TABLE=P.ABT
BLISS_MAP_TABLE=    P.ABU
BLISS_HIER_TABLE=   P.ABV
BLISS_HIERD_TABLE=  P.ABW
BLISS_ADD_TABLE=    P.ABX
BLISS_SUB_TABLE=    P.ABY
BLISS_MUL_TABLE=    P.ABZ
BLISS_DIV_TABLE=    P.ACA
BLISS_MOD_TABLE=    P.ACB
BLISS_SHIFT_TABLE=  P.ACC
BLISS_EQUAL_TABLE=  P.ACD
BLISS_NOT_EQUAL_TABLE=
```

```
                                P.ACE
BLISS_LSS_THAN_TABLE=
                                P.ACF
BLISS_LSSU_THAN_TABLE=
                                P.ACG
BLISS_GTR_THAN_TABLE=
                                P.ACH
BLISS_GTRU_THAN_TABLE=
                                P.ACI
BLISS_LSS_EQUAL_TABLE=
                                P.ACJ
BLISS_LSSU_EQUAL_TABLE=
                                P.ACK
BLISS_GTR_EQUAL_TABLE=
                                P.ACL
BLISS_GTRU_EQUAL_TABLE=
                                P.ACM
BLISS_BIT_AND_TABLE=P.ACN
BLISS_BIT_OR_TABLE= P.ACO
BLISS_BIT_XOR_TABLE=P.ACP
BLISS_BIT_EQV_TABLE=P.ACQ
BLISS_UNARY_PLUS_TABLE=
                                P.ACR
BLISS_UNARY_MINUS_TABLE=
                                P.ACS
BLISS_BIT_NOT_TABLE=P.ACT
BLISS_BITSELECT_TABLE=
                                P.ACU
BLISS_INDIRECT_TABLE=
                                P.ACV
C_HIER_TABLE=                   P.ACW
C_HIER0_TABLE=                  P.ACX
C_ADD_TABLE=                    P.ACY
C_SUB_TABLE=                    P.ACZ
C_MUL_TABLE=                    P.ADA
C_DIV_TABLE=                    P.ADB
C_UNARY_MINUS_TABLE=P.ADC
C_UNARY_PLUS_TABLE= P.ADD
C_MOD_TABLE=                    P.ADE
C_EQL_TABLE=                    P.ADF
C_NEQ_TABLE=                    P.ADG
C_LSS_TABLE=                    P.ADH
C_GTR_TABLE=                    P.ADI
C_LEQ_TABLE=                    P.ADJ
C_GEQ_TABLE=                    P.ADK
C_BIT_AND_TABLE=                P.ADL
C_BIT_OR_TABLE=                 P.ADM
C_BIT_XOR_TABLE=                P.ADN
C_BIT_NOT_TABLE=                P.ADO
C_AND_TABLE=                    P.ADP
C_OR_TABLE=                     P.ADQ
C_NOT_TABLE=                    P.ADR
C_SHIFT_LEFT_TABLE=             P.ADS
C_SHIFT_RT_TABLE=               P.ADT
C_PRE_INCR_TABLE=               P.ADU
C_POST_INCR_TABLE=              P.ADV
C_PRE_DECR_TABLE=               P.ADW
```

```
C_POST_DECR_TABLE=  P.ADX
C_ADDRESS_TABLE=    P.ADY
C_SIZEOF_TABLE=     P.ADZ
C_INDIRECT_TABLE=   P.AEA
COBOL_CVT_TABLE=    P.AEB
COBOL_HIER_TABLE=   P.AEC
COBOL_HIERD_TABLE=  P.AED
COBOL_ADD_TABLE=    P.AEE
COBOL_SUB_TABLE=    P.AEF
COBOL_MUL_TABLE=    P.AEG
COBOL_DIV_TABLE=    P.AEH
COBOL_UNARY_PLUS_TABLE=
                    P.AEI
COBOL_UNARY_MINUS_TABLE=
                    P.AEJ
COBOL_EQL_TABLE=    P.AEK
COBOL_NEQ_TABLE=    P.AEL
COBOL_GTR_TABLE=    P.AEM
COBOL_GEQ_TABLE=    P.AEN
COBOL_LSS_TABLE=    P.AEO
COBOL_LEQ_TABLE=    P.AEP
COBOL_NOT_TABLE=    P.AEQ
COBOL_AND_TABLE=    P.AER
COBOL_OR_TABLE=     P.AES
FORTRAN_MAP_TABLE=  P.AET
FORTRAN_HIER1_TABLE=P.AEU
FORTRAN_HIER2_TABLE=P.AEV
FORTRAN_HIER3_TABLE=P.AEW
FORTRAN_HIERD_TABLE=P.AEX
FORTRAN_INCOMP_TABLE=
                    P.AEY
FORTRAN_ADD_TABLE=  P.AEZ
FORTRAN_SUB_TABLE=  P.AFA
FORTRAN_MUL_TABLE=  P.AFB
FORTRAN_DIV_TABLE=  P.AFC
FORTRAN_UNARY_PLUS_TABLE=
                    P.AFD
FORTRAN_UNARY_MINUS_TABLE=
                    P.AFE
FORTRAN_POWER_TABLE=P.AFF
FORTRAN_CONCAT_TABLE=
                    P.AFG
FORTRAN_EQL_TABLE=  P.AFH
FORTRAN_NEQ_TABLE=  P.AFI
FORTRAN_GTR_TABLE=  P.AFJ
FORTRAN_GEQ_TABLE=  P.AFK
FORTRAN_LSS_TABLE=  P.AFL
FORTRAN_LEQ_TABLE=  P.AFM
FORTRAN_BIT_NOT_TABLE=
                    P.AFN
FORTRAN_BIT_AND_TABLE=
                    P.AFO
FORTRAN_BIT_OR_TABLE=
                    P.AFP
FORTRAN_BIT_XOR_TABLE=
                    P.AFQ
FORTRAN_BIT_EQV_TABLE=
```

```
                                P.AFR
MACRO_MAP_TABLE=                P.AFS
MACRO_HIER_TABLE=               P.AFT
MACRO_HIERD_TABLE=              P.AFU
MACRO_ADD_TABLE=                P.AFV
MACRO_SUB_TABLE=                P.AFW
MACRO_MUL_TABLE=                P.AFX
MACRO_DIV_TABLE=                P.AFY
MACRO_SHIFT_TABLE=              P.AFZ
MACRO_MOD_TABLE=                P.AGA
MACRO_UNARY_PLUS_TABLE=
                                P.AGB
MACRO_UNARY_MINUS_TABLE=
                                P.AGC
MACRO_EQUAL_TABLE=              P.AGD
MACRO_NOT_EQUAL_TABLE=
                                P.AGE
MACRO_LSS_THAN_TABLE=
                                P.AGF
MACRO_GTR_THAN_TABLE=
                                P.AGG
MACRO_LSS_EQUAL_TABLE=
                                P.AGH
MACRO_GTR_EQUAL_TABLE=
                                P.AGI
MACRO_LSSU_THAN_TABLE=
                                P.AGJ
MACRO_GTRU_THAN_TABLE=
                                P.AGK
MACRO_LSSU_EQUAL_TABLE=
                                P.AGL
MACRO_GTRU_EQUAL_TABLE=
                                P.AGM
MACRO_BIT_AND_TABLE=            P.AGN
MACRO_BIT_OR_TABLE=             P.AGO
MACRO_BIT_XOR_TABLE=            P.AGP
MACRO_BIT_EQV_TABLE=            P.AGQ
MACRO_BIT_NOT_TABLE=            P.AGR
MACRO_BITSELECT_TABLE=
                                P.AGS
MACRO_INDIRECT_TABLE=
                                P.AGT
PASCAL_MAP_TABLE=               P.AGU
PASCAL_HIER_TABLE=              P.AGV
PASCAL_HIERT_TABLE=             P.AGW
PASCAL_HIERD_TABLE=             P.AGX
PASCAL_INCOMP_TABLE=            P.AGY
PASCAL_ADD_TABLE=               P.AGZ
PASCAL_SUB_TABLE=               P.AHA
PASCAL_MUL_TABLE=               P.AHB
PASCAL_POWER_TABLE=             P.AHC
PASCAL_DIV_TABLE=               P.AHD
PASCAL_UNARY_PLUS_TABLE=
                                P.AHE
PASCAL_UNARY_MINUS_TABLE=
                                P.AHF
PASCAL_INTDIV_TABLE=            P.AHG
```

```
PASCAL_MOD_TABLE=       P.AHH
PASCAL_REM_TABLE=       P.AHI
PASCAL_EQL_TABLE=       P.AHJ
PASCAL_NEQ_TABLE=       P.AHK
PASCAL_GEQ_TABLE=       P.AHL
PASCAL_GTR_TABLE=       P.AHM
PASCAL_LSS_TABLE=       P.AHN
PASCAL_LEQ_TABLE=       P.AHO
PASCAL_NOT_TABLE=       P.AHP
PASCAL_AND_TABLE=       P.AHQ
PASCAL_OR_TABLE=        P.AHR
PASCAL_IN_TABLE=        P.AHS
PASCAL_SUCCESSOR_TABLE=
                        P.AHT
PASCAL_PREDECESSOR_TABLE=
                        P.AHU
PLI_MAP_TABLE=          P.AHV
PLI_INCOMP_TABLE=       P.AHW
PLI_CVT_TABLE=          P.AHX
PLI_HIER1_TABLE=        P.AHY
PLI_HIER2_TABLE=        P.AHZ
PLI_HIER3_TABLE=        P.AIA
PLI_HIER4_TABLE=        P.AIB
PLI_HIER5_TABLE=        P.AIC
PLI_HIERD_TABLE=        P.AID
PLI_ADD_TABLE=          P.AIE
PLI_SUB_TABLE=          P.AIF
PLI_MUL_TABLE=          P.AIG
PLI_DIV_TABLE=          P.AIH
PLI_UNARY_PLUS_TABLE=
                        P.AII
PLI_UNARY_MINUS_TABLE=
                        P.AIJ
PLI_POWER_TABLE=        P.AIK
PLI_CONCAT_TABLE=       P.AIL
PLI_EQL_TABLE=          P.AIM
PLI_NEQ_TABLE=          P.AIN
PLI_GTR_TABLE=          P.AIO
PLI_GEQ_TABLE=          P.AIP
PLI_LSS_TABLE=          P.AIQ
PLI_LEQ_TABLE=          P.AIR
PLI_BIT_NOT_TABLE=      P.AIS
PLI_BIT_AND_TABLE=      P.AIT
PLI_BIT_OR_TABLE=       P.AIU
RPG_HIER_TABLE=         P.AIV
RPG_HIERD_TABLE=        P.AIW
RPG_ADD_TABLE=          P.AIX
RPG_SUB_TABLE=          P.AIY
RPG_MUL_TABLE=          P.AIZ
RPG_DIV_TABLE=          P.AJA
RPG_UNARY_PLUS_TABLE=
                        P.AJB
RPG_UNARY_MINUS_TABLE=
                        P.AJC
RPG_EQL_TABLE=          P.AJD
RPG_NEQ_TABLE=          P.AJE
RPG_GTR_TABLE=          P.AJF
```

```
RPG_GEQ_TABLE=      P.AJG
RPG_LSS_TABLE=      P.AJH
RPG_LEQ_TABLE=      P.AJI
RPG_NOT_TABLE=      P.AJJ
RPG_AND_TABLE=      P.AJK
RPG_OR_TABLE=       P.AJL
UNKNOWN_HIER_TABLE= P.AJM
UNKNOWN_HIERD_TABLE=P.AJN
UNKNOWN_INCOMP_TABLE=
                    P.AJO
UNKNOWN_ADD_TABLE=  P.AJP
UNKNOWN_SUB_TABLE=  P.AJQ
UNKNOWN_MUL_TABLE=  P.AJR
UNKNOWN_DIV_TABLE=  P.AJS
UNKNOWN_UNARY_PLUS_TABLE=
                    P.AJT
UNKNOWN_UNARY_MINUS_TABLE=
                    P.AJU
UNKNOWN_POWER_TABLE=P.AJV
UNKNOWN_EQL_TABLE=  P.AJW
UNKNOWN_NEQ_TABLE=  P.AJX
UNKNOWN_LSS_THAN_TABLE=
                    P.AJY
UNKNOWN_GTR_THAN_TABLE=
                    P.AJZ
UNKNOWN_LEQ_TABLE=  P.AKA
UNKNOWN_GEQ_TABLE=  P.AKB
UNKNOWN_AND_TABLE=  P.AKC
UNKNOWN_OR_TABLE=   P.AKD
UNKNOWN_XOR_TABLE=  P.AKE
UNKNOWN_EQV_TABLE=  P.AKF
UNKNOWN_NOT_TABLE=  P.AKG
UNKNOWN_CONCATENATE_TABLE=
                    P.AKH
        .EXTRN  FOR$CVT_D_TE, FOR$CVT_G_TE
        .EXTRN  FOR$CVT_H_TE, MTH$JNOT
        .EXTRN  OTS$CVT_TB_L, OTS$CVT_TI_L
        .EXTRN  OTS$CVT_T_F, OTS$CVT_T_D
        .EXTRN  OTS$CVT_T_G, OTS$CVT_T_H
        .EXTRN  OTS$CVT_TO_L, OTS$CVT_TZ_L
        .EXTRN  PLI$CHARABIT_R6
        .EXTRN  PLI$CVRT_ANY, DBG$COLLECT
        .EXTRN  DBG$COVER_DX_DX
        .EXTRN  DBG$CVT_DX_DX, DBG$CVT_TQUADWORD_TO_VALUE
        .EXTRN  DBG$CVT_TUQUADWORD_TO_VALUE
        .EXTRN  DBG$CVT_TOCTAWORD_TO_VALUE
        .EXTRN  DBG$CVT_TRFA_TO_VALUE
        .EXTRN  DBG$GET_DST_NAME
        .EXTRN  DBG$GET_TEMPMEM
        .EXTRN  DBG$MAKE_SKELETON_DESC
        .EXTRN  DBG$MAKE_VAL_DESC
        .EXTRN  DBG$MAKE_VMS_DESC
        .EXTRN  DBG$NEWLINE, DBG$PERFORM_OPERATOR
        .EXTRN  DBG$PRIM_TO_ADDR
        .EXTRN  DBG$PRIM_TO_VAL
        .EXTRN  DBG$PRINT, DBG$STA_SYMVALUE
        .EXTRN  DBG$STA_TYP_ATOMIC
```

```
                                        .EXTRN  DBG$STA_TYP_ENUM
                                        .EXTRN  DBG$STA_TYP_PICT
                                        .EXTRN  DBG$STA_TYP_SET
                                        .EXTRN  DBG$STA_TYP_SUBRNG
                                        .EXTRN  DBG$STA_TYP_TYPEDPTR
                                        .EXTRN  DBG$TYPEID_FOR_SET
                                        .EXTRN  DBG$GB_LANGUAGE
                                        .EXTRN  DBG$GL_NEG_CONST_TOKEN
                                        .EXTRN  DBG$GL_POS_CONST_TOKEN
                                        .EXTRN  DBG$GL_NEG_SIGN_TOKEN
                                        .EXTRN  DBG$GL_POS_SIGN_TOKEN
                                        .EXTRN  DBG$GL_DEVELOPER

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

                              0000 00000 AAA_DUMMY:
                                        .WORD   Save nothing                    ; 5709
                           50   D4 00002         CLRL    R0                      ; 5717
                                04 00004         RET                             ;

; Routine Size:  5 bytes,    Routine Base:  DBG$CODE + 0000
```

```
;  5607          5718  1 GLOBAL ROUTINE DBG$BLISS_BITSELECT (OPERATOR, ARG_DESC, RESULT_DESC): NOVALUE =
;  5608          5719  1 !
;  5609          5720  1 ! FUNCTION
;  5610          5721  1 !
;  5611          5722  1 !       This routine is called from DBG$PERFORM_OPERATOR to do the
;  5612          5723  1 !       BLISS bit select operation (e.g, X<p,s,e>). It receives
;  5613          5724  1 !       the operator token containing the p,s,e information and
;  5614          5725  1 !       a pointer to the descriptor for X. It builds a new descriptor with
;  5615          5726  1 !       the new bit offset, length, and sign extension information.
;  5616          5727  1 !
;  5617          5728  1 ! INPUTS
;  5618          5729  1 !
;  5619          5730  1 !       OPERATOR        - points to the operator token for the BITSELECT operator.
;  5620          5731  1 !                         The p,s,e information can be extracted from the token
;  5621          5732  1 !                         in the BIT_OFFSET, BIT_LENGTH, and SGNEXT fields.
;  5622          5733  1 !       ARG_DESC        - points to the VMS descriptor representing the argument
;  5623          5734  1 !                         of the bit-select operator.
;  5624          5735  1 !       RESULT_DESC     - points to the VMS descriptor representing the result.
;  5625          5736  1 !
;  5626          5737  1 ! OUTPUTS
;  5627          5738  1 !
;  5628          5739  1 !       The result VMS descriptor is filled in.
;  5629          5740  1 !       No value is returned.
;  5630          5741  1 !
;  5631          5742  2     BEGIN
;  5632          5743  2     MAP
;  5633          5744  2         OPERATOR        : REF TOKEN$ENTRY,
;  5634          5745  2         ARG_DESC        : REF DBG$STG_DESC,
;  5635          5746  2         RESULT_DESC     : REF DBG$STG_DESC;
;  5636          5747  2
;  5637          5748  2     LOCAL
;  5638          5749  2         ADDRESS,                    ! The address of the data
;  5639          5750  2         BIT_OFFSET,                 ! New bit offset from byte address
;  5640          5751  2         DTYPE;                      ! New dtype
;  5641          5752  2
;  5642          5753  2     ! Obtain the address.
;  5643          5754  2     !
;  5644          5755  2     ADDRESS = ..ARG_DESC[DSC$A_POINTER];
;  5645          5756  2
;  5646          5757  2     ! Obtain the bit offsets.
;  5647          5758  2     !
;  5648          5759  2     BIT_OFFSET = .OPERATOR [TOKEN$W_BIT_OFFSET];
;  5649          5760  2
;  5650          5761  2     ! Compute the new byte address.
;  5651          5762  2     !
;  5652          5763  2     ADDRESS = .ADDRESS + .BIT_OFFSET / 8;
;  5653          5764  2
;  5654          5765  2     ! Compute the bit offset. From it and the sign extension bit,
;  5655          5766  2     ! determine the new class and dtype.
;  5656          5767  2     !
;  5657          5768  2     BIT_OFFSET = .BIT_OFFSET MOD 8;
;  5658          5769  2     IF .BIT_OFFSET EQL 0
;  5659          5770  2     THEN
;  5660          5771  3         BEGIN
;  5661          5772  3         IF .OPERATOR [TOKEN$V_SGNEXT]
;  5662          5773  3         THEN
;  5663          5774  3             DTYPE = DSC$K_DTYPE_SV
```

```
;  5664          5775  3          ELSE
;  5665          5776  3              DTYPE = DSC$K_DTYPE_V;
;  5666          5777  3          SAVE_RESULT_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  5667          5778  3          END
;  5668          5779  2      ELSE
;  5669          5780  3          BEGIN
;  5670          5781  3          IF .OPERATOR [TOKEN$V_SGNEXT]
;  5671          5782  3          THEN
;  5672          5783  3              DTYPE = DSC$K_DTYPE_SVU
;  5673          5784  3          ELSE
;  5674          5785  3              DTYPE = DSC$K_DTYPE_VU;
;  5675          5786  3          SAVE_RESULT_DESC[DSC$B_CLASS] = DSC$K_CLASS_UBS;
;  5676          5787  2          END;
;  5677          5788  2
;  5678          5789  2      ! Save away the new dtype, bit offset, and length.
;  5679          5790  2      !
;  5680          5791  2      SAVE_RESULT_DESC [DSC$B_DTYPE] = .DTYPE;
;  5681          5792  2      SAVE_RESULT_DESC [DSC$W_LENGTH] = .OPERATOR [TOKEN$W_BIT_LENGTH];
;  5682          5793  2      SAVE_RESULT_DESC [DSC$A_POINTER] = .ADDRESS;
;  5683          5794  2      SAVE_RESULT_DESC [DSC$L_POS] = .BIT_OFFSET;
;  5684          5795  2
;  5685          5796  2      ! Fill in the new address.
;  5686          5797  2      !
;  5687          5798  2      .RESULT_DESC[DSC$A_POINTER] = .ADDRESS;
;  5688          5799  2
;  5689          5800  2      ! Set a flag saying we've done a BLISS bit selection.
;  5690          5801  2      !
;  5691          5802  2      BLISS_BITSELECTION_FLAG1 = TRUE;
;  5692          5803  1      END;


                                001C 00000          .ENTRY  DBG$BLISS_BITSELECT, Save R2,R3,R4                  ; 5718
                    54 00000000'  EF  9E 00002          MOVAB   SAVE_RESULT_DESC+3, R4                              ;
                    50       08   AC  D0 00009          MOVL    ARG_DESC, R0                                        ; 5755
                    53       04   B0  D0 0000D          MOVL    @4(R0), ADDRESS                                     ;
                    51       04   AC  D0 00011          MOVL    OPERATOR, R1                                        ; 5759
                    50       08   A1  3C 00015          MOVZWL  8(R1), BIT_OFFSET                                   ;
                52  50             08  C7 00019          DIVL3   #8, BIT_OFFSET, R2                                  ; 5763
                    53             52  C0 0001D          ADDL2   R2, ADDRESS                                         ;
          7E    00  50             01  7A 00020          EMUL    #1, BIT_OFFSET, #0, -(SP)                           ; 5768
          50    50  8E             08  7B 00025          EDIV    #8, (SP)+, BIT_OFFSET, BIT_OFFSET                   ;
                                   50  D5 0002A          TSTL    BIT_OFFSET                                          ; 5769
                                   11  12 0002C          BNEQ    3$                                                  ;
                05  61             0A  E1 0002E          BBC     #10, (R1), 1$                                       ; 5772
                    52             29  D0 00032          MOVL    #41, DTYPE                                          ; 5774
                                   03  11 00035          BRB     2$                                                  ;
                    52             01  D0 00037  1$:     MOVL    #1, DTYPE                                           ; 5776
                    64             01  90 0003A  2$:     MOVB    #1, SAVE_RESULT_DESC+3                              ; 5777
                                   0F  11 0003D          BRB     6$                                                  ; 5769
                05  61             0A  E1 0003F  3$:     BBC     #10, (R1), 4$                                       ; 5781
                    52             2A  D0 00043          MOVL    #42, DTYPE                                          ; 5783
                                   03  11 00046          BRB     5$                                                  ;
                    52             22  D0 00048  4$:     MOVL    #34, DTYPE                                          ; 5785
                    64             0D  90 0004B  5$:     MOVB    #13, SAVE_RESULT_DESC+3                             ; 5786
```

```
                FF    A4          52  90 0004E 6$:    MOVB    DTYPE, SAVE_RESULT_DESC+2                ; 5791
                FD    A4    0A    A1  B0 00052       MOVW    10(R1), SAVE_RESULT_DESC                 ; 5792
                01    A4          53  D0 00057       MOVL    ADDRESS, SAVE_RESULT_DESC+4              ; 5793
                05    A4          50  D0 0005B       MOVL    BIT_OFFSET, SAVE_RESULT_DESC+8           ; 5794
                      51    0C    AC  D0 0005F       MOVL    RESULT_DESC, R1                          ; 5798
                04    B1          53  D0 00063       MOVL    ADDRESS, @4(R1)                          ;
                D1    A4          01  D0 00067       MOVL    #1, BLISS_BITSELECTION_FLAG1             ; 5802
                                      04 0006B       RET                                              ; 5803

; Routine Size:  108 bytes,    Routine Base:  DBG$CODE + 0005
```

```
;  5694          5804  1 GLOBAL ROUTINE DBG$BLISS_INDIRECTION (ARG_DESC) =
;  5695          5805  1 !
;  5696          5806  1 ! FUNCTION
;  5697          5807  1 !       This routine is called from DBG$PERFORM_OPERATOR to perform a
;  5698          5808  1 !       BLISS indirection. The routine just fetches the object
;  5699          5809  1 !       given by the VMS descriptor, and returns the object as a longword
;  5700          5810  1 !       value.
;  5701          5811  1 !
;  5702          5812  1 ! INPUTS
;  5703          5813  1 !       ARG_DESC - A VMS descriptor representing the argument
;  5704          5814  1 !
;  5705          5815  1 ! OUTPUTS
;  5706          5816  1 !       The object that the VMS descriptor points to is fetched.
;  5707          5817  1 !       This value is returned. If there was no read access to the
;  5708          5818  1 !       address in the descriptor, the NOACCESSR message is
;  5709          5819  1 !       signalled.
;  5710          5820  1 !
;  5711          5821  1
;  5712          5822  2     BEGIN
;  5713          5823  2     BUILTIN
;  5714          5824  2         PROBER;
;  5715          5825  2
;  5716          5826  2     LOCAL
;  5717          5827  2         VALDESC: REF DBG$VALDESC;               ! Pointer to a value descriptor
;  5718          5828  2
;  5719          5829  2     ! Turn the VMS desc into a value descriptor
;  5720          5830  2     !
;  5721          5831  2     IF .BLISS_BITSELECTION_FLAG2
;  5722          5832  2     THEN
;  5723          5833  2         VALDESC = DBG$MAKE_VAL_DESC (SAVE_RESULT_DESC, DBG$K_VALUE_DESC)
;  5724          5834  2     ELSE
;  5725          5835  2         VALDESC = DBG$MAKE_VAL_DESC (.ARG_DESC, DBG$K_VALUE_DESC);
;  5726          5836  2
;  5727          5837  2     ! If we have already done the fetch,
;  5728          5838  2     ! return the value in the descriptor.
;  5729          5839  2     !
;  5730          5840  2     IF .BLISS_INDIRECTION_FLAG OR .BLISS_BITSELECTION_FLAG2
;  5731          5841  2     THEN
;  5732          5842  2         RETURN .VALDESC[DBG$L_VALUE_VALUE0]
;  5733          5843  2
;  5734          5844  2     ELSE
;  5735          5845  2
;  5736          5846  2         ! Check for read access.
;  5737          5847  2         !
;  5738          5848  2         IF PROBER (%REF(0), %REF(1), .VALDESC[DBG$L_VALUE_VALUE0])
;  5739          5849  2         THEN
;  5740          5850  2             RETURN ..VALDESC[DBG$L_VALUE_VALUE0]
;  5741          5851  2
;  5742          5852  2         ELSE
;  5743          5853  2             SIGNAL (DBG$_NOACCESSR, 1, .VALDESC[DBG$L_VALUE_VALUE0])
;  5744          5854  1     END;


                                0004 00000         .ENTRY  DBG$BLISS_INDIRECTION, Save R2                ; 5804
```

```
                        52 00000000'   EF  9E 00002          MOVAB   BLISS_BITSELECTION_FLAG2, R2                    ;
                        09             62  E9 00009          BLBC    BLISS_BITSELECTION_FLAG2, 1$                  ; 5831
                        7E       7A    8F  9A 0000C          MOVZBL  #122, -(SP)                                   ; 5833
                                 28    A2  9F 00010          PUSHAB  SAVE_RESULT_DESC                              ;
                                       07  11 00013          BRB     2$                                            ;
                        7E       7A    8F  9A 00015 1$:      MOVZBL  #122, -(SP)                                   ; 5835
                                 04    AC  DD 00019          PUSHL   ARG_DESC                                      ;
                00000000G  00          02  FB 0001C 2$:      CALLS   #2, DBG$MAKE_VAL_DESC                         ;
                        03       04    A2  E8 00023          BLBS    BLISS_INDIRECTION_FLAG, 3$                    ; 5840
                        05             62  E9 00027          BLBC    BLISS_BITSELECTION_FLAG2, 4$                  ;
                        50       20    A0  D0 0002A 3$:      MOVL    32(VALDESC), R0                               ; 5842
                                       04  0002E              RET                                                   ;
              20  B0                01        00  0C 0002F 4$:      PROBER  #0, #1, @32(VALDESC)                  ; 5848
                                       C?  13 00034          BEQL    5$                                            ;
                        50       20    B0  D0 00036          MOVL    @32(VALDESC), R0                              ; 5850
                                       04  0003A              RET                                                   ;
                                 20    A0  DD 0003B 5$:      PUSHL   32(VALDESC)                                   ; 5853
                                       01  DD 0003E          PUSHL   #1                                            ;
                           00028228    8F  DD 00040          PUSHL   #164392                                       ;
                00000000G  00          03  FB 00046          CALLS   #3, LIB$SIGNAL                                ;
                                       04  0004D              RET                                                   ; 5854

; Routine Size:  78 bytes,    Routine Base:  DBG$CODE + 0071
```

```
;  5746          5855  1 GLOBAL ROUTINE DBG$CONV_TEXT_VALUE(VALUE1, VALUE2, NEW_TYPE) =
;  5747          5856  1 !
;  5748          5857  1 ! FUNCTION
;  5749          5858  1 !       Perform type conversion from an unconverted string to a value.
;  5750          5859  1 !       Note: this routine accepts unsigned dtype, and treat the
;  5751          5860  1 !       unsigned value as signed value (in other words, the T --> value
;  5752          5861  1 !       is signed.).
;  5753          5862  1 !
;  5754          5863  1 ! INPUTS
;  5755          5864  1 !       VALUE1  - Pointer to a value descriptor to be type-converted.
;  5756          5865  1 !       VALUE2  - Pointer to the target value descriptor.
;  5757          5866  1 !       NEW_TYPE- The eventual type of the value
;  5758          5867  1 !
;  5759          5868  1 ! OUTPUTS
;  5760          5869  1 !       Pointer to VALUE2 is returned.
;  5761          5870  1 !
;  5762          5871  2     BEGIN
;  5763          5872  2
;  5764          5873  2     MAP
;  5765          5874  2         VALUE1: REF DBG$VALDESC,        ! Pointer to Value Descr to convert
;  5766          5875  2         VALUE2: REF DBG$VALDESC;        ! Pointer to Target Descr
;  5767          5876  2
;  5768          5877  2     LOCAL
;  5769          5878  2         LENGTH,                         ! Length of the input data.
;  5770          5879  2         STATUS,                         ! Return code from library routines
;  5771          5880  2         STG_DESC: DBG$STG_DESC,         ! String descriptor for the input
;  5772          5881  2         STR_PTR,                        ! ASCIC pointer
;  5773          5882  2         TARGET_DTYPE,                   ! Target dtype
;  5774          5883  2         TARGET_LENGTH;                  ! Target length
;  5775          5884  2
;  5776          5885  2
;  5777          5886  2     ! Fill in a string descriptor to be used as the source for the
;  5778          5887  2     ! conversion.
;  5779          5888  2     !
;  5780          5889  2     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  5781          5890  2     STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  5782          5891  2     LENGTH = .VALUE1[DBG$W_VALUE_LENGTH];
;  5783          5892  2     STG_DESC[DSC$W_LENGTH] = .LENGTH;
;  5784          5893  2     STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
;  5785          5894  2
;  5786          5895  2     ! If language is ADA then we may have to strip underscores from the
;  5787          5896  2     ! text.
;  5788          5897  2     !
;  5789          5898  2     IF .DBG$GB_LANGUAGE EQL DBG$K_ADA
;  5790          5899  2     THEN
;  5791          5900  3         BEGIN
;  5792          5901  3         LOCAL
;  5793          5902  3             IN_PTR: REF VECTOR[,BYTE],
;  5794          5903  3             OUT_PTR: REF VECTOR[,BYTE];
;  5795          5904  3         IN_PTR = .VALUE1[DBG$L_VALUE_POINTER];
;  5796          5905  3         OUT_PTR = .STG_DESC[DSC$A_POINTER];
;  5797          5906  3         INCR I FROM 1 TO .LENGTH DO
;  5798          5907  4             BEGIN
;  5799          5908  4             IF .IN_PTR[0] NEQ '_' THEN
;  5800          5909  5                 BEGIN
;  5801          5910  5                 OUT_PTR[0] = .IN_PTR[0];
;  5802          5911  5                 OUT_PTR = .OUT_PTR + 1;
```

```
;  5803          5912  4                END;
;  5804          5913  4            IN_PTR = .IN_PTR + 1;
;  5805          5914  3            END;
;  5806          5915  3        STG_DESC[DSC$W_LENGTH] = .OUT_PTR - .STG_DESC[DSC$A_POINTER];
;  5807          5916  3        END
;  5808          5917  2    ELSE
;  5809          5918  2        CH$MOVE(.LENGTH, .VALUE1[DBG$L_VALUE_POINTER], .STG_DESC[DSC$A_POINTER]);
;  5810          5919  2
;  5811          5920  2
;  5812          5921  2    ! Case on the dtype to decide which conversion routine to call.
;  5813          5922  2    !
;  5814          5923  2    CASE .NEW_TYPE FROM DBG$K_MINIMUM_DTYPE TO DBG$K_MAXIMUM_DTYPE OF
;  5815          5924  2        SET
;  5816          5925  2
;  5817          5926  2
;  5818          5927  2        ! Short integers. If the string input looked like an integer
;  5819          5928  2        ! then convert it to longword.
;  5820          5929  2        !
;  5821          5930  2        [DSC$K_DTYPE_B, DSC$K_DTYPE_BU, DSC$K_DTYPE_W, DSC$K_DTYPE_WU,
;  5822          5931  2         DSC$K_DTYPE_L]:
;  5823          5932  3            BEGIN
;  5824          5933  3            IF .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_INTEGER OR
;  5825          5934  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_HEX_INTEGER OR
;  5826          5935  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_BIN_INTEGER OR
;  5827          5936  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_OCT_INTEGER
;  5828          5937  3            THEN
;  5829          5938  3                TARGET_DTYPE = DSC$K_DTYPE_L
;  5830          5939  3            ELSE
;  5831          5940  3                TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
;  5832          5941  2            END;
;  5833          5942  2
;  5834          5943  2
;  5835          5944  2        ! If the target type is long integer and the input string looked
;  5836          5945  2        ! like an integer then convert the string directly to the long
;  5837          5946  2        ! integer. This ensures that we accept things like
;  5838          5947  2        ! DEP/QUAD X = 11111111111
;  5839          5948  2        ! If we first converted to longword and then to quad we would
;  5840          5949  2        ! overflow.
;  5841          5950  2        !
;  5842          5951  2        [DSC$K_DTYPE_LU, DSC$K_DTYPE_Q, DSC$K_DTYPE_QU,
;  5843          5952  2         DSC$K_DTYPE_O, DSC$K_DTYPE_OU]:
;  5844          5953  3            BEGIN
;  5845          5954  3            IF .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_INTEGER OR
;  5846          5955  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_HEX_INTEGER OR
;  5847          5956  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_BIN_INTEGER OR
;  5848          5957  3               .VALUE1[DBG$W_VALUE_TOKENCODE] EQL TOKEN$K_OCT_INTEGER
;  5849          5958  3            THEN
;  5850          5959  3                TARGET_DTYPE = .NEW_TYPE
;  5851          5960  3            ELSE
;  5852          5961  3                TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
;  5853          5962  2            END;
;  5854          5963  2
;  5855          5964  2        ! If the target type is float, if the input string is in hex, bin,
;  5856          5965  2        ! or oct radix, then hex, bin, oct always goes to Longword integer.
;  5857          5966  2        ! For if we DEP f = %hex 1, we want to convert text in hex to integer
;  5858          5967  2        ! value, then coverts 1 to float 1, then performs the DEP.
;  5859          5968  2        !
```

```
; 5860          5969  2          [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H,
; 5861          5970  2           DSC$K_DTYPE_P]:
; 5862          5971  3              BEGIN
; 5863          5972  3              IF .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_HEX_INTEGER AND
; 5864          5973  3                 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_BIN_INTEGER AND
; 5865          5974  3                 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_OCT_INTEGER AND
; 5866          5975  3                 .VALUE1[DBG$W_VALUE_TOKENCODE] NEQ TOKEN$K_COMPLEMENT
; 5867          5976  3              THEN
; 5868          5977  3                  TARGET_DTYPE = .NEW_TYPE
; 5869          5978  3              ELSE
; 5870          5979  3                  TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
; 5871          5980  2              END;
; 5872          5981  2
; 5873          5982  2
; 5874          5983  2          ! All the other types are either unsupported or fixed type conversion.
; 5875          5984  2          ! For example, bit-string.
; 5876          5985  2          !
; 5877          5986  2          [INRANGE, OUTRANGE]:
; 5878          5987  2              TARGET_DTYPE = .VALUE1[DBG$B_VALUE_DTYPE];
; 5879          5988  2
; 5880          5989  2          TES;
; 5881          5990  2
; 5882          5991  2      STR_PTR = UPLIT BYTE(%ASCIC '');
; 5883          5992  2      CASE .VALUE1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
; 5884          5993  2                                          TO   TOKEN$K_MAX_OPERAND OF
; 5885          5994  2          SET
; 5886          5995  2          [TOKEN$K_INTEGER]:
; 5887          5996  2              STR_PTR = UPLIT BYTE(%ASCIC 'decimal ');
; 5888          5997  2
; 5889          5998  2          [TOKEN$K_HEX_INTEGER]:
; 5890          5999  2              STR_PTR = UPLIT BYTE(%ASCIC 'hexadecimal ');
; 5891          6000  2
; 5892          6001  2          [TOKEN$K_FLOATING]:
; 5893          6002  2              STR_PTR = UPLIT BYTE(%ASCIC 'f_float ');
; 5894          6003  2
; 5895          6004  2          [TOKEN$K_EXP_E_FLOAT]:
; 5896          6005  2              STR_PTR = UPLIT BYTE(%ASCIC 'f_float ');
; 5897          6006  2
; 5898          6007  2          [TOKEN$K_EXP_D_FLOAT]:
; 5899          6008  2              STR_PTR = UPLIT BYTE(%ASCIC 'd_float ');
; 5900          6009  2
; 5901          6010  2          [TOKEN$K_EXP_G_FLOAT]:
; 5902          6011  2              STR_PTR = UPLIT BYTE(%ASCIC 'h_float ');
; 5903          6012  2
; 5904          6013  2          [TOKEN$K_BIN_INTEGER]:
; 5905          6014  2              STR_PTR = UPLIT BYTE(%ASCIC 'binary ');
; 5906          6015  2
; 5907          6016  2          [TOKEN$K_OCT_INTEGER]:
; 5908          6017  2              STR_PTR = UPLIT BYTE(%ASCIC 'octal ');
; 5909          6018  2
; 5910          6019  2          [TOKEN$K_BIT_STRING]:
; 5911          6020  2              STR_PTR = UPLIT BYTE(%ASCIC 'bit-string ');
; 5912          6021  2
; 5913          6022  2          [TOKEN$K_PACK_DECIMAL]:
; 5914          6023  2              STR_PTR = UPLIT BYTE(%ASCIC 'packed decimal ');
; 5915          6024  2
; 5916          6025  2          [TOKEN$K_EXP_G_FLOAT]:
```

```
;  5917          6026  2              STR_PTR = UPLIT BYTE(%ASCIC 'g_float ');
;  5918          6027  2
;  5919          6028  2          [INRANGE, OUTRANGE]:
;  5920          6029  2              0;
;  5921          6030  2
;  5922          6031  2          TES;
;  5923          6032  2
;  5924          6033  2
;  5925          6034  2      CASE .TARGET_DTYPE FROM DBG$K_MINIMUM_DTYPE
;  5926          6035  2                         TO   DBG$K_MAXIMUM_DTYPE OF
;  5927          6036  2          SET
;  5928          6037  2
;  5929          6038  2
;  5930          6039  2          ! Signed integers. Call a runtime routine to do the conversion
;  5931          6040  2          ! from text to integer.
;  5932          6041  2          !
;  5933          6042  2          ! Notes - known problems with this code:
;  5934          6043  2          ! 1) In here, we treat the signed and unsigned
;  5935          6044  2          ! are the same, this may need to be fixed in the future to do the
;  5936          6045  2          ! the unsigned conversion. This only affects unsigned constants
;  5937          6046  2          ! between 2*31-1 and 2*32-1 (we treat these as signed longword,
;  5938          6047  2          ! so they overflow).
;  5939          6048  2          ! 2) Another problem is that
;  5940          6049  2          ! the largest negative longword integer will overflow because
;  5941          6050  2          ! we are converting it to longword and then negating it at the end.
;  5942          6051  2          !
;  5943          6052  2          ! Since these problems only affect very large constants in
;  5944          6053  2          ! infrequent situations, they are being ignored at the present time.
;  5945          6054  2          !
;  5946          6055  2          [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L,
;  5947          6056  2           DSC$K_DTYPE_BU, DSC$K_DTYPE_WU, DSC$K_DTYPE_LU]:
;  5948          6057  3              BEGIN
;  5949          6058  3              SELECTONE .TARGET_DTYPE OF
;  5950          6059  3                  SET
;  5951          6060  3                  [DSC$K_DTYPE_B, DSC$K_DTYPE_BU]:
;  5952          6061  3                      TARGET_LENGTH = 1;
;  5953          6062  3
;  5954          6063  3                  [DSC$K_DTYPE_W, DSC$K_DTYPE_WU]:
;  5955          6064  3                      TARGET_LENGTH = 2;
;  5956          6065  3
;  5957          6066  3                  [DSC$K_DTYPE_L, DSC$K_DTYPE_LU]:
;  5958          6067  3                      TARGET_LENGTH = 4;
;  5959          6068  3
;  5960          6069  3                  TES;
;  5961          6070  3
;  5962          5071  3              CASE .VALUE1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
;  5963          6072  3                                                  TO   TOKEN$K_MAX_OPERAND OF
;  5964          6073  3                  SET
;  5965          6074  3                  [TOKEN$K_INTEGER]:
;  5966          6075  4                      BEGIN
;  5967          6076  4                      STATUS = OTS$CVT_TI_L(STG_DESC,
;  5968          6077  4                          .VALUE2[DBG$L_VALUE_POINTER], .TARGET_LENGTH);
;  5969          6078  3                      END;
;  5970          6079  3
;  5971          6080  3                  [TOKEN$K_HEX_INTEGER]:
;  5972          6081  4                      BEGIN
;  5973          6082  4                      STATUS = OTS$CVT_TZ_L(STG_DESC,
```

```
;  5974          6083  4                  .VALUE2[DBG$L_VALUE_POINTER], .TARGET_LENGTH);
;  5975          6084  4              END;
;  5976          6085
;  5977          6086  3          [TOKEN$K_BIN_INTEGER]:
;  5978          6087  4              BEGIN
;  5979          6088  4              STATUS = OTS$CVT_TB_L(STG_DESC,
;  5980          6089  4                  .VALUE2[DBG$L_VALUE_POINTER], .TARGET_LENGTH);
;  5981          6090  4              END;
;  5982          6091
;  5983          6092  3          [TOKEN$K_OCT_INTEGER]:
;  5984          6093  4              BEGIN
;  5985          6094  4              STATUS = OTS$CVT_TO_L(STG_DESC,
;  5986          6095  4                  .VALUE2[DBG$L_VALUE_POINTER], .TARGET_LENGTH);
;  5987          6096  4              END;
;  5988          6097
;  5989          6098  3          [TOKEN$K_COMPLEMENT]:
;  5990          6099  4              BEGIN
;  5991          6100  4              STATUS = 1;
;  5992          6101  4              .VALUE2[DBG$L_VALUE_POINTER] =
;  5993          6102  4                  MTH$JNOT(.STG_DESC[DSC$A_POINTER]);
;  5994          6103  4              END;
;  5995          6104
;  5996          6105  3          [INRANGE, OUTRANGE]:
;  5997          6106  3              $DBG_ERROR('DBGEVALOP\DBG$CONV_TEXT_VALUE');
;  5998          6107
;  5999          6108  3          TES;
;  6000          6109
;  6001          6110  3      IF NOT .STATUS
;  6002          6111  3      THEN
;  6003          6112  4          BEGIN
;  6004          6113  4          SELECTONE .TARGET_DTYPE OF
;  6005          6114  4              SET
;  6006          6115  4              [DSC$K_DTYPE_B]:
;  6007          6116  4                  SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6008          6117  4                         .STG_DESC[DSC$A_POINTER],
;  6009          6118  4                         UPLIT BYTE (%ASCIC 'byte integer'),
;  6010          6119  4                         .STATUS);
;  6011          6120  4
;  6012          6121  4              [DSC$K_DTYPE_BU]:
;  6013          6122  4                  SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6014          6123  4                         .STG_DESC[DSC$A_POINTER],
;  6015          6124  4                         UPLIT BYTE (%ASCIC 'byte logical'),
;  6016          6125  4                         .STATUS);
;  6017          6126  4
;  6018          6127  4              [DSC$K_DTYPE_W]:
;  6019          6128  4                  SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6020          6129  4                          .STG_DESC[DSC$A_POINTER],
;  6021          6130  4                          UPLIT BYTE (%ASCIC 'word integer'),
;  6022          6131  4                          .STATUS);
;  6023          6132  4
;  6024          6133  4              [DSC$K_DTYPE_WU]:
;  6025          6134  4                  SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6026          6135  4                          .STG_DESC[DSC$A_POINTER],
;  6027          6136  4                          UPLIT BYTE (%ASCIC 'word logical'),
;  6028          6137  4                          .STATUS);
;  6029          6138  4
;  6030          6139  4              [DSC$K_DTYPE_L]:
```

```
; 6031          6140  4                       SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6032          6141  4                               .STG_DESC[DSC$A_POINTER],
; 6033          6142  4                               UPLIT BYTE (%ASCIC 'longword integer'),
; 6034          6143  4                               .STATUS);
; 6035          6144  4
; 6036          6145  4                   [DSC$K_DTYPE_LU]:
; 6037          6146  4                       SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6038          6147  4                               .STG_DESC[DSC$A_POINTER],
; 6039          6148  4                               UPLIT BYTE (%ASCIC 'longword logical'),
; 6040          6149  4                               .STATUS);
; 6041          6150  4                       TES;
; 6042          6151  3                   END;
; 6043          6152  3
; 6044          6153  3               VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
; 6045          6154  3               VALUE2[DBG$W_VALUE_LENGTH] = .TARGET_LENGTH;
; 6046          6155  2               END;
; 6047          6156  2
; 6048          6157  2           [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
; 6049          6158  3               BEGIN
; 6050          6159  3               STATUS =
; 6051          6160  3                   DBG$CONV_TQUADWORD_VALUE(.VALUE1, .VALUE2);
; 6052          6161  3               IF NOT .STATUS
; 6053          6162  3               THEN
; 6054          6163  4                   BEGIN
; 6055          6164  4                   SELECTONE .TARGET_DTYPE OF
; 6056          6165  4                       SET
; 6057          6166  4                       [DSC$K_DTYPE_Q]:
; 6058          6167  4                           SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6059          6168  4                                   .STG_DESC[DSC$A_POINTER],
; 6060          6169  4                                   UPLIT BYTE (%ASCIC 'quadword integer'),
; 6061          6170  4                                   .STATUS);
; 6062          6171  4
; 6063          6172  4                       [DSC$K_DTYPE_QU]:
; 6064          6173  4                           SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6065          6174  4                                   .STG_DESC[DSC$A_POINTER],
; 6066          6175  4                                   UPLIT BYTE (%ASCIC 'quadword logical'),
; 6067          6176  4                                   .STATUS);
; 6068          6177  4                       TES;
; 6069          6178  3                   END;
; 6070          6179  3
; 6071          6180  3               VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
; 6072          6181  3               VALUE2[DBG$W_VALUE_LENGTH] = 8;
; 6073          6182  2               END;
; 6074          6183  2
; 6075          6184  2
; 6076          6185  2           [DSC$K_DTYPE_O, DSC$K_DTYPE_OU]:
; 6077          6186  3               BEGIN
; 6078          6187  3               STATUS =
; 6079          6188  3                   DBG$CONV_TOCTAWORD_VALUE(.VALUE1, .VALUE2);
; 6080          6189  3               IF NOT .STATUS
; 6081          6190  3               THEN
; 6082          6191  4                   BEGIN
; 6083          6192  4                   SELECTONE .TARGET_DTYPE OF
; 6084          6193  4                       SET
; 6085          6194  4                       [DSC$K_DTYPE_O]:
; 6086          6195  4                           SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6087          6196  4                                   .STG_DESC[DSC$A_POINTER],
```

```
;  6088          6197  4                          UPLIT BYTE (%ASCIC 'octaword integer'),
;  6089          6198  4                          .STATUS);
;  6090          6199  4
;  6091          6200  4                      [DSC$K_DTYPE_OU]:
;  6092          6201  4                          SIGNAL(DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6093          6202  4                                  .STG_DESC[DSC$A_POINTER],
;  6094          6203  4                                  UPLIT BYTE (%ASCIC 'octaword logical'),
;  6095          6204  4                                  .STATUS);
;  6096          6205  4                      TES;
;  6097          6206  3                  END;
;  6098          6207  3
;  6099          6208  3                  VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
;  6100          6209  3                  VALUE2[DBG$W_VALUE_LENGTH] = 16;
;  6101          6210  2                  END;
;  6102          6211  2
;  6103          6212  2              ! Floating point. Call a runtime routine to do the conversion
;  6104          6213  2              ! from text to float value.
;  6105          6214  2              !
;  6106          6215  2              [DSC$K_DTYPE_F] :
;  6107          6216  3                  BEGIN
;  6108          6217  3                  STATUS = OTS$CVT_T_F(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
;  6109          6218  3                  IF NOT  .STATUS
;  6110          6219  3                  THEN
;  6111          6220  3                      SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6112          6221  3                              .STG_DESC[DSC$A_POINTER],
;  6113          6222  3                              UPLIT BYTE (%ASCIC 'floating'),
;  6114          6223  3                              .STATUS);
;  6115          6224  3
;  6116          6225  3                  VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
;  6117          6226  3                  VALUE2[DBG$W_VALUE_LENGTH] = 4;
;  6118          6227  2                  END;
;  6119          6228  2
;  6120          6229  2              ! Double floating point. Call a runtime routine to do the conversion
;  6121          6230  2              ! from text to double float value.
;  6122          6231  2              !
;  6123          6232  2              [DSC$K_DTYPE_D] :
;  6124          6233  3                  BEGIN
;  6125          6234  3                  STATUS = OTS$CVT_T_D(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
;  6126          6235  3                  IF NOT  .STATUS
;  6127          6236  3                  THEN
;  6128          6237  3                      SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6129          6238  3                              .STG_DESC[DSC$A_POINTER],
;  6130          6239  3                              UPLIT BYTE (%ASCIC 'double floating'),
;  6131          6240  3                              .STATUS);
;  6132          6241  3
;  6133          6242  3                  VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
;  6134          6243  3                  VALUE2[DBG$W_VALUE_LENGTH] = 8;
;  6135          6244  2                  END;
;  6136          6245  2
;  6137          6246  2              ! G Floating point. Call a runtime routine to do the conversion
;  6138          6247  2              ! from text to G float value.
;  6139          6248  2              !
;  6140          6249  2              [DSC$K_DTYPE_G] :
;  6141          6250  3                  BEGIN
;  6142          6251  3                  STATUS = OTS$CVT_T_G(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
;  6143          6252  3                  IF NOT  .STATUS
;  6144          6253  3                  THEN
```

```
; 6145          6254  3                 SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6146          6255                          .STG_DESC[DSC$A_POINTER],
; 6147          6256                          UPLIT BYTE (%ASCIC 'g_floating'),
; 6148          6257                          .STATUS);
; 6149          6258
; 6150          6259                  VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
; 6151          6260                  VALUE2[DBG$W_VALUE_LENGTH] = 8;
; 6152          6261                  END;
; 6153          6262
; 6154          6263              ! H Floating point. Call a runtime routine to do the conversion
; 6155          6264              ! from text to H float value.
; 6156          6265              !
; 6157          6266              [DSC$K_DTYPE_H] :
; 6158          6267                  BEGIN
; 6159          6268                  STATUS = OTS$CVT_T_H(STG_DESC, .VALUE2[DBG$L_VALUE_POINTER]);
; 6160          6269                  IF NOT  .STATUS
; 6161          6270                  THEN
; 6162          6271                      SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
; 6163          6272                              .STG_DESC[DSC$A_POINTER],
; 6164          6273                              UPLIT BYTE (%ASCIC 'h_floating'),
; 6165          6274                              .STATUS);
; 6166          6275
; 6167          6276                  VALUE2[DBG$B_VALUE_DTYPE] = .TARGET_DTYPE;
; 6168          6277                  VALUE2[DBG$W_VALUE_LENGTH] = 16;
; 6169          6278                  END;
; 6170          6279
; 6171          6280              ! Pack decimal.  Call a user routine to do the conversion
; 6172          6281              ! from text to pack decimal.
; 6173          6282              !
; 6174          6283              [DSC$K_DTYPE_P]:
; 6175          6284                  VALUE2 = CONV_TEXT_PACK_VALUE(.VALUE1);
; 6176          6285
; 6177          6286              ! Text string. There is no conversion to be done here since the value
; 6178          6287              ! is already in the right format.
; 6179          6288              !
; 6180          6289              [DSC$K_DTYPE_T] :
; 6181          6290                  0;
; 6182          6291
; 6183          6292              ! Bit-string.  Call a Debug routine to do the conversion from text to
; 6184          6293              ! bit-string format.  The Debug routine actually calls a PL/I
; 6185          6294              ! run-time routine to do the conversion; it is written in macro to
; 6186          6295              ! simplify the interface (PL/I run-time routines notoriously do not
; 6187          6296              ! adhere to the VAX calling standard).  The bit-string format is thus
; 6188          6297              ! PL/I specific.
; 6189          6298              !
; 6190          6299              [DSC$K_DTYPE_V]:
; 6191          6300                  BEGIN
; 6192          6301                  LOCAL
; 6193          6302                      STG_PTR:  REF VECTOR[, BYTE];
; 6194          6303
; 6195          6304
; 6196          6305                  ! At present, allow only binary constants.  Catch errors here,
; 6197          6306                  ! because PL/I run-time routines do not return a status.
; 6198          6307                  !
; 6199          6308                  STG_PTR = .STG_DESC[DSC$A_POINTER];
; 6200          6309                  INCR I FROM 0 TO .LENGTH-1 DO
; 6201          6310  4                   BEGIN
```

```
;  6202          6311  4                IF .STG_PTR[.I] NEQ %C'0' AND .STG_PTR[.I] NEQ %C'1'
;  6203          6312  4                THEN
;  6204          6313  4                    SIGNAL (DBG$_UNACVT, 4, .STR_PTR, .LENGTH,
;  6205          6314  4                            .STG_DESC[DSC$A_POINTER],
;  6206          6315  4                            UPLIT BYTE (%ASCIC 'bit-string'), .STATUS);
;  6207          6316  3                END;
;  6208          6317  3
;  6209          6318  3            PLI$CHARABIT_R6(.STG_DESC[DSC$A_POINTER], .LENGTH,
;  6210          6319  3                .VALUE2[DBG$L_VALUE_POINTER], .VALUE2[DBG$W_VALUE_LENGTH]);
;  6211          6320  2            END;
;  6212          6321  2
;  6213          6322  2
;  6214          6323  2        ! We do not expect any other dtype, so signal an error if we see one.
;  6215          6324  2        !
;  6216          6325  2        [INRANGE, OUTRANGE] :
;  6217          6326  2            $DBG_ERROR ('DBGEVALOP\DBG$CONV_TEXT_VALUE');
;  6218          6327  2
;  6219          6328  2        TES;
;  6220          6329  2
;  6221          6330  2
;  6222          6331  2      ! The conversion has been completed.  Turn off the "unconverted" bit in
;  6223          6332  2      ! the value descriptor.  Return a pointer to the Value Descriptor which
;  6224          6333  2      ! contains the converted value.
;  6225          6334  2      !
;  6226          6335  2      VALUE2[DBG$V_DHDR_UNCVT] = 0;
;  6227          6336  2      VALUE2[DBG$W_VALUE_TOKENCODE] = 0;
;  6228          6337  2
;  6229          6338  2
;  6230          6339  2      ! Take care of the sign.
;  6231          6340  2      !
;  6232          6341  2      IF .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST OR
;  6233          6342  2         .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_POSCONST
;  6234          6343  2      THEN
;  6235          6344  3          BEGIN
;  6236          6345  3          IF .VALUE1[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
;  6237          6346  3          THEN
;  6238          6347  4              BEGIN
;  6239          6348  4              VALUE2[DBG$W_VALUE_SIGN_CODE] = 0;
;  6240          6349  4              VALUE2 = DBG$EVAL_UNARY_OPERATOR(DBG$GL_NEG_CONST_TOKEN,
;  6241          6350  4                                  .VALUE2, 0);
;  6242          6351  4              END
;  6243          6352  4
;  6244          6353  3          ELSE
;  6245          6354  4              BEGIN
;  6246          6355  4              VALUE2[DBG$W_VALUE_SIGN_CODE] = 0;
;  6247          6356  4              VALUE2 = DBG$EVAL_UNARY_OPERATOR(DBG$GL_POS_CONST_TOKEN,
;  6248          6357  4                                  .VALUE2, 0);
;  6249          6358  3              END;
;  6250          6359  3
;  6251          6360  2          END;
;  6252          6361  2
;  6253          6362  2      RETURN .VALUE2;
;  6254          6363  2
;  6255          6364  1      END;
; INFO#250                  L1:6315
; Referenced LOCAL symbol STATUS is probably not initialized
```

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

                                                       00  05A74 P.AKI:  .ASCII  <0>                                          ;
                         20  6C  61  6D  69  63  65  64  08  05A75 P.AKJ:  .ASCII  <8>\decimal \                               ;
          20  6C  61  6D  69  63  65  64  61  78  65  68  0C  05A7E P.AKK:  .ASCII  <12>\hexadecimal \                          ;
                         20  74  61  6F  6C  66  5F  66  08  05A8B P.AKL:  .ASCII  <8>\f_float \                               ;
                         20  74  61  6F  6C  66  5F  66  08  05A94 P.AKM:  .ASCII  <8>\f_float \                               ;
                         20  74  61  6F  6C  66  5F  64  08  05A9D P.AKN:  .ASCII  <8>\d_float \                               ;
                         20  74  61  6F  6C  66  5F  68  08  05AA6 P.AKO:  .ASCII  <8>\h_float \                               ;
                             20  79  72  61  6E  69  62  07  05AAF P.AKP:  .ASCII  <7>\binary \                                ;
                                 20  6C  61  74  63  6F  06  05AB7 P.AKQ:  .ASCII  <6>\octal \                                 ;
                 20  67  6E  69  72  74  73  2D  74  69  62  0B  05ABE P.AKR:  .ASCII  <11>\bit-string \                        ;
  6C  61  6D  69  63  65  64  20  64  65  6B  63  61  70  0F  05ACA P.AKS:  .ASCII  <15>\packed decimal \                     ;
                                                       20  05AD9                                                              ;
                         20  74  61  6F  6C  66  5F  67  08  05ADA P.AKT:  .ASCII  <8>\g float \                               ;
  24  47  42  44  5C  50  4F  4C  41  56  45  47  42  44  1D  05AE3 P.AKU:  .ASCII  <29>\DBGEVALOP\<92>\DBG$CONV_TEXT_VALU\     ;
      55  4C  41  56  5F  54  58  45  54  5F  56  4E  4F  43  05AF2                                                         ;
                                                       45  05B00         .ASCII  \E\                                      ;
          72  65  67  65  74  6E  69  20  65  74  79  62  0C  05B01 P.AKV:  .ASCII  <12>\byte integer\                         ;
          6C  61  63  69  67  6F  6C  20  65  74  79  62  0C  05B0E P.AKW:  .ASCII  <12>\byte logical\                         ;
          72  65  67  65  74  6E  69  20  64  72  6F  77  0C  05B1B P.AKX:  .ASCII  <12>\word integer\                         ;
          6C  61  63  69  67  6F  6C  20  64  72  6F  77  0C  05B28 P.AKY:  .ASCII  <12>\word logical\                         ;
  67  65  74  6E  69  20  64  72  6F  77  67  6E  6F  6C  10  05B35 P.AKZ:  .ASCII  <16>\longword integer\                     ;
                                                   72  65  05B44                                                              ;
  63  69  67  6F  6C  20  64  72  6F  77  67  6E  6F  6C  10  05B46 P.ALA:  .ASCII  <16>\longword logical\                     ;
                                                   6C  61  05B55                                                              ;
  67  65  74  6E  69  20  64  72  6F  77  64  61  75  71  10  05B57 P.ALB:  .ASCII  <16>\quadword integer\                     ;
                                                   72  65  05B66                                                              ;
  63  69  67  6F  6C  20  64  72  6F  77  64  61  75  71  10  05B68 P.ALC:  .ASCII  <16>\quadword logical\                     ;
                                                   6C  61  05B77                                                              ;
  67  65  74  6E  69  20  64  72  6F  77  61  74  63  6F  10  05B79 P.ALD:  .ASCII  <16>\octaword integer\                     ;
                                                   72  65  05B88                                                              ;
  63  69  67  6F  6C  20  64  72  6F  77  61  74  63  6F  10  05B8A P.ALE:  .ASCII  <16>\octaword logical\                     ;
                                                   6C  61  05B99                                                              ;
                         67  6E  69  74  61  6F  6C  66  08  05B9B P.ALF:  .ASCII  <8>\floating\                               ;
  6E  69  74  61  6F  6C  66  20  65  6C  62  75  6F  64  0F  05BA4 P.ALG:  .ASCII  <15>\double floating\                      ;
                                                       67  05BB3                                                              ;
                 67  6E  69  74  61  6F  6C  66  5F  67  0A  05BB4 P.ALH:  .ASCII  <10>\g_floating\                            ;
                 67  6E  69  74  61  6F  6C  66  5F  68  0A  05BBF P.ALI:  .ASCII  <10>\h_floating\                            ;
                 67  6E  69  72  74  73  2D  74  69  62  0A  05BCA P.ALJ:  .ASCII  <10>\bit-string\                            ;
  24  47  42  44  5C  50  4F  4C  41  56  45  47  42  44  1D  05BD5 P.ALK:  .ASCII  <29>\DBGEVALOP\<92>\DBG$CONV_TEXT_VALU\     ;
      55  4C  41  56  5F  54  58  45  54  5F  56  4E  4F  43  05BE4                                                         ;
                                                       45  05BF2         .ASCII  \E\                                      ;


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

                                                  07FC 00000             .ENTRY  DBG$CONV_TEXT_VALUE, Save R2,R3,R4,R5,R6,-   ; 5855
                                                                                 R7,R8,R9,R10                                 ;
                              5A 00000000G   00   9E 00002             MOVAB   LIB$SIGNAL, R10                              ;
                              59 00000000'   EF   9E 00009             MOVAB   P.AKI, R9                                    ;
                              5E             0C   C2 00010             SUBL2   #12, SP                                      ;
                        02    AE      010E   8F   B0 00013             MOVW    #270, STG_DESC+2                             ; 5890
                              56        04   AC   D0 00019             MOVL    VALUE1, R6                                   ; 5891
```

```
                                58        14    A6  9E 0001D           MOVAB   20(R6), R8                                      ;
                                57              68  3C 00021           MOVZWL  (R8), LENGTH                                    ;
                                6E              57  B0 00024           MOVW    LENGTH, STG_DESC                                ; 5892
                                51        03    A7  9E 00027           MOVAB   3(R7), R1                                       ; 5893
                             7E                 51  C7 0002B           DIVL3   #4, R1, -(SP)                                   ;
                   00000000G                    01  FB 0002F           CALLS   #1, DBG$GET_TEMPMEM                             ;
                          04                    50  D0 00036           MOVL    R0, STG_DESC+4                                  ;
                             09 00000000G       00  91 0003A           CMPB    DBG$GB_LANGUAGE, #9                             ; 5898
                                               22  12 00041           BNEQ    4$                                              ;
                                51        18    A6  D0 00043           MOVL    24(R6), IN_PTR                                  ; 5904
                                50        04    AE  D0 00047           MOVL    STG_DESC+4, OUT_PTR                             ; 5905
                                               52  D4 0004B           CLRL    I                                               ; 5906
                                               0B  11 0004D           BRB     3$                                              ;
                          5F    8F             61  91 0004F 1$:       CMPB    (IN_PTR), #95                                   ; 5908
                                               03  13 00053           BEQL    2$                                              ;
                                80             61  90 00055           MOVB    (IN_PTR), (OUT_PTR)+                            ; 5910
                                               51  D6 00058 2$:       INCL    IN_PTR                                          ; 5913
                             F1                52  F3 0005A 3$:       AOBLEQ  LENGTH, I, 1$                                   ; 5906
                             6E                50        04    AE  A3 0005E           SUBW3   STG_DESC+4, OUT_PTR, STG_DESC           ; 5915
                                               06  11 00063           BRB     5$                                              ; 5898
                      04    BE        18      B6                57  28 00065 4$:       MOVC3   LENGTH, @24(R6), @STG_DESC+4            ; 5918
                            2B                00        0C    AC  CF 0006B 5$:       CASEL   NEW_TYPE, #0, #43                       ; 5923
                  005A              005A              00AF              00AF        00070 6$:       .WORD   12$-6$,-
                  005A              005A              0077              0077        00078           12$-6$,-
                  0091              0091              0077              005A        00080           7$-6$,-
                  00AF              00AF              00AF              00AF        00088           7$-6$,-
                  00AF              00AF              00AF              00AF        00090           9$-6$,-
                  00AF              00AF              0091              00AF        00098           9$-6$,-
                  0091              0077              0077              00AF        000A0           7$-6$,-
                  00AF              00AF              00AF              0091        000A8           7$-6$,-
                  00AF              00AF              00AF              00AF        000B0           7$-6$,-
                  00AF              00AF              00AF              00AF        000B8           9$-6$,-
                  00AF              00AF              00AF              00AF        000C0           10$-6$,-
                                                                                                  10$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  10$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  9$-6$,-
                                                                                                  9$-6$,-
                                                                                                  10$-6$,-
                                                                                                  10$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
                                                                                                  12$-6$,-
```

```
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$,-                          ;
                                                                 12$-6$                            ;
                      55  11  000C8          BRB     12$                                           ; 5987
        04     10     A6  B1  000CA  7$:     CMPW    16(R6), #4                                    ; 5933
                      12  13  000CE          BEQL    8$                                            ;
        05     10     A6  B1  000D0          CMPW    16(R6), #5                                    ; 5934
                      0C  13  000D4          BEQL    8$                                            ;
        0A     10     A6  B1  000D6          CMPW    16(R6), #10                                   ; 5935
                      06  13  000DA          BEQL    8$                                            ;
        0B     10     A6  B1  000DC          CMPW    16(R6), #11                                   ; 5936
                      3D  12  000E0          BNEQ    12$                                           ;
        55            08  D0  000E2  8$:     MOVL    #8, TARGET_DTYPE                              ; 5938
                      3C  11  000E5          BRB     13$                                           ;
        04     10     A6  B1  000E7  9$:     CMPW    16(R6), #4                                    ; 5954
                      2C  13  000EB          BEQL    11$                                           ;
        05     10     A6  B1  000ED          CMPW    16(R6), #5                                    ; 5955
                      26  13  000F1          BEQL    11$                                           ;
        0A     10     A6  B1  000F3          CMPW    16(R6), #10                                   ; 5956
                      20  13  000F7          BEQL    11$                                           ;
        0B     10     A6  B1  000F9          CMPW    16(R6), #11                                   ; 5957
                      1A  13  000FD          BEQL    11$                                           ;
                      1E  11  000FF          BRB     12$                                           ; 5961
        05     10     A6  B1  00101  10$:    CMPW    16(R6), #5                                    ; 5972
                      18  13  00105          BEQL    12$                                           ;
        0A     10     A6  B1  00107          CMPW    16(R6), #10                                   ; 5973
                      12  13  0010B          BEQL    12$                                           ;
        0B     10     A6  B1  0010D          CMPW    16(R6), #11                                   ; 5974
                      0C  13  00111          BEQL    12$                                           ;
        0C     10     A6  B1  00113          CMPW    16(R6), #12                                   ; 5975
                      06  13  00117          BEQL    12$                                           ;
        55     0C     AC  D0  00119  11$:    MOVL    NEW_TYPE, TARGET_DTYPE                        ; 5977
                      04  11  0011D          BRB     13$                                           ;
        55     02     A8  9A  0011F  12$:    MOVZBL  2(R8), TARGET_DTYPE                           ; 5979
        53            69  9E  00123  13$:    MOVAB   P.AKI, STR_PTR                                ; 5991
    0F  01     10     A6  AF  00126          CASEW   16(R6), #1, #15                               ; 5992
0022  0062  0062      0062    0012B  14$:    .WORD   26$-14$,-                                     ;
003A  0034  002E      0028    00133                  26$-14$,-                                     ;
0062  004C  0046      0040    0013B                  26$-14$,-                                     ;
0062  005E  0058      0052    00143                  15$-14$,-                                     ;
                                                     16$-14$,-                                     ;
                                                     17$-14$,-                                     ;
                                                     18$-14$,-                                     ;
                                                     19$-14$,-                                     ;
                                                     20$-14$,-                                     ;
                                                     21$-14$,-                                     ;
                                                     22$-14$,-                                     ;
                                                     26$-14$,-                                     ;
                                                     23$-14$,-                                     ;
                                                     24$-14$,-                                     ;
                                                     25$-14$,-                                     ;
```

```
                                                                  26$-14$                        ;
                                  40  11 0014B          BRB       26$                            ;
                          53  01  A9  9E 0014D 15$:     MOVAB     P.AKJ, STR_PTR                 ; 5996
                                  3A  11 00151          BRB       26$                            ;
                          53  0A  A9  9E 00153 16$:     MOVAB     P.AKK, STR_PTR                 ; 5999
                                  34  11 00157          BRB       26$                            ;
                          53  17  A9  9E 00159 17$:     MOVAB     P.AKL, STR_PTR                 ; 6002
                                  2E  11 0015D          BRB       26$                            ;
                          53  20  A9  9E 0015F 18$:     MOVAB     P.AKM, STR_PTR                 ; 6005
                                  28  11 00163          BRB       26$                            ;
                          53  29  A9  9E 00165 19$:     MOVAB     P.AKN, STR_PTR                 ; 6008
                                  22  11 00169          BRB       26$                            ;
                          53  32  A9  9E 0016B 20$:     MOVAB     P.AKO, STR_PTR                 ; 6011
                                  1C  11 0016F          BRB       26$                            ;
                          53  3B  A9  9E 00171 21$:     MOVAB     P.AKP, STR_PTR                 ; 6014
                                  16  11 00175          BRB       26$                            ;
                          53  43  A9  9E 00177 22$:     MOVAB     P.AKQ, STR_PTR                 ; 6017
                                  10  11 0017B          BRB       26$                            ;
                          53  4A  A9  9E 0017D 23$:     MOVAB     P.AKR, STR_PTR                 ; 6020
                                  0A  11 00181          BRB       26$                            ;
                          53  56  A9  9E 00183 24$:     MOVAB     P.AKS, STR_PTR                 ; 6023
                                  04  11 00187          BRB       26$                            ;
                          53  66  A9  9E 00189 25$:     MOVAB     P.AKT, STR_PTR                 ; 6026
                  2B      00      55  CF 0018D 26$:     CASEL     TARGET_DTYPE, #0, #43          ; 6034
      006A      006A      02DF    0058   00191 27$:     .WORD     28$-27$,-                      ;
      006A      006A      01A6    006A   00199                    74$-27$,-                      ;
      0241      0208      01A6    006A   001A1                    29$-27$,-                      ;
      0058      032A      0058    0058   001A9                    29$-27$,-                      ;
      0058      0058      0058    0058   001B1                    29$-27$,-                      ;
      0058      0058      02D2    0058   001B9                    52$-27$,-                      ;
      0260      01D7      01D7    0058   001C1                    29$-27$,-                      ;
      0058      0058      0058    0299   001C9                    29$-27$,-                      ;
      0058      0058      0058    0058   001D1                    29$-27$,-                      ;
      0058      0058      0058    0058   001D9                    52$-27$,-                      ;
      0058      0058      0058    0058   001E1                    62$-27$,-                      ;
                                                                  64$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  77$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  72$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  57$-27$,-                      ;
                                                                  57$-27$,-                      ;
                                                                  65$-27$,-                      ;
                                                                  69$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
                                                                  28$-27$,-                      ;
```

```
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$,-
                                                28$-27$
        0161    C9  9F 001E9 28$:    PUSHAB  P.ALK                           ;  6326
                01  DD 001ED        PUSHL   #1                              ;
    00028362    8F  DD 001EF        PUSHL   #164706                         ;
6A              03  FB 001F5        CALLS   #3, LIB$SIGNAL                  ;
              02C0  31 001F8        BRW     77$                             ;
02              55  D1 001FB 29$:    CMPL    TARGET_DTYPE, #2                ;  6060
                05  13 001FE        BEQL    30$                             ;
06              55  D1 00200        CMPL    TARGET_DTYPE, #6                ;
                05  12 00203        BNEQ    31$                             ;
58              01  D0 00205 30$:    MOVL    #1, TARGET_LENGTH               ;  6061
                1C  11 00208        BRB     35$                             ;
03              55  D1 0020A 31$:    CMPL    TARGET_DTYPE, #3                ;  6063
                05  13 0020D        BEQL    32$                             ;
07              55  D1 0020F        CMPL    TARGET_DTYPE, #7                ;
                05  12 00212        BNEQ    33$                             ;
58              02  D0 00214 32$:    MOVL    #2, TARGET_LENGTH               ;  6064
                0D  11 00217        BRB     35$                             ;
04              55  D1 00219 33$:    CMPL    TARGET_DTYPE, #4                ;  6066
                05  13 0021C        BEQL    34$                             ;
08              55  D1 0021E        CMPL    TARGET_DTYPE, #8                ;
                03  12 00221        BNEQ    35$                             ;
58              04  D0 00223 34$:    MOVL    #4, TARGET_LENGTH               ;  6067
      0F    01  10  A6  AF 00226 35$:    CASEW   16(R6), #1, #15             ;  6071
0030  0020  0020      0020    0022B 36$:    .WORD   37$-36$,-                   ;
0020  0020  0020      0045    00233         37$-36$,-                   ;
0087  006F  005A      0020    0023B         37$-36$,-                   ;
0020  0020  0020      0020    00243         38$-36$,-                   ;
                                            39$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$,-                   ;
                                            40$-36$,-                   ;
                                            41$-36$,-                   ;
                                            43$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$,-                   ;
                                            37$-36$                     ;
            6F  A9  9F 0024B 37$:    PUSHAB  P.AKU                           ;  6106
                01  DD 0024E        PUSHL   #1                              ;
    00028362    8F  DD 00250        PUSHL   #164706                         ;
6A              03  FB 00256        CALLS   #3, LIB$SIGNAL                  ;
                6C  11 00259        BRB     44$                             ;
                58  DD 0025B 38$:    PUSHL   TARGET_LENGTH                   ;  6077
50          08  AC  D0 0025D        MOVL    VALUE2, R0                      ;
```

```
                    18 A0 DD 00261          PUSHL   24(R0)
                    08 AE 9F 00264          PUSHAB  STG_DESC                        ; 6076
          00000000G   00 03 FB 00267        CALLS   #3, OTS$CVT_TI_L
                       3D 11 0026E          BRB     42$
                       58 DD 00270 39$:     PUSHL   TARGET_LENGTH                   ; 6083
                    50 08 AC D0 00272       MOVL    VALUE2, R0
                    18 A0 DD 00276          PUSHL   24(R0)
                    08 AE 9F 00279          PUSHAB  STG_DESC                        ; 6082
          00000000G   00 03 FB 0027C        CALLS   #3, OTS$CVT_TZ_L
                       28 11 00283          BRB     42$
                       58 DD 00285 40$:     PUSHL   TARGET_LENGTH                   ; 6089
                    50 08 AC D0 00287       MOVL    VALUE2, R0
                    18 A0 DD 0028B          PUSHL   24(R0)
                    08 AE 9F 0028E          PUSHAB  STG_DESC                        ; 6088
          00000000G   00 03 FB 00291        CALLS   #3, OTS$CVT_TB_L
                       13 11 00298          BRB     42$
                       58 DD 0029A 41$:     PUSHL   TARGET_LENGTH                   ; 6095
                    50 08 AC D0 0029C       MOVL    VALUE2, R0
                    18 A0 DD 002A0          PUSHL   24(R0)
                    08 AE 9F 002A3          PUSHAB  STG_DESC                        ; 6094
          00000000G   00 03 FB 002A6        CALLS   #3, OTS$CVT_TO_L
                    52 50 D0 002AD 42$:     MOVL    R0, STATUS
                       15 11 002B0          BRB     44$                             ; 6071
                    52 01 D0 002B2 43$:     MOVL    #1, STATUS                      ; 6100
                    54 08 AC D0 002B5       MOVL    VALUE2, R4                      ; 6101
                    04 AE DD 002B9          PUSHL   STG_DESC+4                      ; 6102
          00000000G   00 01 FB 002BC        CALLS   #1, MTH$JNOT
                 18 B4 50 D0 002C3          MOVL    R0, @24(R4)
                    5E 52 E8 002C7 44$:     BLBS    STATUS, 51$                     ; 6110
                    06 55 D1 002CA          CMPL    TARGET_DTYPE, #6                ; 6115
                       08 12 002CD          BNEQ    45$
                       52 DD 002CF          PUSHL   STATUS                          ; 6119
                 008D  C9 9F 002D1          PUSHAB  P.AKV                           ; 6118
                       3F 11 002D5          BRB     50$                             ; 6117
                 02    55 D1 002D7 45$:     CMPL    TARGET_DTYPE, #2                ; 6121
                       08 12 002DA          BNEQ    46$
                       52 DD 002DC          PUSHL   STATUS                          ; 6125
                 009A  C9 9F 002DE          PUSHAB  P.AKW                           ; 6124
                       32 11 002E2          BRB     50$                             ; 6123
                 07    55 D1 002E4 46$:     CMPL    TARGET_DTYPE, #7                ; 6127
                       08 12 002E7          BNEQ    47$
                       52 DD 002E9          PUSHL   STATUS                          ; 6131
                 00A7  C9 9F 002EB          PUSHAB  P.AKX                           ; 6130
                       25 11 002EF          BRB     50$                             ; 6129
                 03    55 D1 002F1 47$:     CMPL    TARGET_DTYPE, #3                ; 6133
                       08 12 002F4          BNEQ    48$
                       52 DD 002F6          PUSHL   STATUS                          ; 6137
                 00B4  C9 9F 002F8          PUSHAB  P.AKY                           ; 6136
                       18 11 002FC          BRB     50$                             ; 6135
                 08    55 D1 002FE 48$:     CMPL    TARGET_DTYPE, #8                ; 6139
                       08 12 00301          BNEQ    49$
                       52 DD 00303          PUSHL   STATUS                          ; 6143
                 00C1  C9 9F 00305          PUSHAB  P.AKZ                           ; 6142
                       0B 11 00309          BRB     50$                             ; 6141
                 04    55 D1 0030B 49$:     CMPL    TARGET_DTYPE, #4                ; 6145
                       18 12 0030E          BNEQ    51$
                       52 DD 00310          PUSHL   STATUS                          ; 6149
```

```
                00D2   C9  9F 00312          PUSHAB  P.ALA                    ; 6148
                  0C   AE  DD 00316 50$:     PUSHL   STG_DESC+4              ; 6147
                0088   8F  BB 00319          PUSHR   #^M<R3,R7>              ; 6146
                  04   DD  0031D             PUSHL   #4                      ;
          00028E78     8F  DD 0031F          PUSHL   #167544                 ;
          6A           07  FB 00325          CALLS   #7, LIB$SIGNAL          ;
          50     08    AC  D0 00328 51$:     MOVL    VALUE2, R0              ; 6153
    16    A0           55  90 0032C          MOVB    TARGET_DTYPE, 22(R0)    ;
    14    A0           58  B0 00330          MOVW    TARGET_LENGTH, 20(R0)   ; 6154
                0184   31  00334             BRW     77$                     ; 6034
          54     08    AC  D0 00337 52$:     MOVL    VALUE2, R4              ; 6160
                       54  DD 0033B          PUSHL   R4                      ;
                       56  DD 0033D          PUSHL   R6                      ;
  0000V   CF           02  FB 0033F          CALLS   #2, DBG$CONV_TQUADWORD_VALUE ;
          52           50  D0 00344          MOVL    R0, STATUS              ;
          03           52  E9 00347          BLBC    STATUS, 54$             ; 6161
                00D3   31  0034A 53$:        BRW     67$                     ;
          09           55  D1 0034D 54$:     CMPL    TARGET_DTYPE, #9        ; 6166
                       08  12 00350          BNEQ    55$                     ;
                       52  DD 00352          PUSHL   STATUS                  ; 6170
                00E3   C9  9F 00354          PUSHAB  P.ALB                   ; 6169
                       0B  11 00358          BRB     56$                     ; 6168
          05           55  D1 0035A 55$:     CMPL    TARGET_DTYPE, #5        ; 6172
                       EB  12 0035D          BNEQ    53$                     ;
                       52  DD 0035F          PUSHL   STATUS                  ; 6176
                00F4   C9  9F 00361          PUSHAB  P.ALC                   ; 6175
                00A6   31  00365 56$:        BRW     66$                     ; 6174
          54     08    AC  D0 00368 57$:     MOVL    VALUE2, R4              ; 6188
                       54  DD 0036C          PUSHL   R4                      ;
                       56  DD 0036E          PUSHL   R6                      ;
  0000V   CF           02  FB 00370          CALLS   #2, DBG$CONV_TOCTAWORD_VALUE ;
          52           50  D0 00375          MOVL    R0, STATUS              ;
          03           52  E9 00378          BLBC    STATUS, 59$             ; 6189
                00DB   31  0037B 58$:        BRW     71$                     ;
          1A           55  D1 0037E 59$:     CMPL    TARGET_DTYPE, #26       ; 6194
                       08  12 00381          BNEQ    60$                     ;
                       52  DD 00383          PUSHL   STATUS                  ; 6198
                0105   C9  9F 00385          PUSHAB  P.ALD                   ; 6197
                       0B  11 00389          BRB     61$                     ; 6196
          19           55  D1 0038B 60$:     CMPL    TARGET_DTYPE, #25       ; 6200
                       EB  12 0038F          BNEQ    58$                     ;
                       52  DD 00390          PUSHL   STATUS                  ; 6204
                0116   C9  9F 00392          PUSHAB  P.ALE                   ; 6203
                00AE   31  00396 61$:        BRW     70$                     ; 6202
          54     08    AC  D0 00399 62$:     MOVL    VALUE2, R4              ; 6217
                       18  A4  DD 0039D      PUSHL   24(R4)                  ;
                       04  AE  9F 003A0      PUSHAB  STG_DESC                ;
00000000G 00           02  FB 003A3          CALLS   #2, OTS$CVT_T_F         ;
          52           50  D0 003AA          MOVL    R0, STATUS              ;
          18           52  E8 003AD          BLBS    STATUS, 63$             ; 6218
                       52  DD 003B0          PUSHL   STATUS                  ; 6223
                0127   C9  9F 003B2          PUSHAB  P.ALF                   ; 6222
                  0C   AE  DD 003B6          PUSHL   STG_DESC+4              ; 6221
                0088   8F  BB 003B9          PUSHR   #^M<R3,R7>              ; 6220
                  04   DD  003BD             PUSHL   #4                      ;
          00028E78     8F  DD 003BF          PUSHL   #167544                 ;
          6A           07  FB 003C5          CALLS   #7, LIB$SIGNAL          ;
```

```
                    16   A4              55  90 003C8 63$:    MOVB    TARGET_DTYPE, 22(R4)                ; 6225
                    14   A4              04  B0 003CC          MOVW    #4, 20(R4)                          ; 6226
                                         56  11 003D0          BRB     68$                                 ; 6034
                         54        08    AC  D0 003D2 64$:    MOVL    VALUE2, R4                          ; 6234
                                   18    A4  DD 003D6          PUSHL   24(R4)                              ;
                                   04    AE  9F 003D9          PUSHAB  STG_DESC                            ;
              00000000G    00             02  FB 003DC          CALLS   #2, OTS$CVT_T_D                     ;
                           52             50  D0 003E3          MOVL    R0, STATUS                          ;
                           37             52  E8 003E6          BLBS    STATUS, 67$                         ; 6235
                                          52  DD 003E9          PUSHL   STATUS                              ; 6240
                                0130      C9  9F 003EB          PUSHAB  P.ALG                               ; 6239
                                          1D  11 003EF          BRB     66$                                 ; 6238
                         54        08    AC  D0 003F1 65$:    MOVL    VALUE2, R4                          ; 6251
                                   18    A4  DD 003F5          PUSHL   24(R4)                              ;
                                   04    AE  9F 003F8          PUSHAB  STG_DESC                            ;
              00000000G    00             02  FB 003FB          CALLS   #2, OTS$CVT_T_G                     ;
                           52             50  D0 00402          MOVL    R0, STATUS                          ;
                           18             52  E8 00405          BLBS    STATUS, 67$                         ; 6252
                                          52  DD 00408          PUSHL   STATUS                              ; 6257
                                0140      C9  9F 0040A          PUSHAB  P.ALH                               ; 6256
                                  0C      AE  DD 0040E 66$:    PUSHL   STG_DESC+4                          ; 6255
                                0088      8F  BB 00411          PUSHR   #^M<R3,R7>                          ; 6254
                                          04  DD 00415          PUSHL   #4                                  ;
                            00028E78      8F  DD 00417          PUSHL   #167544                             ;
                         6A                07  FB 0041D          CALLS   #7, LIB$SIGNAL                      ;
                    16   A4              55  90 00420 67$:    MOVB    TARGET_DTYPE, 22(R4)                ; 6259
                    14   A4              08  B0 00424          MOVW    #8, 20(R4)                          ; 6260
                                         44  11 00428 68$:    BRB     73$                                 ; 6034
                         54        08    AC  D0 0042A 69$:    MOVL    VALUE2, R4                          ; 6268
                                   18    A4  DD 0042E          PUSHL   24(R4)                              ;
                                   04    AE  9F 00431          PUSHAB  STG_DESC                            ;
              00000000G    00             02  FB 00434          CALLS   #2, OTS$CVT_T_H                     ;
                           52             50  D0 0043B          MOVL    R0, STATUS                          ;
                           18             52  E8 0043E          BLBS    STATUS, 71$                         ; 6269
                                          52  DD 00441          PUSHL   STATUS                              ; 6274
                                014B      C9  9F 00443          PUSHAB  P.ALI                               ; 6273
                                  0C      AE  DD 00447 70$:    PUSHL   STG_DESC+4                          ; 6272
                                0088      8F  BB 0044A          PUSHR   #^M<R3,R7>                          ; 6271
                                          04  DD 0044E          PUSHL   #4                                  ;
                            00028E78      8F  DD 00450          PUSHL   #167544                             ;
                         6A                07  FB 00456          CALLS   #7, LIB$SIGNAL                      ;
                    16   A4              55  90 00459 71$:    MOVB    TARGET_DTYPE, 22(R4)                ; 6276
                    14   A4              10  B0 0045D          MOVW    #16, 20(R4)                         ; 6277
                                         58  11 00461          BRB     77$                                 ; 6034
                                         56  DD 00463 72$:    PUSHL   R6                                  ; 6284
                    0000V    CF             01  FB 00465          CALLS   #1, CONV_TEXT_PACK_VALUE            ;
                      08     AC             50  D0 0046A          MOVL    R0, VALUE2                          ;
                                         4B  11 0046E 73$:    BRB     77$                                 ;
                         54        04    AE  D0 00470 74$:    MOVL    STG_DESC+4, STG_PTR                 ; 6308
                         55              01  CE 00474          MNEGL   #1, I                               ; 6311
                                         24  11 00477          BRB     76$                                 ;
                         30           6544  91 00479 75$:    CMPB    (I)[STG_PTR], #48                   ;
                                         1E  13 0047D          BEQL    76$                                 ;
                         31           6544  91 0047F          CMPB    (I)[STG_PTR], #49                   ;
                                         18  13 00483          BEQL    76$                                 ;
                                         52  DD 00485          PUSHL   STATUS                              ; 6315
                                0156      C9  9F 00487          PUSHAB  P.ALJ                               ;
```

```
                          0C   AE   DD 0048B          PUSHL   STG_DESC+4                          ; 6314
                        0088   8F   BB 0048E          PUSHR   #^M<R3,R7>                          ; 6313
                               04   DD 00492          PUSHL   #4                                  ;
                    00028E78   8F   DD 00494          PUSHL   #167544                             ;
                          6A        07   FB 0049A          CALLS   #7, LIB$SIGNAL                      ;
D8                        55        57   F2 0049D 76$:     AOBLSS  LENGTH, I, 75$                      ; 6309
                          54   08   AC   D0 004A1          MOVL    VALUE2, R4                          ; 6319
                          53   14   A4   3C 004A5          MOVZWL  20(R4), R3                          ; 6318
                          52   18   A4   D0 004A9          MOVL    24(R4), R2                          ;
                          51        57   D0 004AD          MOVL    LENGTH, R1                          ;
                          50   04   AE   D0 004B0          MOVL    STG_DESC+4, R0                      ;
              00000000G   00        00   FB 004B4          CALLS   #0, PLI$CHARABIT_R6                 ;
                          50   08   AC   D0 004BB 77$:     MOVL    VALUE2, R0                          ; 6335
                    04    A0        20   8A 004BF          BICB2   #32, 4(R0)                          ;
                               10   A0   B4 004C3          CLRW    16(R0)                              ; 6336
                               51   D4 004C6              CLRL    R1                                  ; 6341
                0042   8F   12   A6   B1 004C8          CMPW    18(R6), #66                         ;
                               04   12 004CE              BNEQ    78$                                 ;
                               51   D6 004D0              INCL    R1                                  ;
                               08   11 004D2              BRB     79$                                 ;
                0043   8F   12   A6   B1 004D4 78$:     CMPW    18(R6), #67                         ; 6342
                               28   12 004DA              BNEQ    82$                                 ;
                          0F        51   E9 004DC 79$:     BLBC    R1, 80$                             ; 6345
                               12   A0   B4 004DF          CLRW    18(R0)                              ; 6348
                               7E   D4 004E2              CLRL    -(SP)                               ; 6349
                               50   DD 004E4              PUSHL   R0                                  ; 6350
                    00000000G   00   9F 004E6          PUSHAB  DBG$GL_NEG_CONST_TOKEN              ; 6349
                               0D   11 004EC              BRB     81$                                 ;
                               12   A0   B4 004EE 80$:     CLRW    18(R0)                              ; 6355
                               7E   D4 004F1              CLRL    -(SP)                               ; 6356
                               50   DD 004F3              PUSHL   R0                                  ; 6357
                    00000000G   00   9F 004F5          PUSHAB  DBG$GL_POS_CONST_TOKEN              ; 6356
              0000V   CF        03   FB 004FB 81$:     CALLS   #3, DBG$EVAL_LANG_OPERATOR          ;
                08    AC        50   D0 00500          MOVL    R0, VALUE2                          ;
                          50   08   AC   D0 00504 82$:     MOVL    VALUE2, R0                          ; 6362
                                    04 00508              RET                                 ; 6364

; Routine Size:  1289 bytes,    Routine Base:  DBG$CODE + 00BF
```

```
; 6257          6365  1 GLOBAL ROUTINE DBG$CONV_TQUADWORD_VALUE(VAL_DESC1, VAL_DESC2) =
; 6258          6366  1 !
; 6259          6367  1 ! FUNCTION
; 6260          6368  1 !       This routine takes in unconverted ascii string and converts it into
; 6261          6369  1 !       8 bytes Quardword value according to the given radix
; 6262          6370  1 !       (tokencode).
; 6263          6371  1 !
; 6264          6372  1 ! INPUTS
; 6265          6373  1 !       VAL_DESC1 - Pointer to a value descriptor to be converted.
; 6266          6374  1 !
; 6267          6375  1 !       VAL_DESC2 - Pointer to a value descriptor to be returned.
; 6268          6376  1 !
; 6269          6377  1 ! OUTPUTS
; 6270          6378  1 !       Overflow status is returned.
; 6271          6379  1 !
; 6272          6380  1
; 6273          6381  2     BEGIN
; 6274          6382  2
; 6275          6383  2     MAP
; 6276          6384  2         VAL_DESC1: REF DBG$VALDESC,     ! Pointer to value descriptor
; 6277          6385  2         VAL_DESC2: REF DBG$VALDESC;     ! Pointer to value descriptor
; 6278          6386  2
; 6279          6387  2     LOCAL
; 6280          6388  2         CHAR_VALUE: VECTOR[8, BYTE],    ! Value of the character
; 6281          6389  2         LENGTH,                         ! Lenght of the input data
; 6282          6390  2         PTR: REF VECTOR[,BYTE],         ! Pointer to a vector of bytes
; 6283          6391  2         QUADWORD: VECTOR[8, BYTE],      ! 8 bytes of QUADWORD value
; 6284          6392  2         STATUS,                         ! Return status from conversion
; 6285          6393  2         STG_DESC: DBG$STG_DESC;         ! String descriptor for signal
; 6286          6394  2
; 6287          6395  2
; 6288          6396  2     ! Prepare the string descriptor for signaling.
; 6289          6397  2     !
; 6290          6398  2     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
; 6291          6399  2     STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
; 6292          6400  2     LENGTH = .VAL_DESC1[DBG$W_VALUE_LENGTH];
; 6293          6401  2     STG_DESC[DSC$W_LENGTH] = .LENGTH;
; 6294          6402  2     STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
; 6295          6403  2     CH$MOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC1[DBG$L_VALUE_POINTER],
; 6296          6404  2         .STG_DESC[DSC$A_POINTER]);
; 6297          6405  2
; 6298          6406  2
; 6299          6407  2     INCR I FROM 0 TO 7 DO
; 6300          6408  3         BEGIN
; 6301          6409  3         QUADWORD[.I] = 0;
; 6302          6410  3         CHAR_VALUE[.I] = 0;
; 6303          6411  2         END;
; 6304          6412  2
; 6305          6413  2
; 6306          6414  2     ! Convert the value into 8 bytes one by one byte.
; 6307          6415  2     !
; 6308          6416  2     PTR = .VAL_DESC1[DBG$L_VALUE_POINTER];
; 6309          6417  2     DECR I FROM .LENGTH TO 1 BY 1 DO
; 6310          6418  3         BEGIN
; 6311          6419  3         CASE .VAL_DESC1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
; 6312          6420  3                                                TO   TOKEN$K_MAX_OPERAND OF
; 6313          6421  3             SET
```

```
;  6314          6422  3
;  6315          6423  3
;  6316          6424  3
;  6317          6425  3
;  6318          6426  3
;  6319          6427  4
;  6320          6428  4
;  6321          6429  4
;  6322          6430  4
;  6323          6431  4
;  6324          6432  4
;  6325          6433  4
;  6326          6434  3
;  6327          6435  3
;  6328          6436  3
;  6329          6437  3
;  6330          6438  3
;  6331          6439  4
;  6332          6440  4
;  6333          6441  4
;  6334          6442  4
;  6335          6443  4
;  6336          6444  4
;  6337          6445  4
;  6338          6446  4
;  6339          6447  4
;  6340          6448  4
;  6341          6449  4
;  6342          6450  4
;  6343          6451  3
;  6344          6452  3
;  6345          6453  3
;  6346          6454  3
;  6347          6455  3
;  6348          6456  3
;  6349          6457  4
;  6350          6458  4
;  6351          6459  4
;  6352          6460  4
;  6353          6461  4
;  6354          6462  4
;  6355          6463  4
;  6356          6464  3
;  6357          6465  3
;  6358          6466  3
;  6359          6467  3
;  6360          6468  3
;  6361          6469  4
;  6362          6470  4
;  6363          6471  4
;  6364          6472  4
;  6365          6473  4
;  6366          6474  4
;  6367          6475  4
;  6368          6476  3
;  6369          6477  3
;  6370          6478  3

                ! Validate legal 0-9 characters in decimal radix.
                !
                [TOKEN$K_INTEGER]:
                    BEGIN
                    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
                    THEN
                        CHAR_VALUE[0] = .PTR[0] - 48
                    ELSE
                        SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
                    END;

                ! Validate legal 0-9, A-f, a-f characters in hexdecimal radix.
                !
                [TOKEN$K_HEX_INTEGER]:
                    BEGIN
                    SELECTONE .PTR[0] OF
                        SET
                        [48 TO 57]:
                            CHAR_VALUE[0] = .PTR[0] - 48;
                        [65 TO 70]:
                            CHAR_VALUE[0] = .PTR[0] - 55;
                        [97 TO 102]:
                            CHAR_VALUE[0] = .PTR[0] - 87;
                        [OTHERWISE]:
                            SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
                        TES;
                    END;

                ! Validate legal 0-1 characters in binary radix.
                !
                [TOKEN$K_BIN_INTEGER]:
                    BEGIN
                    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
                    THEN
                        CHAR_VALUE[0] = .PTR[0] - 48
                    ELSE
                        SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
                    END;

                ! Validate legal 0-7 characters in octal radix.
                !
                [TOKEN$K_OCT_INTEGER]:
                    BEGIN
                    IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
                    THEN
                        CHAR_VALUE[0] = .PTR[0] - 48
                    ELSE
                        SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
                    END;

                [INRANGE, OUTRANGE]:
                    $DBG_ERROR('DBGEVALOP\DBG$CONV_TQUADWORD_VALUE');
```

```
;  6371          6479  3
;  6372          6480  3              TES;
;  6373          6481
;  6374          6482  3          STATUS =
;  6375          6483  3              DBG$CVT_TQUADWORD_TO_VALUE(QUADWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
;  6376          6484
;  6377          6485  3          ! Test overflow status.
;  6378          6486  3          !
;  6379          6487  3          IF NOT .STATUS
;  6380          6488  3          THEN
;  6381          6489  4              BEGIN
;  6382          6490  4
;  6383          6491  4              ! If we could convert to an unsigned Quadword, do that with message.
;  6384          6492  4              !
;  6385          6493  4              STATUS =
;  6386          6494  4                  DBG$CVT_TUQUADWORD_TO_VALUE(QUADWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
;  6387          6495  4
;  6388          6496  4              IF NOT .STATUS THEN EXITLOOP;
;  6389          6497  4              SIGNAL(DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
;  6390          6498  3              END;
;  6391          6499  3
;  6392          6500  3          PTR = .PTR + 1;
;  6393          6501  2          END;
;  6394          6502  2
;  6395          6503  2      ! Install the value and fix up the value descriptor.
;  6396          6504  2      !
;  6397          6505  2  !    CH$FILL(%C'0', 16, .VAL_DESC1[DBG$L_VALUE_POINTER]);
;  6398          6506  2      CH$MOVE(8, QUADWORD, .VAL_DESC2[DBG$L_VALUE_POINTER]);
;  6399          6507  2  !    .VAL_DESC2 = .VAL_DESC1;
;  6400          6508  2      RETURN .STATUS;
;  6401          6509  1      END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22  05BF3 P.ALL:    .ASCII  \''DBGEVALOP\<92>\DBG$CONV_TQUADWORD_VALU\      ;
5F 44 52 4F 57 44 41 55 51 54 5F 56 4E 4F 43  05C02                                                              ;
                                 55 4C 41 56  05C11                                                              ;
                                           45  05C15          .ASCII  \E\                                         ;


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

                                                03FC 00000      .ENTRY  DBG$CONV_TQUADWORD_VALUE, Save R2,R3,R4,R5,-  ; 6365
                                                                R6,R7,R8,R9                                         ;
                                59 00000000G  00  9E 00002      MOVAB   LIB$SIGNAL, R9                              ;
                                5E            1C  C2 00009      SUBL2   #28, SP                                     ;
                          02    AE      010E  8F  B0 0000C      MOVW    #270, STG_DESC+2                            ; 6399
                                57        04  AC  D0 00012      MOVL    VAL_DESC1, R7                               ; 6400
                                56        14  A7  3C 00016      MOVZWL  20(R7), LENGTH                              ;
                                6E            56  B0 0001A      MOVW    LENGTH, SIG_DESC                            ; 6401
                                50        03  A6  9E 0001D      MOVAB   3(R6), R0                                   ; 6402
                          7E                04  C7 00021      DIVL3   #4, R0, -(SP)                               ;
                00000000G 00                01  FB 00025      CALLS   #1, DBG$GET_TEMPMEM                         ;
```

```
                                04   AE                50   D0 0002C          MOVL    R0, STG_DESC+4                                  ;
                        04   BE 18   B7                6E   28 00030          MOVC3   STG_DESC, @24(R7), @STG_DESC+4                  ; 6404
                                                      50   D4 00036          CLRL    !                                               ; 6407
                                                   0C AE40 94 00038 1$:      CLRB    QUADWORD[I]                                     ; 6409
                                                   14 AE40 94 0003C          CLRB    CHAR_VALUE[I]                                   ; 6410
                                F4                    50   07 F3 00040      AOBLEQ  #7, I, 1$                                       ; 6407
                                                      52   18 A7 D0 00044    MOVL    24(R7), PTR                                     ; 6416
                                                      56   D6 00048          INCL    I                                               ; 6419
                                                   00D9   31 0004A          BRW     17$                                             ;
                                0F                    01   10 A7 AF 0004D 2$:  CASEW   16(R7), #1, #15                             ;
                        0030            0020          0020   0052 3$:       .WORD   4$-3$,-                                         ;
                        0020            0020          003A   005A                 4$-3$,-                                         ;
                        0020            0075          006B   0062                 4$-3$,-                                         ;
                        0020            0020          0020   006A                 5$-3$,-                                         ;
                                                                                  6$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  9$-3$,-                                         ;
                                                                                  10$-3$,-                                        ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$,-                                         ;
                                                                                  4$-3$                                           ;
                                        00000000'    EF   9F 00072 4$:      PUSHAB  P.ALL                                           ; 6478
                                                     01   DD 00078          PUSHL   #1                                              ;
                                        00028362     8F   DD 0007A          PUSHL   #164706                                         ;
                                                     60   11 00080          BRB     14$                                             ;
                                30                   62   91 00082 5$:      CMPB    (PTR), #48                                      ; 6428
                                                     51   1F 00085          BLSSU   13$                                             ;
                                39                   62   91 00087          CMPB    (PTR), #57                                      ;
                                                     43   11 0008A          BRB     11$                                             ;
                                30                   62   91 0008C 6$:      CMPB    (PTR), #48                                      ; 6442
                                                     05   1F 0008F          BLSSU   7$                                              ;
                                39                   62   91 00091          CMPB    (PTR), #57                                      ;
                                                     3B   1B 00094          BLEQU   12$                                             ;
                             41 8F                   62   91 00096 7$:      CMPB    (PTR), #65                                      ; 6444
                                                     0D   1F 0009A          BLSSU   8$                                              ;
                             46 8F                   62   91 0009C          CMPB    (PTR), #70                                      ;
                                                     07   1A 000A0          BGTRU   8$                                              ;
                14   AE          62                 37   83 000A2          SUBB3   #55, (PTR), CHAR_VALUE                          ; 6445
                                                     3C   11 000A7          BRB     15$                                             ;
                             61 8F                   62   91 000A9 8$:      CMPB    (PTR), #97                                      ; 6446
                                                     29   1F 000AD          BLSSU   13$                                             ;
                             66 8F                   62   91 000AF          CMPB    (PTR), #102                                     ;
                                                     23   1A 000B3          BGTRU   13$                                             ;
                14   AE          62            A9   8F   81 000B5          ADDB3   #-87, (PTR), CHAR_VALUE                         ; 6447
                                                     28   11 000BB          BRB     15$                                             ;
                                30                   62   91 000BD 9$:      CMPB    (PTR), #48                                      ; 6458
                                                     16   1F 000C0          BLSSU   13$                                             ;
                                31                   62   91 000C2          CMPB    (PTR), #49                                      ;
                                                     08   11 000C5          BRB     11$                                             ;
                                30                   62   91 000C7 10$:     CMPB    (PTR), #48                                      ; 6470
                                                     0C   1F 000CA          BLSSU   13$                                             ;
                                37                   62   91 000CC          CMPB    (PTR), #55                                      ;
```

```
                                07  1A 000CF 11$:   BGTRU   13$                                  ;
       14  AE             62    30  83 000D1 12$:   SUBB3   #48, (PTR), CHAR_VALUE               ; 6472
                                0D  11 000D6        BRB     15$                                  ;
                                5E  DD 000D8 13$:   PUSHL   SP                                   ; 6474
                                01  DD 000DA        PUSHL   #1                                   ;
                      000281B0  8F  DD 000DC        PUSHL   #164272                              ;
                          69    03  FB 000E2 14$:   CALLS   #3, LIB$SIGNAL                       ;
                          7E    10  A7  3C 000E5 15$: MOVZWL 16(R7), -(SP)                       ; 6483
                                18  AE  9F 000E9    PUSHAB  CHAR_VALUE                           ;
                                14  AE  9F 000EC    PUSHAB  QUADWORD                             ;
               00000000G  00    03  FB 000EF        CALLS   #3, DBG$CVT_TQUADWORD_TO_VALUE       ;
                          58    50  D0 000F6        MOVL    R0, STATUS                           ;
                          28    58  E8 000F9        BLBS    STATUS, 16$                          ; 6487
                          7E    10  A7  3C 000FC    MOVZWL  16(R7), -(SP)                        ; 6494
                                18  AE  9F 00100    PUSHAB  CHAR_VALUE                           ;
                                14  AE  9F 00103    PUSHAB  QUADWORD                             ;
               00000000G  00    03  FB 00106        CALLS   #3, DBG$CVT_TUQUADWORD_TO_VALUE      ;
                          58    50  D0 0010D        MOVL    R0, STATUS                           ;
                          1B    58  E9 00110        BLBC    STATUS, 19$                          ; 6496
                      00000000' EF  DD 00113        PUSHL   DBG$GL_OPCODE_NAME                   ; 6497
                                01  DD 00119        PUSHL   #1                                   ;
                      000286A3  8F  DD 0011B        PUSHL   #165539                              ;
                          69    03  FB 00121        CALLS   #3, LIB$SIGNAL                       ;
                                52  D6 00124 16$:   INCL    PTR                                  ; 6500
                          02    56  F5 00126 17$:   SOBGTR  I, 18$                               ; 6417
                                03  11 00129        BRB     19$                                  ;
                          FF1F  31 0012B 18$:       BRW     2$                                   ;
                          50    08  AC  D0 0012E 19$: MOVL  VAL_DESC2, R0                        ; 6507
       18  B0      0C     AE    08  28 00132        MOVC3   #8, QUADWORD, @24(R0)                ;
                          50    58  D0 00138        MOVL    STATUS, R0                           ; 6509
                                04 0013B            RET                                          ; 6510

; Routine Size:  316 bytes,    Routine Base:  DBG$CODE + 05C8
```

```
;  6404          6511  1 GLOBAL ROUTINE DBG$CONV_TOCTAWORD_VALUE(VAL_DESC1, VAL_DESC2) =
;  6405          6512  1 !
;  6406          6513  1 ! FUNCTION
;  6407          6514  1 !       This routine takes in unconverted ascii string and converts it into
;  6408          6515  1 !       16 bytes octaword value according to the given radix
;  6409          6516  1 !       (tokencode).
;  6410          6517  1 !
;  6411          6518  1 ! INPUTS
;  6412          6519  1 !       VAL_DESC1 - Pointer to a value descriptor to be converted.
;  6413          6520  1 !
;  6414          6521  1 !       VAL_DESC2 - Pointer to a value descriptor to be returned.
;  6415          6522  1 !
;  6416          6523  1 ! OUTPUTS
;  6417          6524  1 !       Overflow status is returned.
;  6418          6525  1 !
;  6419          6526  1
;  6420          6527  2     BEGIN
;  6421          6528  2
;  6422          6529  2     MAP
;  6423          6530  2         VAL_DESC1: REF DBG$VALDESC,     ! Pointer to value descriptor
;  6424          6531  2         VAL_DESC2: REF DBG$VALDESC;     ! Pointer to value descriptor
;  6425          6532  2
;  6426          6533  2     LOCAL
;  6427          6534  2         CHAR_VALUE: VECTOR[16, BYTE],   ! Value of the character
;  6428          6535  2         LENGTH,                         ! Lenght of the input data
;  6429          6536  2         PTR: REF VECTOR[,BYTE],         ! Pointer to a vector of bytes
;  6430          6537  2         OCTAWORD: VECTOR[16, BYTE],     ! 16 bytes of octaword value
;  6431          6538  2         STATUS,                         ! Return status from conversion
;  6432          6539  2         STG_DESC: DBG$STG_DESC;         ! String descriptor for signal
;  6433          6540  2
;  6434          6541  2
;  6435          6542  2     ! Prepare the string descriptor for signaling.
;  6436          6543  2     !
;  6437          6544  2     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  6438          6545  2     STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  6439          6546  2     LENGTH = .VAL_DESC1[DBG$W_VALUE_LENGTH];
;  6440          6547  2     STG_DESC[DSC$W_LENGTH] = .LENGTH;
;  6441          6548  2     STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
;  6442          6549  2     CH$MOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC1[DBG$L_VALUE_POINTER],
;  6443          6550  2         .STG_DESC[DSC$A_POINTER]);
;  6444          6551  2
;  6445          6552  2     INCR I FROM 0 TO 15 DO
;  6446          6553  3         BEGIN
;  6447          6554  3         OCTAWORD[.I] = 0;
;  6448          6555  3         CHAR_VALUE[.I] = 0;
;  6449          6556  2         END;
;  6450          6557  2
;  6451          6558  2
;  6452          6559  2     ! Convert the value into 16 bytes one by one byte.
;  6453          6560  2     !
;  6454          6561  2     PTR = .VAL_DESC1[DBG$L_VALUE_POINTER];
;  6455          6562  2     DECR I FROM .LENGTH TO 1 BY 1 DO
;  6456          6563  3         BEGIN
;  6457          6564  3         CASE .VAL_DESC1[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
;  6458          6565  3                                                TO   TOKEN$K_MAX_OPERAND OF
;  6459          6566  3             SET
;  6460          6567  3
```

```
;  6461          6568  3
;  6462          6569  3          ! Validate legal 0-9 characters in decimal radix.
;  6463          6570  3          !
;  6464          6571  3          [TOKEN$K_INTEGER]:
;  6465          6572  4              BEGIN
;  6466          6573  4              IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
;  6467          6574  4              THEN
;  6468          6575  4                  CHAR_VALUE[0] = .PTR[0] - 48
;  6469          6576  4              ELSE
;  6470          6577  4                  SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6471          6578  3              END;
;  6472          6579  3
;  6473          6580  3
;  6474          6581  3          ! Validate legal 0-9, A-F, a-f characters in hexdecimal radix.
;  6475          6582  3          !
;  6476          6583  3          [TOKEN$K_HEX_INTEGER]:
;  6477          6584  4              BEGIN
;  6478          6585  4              SELECTONE .PTR[0] OF
;  6479          6586  4                  SET
;  6480          6587  4                  [48 TO 57]:
;  6481          6588  4                      CHAR_VALUE[0] = .PTR[0] - 48;
;  6482          6589  4                  [65 TO 70]:
;  6483          6590  4                      CHAR_VALUE[0] = .PTR[0] - 55;
;  6484          6591  4                  [97 TO 102]:
;  6485          6592  4                      CHAR_VALUE[0] = .PTR[0] - 87;
;  6486          6593  4                  [OTHERWISE]:
;  6487          6594  4                      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6488          6595  4                  TES;
;  6489          6596  3              END;
;  6490          6597  3
;  6491          6598  3
;  6492          6599  3          ! Validate legal 0-1 characters in binary radix.
;  6493          6600  3          !
;  6494          6601  3          [TOKEN$K_BIN_INTEGER]:
;  6495          6602  4              BEGIN
;  6496          6603  4              IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
;  6497          6604  4              THEN
;  6498          6605  4                  CHAR_VALUE[0] = .PTR[0] - 48
;  6499          6606  4              ELSE
;  6500          6607  4                  SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6501          6608  3              END;
;  6502          6609  3
;  6503          6610  3
;  6504          6611  3          ! Validate legal 0-7 characters in octal radix.
;  6505          6612  3          !
;  6506          6613  3          [TOKEN$K_OCT_INTEGER]:
;  6507          6614  4              BEGIN
;  6508          6615  4              IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
;  6509          6616  4              THEN
;  6510          6617  4                  CHAR_VALUE[0] = .PTR[0] - 48
;  6511          6618  4              ELSE
;  6512          6619  4                  SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6513          6620  3              END;
;  6514          6621  3
;  6515          6622  3          [INRANGE, OUTRANGE]:
;  6516          6623  3              $DBG_ERROR('DBGEVALOP\DBG$CONV_TOCTAWORD_VALUE');
;  6517          6624  3
```

```
;  6518          6625  3              TES;
;  6519          6626  3
;  6520          6627  3          STATUS =
;  6521          6628  3              DBG$CVT_TOCTAWORD_TO_VALUE(OCTAWORD, CHAR_VALUE, .VAL_DESC1[DBG$W_VALUE_TOKENCODE]);
;  6522          6629  3
;  6523          6630  3          ! Test overflow status.
;  6524          6631  3          !
;  6525          6632  3          IF NOT .STATUS THEN EXITLOOP;
;  6526          6633  3
;  6527          6634  3          PTR = .PTR + 1;
;  6528          6635  2          END;
;  6529          6636  2
;  6530          6637  2      ! Check for integer overflow on the sign bit.
;  6531          6638  2      !
;  6532          6639  2      IF .OCTAWORD[15] GEQ 128
;  6533          6640  2      THEN
;  6534          6641  2          SIGNAL(DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
;  6535          6642  2
;  6536          6643  2      ! Install the value and fix up the value descriptor.
;  6537          6644  2      !
;  6538          6645  2      CH$MOVE(16, OCTAWORD, .VAL_DESC2[DBG$L_VALUE_POINTER]);
;  6539          6646  2      RETURN .STATUS;
;  6540          6647  1      END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22 05C16 P.ALM:   .ASCII  \"DBGEVALOP\<92>\DBG$CONV_TOCTAWORD_VALU\   ;
5F 44 52 4F 57 41 54 43 4F 54 5F 56 4E 4F 43 05C25                                                          ;
                              55 4C 41 56 05C34                                                             ;
                                          45 05C38          .ASCII  \E\                                     ;


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

                                                  01FC 00000        .ENTRY  DBG$CONV_TOCTAWORD_VALUE, Save R2,R3,R4,R5,-; 6511
                                                                            R6,R7,R8                                    ;
                                58 00000000G      00  9E 00002        MOVAB   LIB$SIGNAL, R8                          ;
                                5E                2C  C2 00009        SUBL2   #44, SP                                 ;
                          02    AE      010E      8F  B0 0000C        MOVW    #270, STG_DESC+2                        ; 6545
                                57        04      AC  D0 00012        MOVL    VAL_DESC1, R7                           ; 6546
                                56        14      A7  3C 00016        MOVZWL  20(R7), LENGTH                          ;
                                6E                56  B0 0001A        MOVW    LENGTH, STG_DESC                        ; 6547
                                52        03      A6  9E 0001D        MOVAB   3(R6), R2                               ; 6548
                          7E                      52  04 C7 00021     DIVL3   #4, R2, -(SP)                           ;
                   00000000G      00              01  FB 00025        CALLS   #1, DBG$GET_TEMPMEM                     ;
                             04   AE              50  D0 0002C        MOVL    R0, STG_DESC+4                          ;
                    04    BE       18   B7              6E 28 00030     MOVC3   STG_DESC, @24(R7), @STG_DESC+4          ; 6550
                                                      50  D4 00036     CLRL    I                                       ; 6552
                                                  0C AE40 94 00038 1$:  CLRB    OCTAWORD[I]                             ; 6554
                                                  1C AE40 94 0003C      CLRB    CHAR_VALUE[I]                           ; 6555
                          F4                      50  0F F3 00040     AOBLEQ  #15, I, 1$                              ; 6552
                                50        18      A7  D0 00044        MOVL    24(R7), PTR                             ; 6561
                                52        01      A6  9E 00048        MOVAB   1(R6), I                                ; 6564
                                              00B1  31 0004C        BRW     16$                                     ;
```

```
                0F      01      10  A7  AF 0004F 2$:    CASEW   16(R7), #1, #15
          0030  0020    0020        0020    00054 3$:    .WORD   4$-3$,-
          0020  0020    0020        003A    0005C                4$-3$,-
          0020  0075    006B        0020    00064                4$-3$,-
          0020  0020    0020        0020    0006C                5$-3$,-
                                                                 6$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 9$-3$,-
                                                                 10$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$,-
                                                                 4$-3$
                    00000000'   EF  9F 00074 4$:    PUSHAB  P.ALM                       6623
                                01  DD 0007A        PUSHL   #1
                    00028362    8F  DD 0007C        PUSHL   #164706
                                60  11 00082        BRB     14$
                30              63  91 00084 5$:    CMPB    (PTR), #48                  6573
                                51  1F 00087        BLSSU   13$
                39              63  91 00089        CMPB    (PTR), #57
                                43  11 0008C        BRB     11$
                30              63  91 0008E 6$:    CMPB    (PTR), #48                  6587
                                05  1F 00091        BLSSU   7$
                39              63  91 00093        CMPB    (PTR), #57
                                3B  1B 00096        BLEQU   12$
             41 8F              63  91 00098 7$:    CMPB    (PTR), #65                  6589
                                0D  1F 0009C        BLSSU   8$
             46 8F              63  91 0009E        CMPB    (PTR), #70
                                07  1A 000A2        BGTRU   8$
          1C AE              63  37  83 000A4        SUBB3   #55, (PTR), CHAR_VALUE      6590
                                3C  11 000A9        BRB     15$
             61 8F              63  91 000AB 8$:    CMPB    (PTR), #97                  6591
                                29  1F 000AF        BLSSU   13$
             66 8F              63  91 000B1        CMPB    (PTR), #102
                                23  1A 000B5        BGTRU   13$
          1C AE              63  8F  81 000B7        ADDB3   #-87, (PTR), CHAR_VALUE     6592
                                28  11 000BD        BRB     15$
                30              63  91 000BF 9$:    CMPB    (PTR), #48                  6603
                                16  1F 000C2        BLSSU   13$
                31              63  91 000C4        CMPB    (PTR), #49
                                08  11 000C7        BRB     11$
                30              63  91 000C9 10$:   CMPB    (PTR), #48                  6615
                                0C  1F 000CC        BLSSU   13$
                37              63  91 000CE        CMPB    (PTR), #55
                                07  1A 000D1 11$:   BGTRU   13$
          1C AE              63  30  83 000D3 12$:   SUBB3   #48, (PTR), CHAR_VALUE     6617
                                0D  11 000D8        BRB     15$
                                5E  DD 000DA 13$:   PUSHL   SP                          6619
                                01  DD 000DC        PUSHL   #1
                    00028180    8F  DD 000DE        PUSHL   #164272
                68              03  FB 000E4 14$:   CALLS   #3, LIB$SIGNAL
                7E      10      A7  3C 000E7 15$:   MOVZWL  16(R7), -(SP)               6628
                        20      AE  9F 000EB        PUSHAB  CHAR_VALUE
```

```
                                    14    AE  9F 000EE             PUSHAB  OCTAWORD                                        ;
                          00000000G 00          03  FB 000F1       CALLS   #3, DBG$CVT_TOCTAWORD_TO_VALUE                  ;
                                    56          50  D0 000F8       MOVL    R0, STATUS                                      ;
                                    0A          56  E9 000FB       BLBC    STATUS, 18$                                     ; 6632
                                                53  D6 000FE       INCL    PTR                                             ; 6634
                                    02          52  F5 00100 16$:   SOBGTR  I, 17$                                         ; 6562
                                                03  11 00103       BRB     18$                                             ;
                                            FF47    31 00105 17$:   BRW     2$                                              ;
                                80  8F      1B    AE  91 00108 18$:   CMPB    OCTAWORD+15, #128                           ; 6639
                                                11  1F 0010D       BLSSU   19$                                             ;
                                    00000000'   EF  DD 0010F       PUSHL   DBG$GL_OPCODE_NAME                              ; 6641
                                                01  DD 00115       PUSHL   #1                                              ;
                                    000286A3    8F  DD 00117       PUSHL   #165539                                         ;
                                    68          03  FB 0011D       CALLS   #3, LIB$SIGNAL                                  ;
                                    50      08  AC  D0 00120 19$:   MOVL    VAL_DESC2, R0                               ; 6645
                              18    B0      0C  AE          10  28 00124   MOVC3   #16, OCTAWORD, @24(R0)                   ;
                                    50          56  D0 0012A       MOVL    STATUS, R0                                      ; 6646
                                                04  00 0012D       RET                                                     ; 6647

; Routine Size:  302 bytes,    Routine Base:  DBG$CODE + 0704
```

```
;  6542          6648  1 GLOBAL ROUTINE DBG$CONV_TRFA_VALUE(VAL_DESC) =
;  6543          6649  1 !
;  6544          6650  1 ! FUNCTION
;  6545          6651  1 !       This routine takes in unconverted ascii string and converts it into
;  6546          6652  1 !       6 bytes Record File Address value according to the given radix
;  6547          6653  1 !       (tokencode).
;  6548          6654  1 !
;  6549          6655  1 ! INPUTS
;  6550          6656  1 !       VAL_DESC - Pointer to a value descriptor to be converted.
;  6551          6657  1 !
;  6552          6658  1 ! OUTPUTS
;  6553          6659  1 !       Converted value is returned in VAL_DESC.
;  6554          6660  1 !
;  6555          6661  1
;  6556          6662  2     BEGIN
;  6557          6663  2
;  6558          6664  2     MAP
;  6559          6665  2         VAL_DESC: REF DBG$VALDESC;      ! Pointer to value descriptor
;  6560          6666  2
;  6561          6667  2     LOCAL
;  6562          6668  2         CHAR_VALUE: VECTOR[6, BYTE],    ! Value of the character
;  6563          6669  2         LENGTH,                         ! Lenght of the input data
;  6564          6670  2         PTR: REF VECTOR[,BYTE],         ! Pointer to a vector of bytes
;  6565          6671  2         RFA: VECTOR[6, BYTE],           ! 6 bytes of RFA value
;  6566          6672  2         STATUS,                         ! Return status from conversion
;  6567          6673  2         STG_DESC: DBG$STG_DESC;         ! String descriptor for signal
;  6568          6674  2
;  6569          6675  2
;  6570          6676  2     ! Won't accept the negative constant.
;  6571          6677  2     !
;  6572          6678  2     IF .VAL_DESC[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
;  6573          6679  2     THEN
;  6574          6680  2         SIGNAL(DBG$_CVTNEGUNS, 1, .DBG$GL_OPCODE_NAME);
;  6575          6681  2
;  6576          6682  2
;  6577          6683  2     ! Prepare the string descriptor for signaling.
;  6578          6684  2     !
;  6579          6685  2     STG_DESC[DSC$B_CLASS] = DSC$K_CLASS_S;
;  6580          6686  2     STG_DESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  6581          6687  2     LENGTH = .VAL_DESC[DBG$W_VALUE_LENGTH];
;  6582          6688  2     STG_DESC[DSC$W_LENGTH] = .LENGTH;
;  6583          6689  2     STG_DESC[DSC$A_POINTER] = DBG$GET_TEMPMEM((3+.LENGTH)/4);
;  6584          6690  2     CH$MOVE(.STG_DESC[DSC$W_LENGTH], .VAL_DESC[DBG$L_VALUE_POINTER],
;  6585          6691  2         .STG_DESC[DSC$A_POINTER]);
;  6586          6692  2
;  6587          6693  2
;  6588          6694  2     INCR I FROM 0 TO 5 DO
;  6589          6695  3         BEGIN
;  6590          6696  3         RFA[.I] = 0;
;  6591          6697  3         CHAR_VALUE[.I] = 0;
;  6592          6698  2         END;
;  6593          6699  2
;  6594          6700  2
;  6595          6701  2     ! Convert the value into 6 bytes one by one byte.
;  6596          6702  2     !
;  6597          6703  2     PTR = .VAL_DESC[DBG$L_VALUE_POINTER];
;  6598          6704  2     DECR I FROM .LENGTH TO 1 BY 1 DO
```

```
;  6599          6705  3          BEGIN
;  6600          6706  3          CASE .VAL_DESC[DBG$W_VALUE_TOKENCODE] FROM TOKEN$K_MIN_OPERAND
;  6601          6707  3                                                TO   TOKEN$K_MAX_OPERAND OF
;  6602          6708  3              SET
;  6603          6709  3
;  6604          6710  3
;  6605          6711  3              ! Validate legal 0-9 characters in decimal radix.
;  6606          6712  3              !
;  6607          6713  3              [TOKEN$K_INTEGER]:
;  6608          6714  4                  BEGIN
;  6609          6715  4                  IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 57
;  6610          6716  4                  THEN
;  6611          6717  4                      CHAR_VALUE[0] = .PTR[0] - 48
;  6612          6718  4                  ELSE
;  6613          6719  4                      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6614          6720  3                  END;
;  6615          6721  3
;  6616          6722  3
;  6617          6723  3              ! Validate legal 0-9, A-F, a-f characters in hexdecimal radix.
;  6618          6724  3              !
;  6619          6725  3              [TOKEN$K_HEX_INTEGER]:
;  6620          6726  4                  BEGIN
;  6621          6727  4                  SELECTONE .PTR[0] OF
;  6622          6728  4                      SET
;  6623          6729  4                      [48 TO 57]:
;  6624          6730  4                          CHAR_VALUE[0] = .PTR[0] - 48;
;  6625          6731  4                      [65 TO 70]:
;  6626          6732  4                          CHAR_VALUE[0] = .PTR[0] - 55;
;  6627          6733  4                      [97 TO 102]:
;  6628          6734  4                          CHAR_VALUE[0] = .PTR[0] - 87;
;  6629          6735  4                      [OTHERWISE]:
;  6630          6736  4                          SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6631          6737  4                      TES;
;  6632          6738  3                  END;
;  6633          6739  3
;  6634          6740  3
;  6635          6741  3              ! Validate legal 0-1 characters in binary radix.
;  6636          6742  3              !
;  6637          6743  3              [TOKEN$K_BIN_INTEGER]:
;  6638          6744  4                  BEGIN
;  6639          6745  4                  IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 49
;  6640          6746  4                  THEN
;  6641          6747  4                      CHAR_VALUE[0] = .PTR[0] - 48
;  6642          6748  4                  ELSE
;  6643          6749  4                      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
;  6644          6750  3                  END;
;  6645          6751  3
;  6646          6752  3
;  6647          6753  3              ! Validate legal 0-7 characters in octal radix.
;  6648          6754  3              !
;  6649          6755  3              [TOKEN$K_OCT_INTEGER]:
;  6650          6756  4                  BEGIN
;  6651          6757  4                  IF .PTR[0] GEQ 48 AND .PTR[0] LEQ 55
;  6652          6758  4                  THEN
;  6653          6759  4                      CHAR_VALUE[0] = .PTR[0] - 48
;  6654          6760  4                  ELSE
;  6655          6761  4                      SIGNAL(DBG$_INVNUMBER, 1, STG_DESC);
```

```
;  6656          6762  3                END;
;  6657          6763
;  6658          6764  3            [INRANGE, OUTRANGE]:
;  6659          6765  3                $DBG_ERROR('DBGEVALOP\DBG$CONV_TRFA_VALUE');
;  6660          6766
;  6661          6767  3                TES;
;  6662          6768
;  6663          6769  3            STATUS =
;  6664          6770  3                DBG$CVT_TRFA_TO_VALUE(RFA, CHAR_VALUE, .VAL_DESC[DBG$W_VALUE_TOKENCODE]);
;  6665          6771
;  6666          6772
;  6667          6773  3            ! Test overflow status.
;  6668          6774  3            !
;  6669          6775  3            IF NOT .STATUS
;  6670          6776  3            THEN
;  6671          6777  4                BEGIN
;  6672          6778  4                SIGNAL(DBG$_IRFAOVF, 1, .DBG$GL_OPCODE_NAME);
;  6673          6779  4                EXITLOOP;
;  6674          6780  3                END;
;  6675          6781  3
;  6676          6782  3            PTR = .PTR + 1;
;  6677          6783  2            END;
;  6678          6784  2
;  6679          6785  2        ! Install the value and fix up the value descriptor.
;  6680          6786  2        !
;  6681          6787  2        VAL_DESC[DBG$V_DHDR_UNCVT] = 0;
;  6682          6788  2        VAL_DESC[DBG$B_DHDR_FCODE] = RST$K_TYPE_RFA;
;  6683          6789  2        VAL_DESC[DBG$B_VALUE_CLASS] = 0;
;  6684          6790  2        VAL_DESC[DBG$B_VALUE_DTYPE] = 0;
;  6685          6791  2        VAL_DESC[DBG$W_VALUE_LENGTH] = 6;
;  6686          6792  2        CH$FILL(%C'0', 16, .VAL_DESC[DBG$L_VALUE_POINTER]);
;  6687          6793  2        CH$MOVE(6, RFA, .VAL_DESC[DBG$L_VALUE_POINTER]);
;  6688          6794  2        VAL_DESC[DBG$W_VALUE_TOKENCODE] = 0;
;  6689          6795  2        VAL_DESC[DBG$W_VALUE_SIGN_CODE] = 0;
;  6690          6796  2        RETURN .VAL_DESC;
;  6691          6797  1        END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 1D  05C39 P.ALN:    .ASCII  <29>\DBGEVALOP\<92>\DBG$CONV_TRFA_VALU\     ;
   55 4C 41 56 5F 41 46 52 54 5F 56 4E 4F 43  05C48                                                          ;
                                              45  05C56          .ASCII  \E\                                  ;


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

                                                  03FC 00000    .ENTRY  DBG$CONV_TRFA_VALUE, Save R2,R3,R4,R5,R6,-  ; 6648
                                                                        R7,R8,R9                                    ;
                                  59 00000000'    EF   9E 00002 MOVAB   DBG$GL_OPCODE_NAME, R9                      ;
                                  58 00000000G    00   9E 00009 MOVAB   LIB$SIGNAL, R8                            ;
                                  5E              1C   C2 00010 SUBL2   #28, SP                                   ;
                                  57        04    AC   D0 00013 MOVL    VAL_DESC, R7                              ; 6678
                            0042  8F        12    A7   B1 00017 CMPW    18(R7), #6                                ;
```

```
                                                 0D  12 0001D           BNEQ    1$                                                      ;
                                                 69  DD 0001F           PUSHL   DBG$GL_OPCODE_NAME                                      ; 6680
                                                 01  DD 00021           PUSHL   #1                                                      ;
                                      00028EF0   8F  DD 00023           PUSHL   #167664                                                 ;
                                          68     03  FB 00029           CALLS   #3, LIB$SIGNAL                                          ;
                                       02 AE  010E   8F  B0 0002C 1$:     MOVW    #270, STG_DESC+2                                        ; 6686
                                          56  14   A7  3C 00032           MOVZWL  20(R7), LENGTH                                          ; 6687
                                          6E       56  B0 00036           MOVW    LENGTH, STG_DESC                                        ; 6688
                                          50  03   A6  9E 00039           MOVAB   3(R6), R0                                               ; 6689
                                   7E             50  C7 0003D           DIVL3   #4, R0, -(SP)                                           ;
                                00000000G         00  FB 00041           CALLS   #1, DBG$GET_TEMPMEM                                     ;
                                       04 AE          50  D0 00048           MOVL    R0, STG_DESC+4                                          ;
                                04 BE  18 B7           6E  28 0004C           MOVC3   STG_DESC, @24(R7), @STG_DESC+4                          ; 6691
                                                 50  D4 00052           CLRL    I                                                       ; 6694
                                              0C AE40  94 00054 2$:     CLRB    RFA[I]                                                  ; 6696
                                              14 AE40  94 00058         CLRB    CHAR_VALUE[I]                                           ; 6697
                                   F4            50  05  F3 0005C           AOBLEQ  #5, I, 2$                                               ; 6694
                                                 52  18   A7  D0 00060     MOVL    24(R7), PTR                                             ; 6703
                                                 56  D6 00064           INCL    I                                                       ; 6704
                                              00C0  31 00066           BRW     18$                                                     ;
                                   0F            01  10   A7  AF 00069 3$:     CASEW   16(R7), #1, #15                                         ; 6706
                0030          0020          0020          0020      0006E 4$:     .WORD   5$-4$,-                                                 ;
                0020          0020          0020          003A      00076         5$-4$,-                                                 ;
                0020          0075          006B          0020      0007E         5$-4$,-                                                 ;
                0020          0020          0020          0020      00086         6$-4$,-                                                 ;
                                                                                  7$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  10$-4$,-                                                ;
                                                                                  11$-4$,-                                                ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$,-                                                 ;
                                                                                  5$-4$                                                   ;
                                      00000000'  EF  9F 0008E 5$:       PUSHAB  P.ALN                                                   ; 6765
                                                 01  DD 00094           PUSHL   #1                                                      ;
                                      00028362   8F  DD 00096           PUSHL   #164706                                                 ;
                                                 60  11 0009C           BRB     15$                                                     ;
                                          30         62  91 0009E 6$:      CMPB    (PTR), #48                                              ; 6715
                                                     51  1F 000A1           BLSSU   14$                                                     ;
                                          39         62  91 000A3           CMPB    (PTR), #57                                              ;
                                                     43  11 000A6           BRB     12$                                                     ;
                                          30         62  91 000A8 7$:      CMPB    (PTR), #48                                              ; 6729
                                                     05  1F 000AB           BLSSU   8$                                                      ;
                                          39         62  91 000AD           CMPB    (PTR), #57                                              ;
                                                     3B  1B 000B0           BLEQU   13$                                                     ;
                                       41 8F         62  91 000B2 8$:      CMPB    (PTR), #65                                              ; 6731
                                                     0D  1F 000B6           BLSSU   9$                                                      ;
                                       46 8F         62  91 000B8           CMPB    (PTR), #70                                              ;
                                                     07  1A 000BC           BGTRU   9$                                                      ;
                                14 AE                62  37  83 000BE       SUBB3   #55, (PTR), CHAR_VALUE                                  ; 6732
                                                     3C  11 000C3           BRB     16$                                                     ;
                                       61 8F         62  91 000C5 9$:      CMPB    (PTR), #97                                              ; 6733
                                                     29  1F 000C9           BLSSU   14$                                                     ;
```

```
                    66  8F          62 91 000CB        CMPB    (PTR), #102                          ;
                                    23 1A 000CF        BGTRU   14$                                  ;
          14  AE        62      A9  8F 81 000D1        ADDB3   #-87, (PTR), CHAR_VALUE              ; 6734
                                    28 11 000D7        BRB     16$                                  ;
                        30          62 91 000D9 10$:   CMPB    (PTR), #48                           ; 6745
                                    16 1F 000DC        BLSSU   14$                                  ;
                        31          62 91 000DE        CMPB    (PTR), #49                           ;
                                    08 11 000E1        BRB     12$                                  ;
                        30          62 91 000E3 11$:   CMPB    (PTR), #48                           ; 6757
                                    0C 1F 000E6        BLSSU   14$                                  ;
                        37          62 91 000E8        CMPB    (PTR), #55                           ;
                                    07 1A 000EB 12$:   BGTRU   14$                                  ;
          14  AE        62          30 83 000ED 13$:   SUBB3   #48, (PTR), CHAR_VALUE               ; 6759
                                    0D 11 000F2        BRB     16$                                  ;
                                    5E DD 000F4 14$:   PUSHL   SP                                   ; 6761
                                    01 DD 000F6        PUSHL   #1                                   ;
                          000281B0  8F DD 000F8        PUSHL   #164272                              ;
                        68          03 FB 000FE 15$:   CALLS   #3, LIB$SIGNAL                       ;
                        7E      10  A7 3C 00101 16$:   MOVZWL  16(R7), -(SP)                        ; 6770
                                18  AE 9F 00105        PUSHAB  CHAR_VALUE                           ;
                                14  AE 9F 00108        PUSHAB  RFA                                  ;
                   00000000G    00  03 FB 0010B        CALLS   #3, DBG$CVT_TRFA_TO_VALUE            ;
                                53  50 D0 00112        MOVL    R0, STATUS                           ;
                                0F  53 E8 00115        BLBS    STATUS, 17$                          ; 6775
                                    69 DD 00118        PUSHL   DBG$GL_OPCODE_NAME                   ; 6778
                                    01 DD 0011A        PUSHL   #1                                   ;
                          00028FCB  8F DD 0011C        PUSHL   #167883                              ;
                        68          03 FB 00122        CALLS   #3, LIB$SIGNAL                       ;
                                    0A 11 00125        BRB     20$                                  ; 6777
                                    52 D6 00127 17$:   INCL    PTR                                  ; 6782
                        02          56 F5 00129 18$:   SOBGTR  I, 19$                               ; 6704
                                    03 11 0012C        BRB     20$                                  ;
                                FF38 31 0012E 19$:     BRW     3$                                   ;
                    04  A7          20 8A 00131 20$:   BICB2   #32, 4(R7)                           ; 6788
                    06  A7          14 90 00135        MOVB    #20, 6(R7)                           ; 6789
                    14  A7          06 D0 00139        MOVL    #6, 20(R7)                           ; 6792
     10           30    6E          00 2C 0013D        MOVC5   #0, (SP), #48, #16, @24(R7)          ; 6793
                                18  B7    00142                                                     ;
          18  B7    0C  AE          06 28 00144        MOVC3   #6, RFA, @24(R7)                     ; 6794
                                10  A7 D4 0014A        CLRL    16(R7)                               ; 6795
                        50          57 D0 0014D        MOVL    R7, R0                               ; 6797
                                       04 00150        RET                                          ; 6798

; Routine Size:  337 bytes,    Routine Base:  DBG$CODE + 0832
```

```
;  6694          6799  1
;  6695          6800  1 GLOBAL ROUTINE DBG$DO_MAPPING (LEFT_ARG) : NOVALUE =
;  6696          6801  1 !
;  6697          6802  1 ! FUNCTION
;  6698          6803  1 !       Perform a language-specific "type mapping" on the given
;  6699          6804  1 !       value descriptor. For example, in FORTRAN, we map unsigned integer
;  6700          6805  1 !       types to signed integer types, because the compiler is given us
;  6701          6806  1 !       the BU, WU, or LU types, while the language really does signed
;  6702          6807  1 !       arithmetic on these.
;  6703          6808  1 !
;  6704          6809  1 ! INPUTS
;  6705          6810  1 !       LEFT_ARG        - points to the value descriptor to be mapped.
;  6706          6811  1 !
;  6707          6812  1 ! OUTPUTS
;  6708          6813  1 !       The value descriptor pointed to by LEFT_ARG may be modified.
;  6709          6814  1 !
;  6710          6815  2     BEGIN
;  6711          6816  2     MAP
;  6712          6817  2         LEFT_ARG: REF DBG$VALDESC;
;  6713          6818  2
;  6714          6819  2     LOCAL
;  6715          6820  2         COMPLIST: REF VECTOR[,LONG],
;  6716          6821  2         DSTPTR,
;  6717          6822  2         LEFT_TYPE,                          ! Dtype for original left operand type
;  6718          6823  2         MAKE_BOOLEAN_FLAG,
;  6719          6824  2         MAP_TBL_ENTRY: TYPE_GRAPH$ENTRY,    ! An entry in the Type Mapping Table
;  6720          6825  2         NAME: REF VECTOR[,BYTE],            ! Name corresponding to typeid
;  6721          6826  2         SYMID: REF RST$ENTRY,
;  6722          6827  2         TYPEID: REF RST$ENTRY;              ! Typeid in descriptor
;  6723          6828  2
;  6724          6829  2
;  6725          6830  2     ! For ADA, we map the enumeration type BOOLEAN into the dtype
;  6726          6831  2     ! DSC$K_DTYPE_TF. The reason for this is that we want the logical
;  6727          6832  2     ! operators AND, OR, XOR, NOT to work on BOOLEAN (but not other
;  6728          6833  2     ! enumeration types).
;  6729          6834  2     !
;  6730          6835  2     IF .DBG$GB_LANGUAGE EQL DBG$K_ADA
;  6731          6836  2     THEN
;  6732          6837  3         BEGIN
;  6733          6838  3         IF .LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ENUM
;  6734          6839  3         THEN
;  6735          6840  4             BEGIN
;  6736          6841  4             TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];
;  6737          6842  4             IF .TYPEID NEQ 0
;  6738          6843  4             THEN
;  6739          6844  5                 BEGIN
;  6740          6845  5                 DSTPTR = .TYPEID[RST$L_DSTPTR];
;  6741          6846  5                 NAME = DBG$GET_DST_NAME(.DSTPTR);
;  6742          6847  5                 IF .NAME[0] EQL 7
;  6743          6848  5                 THEN
;  6744          6849  6                     BEGIN
;  6745          6850  6                     IF CH$EQL(7, NAME[1], 7, UPLIT (%ASCII 'BOOLEAN'))
;  6746          6851  6                     THEN
;  6747          6852  7                         BEGIN
;  6748          6853  7
;  6749          6854  7                         ! We have determined we have data of type 'BOOLEAN'.
;  6750          6855  7                         ! Do some additional checking here:
```

```
;  6751          6856  7                ! check that the enumeration elements are
;  6752          6857  7                ! (FALSE, TRUE). The reason is that the user may
;  6753          6858  7                ! have redefined BOOLEAN so we may not be looking
;  6754          6859  7                ! at a "real" boolean type.
;  6755          6860  7                !
;  6756          6861  7                IF .TYPEID[RST$L_TYPCOMPCNT] EQL 2
;  6757          6862  7                THEN
;  6758          6863  8                    BEGIN
;  6759          6864  8                    COMPLIST = TYPEID[RST$A_TYPCOMPLST];
;  6760          6865  8                    SYMID = .COMPLIST[0];
;  6761          6866  8                    DSTPTR = .SYMID[RST$L_DSTPTR];
;  6762          6867  8                    NAME = DBG$GET_DST_NAME(.DSTPTR);
;  6763          6868  8                    IF .NAME[0] EQL 5
;  6764          6869  8                    THEN
;  6765          6870  9                        BEGIN
;  6766          6871  9                        IF CH$EQL(5, NAME[1],
;  6767          6872  9                                  5, UPLIT (%ASCII 'FALSE'))
;  6768          6873  9                        THEN
;  6769          6874 10                            BEGIN
;  6770          6875 10                            SYMID = .COMPLIST[1];
;  6771          6876 10                            DSTPTR = .SYMID[RST$L_DSTPTR];
;  6772          6877 10                            NAME = DBG$GET_DST_NAME(.DSTPTR);
;  6773          6878 10                            IF .NAME[0] EQL 4
;  6774          6879 10                            THEN
;  6775          6880 11                                BEGIN
;  6776          6881 11                                IF CH$EQL(4, NAME[1],
;  6777          6882 11                                          4, UPLIT (%ASCII 'TRUE'))
;  6778          6883 11                                THEN
;  6779          6884 12                                    BEGIN
;  6780          6885 12
;  6781          6886 12                                    ! We have finally decided that it
;  6782          6887 12                                    ! is OK to change the dtype to TF.
;  6783          6888 12                                    !
;  6784          6889 12                                    LEFT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
;  6785          6890 12                                    LEFT_ARG[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  6786          6891 12                                    LEFT_ARG[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_TF;
;  6787          6892 12                                    LEFT_ARG[DBG$W_VALUE_LENGTH] = 1;
;  6788          6893 11                                    END;
;  6789          6894 10                                END;
;  6790          6895  9                            END;
;  6791          6896  8                        END;
;  6792          6897  7                    END;
;  6793          6898  6                END;
;  6794          6899  5            END;
;  6795          6900  4        END;
;  6796          6901  3    END;
;  6797          6902  2    END;
;  6798          6903  2
;  6799          6904  2
;  6800          6905  2    ! Consult the Type Mapping Table to see if we need to change the dtype
;  6801          6906  2    ! of either the left or the right operand.
;  6802          6907  2    !
;  6803          6908  2    LEFT_TYPE = DBG$GET_DTYPE (.LEFT_ARG);
;  6804          6909  2    IF .MAP_TBL NEQ TABLEBASE
;  6805          6910  2    THEN
;  6806          6911  3        BEGIN
;  6807          6912  3        INCR I FROM 0 TO .MAP_TBL_SIZE - 1 DO
```

```
;  6808          6913  4              BEGIN
;  6809          6914  4              MAP_TBL_ENTRY = .MAP_TBL[.I];
;  6810          6915  4              IF .MAP_TBL_ENTRY EQL 0 THEN EXITLOOP;
;  6811          6916  4              IF .MAP_TBL_ENTRY[TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE
;  6812          6917  4              THEN
;  6813          6918  5                  BEGIN
;  6814          6919  5
;  6815          6920  5                  ! We have found a mapping. Change all the information
;  6816          6921  5                  ! in the descriptor accordingly.
;  6817          6922  5                  !
;  6818          6923  5                  LEFT_TYPE = .MAP_TBL_ENTRY[TYPE_GRAPH$B_HIGHER_TYPE];
;  6819          6924  5
;  6820          6925  5                  ! If we are mapping to a non-vax_standard type, we need
;  6821          6926  5                  ! to change the fcode.
;  6822          6927  5                  !
;  6823          6928  5                  IF .LEFT_TYPE GTR DBG$K_MAXIMUM_DTYPE
;  6824          6929  5                  THEN
;  6825          6930  6                      BEGIN
;  6826          6931  6                      LEFT_ARG[DBG$B_DHDR_FCODE] = .LEFT_TYPE - DBG$K_MAXIMUM_DTYPE;
;  6827          6932  6                      LEFT_ARG[DBG$B_VALUE_CLASS] = DSC$K_CLASS_Z;
;  6828          6933  6                      LEFT_ARG[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_Z;
;  6829          6934  6                      END
;  6830          6935  5                  ELSE
;  6831          6936  6                      BEGIN
;  6832          6937  6                      LEFT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
;  6833          6938  6                      LEFT_ARG[DBG$B_VALUE_CLASS]  = DBG$MAP_DTYPE_CLASS(
;  6834          6939  6                          .LEFT_TYPE,
;  6835          6940  7                          (IF .LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  6836          6941  6                           THEN TRUE ELSE FALSE));
;  6837          6942  6                      LEFT_ARG[DBG$B_VALUE_DTYPE]  = .LEFT_TYPE;
;  6838          6943  5                      END;
;  6839          6944  4                  END;
;  6840          6945  3              END;                        ! End of INCR.
;  6841          6946  2          END;
;  6842          6947  1      END;        ! End of DBG$DO_MAPPING


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

                                                    05C57       .BLKB   1
00  4E  41  45  4C  4F  4F  42                         05C58 P.ALO:  .ASCII  \BOOLEAN\<0>                                  ;
00  00  00  45  53  4C  41  46                         05C60 P.ALP:  .ASCII  \FALSE\<0><0><0>                              ;
                45  55  52  54                          05C68 P.ALQ:  .ASCII  \TRUE\                                        ;


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

                                                0FFC 00000      .ENTRY  DBG$DO_MAPPING, Save R2,R3,R4,R5,R6,R7,R8,-         ; 6800
                                                                        R9,R10,R11                                          ;
                5B 00000000'   EF   9E 00002                    MOVAB   MAP_TBL, R11                                    ;
                5A 00000000G   00   9E 00009                    MOVAB   DBG$GET_DST_NAME, R10                           ;
                59 00000000'   EF   9E 00010                    MOVAB   P.ALO, R9                                       ;
                09 00000000G   00   91 00017                    CMPB    DBG$GB_LANGUAGE, #9                             ; 6835
                               75   12 0001E                    BNEQ    1$                                              ;
                57       04    AC   D0 00020                    MOVL    LEFT_ARG, R7                                    ; 6838
```

```
                                04        06    A7  91 00024          CMPB    6(R7), #4                                       ;
                                                  6B  12 00028          BNEQ    1$                                              ;
                                54        08    A7  D0 0002A          MOVL    8(R7), TYPEID                                   ; 6841
                                                  65  13 0002E          BEQL    1$                                              ; 6842
                                58        0C    A4  D0 00030          MOVL    12(TYPEID), DSTPTR                              ; 6845
                                                  58  DD 00034          PUSHL   DSTPTR                                          ; 6846
                                6A                01  FB 00036          CALLS   #1, DBG$GET_DST_NAME                            ;
                                56                50  D0 00039          MOVL    R0, NAME                                        ;
                                07                66  91 0003C          CMPB    (NAME), #7                                      ; 6847
                                                  54  12 0003F          BNEQ    1$                                              ;
                          69    01        A6        07  29 00041          CMPC3   #7, 1(NAME), P.ALO                              ; 6850
                                                  4D  12 00046          BNEQ    1$                                              ;
                                02        28    A4  D1 00048          CMPL    40(TYPEID), #2                                  ; 6861
                                                  47  12 0004C          BNEQ    1$                                              ;
                                54                2C  C0 0004E          ADDL2   #44, COMPLIST                                   ; 6864
                                55                64  D0 00051          MOVL    (COMPLIST), SYMID                               ; 6865
                                58        0C    A5  D0 00054          MOVL    12(SYMID), DSTPTR                               ; 6866
                                                  58  DD 00058          PUSHL   DSTPTR                                          ; 6867
                                6A                01  FB 0005A          CALLS   #1, DBG$GET_DST_NAME                            ;
                                56                50  D0 0005D          MOVL    R0, NAME                                        ;
                                05                66  91 00060          CMPB    (NAME), #5                                      ; 6868
                                                  30  12 00063          BNEQ    1$                                              ;
                08  A9    01        A6        05  29 00065          CMPC3   #5, 1(NAME), P.ALP                              ; 6871
                                                  28  12 0006B          BNEQ    1$                                              ;
                                55        04    A4  D0 0006D          MOVL    4(COMPLIST), SYMID                              ; 6875
                                58        0C    A5  D0 00071          MOVL    12(SYMID), DSTPTR                               ; 6876
                                                  58  DD 00075          PUSHL   DSTPTR                                          ; 6877
                                6A                01  FB 00077          CALLS   #1, DBG$GET_DST_NAME                            ;
                                56                50  D0 0007A          MOVL    R0, NAME                                        ;
                                04                66  91 0007D          CMPB    (NAME), #4                                      ; 6878
                                                  13  12 00080          BNEQ    1$                                              ;
                          10    A9        01    A6  D1 00082          CMPL    1(NAME), P.ALQ                                  ; 6881
                                                  0C  12 00087          BNEQ    1$                                              ;
                          06    A7                02  90 00089          MOVB    #2, 6(R7)                                       ; 6889
                          14    A7  01280001      8F  D0 0008D          MOVL    #19398657, 20(R7)                               ; 6892
                                        04        AC  DD 00095  1$:     PUSHL   LEFT_ARG                                        ; 6908
                          0000V CF                01  FB 00098          CALLS   #1, DBG$GET_DTYPE                               ;
                                54                50  D0 0009D          MOVL    R0, LEFT_TYPE                                   ;
                                50        A3A8    C9  9E 000A0          MOVAB   TABLEBASE, R0                                   ; 6909
                                50                6B  D1 000A5          CMPL    MAP_TBL, R0                                     ;
                                                  5A  13 000A8          BEQL    7$                                              ;
                                56        04      AB  D0 000AA          MOVL    MAP_TBL_SIZE, R6                                ; 6912
                                53                01  CE 000AE          MNEGL   #1, I                                           ;
                                                  4D  11 000B1          BRB     6$                                              ;
                                55        00 BB43 B0  D0 000B3  2$:     MOVW    @MAP_TBL[I], MAP_TBL_ENTRY                      ; 6914
                                                  4A  13 000B8          BEQL    7$                                              ; 6915
          54        55              08              00  ED 000BA          CMPZV   #0, #8, MAP_TBL_ENTRY, LEFT_TYPE                ; 6916
                                                  3F  12 000BF          BNEQ    6$                                              ;
          54        55              08              08  EF 000C1          EXTZV   #8, #8, MAP_TBL_ENTRY, LEFT_TYPE                ; 6923
                                50        04      AC  D0 000C6          MOVL    LEFT_ARG, R0                                    ; 6931
                                51        04      AC  D0 000CA          MOVL    LEFT_ARG, R1                                    ; 6932
                                52        14      A1  9E 000CE          MOVAB   20(R1), R2                                      ;
                                2B                54  D1 000D2          CMPL    LEFT_TYPE, #43                                  ; 6928
                                                  0A  15 000D5          BLEQ    3$                                              ;
                06  A0              54              2B  83 000D7          SUBB3   #43, LEFT_TYPE, 6(R0)                           ; 6931
                                        02        A2  B4 000DC          CLRW    2(R2)                                           ; 6933
                                                  1F  11 000DF          BRB     6$                                              ; 6928
```

```
                      06   A0          02  90 000E1 3$:    MOVB    #2, 6(R0)                       ; 6937
                           09    03     A2  91 000E5        CMPB    3(R2), #9                       ; 6940
                                        04  12 000E9        BNEQ    4$                              ;
                                        01  DD 000EB        PUSHL   #1                              ;
                                        02  11 000ED        BRB     5$                              ;
                                        7E  D4 000EF 4$:     CLRL    -(SP)                           ;
                                        54  DD 000F1 5$:     PUSHL   LEFT_TYPE                       ; 6939
                    0000V   CF          02  FB 000F3        CALLS   #2, DBG$MAP_DTYPE_CLASS         ;
                      03    A2          50  90 000F8        MOVB    R0, 3(R2)                       ;
                      02    A2          54  90 000FC        MOVB    LEFT_TYPE, 2(R2)                ; 6942
                  AF        53          56  F2 00100 6$:     AOBLSS  R6, T, 2$                       ; 6912
                                            04 00104 7$:     RET                                     ; 6947

; Routine Size:  261 bytes,    Routine Base:  DBG$CODE + 0983
```

```
;  6844          6948  1 GLOBAL ROUTINE DBG$EVALOP_SET_LANGUAGE (LANGUAGE) : NOVALUE =
;  6845          6949  1 !
;  6846          6950  1 ! FUNCTION
;  6847          6951  1 !       This routine gets called during processing of the SET LANGUAGE
;  6848          6952  1 !       command. It sets up the pointer to the Operator Information Table
;  6849          6953  1 !       and Type Conversion Information Table for the language being set.
;  6850          6954  1 !
;  6851          6955  1 ! INPUTS
;  6852          6956  1 !       LANGUAGE        - The language code for the language being set
;  6853          6957  1 !
;  6854          6958  1 ! OUTPUTS
;  6855          6959  1 !       NONE
;  6856          6960  1 !
;  6857          6961  2     BEGIN
;  6858          6962  2
;  6859          6963  2
;  6860          6964  2     ! Case on the language, and set the own variable OPINFO_TABLE and
;  6861          6965  2     ! CVTINFO_TABLE to point to the Operator Information Table and Type
;  6862          6966  2     ! Conversion Table for that language.
;  6863          6967  2     !
;  6864          6968  2     CASE .LANGUAGE FROM DBG$K_MIN_LANGUAGE TO DBG$K_MAX_LANGUAGE OF
;  6865          6969  2         SET
;  6866          6970  2         [DBG$K_ADA]:
;  6867          6971  3                 BEGIN
;  6868          6972  3                 OPINFO_TABLE = ADA_OPINFO_TABLE;
;  6869          6973  3                 CVTINFO_TABLE = ADA_CVTINFO_TABLE;
;  6870          6974  2                 END;
;  6871          6975  2
;  6872          6976  2         [DBG$K_BASIC]:
;  6873          6977  3                 BEGIN
;  6874          6978  3                 OPINFO_TABLE = BASIC_OPINFO_TABLE;
;  6875          6979  3                 CVTINFO_TABLE = BASIC_CVTINFO_TABLE;
;  6876          6980  2                 END;
;  6877          6981  2
;  6878          6982  2         [DBG$K_BLISS]:
;  6879          6983  3                 BEGIN
;  6880          6984  3                 OPINFO_TABLE = BLISS_OPINFO_TABLE;
;  6881          6985  3                 CVTINFO_TABLE = BLISS_CVTINFO_TABLE;
;  6882          6986  2                 END;
;  6883          6987  2
;  6884          6988  2         [DBG$K_C]:
;  6885          6989  3                 BEGIN
;  6886          6990  3                 OPINFO_TABLE = C_OPINFO_TABLE;
;  6887          6991  3                 CVTINFO_TABLE = C_CVTINFO_TABLE;
;  6888          6992  2                 END;
;  6889          6993  2
;  6890          6994  2         [DBG$K_COBOL]:
;  6891          6995  3                 BEGIN
;  6892          6996  3                 OPINFO_TABLE = COBOL_OPINFO_TABLE;
;  6893          6997  3                 CVTINFO_TABLE = COBOL_CVTINFO_TABLE;
;  6894          6998  2                 END;
;  6895          6999  2
;  6896          7000  2         [DBG$K_FORTRAN]:
;  6897          7001  3                 BEGIN
;  6898          7002  3                 OPINFO_TABLE = FORTRAN_OPINFO_TABLE;
;  6899          7003  3                 CVTINFO_TABLE = FORTRAN_CVTINFO_TABLE;
;  6900          7004  2                 END;
```

```
;  6901          7005  2
;  6902          7006  2          [DBG$K_MACRO]:
;  6903          7007  3                  BEGIN
;  6904          7008  3                  OPINFO_TABLE = MACRO_OPINFO_TABLE;
;  6905          7009  3                  CVTINFO_TABLE = MACRO_CVTINFO_TABLE;
;  6906          7010  2                  END;
;  6907          7011  2
;  6908          7012  2          [DBG$K_PASCAL]:
;  6909          7013  3                  BEGIN
;  6910          7014  3                  OPINFO_TABLE = PASCAL_OPINFO_TABLE;
;  6911          7015  3                  CVTINFO_TABLE = PASCAL_CVTINFO_TABLE;
;  6912          7016  2                  END;
;  6913          7017  2
;  6914          7018  2          [DBG$K_PLI]:
;  6915          7019  3                  BEGIN
;  6916          7020  3                  OPINFO_TABLE = PLI_OPINFO_TABLE;
;  6917          7021  3                  CVTINFO_TABLE = PLI_CVTINFO_TABLE;
;  6918          7022  2                  END;
;  6919          7023  2
;  6920          7024  2          [DBG$K_RPG]:
;  6921          7025  3                  BEGIN
;  6922          7026  3                  OPINFO_TABLE = RPG_OPINFO_TABLE;
;  6923          7027  3                  CVTINFO_TABLE = RPG_CVTINFO_TABLE;
;  6924          7028  2                  END;
;  6925          7029  2
;  6926          7030  2          [DBG$K_UNKNOWN]:
;  6927          7031  3                  BEGIN
;  6928          7032  3                  OPINFO_TABLE = UNKNOWN_OPINFO_TABLE;
;  6929          7033  3                  CVTINFO_TABLE = UNKNOWN_CVTINFO_TABLE;
;  6930          7034  2                  END;
;  6931          7035  2
;  6932          7036  2          [INRANGE]:
;  6933          7037  2                  $DBG_ERROR ('DBGEVALOP\DBG$EVALOP_SET_LANGUAGE');
;  6934          7038  2
;  6935          7039  2          [OUTRANGE]:
;  6936          7040  3                  BEGIN
;  6937          7041  3                  OPINFO_TABLE = UNKNOWN_OPINFO_TABLE;
;  6938          7042  3                  CVTINFO_TABLE = UNKNOWN_CVTINFO_TABLE;
;  6939          7043  2                  END;
;  6940          7044  2
;  6941          7045  2          TES;
;  6942          7046  2
;  6943          7047  2
;  6944          7048  2      ! Load in Type Mapping Table from Type Conversion Information Table
;  6945          7049  2      ! for the current language.
;  6946          7050  2      !
;  6947          7051  2      MAP_TBL = .CVTINFO_TABLE[CVTINFO$L_MAPTBL] + TABLEBASE;
;  6948          7052  2      IF .MAP_TBL NEQ TABLEBASE THEN MAP_TBL_SIZE = .(.MAP_TBL - 4) * 2;
;  6949          7053  2
;  6950          7054  2
;  6951          7055  2      ! Load in Type Conversion Table from Type Conversion Information Table
;  6952          7056  2      ! for the current language.
;  6953          7057  2      !
;  6954          7058  2      CVT_TBL = .CVTINFO_TABLE[CVTINFO$L_CVTTBL] + TABLEBASE;
;  6955          7059  2      IF .CVT_TBL NEQ TABLEBASE THEN CVT_TBL_SIZE = .(.CVT_TBL - 4) * 1;
;  6956          7060  2
;  6957          7061  2
```

```
      2      ! Set up the conversion rounding flag.
      2      !
      2      CVT_ROUND_FLAG = .CVTINFO_TABLE[CVTINFO$V_ROUND];
      2
      2      RETURN;
      1      END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 21 05C6C P.ALR:  .ASCII  \!DBGEVALOP\<92>\DBG$EVALOP_SET_LANGUAGE\   ;
47 4E 41 4C 5F 54 45 53 5F 50 4F 4C 41 56 45 05C7B                                                        ;
                                      45 47 41 55 05C8A                                                        ;


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

                                          001C 00000         .ENTRY  DBG$EVALOP_SET_LANGUAGE, Save R2,R3,R4    ; 6948
                     54 00000000'   EF   9E 00002         MOVAB   TABLEBASE, R4                             ;
                     53 00000000'   EF   9E 00009         MOVAB   CVTINFO_TABLE, R3                         ;
              0A     00       04    AC   CF 00010         CASEL   LANGUAGE, #0, #10                         ; 6968
   004C       0032   0059           0066    00015 1$:     .WORD   8$-1$,-                                   ;
   003F       0073   0080           0025    0001D                 7$-1$,-                                   ;
              009A   0019           008D    00025                 4$-1$,-                                   ;
                                                                  6$-1$,-                                   ;
                                                                  3$-1$,-                                   ;
                                                                  10$-1$,-                                  ;
                                                                  9$-1$,-                                   ;
                                                                  5$-1$,-                                   ;
                                                                  11$-1$,-                                  ;
                                                                  2$-1$,-                                   ;
                                                                  13$-1$                                    ;
                                    0081   31 0002B         BRW     13$                                       ; 7041
                      10  A3   0430   C4   9E 0002E 2$:     MOVAB   ADA_OPINFO_TABLE, OPINFO_TABLE            ; 6972
                          63     04   A4   9E 00034         MOVAB   ADA_CVTINFO_TABLE, CVTINFO_TABLE          ; 6973
                                      73   11 00038         BRB     12$                                       ; 6968
                      10  A3   0D5C   C4   9E 0003A 3$:     MOVAB   BASIC_OPINFO_TABLE, OPINFO_TABLE          ; 6978
                          63   0870   C4   9E 00040         MOVAB   BASIC_CVTINFO_TABLE, CVTINFO_TABLE        ; 6979
                                      73   11 00045         BRB     14$                                       ; 6968
                      10  A3   13E0   C4   9E 00047 4$:     MOVAB   BLISS_OPINFO_TABLE, OPINFO_TABLE          ; 6984
                          63   11A4   C4   9E 0004D         MOVAB   BLISS_CVTINFO_TABLE, CVTINFO_TABLE        ; 6985
                                      66   11 00052         BRB     14$                                       ; 6968
                      10  A3   1CF8   C4   9E 00054 5$:     MOVAB   C_OPINFO_TABLE, OPINFO_TABLE              ; 6990
                          63   1820   C4   9E 0005A         MOVAB   C_CVTINFO_TABLE, CVTINFO_TABLE            ; 6991
                                      59   11 0005F         BRB     14$                                       ; 6968
                      10  A3   23A8   C4   9E 00061 6$:     MOVAB   COBOL_OPINFO_TABLE, OPINFO_TABLE          ; 6996
                          63   2144   C4   9E 00067         MOVAB   COBOL_CVTINFO_TABLE, CVTINFO_TABLE        ; 6997
                                      4C   11 0006C         BRB     14$                                       ; 6968
                      10  A3   2D8C   C4   9E 0006E 7$:     MOVAB   FORTRAN_OPINFO_TABLE, OPINFO_TABLE        ; 7002
                          63   27F4   C4   9E 00074         MOVAB   FORTRAN_CVTINFO_TABLE, CVTINFO_TABLE      ; 7003
                                      3F   11 00079         BRB     14$                                       ; 6968
                      10  A3   3414   C4   9E 0007B 8$:     MOVAB   MACRO_OPINFO_TABLE, OPINFO_TABLE          ; 7008
                          63   31D4   C4   9E 00081         MOVAB   MACRO_CVTINFO_TABLE, CVTINFO_TABLE        ; 7009
                                      32   11 00086         BRB     14$                                       ; 6968
                      10  A3   3D10   C4   9E 00088 9$:     MOVAB   PASCAL_OPINFO_TABLE, OPINFO_TABLE         ; 7014
```

```
                                63      385C    C4  9E 0008E           MOVAB   PASCAL_CVTINFO_TABLE, CVTINFO_TABLE      ; 7015
                                        25  11 00093                   BRB     14$                                     ; 6968
                            10  A3      4778    C4  9E 00095 10$:      MOVAB   PLI_OPINFO_TABLE, OPINFO_TABLE          ; 7020
                                63      4194    C4  9E 0009B           MOVAB   PLI_CVTINFO_TABLE, CVTINFO_TABLE        ; 7021
                                        18  11 000A0                   BRB     14$                                     ; 6968
                            10  A3      4D54    C4  9E 000A2 11$:      MOVAB   RPG_OPINFO_TABLE, OPINFO_TABLE          ; 7026
                                63      4BB8    C4  9E 000A8           MOVAB   RPG_CVTINFO_TABLE, CVTINFO_TABLE        ; 7027
                                        0B  11 000AD 12$:              BRB     14$                                     ; 6968
                            10  A3      5634    C4  9E 000AF 13$:      MOVAB   UNKNOWN_OPINFO_TABLE, OPINFO_TABLE      ; 7032
                                63      5194    C4  9E 000B5           MOVAB   UNKNOWN_CVTINFO_TABLE, CVTINFO_TABLE    ; 7033
                                52              63  D0 000BA 14$:      MOVL    CVTINFO_TABLE, R2                       ; 7051
                                50              64  9E 000BD           MOVAB   TABLEBASE, R0                           ;
                        04  A3  50              62  C1 000C0           ADDL3   (R2), R0, MAP_TBL                       ;
                                50          04  A3  D0 000C5           MOVL    MAP_TBL, R0                             ; 7052
                                51              64  9E 000C9           MOVAB   TABLEBASE, R1                           ;
                                51              50  D1 000CC           CMPL    R0, R1                                  ;
                                        06  13 000CF                   BEQL    15$                                     ;
                        08  A3  FC      A0          01  78 000D1       ASHL    #1, -4(R0), MAP_TBL_SIZE                ;
                                50              64  9E 000D7 15$:      MOVAB   TABLEBASE, R0                           ; 7058
                            F8  A3          04 B240 9E 000DA           MOVAB   @4(R2)[R0], CVT_TBL                     ;
                                50          F8  A3  D0 000E0           MOVL    CVT_TBL, R0                             ; 7059
                                51              64  9E 000E4           MOVAB   TABLEBASE, R1                           ;
                                51              50  D1 000E7           CMPL    R0, R1                                  ;
                                        05  13 000EA                   BEQL    16$                                     ;
                            FC  A3      FC      A0  D0 000EC           MOVL    -4(R0), CVT_TBL_SIZE                    ;
                F4  A3      08  A2          01      00  EF 000F1 16$:   EXTZV   #0, #1, 8(R2), CVT_ROUND_FLAG          ; 7064
                                                    04  000F8           RET                                             ; 7067

; Routine Size:  249 bytes,    Routine Base:  DBG$CODE + 0A88
```

```
;  6965          7068  1 GLOBAL ROUTINE DBG$EVAL_LANG_OPERATOR (OPERATOR, LEFT_ARG, RIGHT_ARG) =
;  6966          7069  1 !
;  6967          7070  1 ! FUNCTION
;  6968          7071  1 !       This routine does the evaluation of a language expression operator.
;  6969          7072  1 !
;  6970          7073  1 !       It determines the operand type or types, it determines what type
;  6971          7074  1 !       conversions must be done on the operands (if any), it calls the
;  6972          7075  1 !       type conversion routines to do whatever conversions are needed, it selects
;  6973          7076  1 !       the actual routine to do the operation based on the final operand data
;  6974          7077  1 !       types, and it calls the DBG$PERFORM_OPERATOR routine to actually perform
;  6975          7078  1 !       the selected operation.
;  6976          7079  1 !
;  6977          7080  1 !       This routine uses the Operator Routine Table for this operator to
;  6978          7081  1 !       determine which actual routine will do the operation on operands of
;  6979          7082  1 !       the given data types. If a routine is not found to handle the operation
;  6980          7083  1 !       on the given types, then the Type Hierarchy Table
;  6981          7084  1 !       is consulted to determine what type conversions
;  6982          7085  1 !       may have to be done. If type conversions need to be done,
;  6983          7086  1 !       DBG$TYPE_CONV is called to actually do the conversions.
;  6984          7087  1 !       DBG$PERFORM_OPERATOR is then called with an index specifying the desired
;  6985          7088  1 !       operator routine--that is where the actual operation is finally done.
;  6986          7089  1 !
;  6987          7090  1 !       The routine result Value Descriptor is then returned as the value
;  6988          7091  1 !       of the DBG$EVAL_LANG_OPERATOR call.
;  6989          7092  1 !
;  6990          7093  1 !       A Primary Descriptor may be returned as the routine value as well,
;  6991          7094  1 !       if the computation involved is address-valued operands as in C.
;  6992          7095  1 !
;  6993          7096  1 ! INPUTS
;  6994          7097  1 !       OPERATOR - A pointer to the Operator Lexical Token Entry for the
;  6995          7098  1 !                  operator to be evaluated.
;  6996          7099  1 !
;  6997          7100  1 !       LEFT_ARG - A pointer to the left argument Descriptor.
;  6998          7101  1 !                  If the operator
;  6999          7102  1 !                  is a unary operator, LEFT_ARG points to the operator's one
;  7000          7103  1 !                  argument.
;  7001          7104  1 !
;  7002          7105  1 !       RIGHT_ARG - A pointer to the right argument Descriptor.
;  7003          7106  1 !                  If the operator
;  7004          7107  1 !                  is a unary operator, RIGHT_ARG must be zero.
;  7005          7108  1 !
;  7006          7109  1 ! OUTPUTS
;  7007          7110  1 !       A pointer to the Value Descriptor which results from the evaluation of
;  7008          7111  1 !                  the operator is returned as this routine's result.  Or,
;  7009          7112  1 !       A pointer to the Primary Descritpor.
;  7010          7113  1 !
;  7011          7114  1
;  7012          7115  2     BEGIN
;  7013          7116  2
;  7014          7117  2     MAP
;  7015          7118  2         OPERATOR: REF TOKEN$ENTRY,      ! Token Entry for operator to perform
;  7016          7119  2         LEFT_ARG: REF DBG$VALDESC,      ! Left operand Token Entry
;  7017          7120  2         RIGHT_ARG: REF DBG$VALDESC;     ! Right operand Token Entry
;  7018          7121  2
;  7019          7122  2     LOCAL
;  7020          7123  2         HIER_TBL: REF VECTOR [,WORD],   ! Pointer to a Type Hierarchy Table
;  7021          7124  2         HIER_TBL_SIZE,                  ! Number of entries in HIER_TBL
```

```
;  7022          7125  2          INCOMP_TBL: REF VECTOR [,WORD],  ! Pointer to a Type Incompatibility Table
;  7023          7126  2          INCOMP_TBL_SIZE,                 ! Number of entries in INCOMP_TBL
;  7024          7127  2          ROUT_TBL: REF ORT$TABLE,         ! Pointer to an Operator Routine Table
;  7025          7128  2          ROUT_TBL_SIZE,                   ! Number of entrie in ROUT_TBL
;  7026          7129  2          BINARY_FLAG: BYTE,               ! A flag set to TRUE for binary operators
;  7027          7130  2          DIGITS,                          ! The number of digits for packed decimal
;  7028          7131  2          LEFT_TYPE,                       ! Dtype for original left operand type
;  7029          7132  2          LENGTH,                          ! Length of the data
;  7030          7133  2          MAP_TBL_ENTRY: TYPE_GRAPH$ENTRY, ! An entry in the Type Mapping Table
;  7031          7134  2          NEW_LEFT_ARG: REF DBG$VALDESC,   ! Pointer to left argument Token Entry
;  7032          7135  2                                           !      after type conversion if needed
;  7033          7136  2          NEW_LEFT_TYPE,                   ! Dtype for converted left operand type
;  7034          7137  2          NEW_RIGHT_ARG: REF DBG$VALDESC,  ! Pointer to right argument Token Entry
;  7035          7138  2                                           !      after type conversion if needed
;  7036          7139  2          NEW_RIGHT_TYPE,                  ! Dtype for converted right operand type
;  7037          7140  2          OPCODE,                          ! Operator code for current operator
;  7038          7141  2          SD_CLASS_FLAG,                   ! Flag to indicate SD class overrides
;  7039          7142  2          RESULT: REF DBG$VALDESC,         ! Pointer to result value descriptor
;  7040          7143  2          RESULT_TYPE,                     ! Dtype of result of operation
;  7041          7144  2          RIGHT_TYPE,                      ! Dtype for original right operand type
;  7042          7145  2          ROUT_TBL_INDEX,                  ! Index into Operator Routine Table
;  7043          7146  2          ROUT_INDEX,                      ! Routine CASE index for code that does
;  7044          7147  2                                           !      operation on operand types
;  7045          7148  2          SYMID: REF RST$ENTRY,            ! Symid in primary descriptor
;  7046          7149  2          TYPEID,                          ! Pointer to a typeid
;  7047          7150  2          TYPEID_INDEX,                    ! Routine CASE index for code that does
;  7048          7151  2                                           !      typeid check
;  7049          7152  2          TYPES: TYPE$PAIR;                ! Type indecies for original operand
;  7050          7153  2                                           !      types in Type Conv Table format
;  7051          7154  2
;  7052          7155  2
;  7053          7156  2      ! Trap out one special case.  If the incoming token is Unary minus or
;  7054          7157  2      ! Unary plus, and the unconverted bit is on, We know we have a
;  7055          7158  2      ! constant, say like -1.  We do not want to turn this -1 into negative
;  7056          7159  2      ! integer right away, for those languages have packed decimal we are
;  7057          7160  2      ! not sure at this point we have an integer or packed decimal number.
;  7058          7161  2      ! We make this decision at the evaluation time or depositting time.
;  7059          7162  2      ! So if we have one of this token, we mark it, then simply returns.
;  7060          7163  2      !
;  7061          7164  2      IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_PLUS OR
;  7062          7165  2         .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_MINUS
;  7063          7166  2      THEN
;  7064          7167  3          BEGIN
;  7065          7168  3          IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7066          7169  3          THEN
;  7067          7170  4              BEGIN
;  7068          7171  4
;  7069          7172  4
;  7070          7173  4              ! In this case, there should not be any right argument.
;  7071          7174  4              !
;  7072          7175  4              IF .RIGHT_ARG NEQ 0
;  7073          7176  4              THEN
;  7074          7177  4                  $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 1');
;  7075          7178  4
;  7076          7179  4
;  7077          7180  4              ! In this case, the left arg. should not be primary or volatile
;  7078          7181  4              ! value descriptor.
```

```
;  7079          7182  4          !
;  7080          7183  4          IF (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
;  7081          7184  5             (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC)
;  7082          7185  4          THEN
;  7083          7186  4              $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 2');
;  7084          7187  4
;  7085          7188  4
;  7086          7189  4          ! Mark the sign and return.  Check to see if it is already marked,
;  7087          7190  4          ! if it is, flip the sign, for we may have -(-1) case, this is
;  7088          7191  4          ! the same as +1.
;  7089          7192  4          !
;  7090          7193  4          IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_MINUS
;  7091          7194  4          THEN
;  7092          7195  5              BEGIN
;  7093          7196  5              IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
;  7094          7197  5              THEN
;  7095          7198  5                  LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST
;  7096          7199  5              ELSE
;  7097          7200  5                  LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_NEGCONST;
;  7098          7201  5              END
;  7099          7202  5
;  7100          7203  4          ELSE
;  7101          7204  4              IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_UNARY_PLUS
;  7102          7205  4              THEN
;  7103          7206  5                  BEGIN
;  7104          7207  5                  IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
;  7105          7208  5                  THEN
;  7106          7209  5                      LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_NEGCONST
;  7107          7210  5                  ELSE
;  7108          7211  5                      LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST;
;  7109          7212  5                  END
;  7110          7213  5
;  7111          7214  4                  ELSE
;  7112          7215  4                      $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, inconsistency 3');
;  7113          7216  4
;  7114          7217  4              RETURN .LEFT_ARG;
;  7115          7218  3              END;
;  7116          7219  2          END;
;  7117          7220  2
;  7118          7221  2
;  7119          7222  2      ! This is called from DBG$EVAL_LANG_OPEARTOR with TOKEN$K_NEGCONST or
;  7120          7223  2      ! TOKEN$K_POSCONST token, replaced it to TOKEN$K_NEG_SIGN or
;  7121          7224  2      ! TOKEN$K_POS_SIGN, so the normal unary minus or unary plus operation
;  7122          7225  2      ! can take place.
;  7123          7226  2      !
;  7124          7227  2      IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_NEGCONST
;  7125          7228  2      THEN
;  7126          7229  2          OPERATOR = DBG$GL_NEG_SIGN_TOKEN;
;  7127          7230  2
;  7128          7231  2      IF .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_POSCONST
;  7129          7232  2      THEN
;  7130          7233  2          OPERATOR = DBG$GL_POS_SIGN_TOKEN;
;  7131          7234  2
;  7132          7235  2
;  7133          7236  2      ! Initialize table pointers and flags.
;  7134          7237  2      !
;  7135          7238  2      MAX_DEPTH = 1;
```

```
;  7136          7239  2     IF .BLISS_BITSELECTION_FLAG1
;  7137          7240  2     THEN
;  7138          7241  2         BLISS_BITSELECTION_FLAG2 = TRUE
;  7139          7242  2     ELSE
;  7140          7243  2         BLISS_BITSELECTION_FLAG2 = FALSE;
;  7141          7244  2     BLISS_BITSELECTION_FLAG1 = FALSE;
;  7142          7245  2     BLISS_INDIRECTION_FLAG = FALSE;
;  7143          7246  2     OPCODE = .OPERATOR[TOKEN$W_CODE];
;  7144          7247  2     DBG$GL_OPCODE_NAME = OPERATOR[TOKEN$B_OPLEN];
;  7145          7248  2     BINARY_FLAG = .OPERATOR[TOKEN$B_KIND] EQL TOKEN$K_INFIX_OP;
;  7146          7249  2     ROUT_TBL = .OPINFO_TABLE[.OPCODE, OPINFO$L_ROUTTBL] + TABLEBASE;
;  7147          7250  2     IF .ROUT_TBL EQL TABLEBASE
;  7148          7251  2     THEN
;  7149          7252  3         BEGIN
;  7150          7253  3         IF .OPCODE NEQ TOKEN$K_DEPOSIT AND .OPCODE NEQ TOKEN$K_CONVERT
;  7151          7254  3         AND .OPCODE NEQ TOKEN$K_IDENTITY
;  7152          7255  3         THEN
;  7153          7256  3             $DBG_ERROR ('DBGEVALOP\DBG$EVAL_LANG_OPERATOR routine table missing');
;  7154          7257  3         END
;  7155          7258  2     ELSE
;  7156          7259  2         ROUT_TBL_SIZE = .(.ROUT_TBL - 4) / 2;
;  7157          7260  2     HIER_TBL = .OPINFO_TABLE[.OPCODE, OPINFO$L_HIERTBL] + TABLEBASE;
;  7158          7261  2     IF .HIER_TBL NEQ TABLEBASE
;  7159          7262  2     THEN
;  7160          7263  3         BEGIN
;  7161          7264  3         HIER_TBL_SIZE = .(.HIER_TBL - 4) * 2;
;  7162          7265  3         MAX_DEPTH = .MAX_DEPTH + .HIER_TBL_SIZE * 2;
;  7163          7266  2         END;
;  7164          7267  2     INCOMP_TBL = .OPINFO_TABLE[.OPCODE, OPINFO$L_INCOMPTBL] + TABLEBASE;
;  7165          7268  2     IF .INCOMP_TBL NEQ TABLEBASE THEN INCOMP_TBL_SIZE = .(.INCOMP_TBL - 4) * 2;
;  7166          7269  2
;  7167          7270  2
;  7168          7271  2     ! Convert primary descriptors or volatile value descriptors
;  7169          7272  2     ! to value descriptors. Something should be done to
;  7170          7273  2     ! clean up the structure of the code below. The basic idea of what
;  7171          7274  2     ! it is doing is:
;  7172          7275  2     ! If we get a Primary for "X", say, or a volatile value descriptor for
;  7173          7276  2     ! address "200" in the user program, then:
;  7174          7277  2     ! - If the language does an implicit fetch on operands (i.e., all languages
;  7175          7278  2     !   except BLISS and MACRO), then we do the fetch here by calling PRIM_TO_VAL.
;  7176          7279  2     ! - For BLISS and MACRO we call PRIM_TO_ADDR in order to get the address
;  7177          7280  2     !   of the operand into a value descriptor.
;  7178          7281  2     ! - Special case for arrays in C.
;  7179          7282  2     ! - Special case for records in COBOL.
;  7180          7283  2     ! (It's these special cases that make the code such a mess.)
;  7181          7284  2     !
;  7182          7285  2     IF (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
;  7183          7286  3        (.LEFT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC)
;  7184          7287  2     THEN
;  7185          7288  2
;  7186          7289  2         ! Most operators in most languages do an implicit fetch on primaries.
;  7187          7290  2         ! E.g., in FORTRAN, if we say EV X we want to fetch the value at X.
;  7188          7291  2         ! DBG$PRIM_TO_VAL is the routine that converts primaries to value
;  7189          7292  2         ! descriptors, and does the fetch in the process.
;  7190          7293  2         ! The exceptions are: in BLISS, we only do the fetch when we see
;  7191          7294  2         ! the fetch operator (.). Also, for the "address of" operator
;  7192          7295  2         ! (& in C), we want to suppress this fetch. For these cases,
```

```
;7193      7296  2          ! we use a routine DBG$PRIM_TO_ADDR which converts a primary
;7194      7297  2          ! descriptor to a value descriptor containing the address of the
;7195      7298  2          ! primary.
;7196      7299  2          !
;7197      7300  2          ! We consult a flag in the Operator Information Table that tells
;7198      7301  2          ! us whether or not to do this fetch.
;7199      7302  2          !
;7200      7303  2          IF .OPINFO_TABLE [.OPCODE, OPINFO$V_FETCH]
;7201      7304  2          THEN
;7202      7305  3              BEGIN
;7203      7306  3              BLISS_INDIRECTION_FLAG = TRUE;
;7204      7307  3
;7205      7308  3              ! Special case for arrays in C. We want to treat un-subscripted
;7206      7309  3              ! arrays as pointers, and be able to add, subtract, and do
;7207      7310  3              ! dereferencing on the addresses.
;7208      7311  3              !
;7209      7312  4              IF  (.DBG$GB_LANGUAGE EQL DBG$K_C)
;7210      7313  4              AND (.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ARRAY)
;7211      7314  3              THEN
;7212      7315  4                  BEGIN
;7213      7316  4
;7214      7317  4                  ! Save the typeid.
;7215      7318  4                  !
;7216      7319  4                  TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];
;7217      7320  4
;7218      7321  4                  ! Build a descriptor with the address of the array.
;7219      7322  4                  !
;7220      7323  4                  IF NOT DBG$PRIM_TO_ADDR(.LEFT_ARG, DSC$K_DTYPE_L, LEFT_ARG)
;7221      7324  4                  THEN
;7222      7325  4                      $DBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');
;7223      7326  4
;7224      7327  4                  ! Dummy in the FCODE as TPTR. Also fill in the typeid.
;7225      7328  4                  !
;7226      7329  4                  LEFT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_TPTR;
;7227      7330  4                  LEFT_ARG[DBG$L_DHDR_TYPEID] = .TYPEID;
;7228      7331  4                  END
;7229      7332  4
;7230      7333  3              ELSE
;7231      7334  4                  BEGIN
;7232      7335  4
;7233      7336  4                  ! If we have an aggregate, first try calling the
;7234      7337  4                  ! routine that turns a 1-dimensional array of characters
;7235      7338  4                  ! into a string.
;7236      7339  4                  !
;7237      7340  4                  IF .LEFT_ARG[DBG$V_DHDR_AGGR]
;7238      7341  4                  THEN
;7239      7342  4                      DBG$COLLECT(.LEFT_ARG);
;7240      7343  4
;7241      7344  4                  ! If we still have an aggregate, we check for the special
;7242      7345  4                  ! case of COBOL records. For all other cases, we signal
;7243      7346  4                  ! an error, because operations on aggregates are not
;7244      7347  4                  ! supported.
;7245      7348  4                  !
;7246      7349  4                  SYMID = 0;
;7247      7350  4                  IF .LEFT_ARG[DBG$V_DHDR_AGGR]
;7248      7351  4                  THEN
;7249      7352  5                      BEGIN
```

```
; 7250          7353  5                 IF .LEFT_ARG[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_RECORD
; 7251          7354  5                 THEN
; 7252          7355  5                     SIGNAL(DBG$_NOVALUE);
; 7253          7356  5
; 7254          7357  5
; 7255          7358  5                 ! This is record aggregate, if the language code is cobol, treated as text string.
; 7256          7359  5                 !
; 7257          7360  5                 SYMID = .LEFT_ARG[DBG$L_DHDR_SYMID0];
; 7258          7361  5                 IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
; 7259          7362  5                 WHILE .SYMID[RST$B_KIND] NEQ RST$K_MODULE DO
; 7260          7363  6                     BEGIN
; 7261          7364  6                     SYMID = .SYMID[RST$L_UPSCOPEPTR];
; 7262          7365  6                     IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
; 7263          7366  5                     END;
; 7264          7367  5                 IF .SYMID[RST$B_LANGUAGE] NEQ DBG$K_COBOL
; 7265          7368  5                 THEN
; 7266          7369  5                     SIGNAL(DBG$_NOVALUE);
; 7267          7370  4                 END;
; 7268          7371  4
; 7269          7372  4             IF NOT DBG$PRIM_TO_VAL(.LEFT_ARG, DBG$K_VALUE_DESC, LEFT_ARG)
; 7270          7373  4             THEN
; 7271          7374  4                 $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to val failed');
; 7272          7375  4
; 7273          7376  3             END;
; 7274          7377  3         END
; 7275          7378  3
; 7276          7379  2     ELSE
; 7277          7380  2         IF NOT DBG$PRIM_TO_ADDR(.LEFT_ARG, DSC$K_DTYPE_L, LEFT_ARG)
; 7278          7381  2         THEN
; 7279          7382  2             $DBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');
; 7280          7383  2
; 7281          7384  2
; 7282          7385  2     ! If this is a binary operator, do the same for the right operand.
; 7283          7386  2     !
; 7284          7387  2     IF .BINARY_FLAG
; 7285          7388  2     THEN
; 7286          7389  2         IF (.RIGHT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_PRIMARY_DESC) OR
; 7287          7390  3            ((.RIGHT_ARG[DBG$B_DHDR_TYPE] EQL DBG$K_V_VALUE_DESC) AND
; 7288          7391  3             (.OPCODE NEQ TOKEN$K_DEPOSIT) AND (.OPCODE NEQ TOKEN$K_CONVERT))
; 7289          7392  2         THEN
; 7290          7393  2             IF .OPINFO_TABLE [.OPCODE, OPINFO$V_FETCH]
; 7291          7394  2             THEN
; 7292          7395  3                 BEGIN
; 7293          7396  3                 BLISS_INDIRECTION_FLAG = TRUE;
; 7294          7397  3
; 7295          7398  3                 ! Special case for arrays in C. We want to treat un-subscripted
; 7296          7399  3                 ! arrays as pointers, and be able to add, subtract, and do
; 7297          7400  3                 ! dereferencing on the addresses.
; 7298          7401  3                 !
; 7299          7402  3                 IF (.DBG$GB_LANGUAGE EQL DBG$K_C) AND
; 7300          7403  4                    (.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ARRAY)
; 7301          7404  3                 THEN
; 7302          7405  4                     BEGIN
; 7303          7406  4
; 7304          7407  4                     ! Save the typeid.
; 7305          7408  4                     !
; 7306          7409  4                     TYPEID = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
```

```
;  7307     7410  4
;  7308     7411  4                    ! Build a descriptor with the address of the array.
;  7309     7412  4                    !
;  7310     7413  4                    IF NOT DBG$PRIM_TO_ADDR(.RIGHT_ARG, DSC$K_DTYPE_L, RIGHT_ARG)
;  7311     7414  4                    THEN
;  7312     7415  4                        $DBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');
;  7313     7416  4
;  7314     7417  4                    ! Dummy in the FCODE as TPTR. Also fill in the typeid.
;  7315     7418  4                    !
;  7316     7419  4                    RIGHT_ARG[DBG$B_DHDR_FCODE] = RST$K_TYPE_TPTR;
;  7317     7420  4                    RIGHT_ARG[DBG$L_DHDR_TYPEID] = .TYPEID;
;  7318     7421  4                    END
;  7319     7422  4
;  7320     7423  3                ELSE
;  7321     7424  4                    BEGIN
;  7322     7425  4
;  7323     7426  4
;  7324     7427  4                    ! If we have an aggregate, first try calling the
;  7325     7428  4                    ! routine that turns a 1-dimensional array of characters
;  7326     7429  4                    ! into a string.
;  7327     7430  4                    !
;  7328     7431  4                    IF .RIGHT_ARG[DBG$V_DHDR_AGGR]
;  7329     7432  4                    THEN
;  7330     7433  4                        DBG$COLLECT(.RIGHT_ARG);
;  7331     7434  4
;  7332     7435  4                    ! If we still have an aggregate, we check for the special
;  7333     7436  4                    ! case of COBOL records. For all other cases, we signal
;  7334     7437  4                    ! an error, because operations on aggregates are not
;  7335     7438  4                    ! supported.
;  7336     7439  4                    !
;  7337     7440  4                    SYMID = 0;
;  7338     7441  4                    IF .RIGHT_ARG[DBG$V_DHDR_AGGR]
;  7339     7442  4                    THEN
;  7340     7443  5                        BEGIN
;  7341     7444  5                        IF .RIGHT_ARG[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_RECORD
;  7342     7445  5                        THEN
;  7343     7446  5                            SIGNAL(DBG$_NOVALUE);
;  7344     7447  5
;  7345     7448  5
;  7346     7449  5                        ! This is record aggregate, if the language code is cobol, treated as text string.
;  7347     7450  5                        !
;  7348     7451  5                        SYMID = .RIGHT_ARG[DBG$L_DHDR_SYMID0];
;  7349     7452  5                        IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
;  7350     7453  5                        WHILE .SYMID[RST$B_KIND] NEQ RST$K_MODULE DO
;  7351     7454  6                            BEGIN
;  7352     7455  6                            SYMID = .SYMID[RST$L_UPSCOPEPTR];
;  7353     7456  6                            IF .SYMID EQL 0 THEN SIGNAL(DBG$_NOVALUE);
;  7354     7457  5                            END;
;  7355     7458  5                        IF .SYMID[RST$B_LANGUAGE] NEQ DBG$K_COBOL
;  7356     7459  5                        THEN
;  7357     7460  5                            SIGNAL(DBG$_NOVALUE);
;  7358     7461  4                        END;
;  7359     7462  4
;  7360     7463  4                    IF NOT DBG$PRIM_TO_VAL(.RIGHT_ARG, DBG$K_VALUE_DESC, RIGHT_ARG)
;  7361     7464  4                    THEN
;  7362     7465  4                        $DBG_ERROR('DBGEVALOP\DBG$EVAL_LANG_OPERATOR, prim to val failed');
;  7363     7466  4
```

```
;  7364          7467  3                    END;
;  7365          7468  3                END
;  7366          7469  2            ELSE
;  7367          7470  2                IF NOT DBG$PRIM_TO_ADDR(.RIGHT_ARG, DSC$K_DTYPE_L, RIGHT_ARG)
;  7368          7471  2                THEN
;  7369          7472  2                    $DBG_ERROR('DBGEVALOP\DBGEVAL_LANG_OPERATOR, prim to addr failed');
;  7370          7473  2
;  7371          7474  2
;  7372          7475  2        ! Take care of DEP rfa data type = constant case.
;  7373          7476  2        !
;  7374          7477  2        IF .OPCODE EQL TOKEN$K_DEPOSIT
;  7375          7478  2        THEN
;  7376          7479  3            BEGIN
;  7377          7480  3            IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7378          7481  3            THEN
;  7379          7482  4                BEGIN
;  7380          7483  4                IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_RFA
;  7381          7484  4                THEN
;  7382          7485  5                    BEGIN
;  7383          7486  5                    LEFT_ARG[DBG$L_DHDR_TYPEID] = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
;  7384          7487  5                    LEFT_ARG = DBG$CONV_TRFA_VALUE(.LEFT_ARG);
;  7385          7488  4                    END;
;  7386          7489  3                END;
;  7387          7490  2            END;
;  7388          7491  2
;  7389          7492  2
;  7390          7493  2        ! At this point we should have value descriptors.  Perform special
;  7391          7494  2        ! NRO Data type map check for RPG.
;  7392          7495  2        !
;  7393          7496  2        IF .DBG$GB_LANGUAGE EQL DBG$K_RPG
;  7394          7497  2        THEN
;  7395          7498  3            BEGIN
;  7396          7499  3            IF .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NRO
;  7397          7500  3            THEN
;  7398          7501  3                LEFT_ARG = MAP_NRO_DTYPE_IN_RPG(.LEFT_ARG);
;  7399          7502  3
;  7400          7503  3            IF .BINARY_FLAG
;  7401          7504  3            THEN
;  7402          7505  4                BEGIN
;  7403          7506  5                IF (.OPCODE NEQ TOKEN$K_DEPOSIT AND
;  7404          7507  5                    .OPCODE NEQ TOKEN$K_CONVERT)
;  7405          7508  4                THEN
;  7406          7509  4                    IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NRO
;  7407          7510  4                    THEN
;  7408          7511  4                        RIGHT_ARG = MAP_NRO_DTYPE_IN_RPG(.RIGHT_ARG);
;  7409          7512  3                END;
;  7410          7513  2            END;
;  7411          7514  2
;  7412          7515  2
;  7413          7516  2        ! Take care of the [] (empty set).
;  7414          7517  2        !
;  7415          7518  2        IF .LEFT_ARG[DBG$B_DHDR_LANG] EQL %X'FF' AND
;  7416          7519  2           .LEFT_ARG[DBG$B_VALUE_CLASS] EQL %X'FF' AND
;  7417          7520  2           .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL %X'FF'
;  7418          7521  2        THEN
;  7419          7522  2            LEFT_ARG = FIXUP_EMPTY_SET(.LEFT_ARG, .RIGHT_ARG);
;  7420          7523  2
```

```
;  7421          7524  2      IF .BINARY_FLAG
;  7422          7525  2      THEN
;  7423          7526  2          IF .RIGHT_ARG[DBG$B_DHDR_LANG] EQL %X'FF' AND
;  7424          7527  2             .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL %X'FF' AND
;  7425          7528  2             .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL %X'FF'
;  7426          7529  2          THEN
;  7427          7530  2              RIGHT_ARG = FIXUP_EMPTY_SET(.RIGHT_ARG, .LEFT_ARG);
;  7428          7531  2
;  7429          7532  2
;  7430          7533  2      ! Perform "type mapping". Do not map left-hand-side of deposit when
;  7431          7534  2      ! override applied.
;  7432          7535  2      !
;  7433          7536  2      DBG$DO_MAPPING(.LEFT_ARG);
;  7434          7537  2      IF .BINARY_FLAG
;  7435          7538  2      THEN
;  7436          7539  2          IF (.OPCODE NEQ TOKEN$K_DEPOSIT) OR
;  7437          7540  3             (NOT .RIGHT_ARG [DBG$V_DHDR_OVERRIDE])
;  7438          7541  2          THEN
;  7439          7542  2              DBG$DO_MAPPING(.RIGHT_ARG);
;  7440          7543  2
;  7441          7544  2
;  7442          7545  2      ! Get a Type for the left operand.
;  7443          7546  2      !
;  7444          7547  2      LEFT_TYPE = DBG$GET_DTYPE (.LEFT_ARG);
;  7445          7548  2      NEW_LEFT_TYPE = .LEFT_TYPE;
;  7446          7549  2
;  7447          7550  2
;  7448          7551  2      ! Get a Type for the right operand.
;  7449          7552  2      !
;  7450          7553  2      IF .BINARY_FLAG THEN
;  7451          7554  3          BEGIN
;  7452          7555  3          RIGHT_TYPE = DBG$GET_DTYPE (.RIGHT_ARG);
;  7453          7556  3          NEW_RIGHT_TYPE = .RIGHT_TYPE;
;  7454          7557  3          END
;  7455          7558  3
;  7456          7559  3      ! Unary operator.
;  7457          7560  3      !
;  7458          7561  2      ELSE
;  7459          7562  3          BEGIN
;  7460          7563  3          RIGHT_TYPE = 0;
;  7461          7564  3          NEW_RIGHT_TYPE = 0;
;  7462          7565  2          END;
;  7463          7566  2
;  7464          7567  2
;  7465          7568  2      ! If we are processing the identity operator, then just return
;  7466          7569  2      ! the left argument here.  Before we return, check to see if any
;  7467          7570  2      ! unconverted is left undone.  Make sure we take care of the
;  7468          7571  2      ! unary minus/unary plus case that was returned from earlier
;  7469          7572  2      ! call.  (Note: if you EV -1, -1 was not converted the first
;  7470          7573  2      ! time around, now we are hit by this token$k_identity and
;  7471          7574  2      ! unconverted constant).
;  7472          7575  2      !
;  7473          7576  2      IF .OPCODE EQL TOKEN$K_IDENTITY
;  7474          7577  2      THEN
;  7475          7578  3          BEGIN
;  7476          7579  3          IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7477          7580  3          THEN
```

```
;  7478          7581  3            LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, 0);
;  7479          7582  3
;  7480          7583  3        RETURN .LEFT_ARG;
;  7481          7584  2        END;
;  7482          7585  2
;  7483          7586  2
;  7484          7587  2        ! Consult the Type Hierarchy Table to see if any data type conversions need
;  7485          7588  2        ! to be done.
;  7486          7589  2        !
;  7487          7590  2        ! There are basically three cases here. They all use the hierarchy table,
;  7488          7591  2        ! but in different ways. The FIND_PATH and FIND_JOIN routines also
;  7489          7592  2        ! use the Operator Routine Table.
;  7490          7593  2        !
;  7491          7594  2        ! 1. Binary operators, such as +, which may do implicit conversions on
;  7492          7595  2        ! their operands. Here we call the FIND_JOIN routine. This routine assumes
;  7493          7596  2        ! we are trying to convert both left and right operand to the same type.
;  7494          7597  2        ! It finds a type that both can be converted to and which is legal for
;  7495          7598  2        ! the given operator. If there are several such types, it chooses the
;  7496          7599  2        ! one with the shortest conversion path.
;  7497          7600  2        !
;  7498          7601  2        ! 2. Unary operators. Here we call the FIND_PATH routine. This tries
;  7499          7602  2        ! to find a type that the operand can be converted to which is legal
;  7500          7603  2        ! for the given operator. If there are several, it chooses the one
;  7501          7604  2        ! with the shortest conversion path.
;  7502          7605  2        !
;  7503          7606  2        ! 3. The ASSIGNMENT operator and the CONVERT operator. Here we know
;  7504          7607  2        ! the target type, and we are just trying to determine if there is
;  7505          7608  2        ! a path from the source type. We use the routine FIND_PATH_DEPOSIT
;  7506          7609  2        ! to determine this.
;  7507          7610  2        !
;  7508          7611  2        IF .BINARY_FLAG
;  7509          7612  2        THEN
;  7510          7613  3            BEGIN
;  7511          7614  3
;  7512          7615  3
;  7513          7616  3            ! If an override was present on the DEPOSIT command, e.g.,
;  7514          7617  3            ! DEPOSIT/FLOAT 1000 = 1.1
;  7515          7618  3            ! then we skip the FIND_PATH_DEPOSIT routine which checks whether
;  7516          7619  3            ! the deposit is legal in the current language. We just call
;  7517          7620  3            ! the TYPE_CONV_DEBUG routine directly to do the deposit in a
;  7518          7621  3            ! language-independent fashion. Also call the routine directly
;  7519          7622  3            ! for instruction deposits.
;  7520          7623  3            !
;  7521          7624  3            IF .OPCODE EQL TOKEN$K_DEPOSIT
;  7522          7625  4            AND (.RIGHT_ARG [DBG$V_DHDR_OVERRIDE]
;  7523          7626  4              OR (.LEFT_TYPE EQL DSC$K_DTYPE_T AND .RIGHT_TYPE EQL DSC$K_DTYPE_ZI))
;  7524          7627  3            THEN
;  7525          7628  4                BEGIN
;  7526          7629  4                IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7527          7630  4                THEN
;  7528          7631  4                    LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, .RIGHT_TYPE);
;  7529          7632  4
;  7530          7633  4                RETURN DBG$COVER_DX_DX (.LEFT_ARG, .RIGHT_ARG, .CVT_ROUND_FLAG);
;  7531          7634  3                END;
;  7532          7635  3
;  7533          7636  3
;  7534          7637  3            ! Check for the special data, FCODE = PICT, DTYPE = T. for DEPOSIT.
```

```
;  7535     7638  3          ! In Type Hierd. table, this case is catched by DBG$K_DTYPE_PICT.
;  7536     7639  3          ! If both operands are text data, then we want to treat this as
;  7537     7640  3          ! standard Text String.
;  7538     7641  3          !
;  7539     7642  3          IF .OPCODE EQL TOKEN$K_DEPOSIT OR
;  7540     7643  3             .OPCODE EQL TOKEN$K_CONVERT
;  7541     7644  3          THEN
;  7542     7645  4              BEGIN
;  7543     7646  4              IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
;  7544     7647  4                 .LEFT_ARG[DBG$B_VALUE_DTYPE] NEQ DSC$K_DTYPE_T AND
;  7545     7648  4                 .DBG$GB_LANGUAGE EQL DBG$K_COBOL
;  7546     7649  4              THEN
;  7547     7650  4                  NEW_RIGHT_TYPE = DBG$K_DTYPE_PICT;
;  7548     7651  3              END;
;  7549     7652  3
;  7550     7653  3
;  7551     7654  3          IF .OPCODE EQL TOKEN$K_DEPOSIT OR .OPCODE EQL TOKEN$K_CONVERT
;  7552     7655  3          THEN
;  7553     7656  3              IF FIND_PATH_DEPOSIT (.NEW_LEFT_TYPE, .NEW_RIGHT_TYPE, 0,
;  7554     7657  3                  .HIER_TBL, .HIER_TBL_SIZE,
;  7555     7658  3                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  7556     7659  3                  .ROUT_TBL, .ROUT_TBL_SIZE)
;  7557     7660  3              THEN
;  7558     7661  4                  BEGIN
;  7559     7662  4
;  7560     7663  4
;  7561     7664  4                  ! Change the dtype for date type PICT, so the LANGUAGE type
;  7562     7665  4                  ! conversion can take place.
;  7563     7666  4                  !
;  7564     7667  4                  IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
;  7565     7668  4                     .LEFT_ARG[DBG$B_VALUE_DTYPE] NEQ DSC$K_DTYPE_T AND
;  7566     7669  4                     .DBG$GB_LANGUAGE EQL DBG$K_COBOL
;  7567     7670  4                  THEN
;  7568     7671  4                      RIGHT_ARG[DBG$B_VALUE_DTYPE] = DBG$K_DTYPE_PICT;
;  7569     7672  4
;  7570     7673  4
;  7571     7674  4                  ! If the target is SD, make the new type to be packed.
;  7572     7675  4                  !
;  7573     7676  4                  IF (.RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND
;  7574     7677  5                     (NOT .RIGHT_ARG[DBG$V_VALUE_FL_BINSCALE])
;  7575     7678  4                  THEN
;  7576     7679  4                      NEW_RIGHT_TYPE = DSC$K_DTYPE_P;
;  7577     7680  4
;  7578     7681  4
;  7579     7682  4                  ! Check for unconverted bit.  At this point, long interger
;  7580     7683  4                  ! string or floating string may be converted to Packed
;  7581     7684  4                  ! decimal if the new_right_type is a Packed decimal data item.
;  7582     7685  4                  !
;  7583     7686  4                  IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7584     7687  4                  THEN
;  7585     7688  5                      BEGIN
;  7586     7689  5
;  7587     7690  5
;  7588     7691  5                      ! This is a special case in COBOL.  We can only take the value
;  7589     7692  5                      ! and deposit it into unsigned variable.
;  7590     7693  5                      ! For example: DEP C4V4 (SD, WU) = -1234.1234.  The value
;  7591     7694  5                      ! in C4V4 is 1234.1234.
```

```
;  7592          7695  5     !
;  7593          7696  5     IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
;  7594          7697  5        .DBG$GB_LANGUAGE EQL DBG$K_RPG
;  7595          7698  5     THEN
;  7596          7699  6         BEGIN
;  7597          7700  6
;  7598          7701  6
;  7599          7702  6         ! Check to see if the target is unsigned data type.
;  7600          7703  6         !
;  7601          7704  6         IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_WU OR
;  7602          7705  6            .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU OR
;  7603          7706  6            .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_QU OR
;  7604          7707  6            .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NU
;  7605          7708  6         THEN
;  7606          7709  7             BEGIN
;  7607          7710  7             IF .LEFT_ARG[DBG$W_VALUE_SIGN_CODE] EQL TOKEN$K_NEGCONST
;  7608          7711  7             THEN
;  7609          7712  8                 BEGIN
;  7610          7713  8                 LEFT_ARG[DBG$W_VALUE_SIGN_CODE] = TOKEN$K_POSCONST;
;  7611          7714  8                 SIGNAL(DBG$_IVALOUTBNDS, T, OPERATOR[TOKEN$B_OPLEN]);
;  7612          7715  7                 END;
;  7613          7716  7
;  7614          7717  6             END;
;  7615          7718  6
;  7616          7719  5         END;
;  7617          7720  5
;  7618          7721  5     LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG,
;  7619          7722  5                                    .RIGHT_TYPE);
;  7620          7723  4     END;
;  7621          7724  4
;  7622          7725  4
;  7623          7726  4     ! There are cases the intermediate data type is needed
;  7624          7727  4     ! before the deposit, for example, DEP P=F, F needs to
;  7625          7728  4     ! convert to Packed decimal before deposit into P.
;  7626          7729  4     !
;  7627          7730  5     IF ((.RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND
;  7628          7731  4         (NOT .RIGHT_ARG[DBG$V_VALUE_FL_BINSCALE])) OR
;  7629          7732  4        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_P) OR
;  7630          7733  4        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NU) OR
;  7631          7734  5        (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DBG$K_DTYPE_PICT)
;  7632          7735  4     THEN
;  7633          7736  4         LEFT_ARG = INTMED_DATA_FOR_DEP(.LEFT_ARG, DSC$K_DTYPE_P,
;  7634          7737  4             FALSE);
;  7635          7738  4
;  7636          7739  4
;  7637          7740  4     ! We catch none-constant cases in here, for the ease of checking
;  7638          7741  4     ! sign for different data types.  In cobol, we should have
;  7639          7742  4     ! the source converted into Packed decimal at this point.
;  7640          7743  4     ! (in intermediate).
;  7641          7744  4     !
;  7642          7745  4     ! This is a special case in COBOL.  We can only take the value
;  7643          7746  4     ! and deposit it into unsigned variable.
;  7644          7747  4     ! For example: DEP C4V4 (SD, WU) = SC4V4 (SD, W)  The value
;  7645          7748  4     ! in C4V4 is 1234.1234, in SC4V4 is -1234.1234.
;  7646          7749  4     !
;  7647          7750  4     IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
;  7648          7751  4        .DBG$GB_LANGUAGE EQL DBG$K_RPG
```

```
;  7649          7752  4                THEN
;  7650          7753  5                    BEGIN
;  7651          7754  5
;  7652          7755  5
;  7653          7756  5                    ! Check to see if the target is unsigned data type.
;  7654          7757  5                    !
;  7655          7758  6                    IF (.RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_WU OR
;  7656          7759  6                        .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU OR
;  7657          7760  6                        .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_QU OR
;  7658          7761  5                        .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_NU) AND
;  7659          7762  6                       (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_P)
;  7660          7763  5                    THEN
;  7661          7764  6                        BEGIN
;  7662          7765  6                        LOCAL DIG, SIGN: BYTE, SIGN_POS: REF VECTOR[,BYTE];
;  7663          7766  6
;  7664          7767  6
;  7665          7768  6                        ! Find out the sign position in number string.
;  7666          7769  6                        !
;  7667          7770  6                        DIG = .LEFT_ARG[DBG$W_VALUE_LENGTH];
;  7668          7771  6                        SIGN_POS = .LEFT_ARG[DBG$L_VALUE_POINTER]
;  7669          7772  6                                + (.DIG / 2 + 1) - 1;
;  7670          7773  6
;  7671          7774  6
;  7672          7775  6                        ! Make it into positive.
;  7673          7776  6                        !
;  7674          7777  6                        SIGN = .SIGN_POS[0] AND %X'0D';
;  7675          7778  6                        IF .SIGN EQL %X'0D'
;  7676          7779  6                        THEN
;  7677          7780  7                            BEGIN
;  7678          7781  7                            SIGN_POS[0] = .SIGN_POS[0] AND %X'FE';
;  7679          7782  7                            SIGNAL(DBG$_IVALOUTBNDS, 1, OPERATOR[TOKEN$B_OPLEN]);
;  7680          7783  6                            END;
;  7681          7784  6
;  7682          7785  5                        END;
;  7683          7786  5
;  7684          7787  4                    END;
;  7685          7788  4
;  7686          7789  4
;  7687          7790  4                ! The following case is done for COBOL only.  In cobol,
;  7688          7791  4                ! if we do DEP sc2v2=999.888, sc2v2 is (SD, W with digits 4 and
;  7689          7792  4                ! scaled -2). the answer we want is 99.88, instead of -310.xx.
;  7690          7793  4                ! However if there is no digits specified in the Descriptor
;  7691          7794  4                ! the base digits used is 5 then you'll get -310.xx as an
;  7692          7795  4                ! answer.
;  7693          7796  4                !
;  7694          7797  4                ! The following code is a hack to force the type convertor
;  7695          7798  4                ! to do the right thing for cobol, for example, if we deposit
;  7696          7799  4                ! into a SD integer type, the way we make the type convertor to
;  7697          7800  4                ! do the right thing is to create an intermediate data type of
;  7698          7801  4                ! (SD, P) for left hand side as well, so the path looks like:
;  7699          7802  4                ! right hand side converts (SD, P), creates a (SD, P) descriptor
;  7700          7803  4                ! for left hand side, performs conversion from right (SD, P) to
;  7701          7804  4                ! left (SD, P), final converts right (SD, P) to right hand side.
;  7702          7805  4                !
;  7703          7806  4                IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
;  7704          7807  4                   .DBG$GB_LANGUAGE EQL DBG$K_RPG
;  7705          7808  4                THEN
```

```
;  7706          7809  5                BEGIN
;  7707          7810  5                IF .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  7708          7811  5                THEN
;  7709          7812  6                    BEGIN
;  7710          7813  6
;  7711          7814  6
;  7712          7815  6                    ! Get the SD, P form intermediate data for the left hand side.
;  7713          7816  6                    ! Create a place holder.  Note, the intermediate may not
;  7714          7817  6                    ! be created, for example, if we deposit P-->P, there
;  7715          7818  6                    ! is no need for it.  So the NEW_RIGHT_ARG may be the
;  7716          7819  6                    ! same as RIGHT_ARG.
;  7717          7820  6                    !
;  7718          7821  6                    NEW_RIGHT_ARG = INTMED_DATA_FOR_DEP(.RIGHT_ARG, DSC$K_DTYPE_P,
;  7719          7822  6                            TRUE);
;  7720          7823  6
;  7721          7824  6
;  7722          7825  6                    ! Perform convert from intermediate to intermediate.  The last
;  7723          7826  6                    ! conversion is convert the intermediate to original left
;  7724          7827  6                    ! hand side data type.
;  7725          7828  6                    !
;  7726          7829  6                    IF .NEW_RIGHT_ARG NEQ .RIGHT_ARG
;  7727          7830  6                    THEN
;  7728          7831  7                        BEGIN
;  7729          7832  7                        LEFT_ARG = DBG$TYPE_CONV(.LEFT_ARG, .NEW_RIGHT_ARG);
;  7730          7833  6                        END;
;  7731          7834  5                    END;
;  7732          7835  4                END;
;  7733          7836  4
;  7734          7837  4            RIGHT_ARG = DBG$TYPE_CONV (.LEFT_ARG, .RIGHT_ARG);
;  7735          7838  4
;  7736          7839  4                IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
;  7737          7840  4                   .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DBG$K_DTYPE_PICT AND
;  7738          7841  4                   .DBG$GB_LANGUAGE EQL DBG$K_COBOL
;  7739          7842  4                THEN
;  7740          7843  4                    RIGHT_ARG[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_T;
;  7741          7844  4
;  7742          7845  4                RETURN .RIGHT_ARG;
;  7743          7846  4                END
;  7744          7847  4
;  7745          7848  3            ELSE
;  7746          7849  3                SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN])
;  7747          7850  3
;  7748          7851  3        ELSE
;  7749          7852  4            BEGIN
;  7750          7853  4            IF .DBG$GB_LANGUAGE EQL DBG$K_BASIC
;  7751          7854  4            THEN
;  7752          7855  5                BEGIN
;  7753          7856  5                IF .NEW_LEFT_TYPE EQL DSC$K_DTYPE_P
;  7754          7857  5                THEN
;  7755          7858  5                    MAP_PACKED (NEW_LEFT_TYPE, .LEFT_ARG[DBG$W_VALUE_LENGTH], .NEW_RIGHT_TYPE)
;  7756          7859  5                ELSE
;  7757          7860  5                    IF .NEW_RIGHT_TYPE EQL DSC$K_DTYPE_P
;  7758          7861  5                    THEN
;  7759          7862  5                        MAP_PACKED (NEW_RIGHT_TYPE, .RIGHT_ARG[DBG$W_VALUE_LENGTH], .NEW_LEFT_TYPE);
;  7760          7863  5
;  7761          7864  4                END;
;  7762          7865  4
```

```
;  7763          7866  4              IF .MAX_DEPTH EQL FIND_JOIN(
;  7764          7867  4                  .NEW_LEFT_TYPE, .NEW_RIGHT_TYPE,
;  7765          7868  4                   NEW_LEFT_TYPE,  NEW_RIGHT_TYPE,
;  7766          7869  4                   ROUT_TBL_INDEX,
;  7767          7870  4                   0, .MAX_DEPTH,
;  7768          7871  4                  .HIER_TBL, .HIER_TBL_SIZE,
;  7769          7872  4                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  7770          7873  4                  .ROUT_TBL, .ROUT_TBL_SIZE)
;  7771          7874  4              THEN
;  7772          7875  4                  SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  7773          7876  4
;  7774          7877  3              END;
;  7775          7878  3
;  7776          7879  3          END
;  7777          7880  3
;  7778          7881  2      ELSE
;  7779          7882  3          BEGIN
;  7780          7883  3          IF .MAX_DEPTH EQL FIND_PATH(
;  7781          7884  3                  .NEW_LEFT_TYPE, NEW_LEFT_TYPE,
;  7782          7885  3                  ROUT_TBL_INDEX,
;  7783          7886  3                  .HIER_TBL, .HIER_TBL_SIZE,
;  7784          7887  3                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  7785          7888  3                  .ROUT_TBL, .ROUT_TBL_SIZE)
;  7786          7889  3          THEN
;  7787          7890  3              SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  7788          7891  3
;  7789          7892  2          END;
;  7790          7893  2
;  7791          7894  2
;  7792          7895  2      ! If one of the operand has T, VT, V, or VU in relational and arithmetic
;  7793          7896  2      ! operation in PLI.  The target type for T, VT, V, or VU depends on
;  7794          7897  2      ! the other operand.  This is not what we do normally.  So, we use
;  7795          7898  2      ! the normal way to validate the operand, and then use this routine
;  7796          7899  2      ! to modify the other information.  This is not the best way to do
;  7797          7900  2      ! it, but there is no other way I can think of is easier.
;  7798          7901  2      !
;  7799          7902  2      IF .BINARY_FLAG AND
;  7800          7903  2         .DBG$GB_LANGUAGE EQL DBG$K_PLI
;  7801          7904  2      THEN
;  7802          7905  2          MODIFY_PLI_TARGET_TYPE(.OPERATOR, .LEFT_TYPE, .RIGHT_TYPE,
;  7803          7906  2                  NEW_LEFT_TYPE, NEW_RIGHT_TYPE, ROUT_TBL_INDEX,
;  7804          7907  2                  .HIER_TBL, .HIER_TBL_SIZE,
;  7805          7908  2                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  7806          7909  2                  .ROUT_TBL, .ROUT_TBL_SIZE);
;  7807          7910  2
;  7808          7911  2
;  7809          7912  2      ! Check for unconverted bit.
;  7810          7913  2      !
;  7811          7914  2      IF .LEFT_ARG[DBG$V_DHDR_UNCVT]
;  7812          7915  2      THEN
;  7813          7916  2          LEFT_ARG = DBG$CONV_TEXT_VALUE(.LEFT_ARG, .LEFT_ARG, .NEW_LEFT_TYPE);
;  7814          7917  2
;  7815          7918  2
;  7816          7919  2      ! For the left operand, check whether we need to convert to a new type.
;  7817          7920  2      ! If so, allocate space to hold the result value descriptor, and then
;  7818          7921  2      ! call a routine to do the conversion.
;  7819          7922  2      !
```

```
;  7820          7923  2     IF .LEFT_TYPE NEQ .NEW_LEFT_TYPE
;  7821          7924  2     THEN
;  7822          7925  3         BEGIN
;  7823          7926  3         LENGTH = GET_DATA_LENGTH(.LEFT_TYPE, .NEW_LEFT_TYPE,
;  7824          7927  3             .LEFT_ARG[DBG$W_VALUE_LENGTH]);
;  7825          7928  3         SD_CLASS_FLAG = FALSE;
;  7826          7929  3         IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
;  7827          7930  3            .DBG$GB_LANGUAGE EQL DBG$K_RPG
;  7828          7931  3         THEN
;  7829          7932  4             BEGIN
;  7830          7933  4             IF .LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  7831          7934  4             THEN
;  7832          7935  4                 SD_CLASS_FLAG = TRUE;
;  7833          7936  3             END;
;  7834          7937  3
;  7835          7938  3         NEW_LEFT_ARG = MAKE_VAL_DESC(.NEW_LEFT_TYPE,
;  7836          7939  3                                      .LENGTH,
;  7837          7940  3                                      .LEFT_ARG,
;  7838          7941  3                                      .SD_CLASS_FLAG);
;  7839          7942  3
;  7840          7943  3         IF .NEW_LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  7841          7944  3         THEN
;  7842          7945  4             BEGIN
;  7843          7946  4
;  7844          7947  4
;  7845          7948  4             ! If left_arg is one of the floating-point data type, we need to
;  7846          7949  4             ! get the exponent value and that is the scaling factor.  For
;  7847          7950  4             ! example F --> P, F = 0.1234567E+04  --> P = 1234.567.
;  7848          7951  4             !
;  7849          7952  4             IF .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_F OR
;  7850          7953  4                .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_D OR
;  7851          7954  4                .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_G OR
;  7852          7955  4                .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_H
;  7853          7956  4             THEN
;  7854          7957  5                 BEGIN
;  7855          7958  5                 NEW_LEFT_ARG[DBG$B_VALUE_SCALE] =  GET_SCALE(.LEFT_ARG, DIGITS);
;  7856          7959  5                 NEW_LEFT_ARG[DBG$B_VALUE_DIGITS] = .DIGITS;
;  7857          7960  5                 NEW_LEFT_ARG[DBG$W_VALUE_LENGTH] = .DIGITS;
;  7858          7961  5                 END
;  7859          7962  5
;  7860          7963  4             ELSE
;  7861          7964  4                 NEW_LEFT_ARG[DBG$B_VALUE_SCALE] = .LEFT_ARG[DBG$B_VALUE_SCALE];
;  7862          7965  3             END;
;  7863          7966  3
;  7864          7967  3         NEW_LEFT_ARG = DBG$TYPE_CONV (.LEFT_ARG, .NEW_LEFT_ARG);
;  7865          7968  3         END
;  7866          7969  3
;  7867          7970  3
;  7868          7971  3     ! If there is no conversion to be done, just copy the left arg
;  7869          7972  3     ! into NEW_LEFT_ARG.
;  7870          7973  3     !
;  7871          7974  2     ELSE
;  7872          7975  2         NEW_LEFT_ARG = .LEFT_ARG;
;  7873          7976  2
;  7874          7977  2
;  7875          7978  2     ! Check for unconverted bit.
;  7876          7979  2     !
```

```
;  7877          7980  2      IF .BINARY_FLAG
;  7878          7981  2      THEN
;  7879          7982  2          IF .RIGHT_ARG[DBG$V_DHDR_UNCVT]
;  7880          7983  2          THEN
;  7881          7984  2              RIGHT_ARG = DBG$CONV_TEXT_VALUE(.RIGHT_ARG, .RIGHT_ARG, .NEW_RIGHT_TYPE);
;  7882          7985  2
;  7883          7986  2
;  7884          7987  2      ! If this is a binary operator, do the same for the right argument.
;  7885          7988  2      !
;  7886          7989  2      IF .BINARY_FLAG
;  7887          7990  2      THEN
;  7888          7991  2          IF .RIGHT_TYPE NEQ .NEW_RIGHT_TYPE
;  7889          7992  2          THEN
;  7890          7993  3              BEGIN
;  7891          7994  3              LENGTH = GET_DATA_LENGTH(.RIGHT_TYPE, .NEW_RIGHT_TYPE,
;  7892          7995  3                  .RIGHT_ARG[DBG$W_VALUE_LENGTH]);
;  7893          7996  3              SD_CLASS_FLAG = FALSE;
;  7894          7997  3              IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL OR
;  7895          7998  3                 .DBG$GB_LANGUAGE EQL DBG$K_RPG
;  7896          7999  3              THEN
;  7897          8000  4                  BEGIN
;  7898          8001  4                  IF .RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  7899          8002  4                  THEN
;  7900          8003  4                      SD_CLASS_FLAG = TRUE;
;  7901          8004  3                  END;
;  7902          8005  3
;  7903          8006  3              NEW_RIGHT_ARG = MAKE_VAL_DESC(.NEW_RIGHT_TYPE,
;  7904          8007  3                                            .LENGTH,
;  7905          8008  3                                            .RIGHT_ARG,
;  7906          8009  3                                            .SD_CLASS_FLAG);
;  7907          8010  3
;  7908          8011  3              IF .NEW_RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  7909          8012  3              THEN
;  7910          8013  4                  BEGIN
;  7911          8014  4
;  7912          8015  4                  ! If right_arg is one of the floating-point data type, we need to
;  7913          8016  4                  ! get the exponent value and that is the scaling factor.  For
;  7914          8017  4                  ! example F --> P, F = 0.1234567E+04  --> P = 1234.567.
;  7915          8018  4                  !
;  7916          8019  4                  IF .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_F OR
;  7917          8020  4                     .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_D OR
;  7918          8021  4                     .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_G OR
;  7919          8022  4                     .RIGHT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_H
;  7920          8023  4                  THEN
;  7921          8024  5                      BEGIN
;  7922          8025  5                      NEW_RIGHT_ARG[DBG$B_VALUE_SCALE] =  GET_SCALE(.RIGHT_ARG, DIGITS);
;  7923          8026  5                      NEW_RIGHT_ARG[DBG$B_VALUE_DIGITS] = .DIGITS;
;  7924          8027  5                      NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH] = .DIGITS;
;  7925          8028  5                      END
;  7926          8029  5
;  7927          8030  4                  ELSE
;  7928          8031  4                      NEW_RIGHT_ARG[DBG$B_VALUE_SCALE] = .RIGHT_ARG[DBG$B_VALUE_SCALE];
;  7929          8032  3                  END;
;  7930          8033  3
;  7931          8034  3              NEW_RIGHT_ARG = DBG$TYPE_CONV (.RIGHT_ARG, .NEW_RIGHT_ARG);
;  7932          8035  3              END
;  7933          8036  3
```

```
;  7934          8037  2          ELSE
;  7935          8038  2              NEW_RIGHT_ARG = .RIGHT_ARG;
;  7936          8039  2
;  7937          8040  2
;  7938          8041  2      ROUT_INDEX = .ROUT_TBL[.ROUT_TBL_INDEX, ORT$W_ROUT];
;  7939          8042  2      TYPEID_INDEX = .ROUT_TBL[.ROUT_TBL_INDEX, ORT$W_TYPEID_ROUT];
;  7940          8043  2      RESULT_TYPE = .ROUT_TBL[.ROUT_TBL_INDEX, ORT$B_RESULT_TYPE];
;  7941          8044  2      IF (.NEW_LEFT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) OR
;  7942          8045  3         (.BINARY_FLAG AND (.NEW_RIGHT_ARG[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD))
;  7943          8046  2      THEN
;  7944          8047  2          SD_CLASS_FLAG = TRUE
;  7945          8048  2      ELSE
;  7946          8049  2          SD_CLASS_FLAG = FALSE;
;  7947          8050  2
;  7948          8051  2      LENGTH = 0;
;  7949          8052  2      IF .ROUT_INDEX EQL ORT$K_CONCAT_T_T OR
;  7950          8053  2         .ROUT_INDEX EQL ORT$K_CONCAT_TF_TF
;  7951          8054  2      THEN
;  7952          8055  2          LENGTH = .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH] +
;  7953          8056  2                   .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH];
;  7954          8057  2
;  7955          8058  2      IF .LENGTH EQL 0
;  7956          8059  2      THEN
;  7957          8060  3          BEGIN
;  7958          8061  3          IF .RESULT_TYPE EQL DSC$K_DTYPE_V OR
;  7959          8062  3             .RESULT_TYPE EQL DSC$K_DTYPE_VU OR
;  7960          8063  3             .RESULT_TYPE EQL DSC$K_DTYPE_SV OR
;  7961          8064  3             .RESULT_TYPE EQL DSC$K_DTYPE_SVU
;  7962          8065  3          THEN
;  7963          8066  4              BEGIN
;  7964          8067  4              LENGTH = .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH];
;  7965          8068  4              IF .BINARY_FLAG
;  7966          8069  4              THEN
;  7967          8070  5                  BEGIN
;  7968          8071  5                  IF .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH] GTR
;  7969          8072  5                     .NEW_LEFT_ARG[DBG$W_VALUE_LENGTH]
;  7970          8073  5                  THEN
;  7971          8074  5                      LENGTH = .NEW_RIGHT_ARG[DBG$W_VALUE_LENGTH];
;  7972          8075  4                  END;
;  7973          8076  4
;  7974          8077  4              END
;  7975          8078  4
;  7976          8079  3          ELSE
;  7977          8080  3              LENGTH = DBG$NUM_BYTES(.RESULT_TYPE);
;  7978          8081  3
;  7979          8082  2          END;
;  7980          8083  2
;  7981          8084  2      RESULT = MAKE_VAL_DESC(.RESULT_TYPE,
;  7982          8085  2                             .LENGTH,
;  7983          8086  2                             0,
;  7984          8087  2                             .SD_CLASS_FLAG);
;  7985          8088  2
;  7986          8089  2
;  7987          8090  2      ! Make sure we did not turn SD_CLASS_FLAG on for the following
;  7988          8091  2      ! data types. (in order words, ignore the incoming data types).
;  7989          8092  2      !
;  7990          8093  2      CASE .RESULT[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_LOWEST
```

```
;  7991     8094  2                                        TO   DSC$K_DTYPE_HIGHEST OF
;  7992     8095  2          SET
;  7993     8096  2          [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H,
;  7994     8097  2           DSC$K_DTYPE_FC, DSC$K_DTYPE_DC, DSC$K_DTYPE_GC, DSC$K_DTYPE_HC]:
;  7995     8098  2              RESULT[DBG$B_VALUE_CLASS] = DSC$K_CLASS_S;
;  7996     8099  2          [INRANGE, OUTRANGE]:
;  7997     8100  2              0;
;  7998     8101  2          TES;
;  7999     8102  2
;  8000     8103  2
;  8001     8104  2      ! Call the routine which performs the operation. We pass in
;  8002     8105  2      ! the routine index, which is used as a case index inside the
;  8003     8106  2      ! routine, the addresses of the operand value descriptors,
;  8004     8107  2      ! and the address of a longword containing a pointer to the result
;  8005     8108  2      ! value descriptor. The reason for the extra level of indirection
;  8006     8109  2      ! for the result is that sometimes DBG$PERFORM_OPERATOR will construct
;  8007     8110  2      ! a new descriptor instead of using the one we pass in.
;  8008     8111  2      !
;  8009     8112  2      DBG$PERFORM_OPERATOR (.OPERATOR,
;  8010     8113  2                            .ROUT_INDEX,
;  8011     8114  2                            .NEW_LEFT_ARG,
;  8012     8115  3                            (IF .BINARY_FLAG
;  8013     8116  3                            THEN .NEW_RIGHT_ARG
;  8014     8117  2                            ELSE 0),
;  8015     8118  2                            RESULT);
;  8016     8119  2
;  8017     8120  2
;  8018     8121  2      ! Call TYPEID routine to do type check on non-atomic data types.  And
;  8019     8122  2      ! also fix up the Result Value Descriptor.
;  8020     8123  2      !
;  8021     8124  2      IF .TYPEID_INDEX NEQ 0
;  8022     8125  2      THEN
;  8023     8126  3          BEGIN
;  8024     8127  3          IF NOT DBG$PERFORM_TYPEID_CHECK(.TYPEID_INDEX, .NEW_LEFT_ARG,
;  8025     8128  3                                          .NEW_RIGHT_ARG, .RESULT)
;  8026     8129  3          THEN
;  8027     8130  3              SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;  8028     8131  3
;  8029     8132  2          END;
;  8030     8133  2
;  8031     8134  2
;  8032     8135  2      ! Return a pointer to the RESULT value descriptor.
;  8033     8136  2      !
;  8034     8137  2      RETURN .RESULT;
;  8035     8138  1      END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 31  05C8E P.ALS:  .ASCII  \1DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\   ;
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45  05C9D                                                           ;
                              2C 52 4F 54     05CAC                                                           ;
20 79 63 6E 65 74 73 69 73 6E 6F 63 6E 69 20  05CB0          .ASCII  \ inconsistency 1\                       ;
                                              31 05CBF                                                        ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 31  05CC0 P.ALT:  .ASCII  \1DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\   ;
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45  05CCF                                                           ;
```

```
                                              2C 52 4F 54 05CDE
20 79 63 6E 65 74 73 69 73 6E 6F 63 6E 69 20 05CE2          .ASCII  \ inconsistency 2\                          ;
                                                       32 05CF1                                                     ;
24 47       44 5C 50 4F 4C 41 56 45 47 42 44 31 05CF2 P.ALU: .ASCII  \1DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\  ;
41 5;    ->  50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05D01                                                          ;
                                              2C 52 4F 54 05D10                                                 ;
20 79 63 6E 65 74 73 69 73 6E 6F 63 6E 69 20 05D14          .ASCII  \ inconsistency 3\                          ;
                                                       33 05D23                                                     ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 36 05D24 P.ALV: .ASCII  \6DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR \  ;
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05D33                                                          ;
                                              20 52 4F 54 05D42                                                 ;
6D 20 65 6C 62 61 74 20 65 6E 69 74 75 6F 72 05D46          .ASCII  \routine table missing\                     ;
                              67 6E 69 73 73 69 05D55                                                       ;
45 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05D5B P.ALW: .ASCII  \4DBGEVALOP\<92>\DBGEVAL_LANG_OPERATOR, \  ;
54 41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 05D6A                                                          ;
                                              20 2C 52 4F 05D79                                                 ;
61 66 20 72 64 64 61 20 6F 74 20 6D 69 72 70 05D7D          .ASCII  \prim to addr failed\                       ;
                                              64 65 6C 69 05D8C                                                 ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05D90 P.ALX: .ASCII  \4DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\  ;
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05D9F                                                          ;
                                              2C 52 4F 54 05DAE                                                 ;
61 66 20 6C 61 76 20 6F 74 20 6D 69 72 70 20 05DB2          .ASCII  \ prim to val failed\                       ;
                                              64 65 6C 69 05DC1                                                 ;
45 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05DC5 P.ALY: .ASCII  \4DBGEVALOP\<92>\DBGEVAL_LANG_OPERATOR, \  ;
54 41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 05DD4                                                          ;
                                              20 2C 52 4F 05DE3                                                 ;
61 66 20 72 64 64 61 20 6F 74 20 6D 69 72 70 05DE7          .ASCII  \prim to addr failed\                       ;
                                              64 65 6C 69 05DF6                                                 ;
45 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05DFA P.ALZ: .ASCII  \4DBGEVALOP\<92>\DBGEVAL_LANG_OPERATOR, \  ;
54 41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 05E09                                                          ;
                                              20 2C 52 4F 05E18                                                 ;
61 66 20 72 64 64 61 20 6F 74 20 6D 69 72 70 05E1C          .ASCII  \prim to addr failed\                       ;
                                              64 65 6C 69 05E2B                                                 ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05E2F P.AMA: .ASCII  \4DBGEVALOP\<92>\DBG$EVAL_LANG_OPERATOR,\  ;
41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 45 05E3E                                                          ;
                                              2C 52 4F 54 05E4D                                                 ;
61 66 20 6C 61 76 20 6F 74 20 6D 69 72 70 20 05E51          .ASCII  \ prim to val failed\                       ;
                                              64 65 6C 69 05E60                                                 ;
45 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 34 05E64 P.AMB: .ASCII  \4DBGEVALOP\<92>\DBGEVAL_LANG_OPERATOR, \  ;
54 41 52 45 50 4F 5F 47 4E 41 4C 5F 4C 41 56 05E73                                                          ;
                                              20 2C 52 4F 05E82                                                 ;
61 66 20 72 64 64 61 20 6F 74 20 6D 69 72 70 05E86          .ASCII  \prim to addr failed\                       ;
                                              64 65 6C 69 05E95                                                 ;


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

                                              0FFC 00000           .ENTRY  DBG$EVAL_LANG_OPERATOR, Save R2,R3,R4,R5,-  ; 7068
                                                                           R6,R7,R8,R9,R10,R11                          ;
                      5E                   24   C2 00002           SUBL2   #36, SP                                      ;
                                           53   D4 00005           CLRL    R3                                           ; 7164
      04    04 BC     10                   10   ED 00007           CMPZV   #16, #16, @OPERATOR, #4                      ;
                                           04   12 0000D           BNEQ    1$                                           ;
                                           53   D6 0000F           INCL    R3                                           ;
                                           0B   11 00011           BRB     3$                                           ;
      05    04 BC     10                   10   ED 00013 1$:       CMPZV   #16, #16, @OPERATOR, #5                      ; 7165
```

```
                                                 03  13 00019          BEQL    3$                                          ;
                                              008A   31 0001B 2$:       BRW     12$                                         ;
                                        52       08   AC  D0 0001E 3$:  MOVL    LEFT_ARG, R2                                ; 7168
                                    F4       04   A2  05  E1 00022      BBC     #5, ^(R2), 2$                               ;
                                             0C   AC  D5 00027          TSTL    RIGHT_ARG                                   ; 7175
                                                 15  13 0002A          BEQL    4$                                          ;
                                00000000'        EF  9F 0002C          PUSHAB  P.ALS                                       ; 7177
                                                 01  DD 00032          PUSHL   #1                                          ;
                                00028362         8F  DD 00034          PUSHL   #164706                                     ;
                            00000000G    00      03  FB 0003A          CALLS   #3, LIB$SIGNAL                              ;
                                  79     8F      02  A2  91 00041 4$:   CMPB    2(R2), #121                                 ; 7183
                                                 07  13 00046          BEQL    5$                                          ;
                                  83     8F      02  A2  91 00048      CMPB    2(R2), #131                                 ; 7184
                                                 15  12 0004D          BNEQ    6$                                          ;
                                00000000'        EF  9F 0004F 5$:      PUSHAB  P.ALT                                       ; 7186
                                                 01  DD 00055          PUSHL   #1                                          ;
                                00028362         8F  DD 00057          PUSHL   #164706                                     ;
                            00000000G    00      03  FB 0005D          CALLS   #3, LIB$SIGNAL                              ;
    05      04  BC                   10          10  ED 00064 6$:      CMPZV   #16, #16, @OPERATOR, #5                     ; 7193
                                                 0A  12 0006A          BNEQ    7$                                          ;
                                0042     8F      12  A2  B1 0006C      CMPW    18(R2), #66                                 ; 7196
                                                 0D  12 00072          BNEQ    8$                                          ;
                                                 12  11 00074          BRB     9$                                          ; 7198
                                        16                             53  E9 00076 7$:  BLBC    R3, 10$                     ; 7204
                                0042     8F      12  A2  B1 00079      CMPW    18(R2), #66                                 ; 7207
                                                 07  12 0007F          BNEQ    9$                                          ;
                                  12     A2      42  8F  9B 00081 8$:  MOVZBW  #66, 18(R2)                                 ; 7209
                                                 1C  11 00086          BRB     11$                                         ;
                                  12     A2      43  8F  9B 00088 9$:  MOVZBW  #67, 18(R2)                                 ; 7211
                                                 15  11 0008D          BRB     11$                                         ; 7204
                                00000000'        EF  9F 0008F 10$:     PUSHAB  P.ALU                                       ; 7215
                                                 01  DD 00095          PUSHL   #1                                          ;
                                00028362         8F  DD 00097          PUSHL   #164706                                     ;
                            00000000G    00      03  FB 0009D          CALLS   #3, LIB$SIGNAL                              ;
                                        50       52  D0 000A4 11$:     MOVL    R2, R0                                      ; 7217
                                                 04  000A7            RET                                                 ;
00000042    8F      04  BC                   10          10  ED 000A8 12$:  CMPZV   #16, #16, @OPERATOR, #66                ; 7227
                                                 08  12 000B2          BNEQ    13$                                         ;
                                      04     AC  00000000G    00  9E 000B4  MOVAB   DBG$GL_NEG_SIGN_TOKEN, OPERATOR          ; 7229
00000043    8F      04  BC                   10          10  ED 000BC 13$:  CMPZV   #16, #16, @OPERATOR, #67                ; 7231
                                                 08  12 000C6          BNEQ    14$                                         ;
                                      04     AC  00000000G    00  9E 000C8  MOVAB   DBG$GL_POS_SIGN_TOKEN, OPERATOR          ; 7233
                                00000000'        EF  01  D0 000D0 14$:  MOVL    #1, MAX_DEPTH                               ; 7238
                                                 09  00000000'  EF  E9 000D7  BLBC    BLISS_BITSELECTION_FLAG1, 15$           ; 7239
                                00000000'        EF          01  D0 000DE  MOVL    #1, BLISS_BITSELECTION_FLAG2                ; 7241
                                                 06  11 000E5          BRB     16$                                         ;
                                                 00000000'   EF  D4 000E7 15$:  CLRL    BLISS_BITSELECTION_FLAG2            ; 7243
                                                 00000000'   EF  D4 000ED 16$:  CLRL    BLISS_BITSELECTION_FLAG1            ; 7244
                                                 00000000'   EF  D4 000F3      CLRL    BLISS_INDIRECTION_FLAG              ; 7245
        52      04  BC                   10          10  EF 000F9      EXTZV   #16, #16, @OPERATOR, OPCODE                 ; 7246
                                      04     AC          0C  C1 000FF      ADDL3   #12, OPERATOR, R11                          ; 7247
                                00000000'        EF          5B  D0 00104  MOVL    R11, DBG$GL_OPCODE_NAME                     ;
                                                 50  D4 0010B          CLRL    R0                                          ; 7248
                                        03       04  BC 91 0010D      CMPB    @OPERATOR, #3                               ;
                                                 02  12 00111          BNEQ    17$                                         ;
                                                 50  D6 00113          INCL    R0                                          ;
                                  0C     AE              50  90 00115 17$:  MOVB    R0, BINARY_FLAG                         ;
```

```
                58              52              04  78 00119          ASHL    #4, OPCODE, R8                          ; 7249
                51              58 00000000'    EF  C1 0011D          ADDL3   OPINFO_TABLE, R8, R1
                                50 00000000'    EF  9E 00125          MOVAB   TABLEBASE, R0
                57              50              61  C1 0012C          ADDL3   (R1), R0, ROUT_TBL
                                50 00000000'    EF  9E 00130          MOVAB   TABLEBASE, R0                           ; 7250
                                50              57  D1 00137          CMPL    ROUT_TBL, R0
                                                26  12 0013A          BNEQ    18$
                                32              52  D1 0013C          CMPL    OPCODE, #50                             ; 7253
                                                27  13 0013F          BEQL    19$
                                33              52  D1 00141          CMPL    OPCODE, #51
                                                22  13 00144          BEQL    19$
                                38              52  D1 00146          CMPL    OPCODE, #56                             ; 7254
                                                1D  13 00149          BEQL    19$
                                    00000000'   EF  9F 0014B          PUSHAB  P.ALV                                   ; 7256
                                                01  DD 00151          PUSHL   #1
                                    00028362    8F  DD 00153          PUSHL   #164706
                    000000000G  00              03  FB 00159          CALLS   #3, LIB$SIGNAL
                                                06  11 00160          BRB     19$                                     ; 7250
            04  AE      FC     A7              02  C7 00162 18$:      DIVL3   #2, -4(ROUT_TBL), ROUT_TBL_SIZE         ; 7259
                                50 00000000'    EF  D0 00168 19$:      MOVL    OPINFO_TABLE, R0                        ; 7260
                                51 00000000'    EF  9E 0016F          MOVAB   TABLEBASE, R1
                                        04 A840 9F  00176             PUSHAB  4(R8)[R0]
                5A              51              9E  C1 0017A          ADDL3   @(SP)+, R1, HIER_TBL
                                51 00000000'    EF  9E 0017E          MOVAB   TABLEBASE, R1                           ; 7261
                                51              5A  D1 00185          CMPL    HIER_TBL, R1
                                                11  13 00188          BEQL    20$
                55          FC  AA              01  78 0018A          ASHL    #1, -4(HIER_TBL), HIER_TBL_SIZE         ; 7264
                00000000'   EF  00000000'FF45   3E  0018F          MOVAW   @MAX_DEPTH[HIER_TBL_SIZE], MAX_DEPTH    ; 7265
                                51 00000000'    EF  9E 0019B 20$:      MOVAB   TABLEBASE, R1                           ; 7267
                                        08 A840 9F  001A2             PUSHAB  8(R8)[R0]
                59              51              9E  C1 001A6          ADDL3   @(SP)+, R1, INCOMP_TBL
                                51 00000000'    EF  9E 001AA          MOVAB   TABLEBASE, R1                           ; 7268
                                51              59  D1 001B1          CMPL    INCOMP_TBL, R1
                                                06  13 001B4          BEQL    21$
        08  AE      FC         A9              01  78 001B6          ASHL    #1, -4(INCOMP_TBL), INCOMP_TBL_SIZE
                                53          08  AC  D0 001BC 21$:      MOVL    LEFT_ARG, R3                            ; 7285
                        79      8F          02  A3  91 001C0          CMPB    2(R3), #121
                                                07  13 001C5          BEQL    22$
                        83      8F          02  A3  91 001C7          CMPB    2(R3), #131                             ; 7286
                                                55  12 001CC          BNEQ    25$
                03          0C A840             00  E0 001CE 22$:      BBS     #0, 12(R8)[R0], 23$                     ; 7303
                                                00C2 31 001D4         BRW     33$
                    00000000'   EF              01  D0 001D7 23$:      MOVL    #1, BLISS_INDIRECTION_FLAG              ; 7306
                                07 000000000G   00  91 001DE          CMPB    DBG$GB_LANGUAGE, #7                     ; 7312
                                                3F  12 001E5          BNEQ    26$
                                01          06  A3  91 001E7          CMPB    6(R3), #1                               ; 7313
                                                39  12 001EB          BNEQ    26$
                                56          08  A3  D0 001ED          MOVL    8(R3), TYPEID                           ; 7319
                                            08  AC  9F 001F1          PUSHAB  LEFT_ARG                                ; 7323
                                                08  DD 001F4          PUSHL   #8
                                                53  DD 001F6          PUSHL   R3
                    000000000G  00              03  FB 001F8          CALLS   #3, DBG$PRIM_TO_ADDR
                                15              50  E8 001FF          BLBS    R0, 24$
                                    00000000'   EF  9F 00202          PUSHAB  P.ALW                                   ; 7325
                                                01  DD 00208          PUSHL   #1
                                    00028362    8F  DD 0020A          PUSHL   #164706
                    000000000G  00              03  FB 00210          CALLS   #3, LIB$SIGNAL
```

```
                50        08   AC  D0 00217 24$:    MOVL    LEFT_ARG, R0                    ; 7329
          06    A0              06  90 0021B         MOVB    #6, 8(R0)
          08    A0              56  D0 0021F         MOVL    TYPEID, 8(R0)                   ; 7330
                          0099  31  00223 25$:       BRW     35$                             ; 7312
                09        04    A3  E9 00226 26$:    BLBC    4(R3), 27$                      ; 7340
                                  53  DD 0022A         PUSHL   R3                              ; 7342
00000000G       00              01  FB 0022C         CALLS   #1, DBG$COLLECT
                                  54  D4 00233 27$:    CLRL    SYMID                           ; 7349
                45        04    A3  E9 00235         BLBC    4(R3), 32$                      ; 7350
                07        06    A3  91 00239         CMPB    6(R3), #7                       ; 7353
                                  0D  13 0023D         BEQL    28$
                    000287F8      8F  DD 0023F         PUSHL   #165880                         ; 7355
00000000G       00              01  FB 00245         CALLS   #1, LIB$SIGNAL
                54        0C    A3  D0 0024C 28$:    MOVL    12(R3), SYMID                   ; 7360
                                  0D  12 00250 29$:    BNEQ    30$                             ; 7361
                    000287F8      8F  DD 00252         PUSHL   #165880
00000000G       00              01  FB 00258         CALLS   #1, LIB$SIGNAL
                01        14    A4  91 0025F 30$:    CMPB    20(SYMID), #1                   ; 7362
                                  06  13 00263         BEQL    31$
                54        10    A4  D0 00265         MOVL    16(SYMID), SYMID                ; 7364
                                  E5  11 00269         BRB     29$                             ; 7365
                03        29    A4  91 0026B 31$:    CMPB    41(SYMID), #3                   ; 7367
                                  0D  13 0026F         BEQL    32$
                    000287F8      8F  DD 00271         PUSHL   #165880                         ; 7369
00000000G       00              01  FB 00277         CALLS   #1, LIB$SIGNAL
                          08    AC  9F 0027E 32$:    PUSHAB  LEFT_ARG                        ; 7372
                7E        7A    8F  9A 00281         MOVZBL  #122, -(SP)
                                  53  DD 00285         PUSHL   R3
00000000G       00              03  FB 00287         CALLS   #3, DBG$PRIM_TO_VAL
                2E              50  E8 0028E         BLBS    R0, 35$
                    00000000'     EF  9F 00291         PUSHAB  P.ALX                           ; 7374
                                  17  11 00297         BRB     34$
                          08    AC  9F 00299 33$:    PUSHAB  LEFT_ARG                        ; 7380
                                  08  DD 0029C         PUSHL   #8
                                  53  DD 0029E         PUSHL   R3
          00000000G       00      03  FB 002A0         CALLS   #3, DBG$PRIM_TO_ADDR
                          15      50  E8 002A7         BLBS    R0, 35$
                        00000000' EF  9F 002AA         PUSHAB  P.ALY                           ; 7382
                                  01  DD 002B0 34$:    PUSHL   #1
                        00028362  8F  DD 002B2         PUSHL   #164706
          00000000G       00      03  FB 002B8         CALLS   #3, LIB$SIGNAL
                          77        0C  AE  E9 002BF 35$:    BLBC    BINARY_FLAG, 39$        ; 7387
                          53        0C  AC  D0 002C3         MOVL    RIGHT_ARG, R3           ; 7389
                    79    8F        02  A3  91 002C7         CMPB    2(R3), #121
                                        11  13 002CC         BEQL    36$
                    83    8F        02  A3  91 002CE         CMPB    2(R3), #131             ; 7390
                                        65  12 002D3         BNEQ    39$
                          32              52  D1 002D5         CMPL    OPCODE, #50             ; 7391
                                          60  13 002D8         BEQL    39$
                          33              52  D1 002DA         CMPL    OPCODE, #51
                                          5B  13 002DD         BEQL    39$
          50              58  00000000'   EF  C1 002DF 36$:    ADDL3   OPINFO_TABLE, R8, R0    ; 7393
                          03        0C    A0  E8 002E7         BLBS    12(R0), 37$
                                      00C2  31  002EB         BRW     47$
                00000000' EF              01  D0 002EE 37$:    MOVL    #1, BLISS_INDIRECTION_FLAG  ; 7396
                          07  00000000G    00  91 002F5         CMPB    DBG$GB_LANGUAGE, #7     ; 7402
                                            3F  12 002FC         BNEQ    40$
```

```
                01      06    A3  91 002FE          CMPB    6(R3), #1                       ; 7403
                                  39 12 00302          BNEQ    40$
                56      08    A3  D0 00304          MOVL    8(R3), TYPEID                   ; 7409
                        0C    AC  9F 00308          PUSHAB  RIGHT_ARG                       ; 7413
                              08  DD 0030B          PUSHL   #8
                              53  DD 0030D          PUSHL   R3
00000000G       00            03  FB 0030F          CALLS   #3, DBG$PRIM_TO_ADDR
                15            50  E8 00316          BLBS    R0, 38$
                    00000000'  EF 9F 00319          PUSHAB  P.ALZ                           ; 7415
                              01  DD 0031F          PUSHL   #1
                    00028362  8F  DD 00321          PUSHL   #164706
00000000G       00            03  FB 00327          CALLS   #3, LIB$SIGNAL
                50      0C    AC  D0 0032E 38$:     MOVL    RIGHT_ARG, R0                   ; 7419
      06        A0            06  90 00332          MOVB    #6, 6(R0)
      08        A0            56  D0 00336          MOVL    TYPEID, 8(R0)                   ; 7420
                          0099  31 0033A 39$:     BRW     49$                             ; 7402
                09      04    A3  E9 0033D 40$:     BLBC    4(R3), 41$                      ; 7431
                              53  DD 00341          PUSHL   R3                              ; 7433
00000000G       00            01  FB 00343          CALLS   #1, DBG$COLLECT
                              54  D4 0034A 41$:     CLRL    SYMID                           ; 7440
                45      04    A3  E9 0034C          BLBC    4(R3), 46$                      ; 7441
                07      06    A3  91 00350          CMPB    6(R3), #7                       ; 7444
                              0D  13 00354          BEQL    42$
                    000287F8  8F  DD 00356          PUSHL   #165880                         ; 7446
00000000G       00            01  FB 0035C          CALLS   #1, LIB$SIGNAL
                54      0C    A3  D0 00363 42$:     MOVL    12(R3), SYMID                   ; 7451
                              0D  12 00367 43$:     BNEQ    44$                             ; 7452
                    000287F8  8F  DD 00369          PUSHL   #165880
00000000G       00            01  FB 0036F          CALLS   #1, LIB$SIGNAL
                01      14    A4  91 00376 44$:     CMPB    20(SYMID), #1                   ; 7453
                              06  13 0037A          BEQL    45$
                54      10    A4  D0 0037C          MOVL    16(SYMID), SYMID                ; 7455
                              E5  11 00380          BRB     43$                             ; 7456
                03      29    A4  91 00382 45$:     CMPB    41(SYMID), #3                   ; 7458
                              0D  13 00386          BEQL    46$
                    000287F8  8F  DD 00388          PUSHL   #165880                         ; 7460
00000000G       00            01  FB 0038E          CALLS   #1, LIB$SIGNAL
                        0C    AC  9F 00395 46$:     PUSHAB  RIGHT_ARG                       ; 7463
                7E      7A    8F  9A 00398          MOVZBL  #122, -(SP)
                              53  DD 0039C          PUSHL   R3
00000000G       00            03  FB 0039E          CALLS   #3, DBG$PRIM_TO_VAL
                2E            50  E8 003A5          BLBS    R0, 49$
                    00000000'  EF 9F 003A8          PUSHAB  P.AMA                           ; 7465
                              17  11 003AE          BRB     48$
                        0C    AC  9F 003B0 47$:     PUSHAB  RIGHT_ARG                       ; 7470
                              08  DD 003B3          PUSHL   #8
                              53  DD 003B5          PUSHL   R3
00000000G       00            03  FB 003B7          CALLS   #3, DBG$PRIM_TO_ADDR
                15            50  E8 003BE          BLBS    R0, 49$
                    00000000'  EF 9F 003C1          PUSHAB  P.AMB                           ; 7472
                              01  DD 003C7 48$:     PUSHL   #1
                    00028362  8F  DD 003C9          PUSHL   #164706
00000000G       00            03  FB 003CF          CALLS   #3, LIB$SIGNAL
                              54  D4 003D6 49$:     CLRL    R4                              ; 7477
                32            52  D1 003D8          CMPL    OPCODE, #50
                              25  12 003DB          BNEQ    50$
                              54  D6 003DD          INCL    R4
```

```
                51          08     AC  D0 003DF          MOVL    LEFT_ARG, R1                                    ; 7480
1A        04    A1                 05  E1 003E3          BBC     #5, 4(R1), 50$
                50          0C     AC  D0 003E8          MOVL    RIGHT_ARG, R0                                   ; 7483
                14          06     A0  91 003EC          CMPB    6(R0), #20
                                   10  12 003F0          BNEQ    50$
          08    A1          08     A0  D0 003F2          MOVL    8(R0), 8(R1)                                    ; 7486
                                   51  DD 003F7          PUSHL   R1                                              ; 7487
        F8B3    CF                 01  FB 003F9          CALLS   #1, DBG$CONV_TRFA_VALUE
          08    AC                 50  D0 003FE          MOVL    R0, LEFT_ARG
                08 00000000G       00  91 00402 50$:     CMPB    DBG$GB_LANGUAGE, #8                             ; 7496
                                   38  12 00409          BNEQ    52$
                50          08     AC  D0 0040B          MOVL    LEFT_ARG, R0                                    ; 7499
                13          16     A0  91 0040F          CMPB    22(R0), #19
                                   0B  12 00413          BNEQ    51$
                                   50  DD 00415          PUSHL   R0                                              ; 7501
      0000V     CF                 01  FB 00417          CALLS   #1, MAP_NRO_DTYPE_IN_RPG
          08    AC                 50  D0 0041C          MOVL    R0, LEFT_ARG
                1F          0C     AE  E9 00420 51$:     BLBC    BINARY_FLAG, 52$                                ; 7503
                32                 52  D1 00424          CMPL    OPCODE, #50                                     ; 7506
                                   1A  13 00427          BEQL    52$
                33                 52  D1 00429          CMPL    OPCODE, #51                                     ; 7507
                                   15  13 0042C          BEQL    52$
                50          0C     AC  D0 0042E          MOVL    RIGHT_ARG, R0                                   ; 7509
                13          16     A0  91 00432          CMPB    22(R0), #19
                                   0B  12 00436          BNEQ    52$
                                   50  DD 00438          PUSHL   R0                                              ; 7511
      0000V     CF                 01  FB 0043A          CALLS   #1, MAP_NRO_DTYPE_IN_RPG
          0C    AC                 50  D0 0043F          MOVL    R0, RIGHT_ARG
                50          08     AC  D0 00443 52$:     MOVL    LEFT_ARG, R0                                    ; 7518
          FF    8F          03     A0  91 00447          CMPB    3(R0), #255
                                   1C  12 0044C          BNEQ    53$
          FF    8F          17     A0  91 0044E          CMPB    23(R0), #255                                    ; 7519
                                   15  12 00453          BNEQ    53$
          FF    8F          16     A0  91 00455          CMPB    22(R0), #255                                    ; 7520
                                   0E  12 0045A          BNEQ    53$
                            0C     AC  DD 0045C          PUSHL   RIGHT_ARG                                       ; 7522
                                   50  DD 0045F          PUSHL   R0
      0000V     CF                 02  FB 00461          CALLS   #2, FIXUP_EMPTY_SET
          08    AC                 50  D0 00466          MOVL    R0, LEFT_ARG
                27          0C     AE  E9 0046A 53$:     BLBC    BINARY_FLAG, 54$                                ; 7524
                50          0C     AC  D0 0046E          MOVL    RIGHT_ARG, R0                                   ; 7526
          FF    8F          03     A0  91 00472          CMPB    3(R0), #255
                                   1C  12 00477          BNEQ    54$
          FF    8F          17     A0  91 00479          CMPB    23(R0), #255                                    ; 7527
                                   15  12 0047E          BNEQ    54$
          FF    8F          16     A0  91 00480          CMPB    22(R0), #255                                    ; 7528
                                   0E  12 00485          BNEQ    54$
                            08     AC  DD 00487          PUSHL   LEFT_ARG                                        ; 7530
                                   50  DD 0048A          PUSHL   R0
      0000V     CF                 02  FB 0048C          CALLS   #2, FIXUP_EMPTY_SET
          0C    AC                 50  D0 00491          MOVL    R0, RIGHT_ARG
                53          08     AC  D0 00495 54$:     MOVL    LEFT_ARG, R3                                    ; 7536
                                   53  DD 00499          PUSHL   R3
        F962    CF                 01  FB 0049B          CALLS   #1, DBG$DO_MAPPING
                16          0C     AE  E9 004A0          BLBC    BINARY_FLAG, 56$                                ; 7537
                32                 52  D1 004A4          CMPL    OPCODE, #50                                     ; 7539
                                   09  12 004A7          BNEQ    55$
```

```
                        50      0C   AC   D0 004A9          MOVL    RIGHT_ARG, R0                           ; 7540
                                04   A0   95 004AD          TSTB    4(R0)                                   ;
                                     08   19 004B0          BLSS    56$                                     ;
                                0C   AC   DD 004B2 55$:     PUSHL   RIGHT_ARG                               ; 7542
                      F948      CF        01 FB 004B5       CALLS   #1, DBG$DO_MAPPING                      ;
                                          53 DD 004BA 56$:    PUSHL   R3                                      ; 7547
                      0000V     CF        01 FB 004BC       CALLS   #1, DBG$GET_DTYPE                       ;
                                56        50 D0 004C1       MOVL    R0, LEFT_TYPE                           ;
                        18      AE        56 D0 004C4       MOVL    LEFT_TYPE, NEW_LEFT_TYPE                ; 7548
                                11   0C   AE E9 004C8       BLBC    BINARY_FLAG, 57$                        ; 7553
                                0C   AC   DD 004CC         PUSHL   RIGHT_ARG                               ; 7555
                      0000V     CF        01 FB 004CF       CALLS   #1, DBG$GET_DTYPE                       ;
                                58        50 D0 004D4       MOVL    R0, RIGHT_TYPE                          ;
                        14      AE        58 D0 004D7       MOVL    RIGHT_TYPE, NEW_RIGHT_TYPE              ; 7556
                                          05 11 004DB       BRB     58$                                     ; 7553
                                          58 D4 004DD 57$:    CLRL    RIGHT_TYPE                          ; 7563
                                14   AE   D4 004DF         CLRL    NEW_RIGHT_TYPE                          ; 7564
                                38        52 D1 004E2 58$:    CMPL    OPCODE, #56                             ; 7576
                                          19 12 004E5       BNEQ    60$                                     ;
                    OF    04    A3        05 E1 004E7       BBC     #5, 4(R3), 59$                          ; 7579
                                          7E D4 004EC       CLRL    -(SP)                                   ; 7581
                                          53 DD 004EE       PUSHL   R3                                      ;
                                          53 DD 004F0       PUSHL   R3                                      ;
                      F047      CF        03 FB 004F2       CALLS   #3, DBG$CONV_TEXT_VALUE                 ;
                        08      AC        50 D0 004F7       MOVL    R0, LEFT_ARG                            ;
                                50   08   AC D0 004FB 59$:    MOVL    LEFT_ARG, R0                        ; 7583
                                          04 004FF          RET                                             ;
                                03   0C   AE E8 00500 60$:    BLBS    BINARY_FLAG, 61$                    ; 7657
                                     0278 31 00504          BRW     90$                                     ;
                                40        54 E9 00507 61$:    BLBC    R4, 65$                             ; 7624
                                50   0C   AC D0 0050A         MOVL    RIGHT_ARG, R0                       ; 7625
                                     04   A0 95 0050E         TSTB    4(R0)                                   ;
                                          0A 19 00511       BLSS    62$                                     ;
                                0E        56 D1 00513       CMPL    LEFT_TYPE, #14                          ; 7626
                                          2F 12 00516       BNEQ    64$                                     ;
                                16        58 D1 00518       CMPL    RIGHT_TYPE, #22                         ;
                                          2A 12 0051B       BNEQ    64$                                     ;
                                50        08 AC D0 0051D 62$:    MOVL    LEFT_ARG, R0                     ; 7629
                    OF    04    A0        05 E1 00521       BBC     #5, 4(R0), 63$                          ;
                                      0101 8F BB 00526       PUSHR   #^M<R0,R8>                              ; 7631
                                          50 DD 0052A       PUSHL   R0                                      ;
                          FOOD      CF        03 FB 0052C       CALLS   #3, DBG$CONV_TEXT_VALUE             ;
                            08      AC        50 D0 00531       MOVL    R0, LEFT_ARG                        ;
                                00000000'     EF DD 00535 63$:    PUSHL   CVT_ROUND_FLAG                  ; 7633
                                7E        08  AC 7D 0053B       MOVQ    LEFT_ARG, -(SP)                     ;
                00000000G       00        03 FB 0053F       CALLS   #3, DBG$COVER_DX_DX                     ;
                                          04 00546          RET                                             ;
                                05        54 E8 00547 64$:    BLBS    R4, 66$                             ; 7642
                                33        52 D1 0054A 65$:    CMPL    OPCODE, #51                         ; 7643
                                          21 12 0054D       BNEQ    67$                                     ;
                                50        0C AC D0 0054F 66$:    MOVL    RIGHT_ARG, R0                    ; 7646
                                05        06 A0 91 00553       CMPB    6(R0), #5                               ;
                                          17 12 00557       BNEQ    67$                                     ;
                                50        08 AC D0 00559       MOVL    LEFT_ARG, R0                        ; 7647
                                0E        16 A0 91 0055D       CMPB    22(R0), #14                             ;
                                          0D 13 00561       BEQL    67$                                     ;
                                03 00000000G   00 91 00563       CMPB    DBG$GB_LANGUAGE, #3                 ; 7648
```

```
                                04  12 0056A          BNEQ    67$                              ;
          14  AE                30  D0 0056C          MOVL    #48, NEW_RIGHT_TYPE              ; 7650
              08                54  E8 00570 67$:     BLBS    R4, 68$                          ; 7654
              33                52  D1 00573          CMPL    OPCODE, #51                      ;
                                03  13 00576          BEQL    68$                              ;
                              01A0  31 00578          BRW     86$                              ;
                      04        AE  DD 0057B 68$:     PUSHL   ROUT_TBL_SIZE                    ; 7659
                                57  DD 0057E          PUSHL   ROUT_TBL                         ;
                      10        AE  DD 00580          PUSHL   INCOMP_TBL_SIZE                  ; 7658
                    0220        8F  BB 00583          PUSHR   #^M<R5,R9>                       ; 7657
                                5A  DD 00587          PUSHL   HIER_TBL                         ;
                                7E  D4 00589          CLRL    -(SP)                            ; 7656
                      30        AE  DD 0058B          PUSHL   NEW_RIGHT_TYPE                   ;
                      38        AE  DD 0058E          PUSHL   NEW_LEFT_TYPE                    ;
        0000V  CF                   09  FB 00591          CALLS   #9, FIND_PATH_DEPOSIT            ;
               03                50  E8 00596          BLBS    R0, 69$                          ;
                              0208  31 00599          BRW     92$                              ;
               53       0C      AC  D0 0059C 69$:     MOVL    RIGHT_ARG, R3                    ; 7667
               05       06      A3  91 005A0          CMPB    6(R3), #5                        ;
                                17  12 005A4          BNEQ    70$                              ;
               50       08      AC  D0 005A6          MOVL    LEFT_ARG, R0                     ; 7668
               0E       16      A0  91 005AA          CMPB    22(R0), #14                      ;
                                0D  13 005AE          BEQL    70$                              ;
               03 00000000G     00  91 005B0          CMPB    DBG$GB_LANGUAGE, #3              ; 7669
                                04  12 005B7          BNEQ    70$                              ;
          16   A3               30  90 005B9          MOVB    #48, 22(R3)                      ; 7671
               54       14      A3  9E 005BD 70$:     MOVAB   20(R3), R4                       ; 7676
                                6E  D4 005C1          CLRL    (SP)                             ;
               09       03      A4  91 005C3          CMPB    3(R4), #9                        ;
                                0B  12 005C7          BNEQ    71$                              ;
                                6E  D6 005C9          INCL    (SP)                             ;
   04     1E   A3               03  E0 005CB          BBS     #3, 30(R3), 71$                  ; 7677
          14   AE               15  D0 005D0          MOVL    #21, NEW_RIGHT_TYPE              ; 7679
               52       08      AC  D0 005D4 71$:     MOVL    LEFT_ARG, R2                     ; 7686
   56     04   A2               05  E1 005D8          BBC     #5, 4(R2), 75$                   ;
               50 00000000G     00  9A 005DD          MOVZBL  DBG$GB_LANGUAGE, R0              ; 7696
               03               50  91 005E4          CMPB    R0, #3                           ;
                                05  13 005E7          BEQL    72$                              ;
               08               50  91 005E9          CMPB    R0, #8                           ; 7697
                                36  12 005EC          BNEQ    74$                              ;
               03       02      A4  91 005EE 72$:     CMPB    2(R4), #3                        ; 7704
                                12  13 005F2          BEQL    73$                              ;
               04       02      A4  91 005F4          CMPB    2(R4), #4                        ; 7705
                                0C  13 005F8          BEQL    73$                              ;
               05       02      A4  91 005FA          CMPB    2(R4), #5                        ; 7706
                                06  13 005FE          BEQL    73$                              ;
               0F       02      A4  91 00600          CMPB    2(R4), #15                       ; 7707
                                1E  12 00604          BNEQ    74$                              ;
        0042   8F       12      A2  B1 00606 73$:     CMPW    18(R2), #66                      ; 7710
                                16  12 0060C          BNEQ    74$                              ;
          12   A2       43      8F  9B 0060E          MOVZBW  #67, 18(R2)                      ; 7713
                                5B  DD 00613          PUSHL   R11                              ; 7714
                                01  DD 00615          PUSHL   #1                               ;
                    0002874B    8F  DD 00617          PUSHL   #165707                          ;
    00000000G  00               03  FB 0061D          CALLS   #3, LIB$SIGNAL                   ;
                        0104    8F  BB 00624 74$:     PUSHR   #^M<R2,R8>                       ; 7721
                                52  DD 00628          PUSHL   R2                               ;
```

```
                EF0F    CF              03 FB 0062A             CALLS   #3, DBG$CONV_TEXT_VALUE
                  08    AC              50 D0 0062F             MOVL    RO, LEFT_ARG
                        05              6E E9 00633 75$:        BLBC    (SP), 76$                       ;7730
          12      1E    A3              03 E1 00636             BBC     #3, 30(R3), 77$                 ;7731
                        15      02      A4 91 0063B 76$:        CMPB    2(R4), #21                      ;7732
                                        0C 13 0063F             BEQL    77$
                        0F      02      A4 91 00641             CMPB    2(R4), #15                      ;7733
                                        06 13 00645             BEQL    77$
                        30      02      A4 91 00647             CMPB    2(R4), #48                      ;7734
                                        0F 12 0064B             BNEQ    78$
                        7E              15 7D 0064D 77$:        MOVQ    #21, -(SP)                      ;7736
                                08      AC DD 00650             PUSHL   LEFT_ARG
                0000V   CF              03 FB 00653             CALLS   #3, INTMED_DATA_FOR_DEP
                  08    AC              50 D0 00658             MOVL    RO, LEFT_ARG
                        50 00000000G    00 9A 0065C 78$:        MOVZBL  DBG$GB_LANGUAGE, RO             ;7750
                        03              50 91 00663             CMPB    RO, #3
                                        05 13 00666             BEQL    79$
                        08              50 91 00668             CMPB    RO, #8                          ;7751
                                        50 12 0066B             BNEQ    81$
                        03      02      A4 91 0066D 79$:        CMPB    2(R4), #3                       ;7758
                                        12 13 00671             BEQL    80$
                        04      02      A4 91 00673             CMPB    2(R4), #4                       ;7759
                                        0C 13 00677             BEQL    80$
                        05      02      A4 91 00679             CMPB    2(R4), #5                       ;7760
                                        06 13 0067D             BEQL    80$
                        0F      02      A4 91 0067F             CMPB    2(R4), #15                      ;7761
                                        38 12 00683             BNEQ    81$
                        50      08      AC D0 00685 80$:        MOVL    LEFT_ARG, RO                    ;7762
                        15      16      A0 91 00689             CMPB    22(RO), #21
                                        2E 12 0068D             BNEQ    81$
                        51      08      AC D0 0068F             MOVL    LEFT_ARG, R1                    ;7770
                        50      14      A1 3C 00693             MOVZWL  20(R1), DIG
                        50              02 C6 00697             DIVL2   #2, RO                          ;7772
                        50      18      A1 C0 0069A             ADDL2   24(R1), RO
          51            60      F2      8F 8B 0069E             BICB3   #-14, (SIGN_POS), SIGN          ;7777
                        0D              51 91 006A3             CMPB    SIGN, #13                       ;7778
                                        15 12 006A6             BNEQ    81$
                        60      01      8F 8A 006A8             BICB2   #-255, (SIGN_POS)               ;7781
                                        5B DD 006AC             PUSHL   R11                             ;7782
                                        01 DD 006AE             PUSHL   #1
                            0002874B    8F DD 006B0             PUSHL   #165707
                00000000G   00          03 FB 006B6             CALLS   #3, LIB$SIGNAL
                        50 00000000G    00 9A 006BD 81$:        MOVZBL  DBG$GB_LANGUAGE, RO             ;7806
                        03              50 91 006C4             CMPB    RO, #3
                                        05 13 006C7             BEQL    82$
                        08              50 91 006C9             CMPB    RO, #8                          ;7807
                                        24 12 006CC             BNEQ    83$
                        21              6E E9 006CE 82$:        BLBC    (SP), 83$                       ;7810
                                        01 DD 006D1             PUSHL   #1                              ;7821
                                        15 DD 006D3             PUSHL   #21
                                        53 DD 006D5             PUSHL   R3
                  0000V CF              03 FB 006D7             CALLS   #3, INTMED_DATA_FOR_DEP
                        52              50 D0 006DC             MOVL    RO, NEW_RIGHT_ARG
                        53              52 D1 006DF             CMPL    NEW_RIGHT_ARG, R3               ;7829
                                        0E 13 006E2             BEQL    83$
                                        52 DD 006E4             PUSHL   NEW_RIGHT_ARG                   ;7832
                                08      AC DD 006E6             PUSHL   LEFT_ARG
```

```
0000V   CF                  02  FB 006E9        CALLS   #2, DBG$TYPE_CONV                 ;
   08   AC                  50  D0 006EE        MOVL    R0, LEFT_ARG                      ;
                            53  DD 006F2 83$:   PUSHL   R3                                ; 7837
                      08    AC  DD 006F4        PUSHL   LEFT_ARG                          ;
0000V   CF                  02  FB 006F7        CALLS   #2, DBG$TYPE_CONV                 ;
   0C   AC                  50  D0 006FC        MOVL    R0, RIGHT_ARG                     ;
        05            06    A0  91 00700        CMPB    6(R0), #5                         ; 7839
                            0D  12 00704        BNEQ    84$                               ;
        30            16    A0  91 00706        CMPB    22(R0), #48                       ; 7840
                            07  12 0070A        BNEQ    84$                               ;
        03 00000000G        00  91 0070C        CMPB    DBG$GB_LANGUAGE, #3               ; 7841
                            01  13 00713 84$:   BEQL    85$                               ;
                                04 00715        RET                                       ;
   16   A0                  0E  90 00716 85$:   MOVB    #14, 22(R0)                       ; 7843
                                04 0071A        RET                                       ; 7845
        04 00000000G        00  91 0071B 86$:   CMPB    DBG$GB_LANGUAGE, #4               ; 7853
                            2F  12 00722        BNEQ    89$                               ;
        15            18    AE  D1 00724        CMPL    NEW_LEFT_TYPE, #21                ; 7856
                            10  12 00728        BNEQ    87$                               ;
                      14    AE  DD 0072A        PUSHL   NEW_RIGHT_TYPE                    ; 7858
        50            08    AC  D0 0072D        MOVL    LEFT_ARG, R0                      ;
        7E            14    A0  3C 00731        MOVZWL  20(R0), -(SP)                     ;
                      20    AE  9F 00735        PUSHAB  NEW_LEFT_TYPE                     ;
                            14  11 00738        BRB     88$                               ;
        15            14    AE  D1 0073A 87$:   CMPL    NEW_RIGHT_TYPE, #21               ; 7860
                            13  12 0073E        BNEQ    89$                               ;
                      18    AE  DD 00740        PUSHL   NEW_LEFT_TYPE                     ; 7862
        50            0C    AC  D0 00743        MOVL    RIGHT_ARG, R0                     ;
        7E            14    A0  3C 00747        MOVZWL  20(R0), -(SP)                     ;
                      1C    AE  9F 0074B        PUSHAB  NEW_RIGHT_TYPE                    ;
0000V   CF                  03  FB 0074E 88$:   CALLS   #3, MAP_PACKED                    ;
                      04    AE  DD 00753 89$:   PUSHL   ROUT_TBL_SIZE                     ; 7873
                            57  DD 00756        PUSHL   ROUT_TBL                          ;
                      10    AE  DD 00758        PUSHL   INCOMP_TBL_SIZE                   ; 7872
                    0220    8F  BB 0075B        PUSHR   #^M<R5,R9>                        ; 7871
                            5A  DD 0075F        PUSHL   HIER_TBL                          ;
               00000000'    EF  DD 00761        PUSHL   MAX_DEPTH                         ; 7870
                            7E  D4 00767        CLRL    -(SP)                             ; 7866
                      30    AE  9F 00769        PUSHAB  ROUT_TBL_INDEX                    ;
                      38    AE  9F 0076C        PUSHAB  NEW_RIGHT_TYPE                    ;
                      40    AE  9F 0076F        PUSHAB  NEW_LEFT_TYPE                     ;
                      40    AE  DD 00772        PUSHL   NEW_RIGHT_TYPE                    ; 7867
                      48    AE  DD 00775        PUSHL   NEW_LEFT_TYPE                     ;
0000V   CF                  0D  FB 00778        CALLS   #13, FIND_JOIN                    ;
                            1C  11 0077D        BRB     91$                               ; 7866
                      04    AE  DD 0077F 90$:   PUSHL   ROUT_TBL_SIZE                     ; 7888
                            57  DD 00782        PUSHL   ROUT_TBL                          ;
                      10    AE  DD 00784        PUSHL   INCOMP_TBL_SIZE                   ; 7887
                    0220    8F  BB 00787        PUSHR   #^M<R5,R9>                        ; 7886
                            5A  DD 0078B        PUSHL   HIER_TBL                          ;
                      28    AE  9F 0078D        PUSHAB  ROUT_TBL_INDEX                    ; 7883
                      34    AE  9F 00790        PUSHAB  NEW_LEFT_TYPE                     ;
                      38    AE  DD 00793        PUSHL   NEW_LEFT_TYPE                     ; 7884
0000V   CF                  09  FB 00796        CALLS   #9, FIND_PATH                     ;
        50 00000000'        EF  D1 0079B 91$:   CMPL    MAX_DEPTH, R0                     ; 7883
                            11  12 007A2        BNEQ    93$                               ;
                            5B  DD 007A4 92$:   PUSHL   R11                               ; 7890
```

```
                              01  DD 007A6       PUSHL   #1                                   ;
                 000289CA     8F  DD 007A8       PUSHL   #166346                              ;
        00000000G  00         03  FB 007AE       CALLS   #3, LIB$SIGNAL                       ;
                   2C      0C AE  E9 007B5 93$:  BLBC    BINARY_FLAG, 94$                     ; 7902
                   05 00000000G 00 91 007B9      CMPB    DBG$GB_LANGUAGE, #5                  ; 7903
                              23  12 007C0       BNEQ    94$                                  ;
                           04 AE  DD 007C2       PUSHL   ROUT_TBL_SIZE                        ; 7909
                              57  DD 007C5       PUSHL   ROUT_TBL                             ;
                           10 AE  DD 007C7       PUSHL   INCOMP_TBL_SIZE                      ; 7908
                         0220 8F  BB 007CA       PUSHR   #^M<R5,R9>                           ; 7907
                              5A  DD 007CE       PUSHL   HIER_TBL                             ;
                           28 AE  9F 007D0       PUSHAB  ROUT_TBL_INDEX                       ; 7905
                           30 AE  9F 007D3       PUSHAB  NEW_RIGHT_TYPE                       ;
                           38 AE  9F 007D6       PUSHAB  NEW_LEFT_TYPE                        ;
                         0140 8F  BB 007D9       PUSHR   #^M<R6,R8>                           ;
                           04 AC  DD 007DD       PUSHL   OPERATOR                             ;
            0000V  CF         0C  FB 007E0       CALLS   #12, MODIFY_PLI_TARGET_TYPE          ;
                   50      08 AC  D0 007E5 94$:  MOVL    LEFT_ARG, R0                         ; 7914
     10     04     A0         05  E1 007E9       BBC     #5, 4(R0), 95$                       ;
                           18 AE  DD 007EE       PUSHL   NEW_LEFT_TYPE                        ; 7916
                              50  DD 007F1       PUSHL   R0                                   ;
                              50  DD 007F3       PUSHL   R0                                   ;
             ED44  CF         03  FB 007F5       CALLS   #3, DBG$CONV_TEXT_VALUE              ;
               08  AC         50  D0 007FA       MOVL    R0, LEFT_ARG                         ;
                   53      08 AC  D0 007FE 95$:  MOVL    LEFT_ARG, R3                         ; 7927
               18  AE         56  D1 00802       CMPL    LEFT_TYPE, NEW_LEFT_TYPE             ; 7923
                              03  12 00806       BNEQ    96$                                  ;
                            0089  31 00808       BRW     102$                                 ;
                   7E      14 A3  3C 0080B 96$:  MOVZWL  20(R3), -(SP)                        ; 7927
                           1C AE  DD 0080F       PUSHL   NEW_LEFT_TYPE                        ; 7926
                              56  DD 00812       PUSHL   LEFT_TYPE                            ;
            0000V  CF         03  FB 00814       CALLS   #3, GET_DATA_LENGTH                  ;
                   55         50  D0 00819       MOVL    R0, LENGTH                           ;
                              59  D4 0081C       CLRL    SD_CLASS_FLAG                        ; 7928
                   50 00000000G 00 9A 0081E      MOVZBL  DBG$GB_LANGUAGE, R0                  ; 7929
                   03         50  91 00825       CMPB    R0, #3                               ;
                              05  13 00828       BEQL    97$                                  ;
                   08         50  91 0082A       CMPB    R0, #8                               ; 7930
                              09  12 0082D       BNEQ    98$                                  ;
                   09      17 A3  91 0082F 97$:  CMPB    23(R3), #9                           ; 7933
                              03  12 00833       BNEQ    98$                                  ;
                   59         01  D0 00835       MOVL    #1, SD_CLASS_FLAG                    ; 7935
                         0208 8F  BB 00838 98$:  PUSHR   #^M<R3,R9>                           ; 7940
                              55  DD 0083C       PUSHL   LENGTH                               ; 7939
                           24 AE  DD 0083E       PUSHL   NEW_LEFT_TYPE                        ; 7938
            0000V  CF         04  FB 00841       CALLS   #4, MAKE_VAL_DESC                    ;
                   5A         50  D0 00846       MOVL    R0, NEW_LEFT_ARG                     ;
                   09      17 AA  91 00849       CMPB    23(NEW_LEFT_ARG), #9                 ; 7943
                              37  12 0084D       BNEQ    101$                                 ;
                   0A      16 A3  91 0084F       CMPB    22(R3), #10                          ; 7952
                              12  13 00853       BEQL    99$                                  ;
                   0B      16 A3  91 00855       CMPB    22(R3), #11                          ; 7953
                              0C  13 00859       BEQL    99$                                  ;
                   1B      16 A3  91 0085B       CMPB    22(R3), #27                          ; 7954
                              06  13 0085F       BEQL    99$                                  ;
                   1C      16 A3  91 00861       CMPB    22(R3), #28                          ; 7955
                              1A  12 00865       BNEQ    100$                                 ;
```

```
                                    1C   AE   9F 00867 99$:    PUSHAB  DIGITS                                  ; 7958
                                    53   DD 0086A              PUSHL   R3                                      ;
                0000V  CF                02   FB 0086C              CALLS   #2, GET_SCALE                           ;
                  1C   AA                50   90 00871              MOVB    R0, 28(NEW_LEFT_ARG)                    ;
                  1D   AA          1C    AE   90 00875              MOVB    DIGITS, 29(NEW_LEFT_ARG)                ; 7959
                  14   AA          1C    AE   B0 0087A              MOVW    DIGITS, 20(NEW_LEFT_ARG)                ; 7960
                                    05   11 0087F              BRB     101$                                    ; 7952
                  1C   AA          1C    A3   90 00881 100$:    MOVB    28(R3), 28(NEW_LEFT_ARG)                ; 7964
                                  0408   8F   BB 00886 101$:    PUSHR   #^M<R3,R10>                             ; 7967
                0000V  CF                02   FB 0088A              CALLS   #2, DBG$TYPE_CONV                       ;
                       5A                50   D0 0088F              MOVL    R0, NEW_LEFT_ARG                        ;
                                    03   11 00892              BRB     103$                                    ; 7923
                       5A                53   D0 00894 102$:    MOVL    R3, NEW_LEFT_ARG                        ; 7975
                       19          0C    AE   E9 00897 103$:    BLBC    BINARY_FLAG, 104$                       ; 7980
                       50          0C    AC   D0 0089B              MOVL    RIGHT_ARG, R0                           ; 7982
              10         04   A0          05   E1 0089F              BBC     #5, 4(R0), 104$                         ;
                                    14   AE   DD 008A4              PUSHL   NEW_RIGHT_TYPE                          ; 7984
                                    50   DD 008A7              PUSHL   R0                                      ;
                                    50   DD 008A9              PUSHL   R0                                      ;
                0EC8E  CF                03   FB 008AB              CALLS   #3, DBG$CONV_TEXT_VALUE                 ;
                  0C   AC                50   D0 008B0              MOVL    R0, RIGHT_ARG                           ;
                       03          0C    AE   E8 008B4 104$:    BLBS    BINARY_FLAG, 105$                       ; 7989
                                 0099   31 008B8              BRW     113$                                    ;
                       53          0C    AC   D0 008BB 105$:    MOVL    RIGHT_ARG, R3                           ; 7995
                  14   AE                58   D1 008BF              CMPL    RIGHT_TYPE, NEW_RIGHT_TYPE              ; 7991
                                    03   12 008C3              BNEQ    106$                                    ;
                                 0089   31 008C5              BRW     112$                                    ;
                       7E         14    A3   3C 008C8 106$:    MOVZWL  20(R3), -(SP)                           ; 7995
                                  18    AE   DD 008CC              PUSHL   NEW_RIGHT_TYPE                          ; 7994
                                    58   DD 008CF              PUSHL   RIGHT_TYPE                              ;
              0000V  CF                03   FB 008D1              CALLS   #3, GET_DATA_LENGTH                     ;
                       55                50   D0 008D6              MOVL    R0, LENGTH                              ;
                                    59   D4 008D9              CLRL    SD_CLASS_FLAG                           ; 7996
                       50 00000000G      00   9A 008DB              MOVZBL  DBG$GB_LANGUAGE, R0                     ; 7997
                       03                50   91 008E2              CMPB    R0, #3                                  ;
                                    05   13 008E5              BEQL    107$                                    ;
                       08                50   91 008E7              CMPB    R0, #8                                  ; 7998
                                    09   12 008EA              BNEQ    108$                                    ;
                       09         17    A3   91 008EC 107$:    CMPB    23(R3), #9                              ; 8001
                                    03   12 008F0              BNEQ    108$                                    ;
                       59                01   D0 008F2              MOVL    #1, SD_CLASS_FLAG                       ; 8003
                                  0208   8F   BB 008F5 108$:    PUSHR   #^M<R3,R9>                              ; 8008
                                    55   DD 008F9              PUSHL   LENGTH                                  ; 8007
                                  20    AE   DD 008FB              PUSHL   NEW_RIGHT_TYPE                          ; 8006
                0000V  CF                04   FB 008FE              CALLS   #4, MAKE_VAL_DESC                       ;
                       52                50   D0 00903              MOVL    R0, NEW_RIGHT_ARG                       ;
                       09         17    A2   91 00906              CMPB    23(NEW_RIGHT_ARG), #9                   ; 8011
                                    37   12 0090A              BNEQ    111$                                    ;
                       0A         16    A3   91 0090C              CMPB    22(R3), #10                             ; 8019
                                    12   13 00910              BEQL    109$                                    ;
                       0B         16    A3   91 00912              CMPB    22(R3), #11                             ; 8020
                                    0C   13 00916              BEQL    109$                                    ;
                       1B         16    A3   91 00918              CMPB    22(R3), #27                             ; 8021
                                    06   13 0091C              BEQL    109$                                    ;
                       1C         16    A3   91 0091E              CMPB    22(R3), #28                             ; 8022
                                    1A   12 00922              BNEQ    110$                                    ;
                                  1C    AE   9F 00924 109$:    PUSHAB  DIGITS                                  ; 8025
```

```
                              53  DD 00927            PUSHL   R3                                             ;
           0000V  CF          02  FB 00929            CALLS   #2, GET_SCALE                                  ;
              1C  A2          50  90 0092E            MOVB    R0, 28(NEW_RIGHT_ARG)                          ;
              1D  A2     1C   AE  90 00932            MOVB    DIGITS, 29(NEW_RIGHT_ARG)                      ; 8026
              14  A2     1C   AE  B0 00937            MOVW    DIGITS, 20(NEW_RIGHT_ARG)                      ; 8027
                              05  11 0093C            BRB     111$                                           ; 8019
              1C  A2     1C   A3  90 0093E 110$:      MOVB    28(R3), 28(NEW_RIGHT_ARG)                      ; 8031
                              52  DD 00943 111$:      PUSHL   NEW_RIGHT_ARG                                  ; 8034
                              53  DD 00945            PUSHL   R3                                             ;
           0000V  CF          02  FB 00947            CALLS   #2, DBG$TYPE_CONV                              ;
                  52          50  D0 0094C            MOVL    R0, NEW_RIGHT_ARG                              ;
                              03  11 0094F            BRB     113$                                           ; 7991
                  52          53  D0 00951 112$:      MOVL    R3, NEW_RIGHT_ARG                              ; 8038
                  50     10   AE  D0 00954 113$:      MOVL    ROUT_TBL_INDEX, R0                             ; 8041
                      02 A740 7F  00958               PUSHAQ  2(ROUT_TBL)[R0]                                ;
                  53          9E  3C 0095C            MOVZWL  @(SP)+, ROUT_INDEX                             ;
                      04 A740 7F  0095F               PUSHAQ  4(ROUT_TBL)[R0]                                ; 8042
                  54          9E  3C 00963            MOVZWL  @(SP)+, TYPEID_INDEX                           ;
                      06 A740 7F  00966               PUSHAQ  6(ROUT_TBL)[R0]                                ; 8043
                  56          9E  9A 0096A            MOVZBL  @(SP)+, RESULT_TYPE                            ;
                  09     17   AA  91 0096D            CMPB    23(NEW_LEFT_ARG), #9                           ; 8044
                              0A  13 00971            BEQL    114$                                           ;
                  0B     0C   AE  E9 00973            BLBC    BINARY_FLAG, 115$                              ; 8045
                  09     17   A2  91 00977            CMPB    23(NEW_RIGHT_ARG), #9                          ;
                              05  12 0097B            BNEQ    115$                                           ;
                  59          01  D0 0097D 114$:      MOVL    #1, SD_CLASS_FLAG                              ; 8047
                              02  11 00980            BRB     116$                                           ;
                              59  D4 00982 115$:      CLRL    SD_CLASS_FLAG                                  ; 8049
                              55  D4 00984 116$:      CLRL    LENGTH                                         ; 8051
        00000052  8F          53  D1 00986            CMPL    ROUT_INDEX, #82                                ; 8052
                              09  13 0098D            BEQL    117$                                           ;
        000000FE  8F          53  D1 0098F            CMPL    ROUT_INDEX, #254                               ; 8053
                              0B  12 00996            BNEQ    118$                                           ;
                  55     14   AA  3C 00998 117$:      MOVZWL  20(NEW_LEFT_ARG), LENGTH                       ; 8056
                  50     14   A2  3C 0099C            MOVZWL  20(NEW_RIGHT_ARG), R0                          ;
                  55          50  C0 009A0            ADDL2   R0, LENGTH                                     ;
                              55  D5 009A3 118$:      TSTL    LENGTH                                         ; 8058
                              33  12 009A5            BNEQ    121$                                           ;
                  01          56  D1 009A7            CMPL    RESULT_TYPE, #1                                ; 8061
                              0F  13 009AA            BEQL    119$                                           ;
                  22          56  D1 009AC            CMPL    RESULT_TYPE, #34                               ; 8062
                              0A  13 009AF            BEQL    119$                                           ;
                  29          56  D1 009B1            CMPL    RESULT_TYPE, #41                               ; 8063
                              05  13 009B4            BEQL    119$                                           ;
                  2A          56  D1 009B6            CMPL    RESULT_TYPE, #42                               ; 8064
                              15  12 009B9            BNEQ    120$                                           ;
                  55     14   AA  3C 009BB 119$:      MOVZWL  20(NEW_LEFT_ARG), LENGTH                       ; 8067
                  17     0C   AE  E9 009BF            BLBC    BINARY_FLAG, 121$                              ; 8068
              14  AA     14   A2  B1 009C3            CMPW    20(NEW_RIGHT_ARG), 20(NEW_LEFT_ARG)            ; 8072
                              10  1B 009C8            BLEQU   121$                                           ;
                  55     14   A2  3C 009CA            MOVZWL  20(NEW_RIGHT_ARG), LENGTH                      ; 8074
                              0A  11 009CE            BRB     121$                                           ; 8061
                              56  DD 009D0 120$:      PUSHL   RESULT_TYPE                                    ; 8080
           0000V  CF          01  FB 009D2            CALLS   #1, DBG$NUM_BYTES                              ;
                  55          50  D0 009D7            MOVL    R0, LENGTH                                     ;
                              59  DD 009DA 121$:      PUSHL   SD_CLASS_FLAG                                  ; 8087
                              7E  D4 009DC            CLRL    -(SP)                                          ; 8084
```

```
                                        55  DD 009DE          PUSHL   LENGTH                                 ; 8085
                                        56  DD 009E0          PUSHL   RESULT_TYPE                            ; 8084
                      0000V  CF         04  FB 009E2          CALLS   #4, MAKE_VAL_DESC                      ;
                         20  AE         50  D0 009E7          MOVL    R0, RESULT                             ;
                  24         01    16   A0  8F 009EB          CASEB   22(R0), #1, #36                        ; 8093
     0050         0050     0050       0050     009F0 122$:    .WORD   124$-122$,-                            ;
     0050         0050     0050       0050     009F8                  124$-122$,-                            ;
     004C         004C     004C       0050     00A00                  124$-122$,-                            ;
     0050         0050     0050       004C     00A08                  124$-122$,-                            ;
     0050         0050     0050       0050     00A10                  124$-122$,-                            ;
     0050         0050     0050       0050     00A18                  124$-122$,-                            ;
     004C         004C     0050       0050     00A20                  124$-122$,-                            ;
     0050         0050     004C       004C     00A28                  124$-122$,-                            ;
     0050         0050     0050       0050     00A30                  124$-122$,-                            ;
                                      0050     00A38                  123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      123$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$,-                            ;
                                                                      124$-122$                              ;
                                        04  11 00A3A          BRB     124$                                   ;
                         17  A0         01  90 00A3C 123$:    MOVB    #1, 23(R0)                             ; 8098
                                   20   AE  9F 00A40 124$:    PUSHAB  RESULT                                 ; 8112
                             04    10   AE  E9 00A43          BLBC    BINARY_FLAG, 125$                      ; 8115
                                        52  DD 00A47          PUSHL   NEW_RIGHT_ARG                          ; 8116
                                        02  11 00A49          BRB     126$                                   ;
                                        7E  D4 00A4B 125$:    CLRL    -(SP)                                  ; 8115
                                 0408   8F  BB 00A4D 126$:    PUSHR   #^M<R3,R10>                            ; 8113
                                   04   AC  DD 00A51          PUSHL   OPERATOR                               ; 8112
                  00000000G  00         05  FB 00A54          CALLS   #5, DBG$PERFORM_OPERATOR               ;
                                        54  D5 00A5B          TSTL    TYPEID_INDEX                           ; 8124
                                        22  13 00A5D          BEQL    127$                                   ;
                                   20   AE  DD 00A5F          PUSHL   RESULT                                 ; 8128
                                        52  DD 00A62          PUSHL   NEW_RIGHT_ARG                          ;
                                 0410   8F  BB 00A64          PUSHR   #^M<R4,R10>                            ; 8127
```

```
                     0000V  CF             04  FB 00A68          CALLS   #4, DBG$PERFORM_TYPEID_CHECK                ;
                            11             50  E8 00A6D          BLBS    R0, 127$                                   ;
                                           5B  DD 00A70          PUSHL   R11                                        ; 8130
                                           01  DD 00A72          PUSHL   #1                                         ;
                                 000289CA  8F  DD 00A74          PUSHL   #166346                                    ;
                00000000G  00            03  FB 00A7A          CALLS   #3, LIB$SIGNAL                             ;
                            50       20  AE  D0 00A81 127$:     MOVL    RESULT, R0                                 ; 8137
                                             04 00A85          RET                                                ; 8138

; Routine Size:  2694 bytes,    Routine Base:  DBG$CODE + 0B81
```

```
;  8037          8139  1 GLOBAL ROUTINE DBG$GET_SET_TYPEID(TYPEID, PARENT_TYPEID) =
;  8038          8140  1 !
;  8039          8141  1 ! FUNCTION
;  8040          8142  1 !       This routine is a recursive routine to get the parent type for SET data
;  8041          8143  1 !       type.
;  8042          8144  1 !
;  8043          8145  1 ! INPUTS
;  8044          8146  1 !       PARENT_TYPEID - Parent typeid. (only used for the subrange type).
;  8045          8147  1 !       TYPEID  - Typeid for a given set data.
;  8046          8148  1 !
;  8047          8149  1 ! OUTPUTS
;  8048          8150  1 !       Pointer to parent typeid of set data.
;  8049          8151  1 !
;  8050          8152  1
;  8051          8153  2     BEGIN
;  8052          8154  2
;  8053          8155  2     MAP
;  8054          8156  2         PARENT_TYPEID: REF VECTOR[1],
;  8055          8157  2         TYPEID: REF RST$ENTRY;
;  8056          8158  2
;  8057          8159  2     LOCAL
;  8058          8160  2         PARENT_TYPE,
;  8059          8161  2         HIGHPTR,
;  8060          8162  2         LOWPTR,
;  8061          8163  2         SIZE;
;  8062          8164  2
;  8063          8165  2
;  8064          8166  2     CASE .TYPEID[RST$B_FCODE] FROM RST$K_TYPE_MINIMUM
;  8065          8167  2                              TO   RST$K_TYPE_MAXIMUM OF
;  8066          8168  2         SET
;  8067          8169  2         [RST$K_TYPE_SET]:
;  8068          8170  3             BEGIN
;  8069          8171  3             DBG$STA_TYP_SET(.TYPEID, PARENT_TYPE, SIZE);
;  8070          8172  3             TYPEID = DBG$GET_SET_TYPEID(.PARENT_TYPE, PARENT_TYPEID[0]);
;  8071          8173  3             RETURN .TYPEID;
;  8072          8174  2             END;
;  8073          8175  2
;  8074          8176  2         [RST$K_TYPE_ATOMIC, RST$K_TYPE_DESCR, RST$K_TYPE_ENUM]:
;  8075          8177  2             RETURN .TYPEID;
;  8076          8178  2
;  8077          8179  2         [RST$K_TYPE_SUBRNG]:
;  8078          8180  3             BEGIN
;  8079          8181  3             PARENT_TYPEID[0] = .TYPEID;
;  8080          8182  3             DBG$STA_TYP_SUBRNG(.TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR, SIZE);
;  8081          8183  3             TYPEID = DBG$GET_SET_TYPEID(.PARENT_TYPE, PARENT_TYPEID[0]);
;  8082          8184  3             RETURN .TYPEID;
;  8083          8185  2             END;
;  8084          8186  2
;  8085          8187  2         [INRANGE]:
;  8086          8188  2             $DBG_ERROR('DBGEVALOP\DBG$GET_SET_TYPEID, fcode cannot be set type');
;  8087          8189  2         TES;
;  8088          8190  2
;  8089          8191  2     RETURN 0;
;  8090          8192  1     END;
```

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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 36 05E99 P.AMC:   .ASCII  \6DBGEVALOP\<92>\DBG$GET_SET_TYPEID, fco\   ;
2C 44 49 45 50 59 54 5F 54 45 53 5F 54 45 47 05EA8                                                                 ;
                                      6F 63 66 20 05EB7                                                              ;
65 73 20 65 62 20 74 6F 6E 6E 61 63 20 65 64 05EBB          .ASCII  \de cannot be set type\                         ;
                              65 70 79 74 20 74 05ECA                                                                ;


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

                                            0004 00000          .ENTRY  DBG$GET_SET_TYPEID, Save R2             ; 8139
                              5E          10  C2 00002          SUBL2   #16, SP                                 ;
                              52      04  AC  D0 00005          MOVL    TYPEID, R2                              ; 8166
                    15        01      18  A2  8F 00009          CASEB   24(R2), #1, #21                         ;
      003C        003C      003C        0067    0000E 1$.       .WORD   6$-1$,-                                 ;
      002C        0067      0067        0067    00016                   3$-1$,-                                 ;
      0067        0067      0067        0040    0001E                   3$-1$,-                                 ;
      0067        0067      0067        0067    00026                   3$-1$,-                                 ;
      0067        0067      0067        0067    0002E                   6$-1$,-                                 ;
                            0067        0067    00036                   6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        2$-1$,-                                 ;
                                                                        4$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$,-                                 ;
                                                                        6$-1$                                   ;
                                          5E  DD 0003A 2$:      PUSHL   SP                                      ; 8171
                                      10  AE  9F 0003C          PUSHAB  PARENT_TYPE                             ;
                                          52  DD 0003F          PUSHL   R2                                      ;
                   00000000G  00          03  FB 00041          CALLS   #3, DBG$STA_TYP_SET                     ;
                                          1C  11 00048          BRB     5$                                      ; 8172
                              50          52  DC 0004A 3$:      MOVL    R2, R0                                  ; 8177
                                              04 0004D          RET                                             ;
                          08  BC          52  D0 0004E 4$:      MOVL    R2, @PARENT_TYPEID                      ; 8181
                                          5E  DD 00052          PUSHL   SP                                      ; 8182
                                      08  AE  9F 00054          PUSHAB  HIGHPTR                                 ;
                                      10  AE  9F 00057          PUSHAB  LOWPTR                                  ;
                                      18  AE  9F 0005A          PUSHAB  PARENT_TYPE                             ;
                                          52  DD 0005D          PUSHL   R2                                      ;
                   00000000G  00          05  FB 0005F          CALLS   #5, DBG$STA_TYP_SUBRNG                  ;
                                      08  AC  DD 00066 5$:      PUSHL   PARENT_TYPEID                           ; 8183
                                      10  AE  DD 00069          PUSHL   PARENT_TYPE                             ;
                          90  AF          02  FB 0006C          CALLS   #2, DBG$GET_SET_TYPEID                  ;
                          04  AC          50  D0 00070          MOVL    R0, TYPEID                              ;
                                              04 00074          RET                                             ; 8184
```

```
                          00000000'  EF  9F 00075 6$:     PUSHAB  P.AMC                           ; 8188
                                     01  DD 0007B         PUSHL   #1                              ;
                          00028362   8F  DD 0007D         PUSHL   #164706                        ;
                00000000G  00        03  FB 00083         CALLS   #3, LIB$SIGNAL                 ;
                                     50  D4 0008A         CLRL    R0                             ; 8191
                                         04 0008C         RET                                    ; 8192

; Routine Size:  141 bytes,    Routine Base:  DBG$CODE + 1607
```

```
;  8092          8193  1 GLOBAL ROUTINE DBG$GET_DTYPE (VALDESC) =
;  8093          8194  1 !
;  8094          8195  1 ! FUNCTION
;  8095          8196  1 !       Given a Value Descriptor, this routine returns a 1-byte type code
;  8096          8197  1 !       for the descriptor. This type code is ordinarily just taken from
;  8097          8198  1 !       the dtype field, but if the fcode is not atomic or descriptor, the
;  8098          8199  1 !       1-byte type code is DBG$K_MAXIMUM_DTYPE + FCODE.
;  8099          8200  1 !
;  8100          8201  1 ! INPUT
;  8101          8202  1 !       VALDESC -       A pointer to the value descriptor for which a
;  8102          8203  1 !                       type code is desired.
;  8103          8204  1 !
;  8104          8205  1 ! OUTPUT
;  8105          8206  1 !       A type code is returned.
;  8106          8207  1 !
;  8107          8208  2     BEGIN
;  8108          8209  2     MAP
;  8109          8210  2         VALDESC: REF DBG$VALDESC;
;  8110          8211  2
;  8111          8212  2
;  8112          8213  2     ! Check for data. Note that record components A.B come back with
;  8113          8214  2     ! KIND = TYPCOMP, and these are also data.
;  8114          8215  2     !
;  8115          8216  2     IF .VALDESC[DBG$B_DHDR_KIND] EQL RST$K_DATA
;  8116          8217  2     OR .VALDESC[DBG$B_DHDR_KIND] EQL RST$K_TYPCOMP
;  8117          8218  2     THEN
;  8118          8219  3         BEGIN
;  8119          8220  3
;  8120          8221  3         ! Cobol and Pli picture are treated differently, in COBOL, picture
;  8121          8222  3         ! is treated as T, in PLI, we want to have FCODE + 200 as the dtype.
;  8122          8223  3         !
;  8123          8224  3         IF .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT
;  8124          8225  3         THEN
;  8125          8226  4             BEGIN
;  8126          8227  4             IF .DBG$GB_LANGUAGE EQL DBG$K_COBOL
;  8127          8228  4             THEN
;  8128          8229  4                 RETURN .VALDESC [DBG$B_VALUE_DTYPE]
;  8129          8230  4             ELSE
;  8130          8231  4                 RETURN .VALDESC [DBG$B_DHDR_FCODE] + DBG$K_MAXIMUM_DTYPE;
;  8131          8232  3             END;
;  8132          8233  3
;  8133          8234  3
;  8134          8235  3         ! The argument is data. Check the fcode.
;  8135          8236  3         !
;  8136          8237  3         IF .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC
;  8137          8238  3         OR .VALDESC [DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR
;  8138          8239  3         THEN
;  8139          8240  3
;  8140          8241  3             ! Special case for ADA fixed point and PL/1 fixed binary.
;  8141          8242  3             ! These come in as
;  8142          8243  3             ! class SD with the BINSCALE bit set. Return a special
;  8143          8244  3             ! dtype for "FIXED" in this case.
;  8144          8245  3             !
;  8145          8246  3             IF (.VALDESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD) AND
;  8146          8247  4                (.VALDESC[DBG$V_VALUE_FL_BINSCALE])
;  8147          8248  3             THEN
;  8148          8249  3                 RETURN DSC$K_DTYPE_FIXED
```

```
;  8149          8250  3
;  8150          8251  3            ELSE
;  8151          8252  3
;  8152          8253  3                ! VAX standard type.
;  8153          8254  3                !
;  8154          8255  3                RETURN .VALDESC [DBG$B_VALUE_DTYPE]
;  8155          8256  3
;  8156          8257  3            ELSE
;  8157          8258  3
;  8158          8259  3                ! Not VAX standard. Represent type as 200 + fcode.
;  8159          8260  3                !
;  8160          8261  3                RETURN .VALDESC [DBG$B_DHDR_FCODE] + DBG$K_MAXIMUM_DTYPE
;  8161          8262  3            END
;  8162          8263  3
;  8163          8264  2        ELSE
;  8164          8265  2
;  8165          8266  2            ! Other kinds include RST$K_ROUTINE, RST$K_LINE, etc.
;  8166          8267  2            ! Some languages (BLISS, MACRO) allow operations on code addresses,
;  8167          8268  2            ! e.g., EVAL SUBR + 100 where SUBR is a subroutine.
;  8168          8269  2            ! In this case, we expect dtype to be ZI or ZEM.
;  8169          8270  2            !
;  8170          8271  2            IF .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_ROUTINE
;  8171          8272  2            OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_BLOCK
;  8172          8273  2            OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_LABEL
;  8173          8274  2            OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_LINE
;  8174          8275  2            OR .VALDESC [DBG$B_DHDR_KIND] EQL RST$K_ENTRY
;  8175          8276  2            THEN
;  8176          8277  2
;  8177          8278  2                ! The type should be ZI or ZEM unless an override was present.
;  8178          8279  2                !
;  8179          8280  2                RETURN .VALDESC [DBG$B_VALUE_DTYPE]
;  8180          8281  2
;  8181          8282  2            ELSE
;  8182          8283  2
;  8183          8284  2                ! Any other kind is an error.
;  8184          8285  2                !
;  8185          8286  3                BEGIN
;  8186          8287  3                $DBG_ERROR ('DBGEVALOP\DBG$GET_TYPECODE unknown rst kind');
;  8187          8288  3                RETURN 0;
;  8188          8289  2                END;
;  8189          8290  1        END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 2B  05ED0 P.AMD:    .ASCII  \+DBGEVALOP\<92>\DBG$GET_TYPECODE unknow\   ;
6E 75 20 45 44 4F 43 45 50 59 54 5F 54 45 47  05EDF                                                          ;
                           77 6F 6E 6B  05EEE                                                                 ;
                  64 6E 69 6B 20 74 73 72 20 6E  05EF2          .ASCII  \n rst kind\                           ;


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

                                                 000C 00000     .ENTRY  DBG$GET_DTYPE, Save R2,R3              ; 8193
                        50      04   AC   D0 00002             MOVL    VALDESC, R0                            ; 8216
```

```
                    51        07   A0   9A 00006         MOVZBL  7(R0), R1                        ;
                    06             51   91 0000A         CMPB    R1, #6                           ;
                                   05   13 0000D         BEQL    1$                               ;
                    0A             51   91 0000F         CMPB    R1, #10                          ; 8217
                                   43   12 00012         BNEQ    7$                               ;
                    52        06   A0   9A 00014 1$:     MOVZBL  6(R0), R2                        ; 8224
                    05             52   91 00018         CMPB    R2, #5                           ;
                                   1A   12 0001B         BNEQ    4$                               ;
                    03 00000000G   00   91 0001D         CMPB    DBG$GB_LANGUAGE, #3              ; 8227
                                   06   12 00024         BNEQ    2$                               ;
                    51        16   A0   9A 00026         MOVZBL  22(R0), R1                       ; 8231
                                   07   11 0002A         BRB     3$                               ;
                    53        2B   A2   9E 0002C 2$:     MOVAB   43(R2), R3                       ;
                    51             53   D0 00030         MOVL    R3, R1                           ;
                    50             51   D0 00033 3$:     MOVL    R1, R0                           ;
                                   04 00036              RET                                      ;
                    02             52   91 00037 4$:     CMPB    R2, #2                           ; 8237
                                   05   13 0003A         BEQL    5$                               ;
                    03             52   91 0003C         CMPB    R2, #3                           ; 8238
                                   0F   12 0003F         BNEQ    6$                               ;
                    09        17   A0   91 00041 5$:     CMPB    23(R0), #9                       ; 8246
                                   29   12 00045         BNEQ    8$                               ;
          24        1E   A0        03   E1 00047         BBC     #3, 30(R0), 8$                   ; 8247
                    50             2B   D0 0004C         MOVL    #43, R0                          ; 8249
                                   04 0004F              RET                                      ;
                    52             2B   C0 00050 6$:     ADDL2   #43, R2                          ; 8261
                    50             52   D0 00053         MOVL    R2, R0                           ;
                                   04 00056              RET                                      ; 8271
                    02             51   91 00057 7$:     CMPB    R1, #2                           ;
                                   14   13 0005A         BEQL    8$                               ;
                    03             51   91 0005C         CMPB    R1, #3                           ; 8272
                                   0F   13 0005F         BEQL    8$                               ;
                    04             51   91 00061         CMPB    R1, #4                           ; 8273
                                   0A   13 00064         BEQL    8$                               ;
                    05             51   91 00066         CMPB    R1, #5                           ; 8274
                                   05   13 00069         BEQL    8$                               ;
                    08             51   91 0006B         CMPB    R1, #8                           ; 8275
                                   05   12 0006E         BNEQ    9$                               ;
                    50        16   A0   9A 00070 8$:     MOVZBL  22(R0), R0                       ; 8280
                                   04 00074              RET                                      ;
                       00000000'   EF   9F 00075 9$:     PUSHAB  P.AMD                            ; 8287
                                   01   DD 0007B         PUSHL   #1                               ;
                       00028362    8F   DD 0007D         PUSHL   #164706                          ;
             00000000G  00              03   FB 00083    CALLS   #3, LIB$SIGNAL                   ;
                                   50   D4 0008A         CLRL    R0                               ; 8288
                                   04 0008C              RET                                      ; 8290

; Routine Size:  141 bytes,    Routine Base:  DBG$CODE + 1694
```

```
;  8191          8291  1 ROUTINE DBG$LANGUAGE_TYPE_CONV (CVT_ROUT_INDEX, VALUE1, VALUE2) =
;  8192          8292  1 !
;  8193          8293  1 ! FUNCTION
;  8194          8294  1 !       Performs language-specific type conversion on the given
;  8195          8295  1 !       descriptors, according to the routine index given
;  8196          8296  1 !       by CVT_ROUT_INDEX.
;  8197          8297  1 !
;  8198          8298  1 ! INPUTS
;  8199          8299  1 !       CVT_ROUT_INDEX  - A routine index indicating which operation
;  8200          8300  1 !                         is to be performed. The possible values for
;  8201          8301  1 !                         this index are given in DBGLIB under CVT$K_
;  8202          8302  1 !       VALUE1          - DEBUG value descriptor for the source
;  8203          8303  1 !       VALUE2          - DEBUG value descriptor for the target
;  8204          8304  1 !
;  8205          8305  1 ! OUTPUTS
;  8206          8306  1 !       A pointer to the result descriptor is returned.
;  8207          8307  1 !
;  8208          8308  2     BEGIN
;  8209          8309  2
;  8210          8310  2     MAP
;  8211          8311  2         VALUE1 : REF DBG$VALDESC,       ! Pointer to the source value descriptor
;  8212          8312  2         VALUE2 : REF DBG$VALDESC;       ! Pointer to the target value descriptor
;  8213          8313  2
;  8214          8314  2
;  8215          8315  2     ! Case on the routine index.
;  8216          8316  2     !
;  8217          8317  2     CASE .CVT_ROUT_INDEX FROM CVT$K_MIN_ROUT TO CVT$K_MAX_ROUT OF
;  8218          8318  2         SET
;  8219          8319  2
;  8220          8320  2         [CVT$K_PLI_CVT]:
;  8221          8321  2             VALUE2 = PLI_TYPE_CONV(.VALUE1, .VALUE2);
;  8222          8322  2
;  8223          8323  2         [CVT$K_COB_PICT]:
;  8224          8324  3             BEGIN
;  8225          8325  3             LOCAL
;  8226          8326  3                 BUFFER: VECTOR[4, LONG],! Buffer area
;  8227          8327  3                 CHANGE_SCALE: BYTE,     ! Change of scale
;  8228          8328  3                 LANGCODE,               ! Language code
;  8229          8329  3                 PICTPTR: REF VECTOR[,BYTE],
;  8230          8330  3                                         ! Pointer to picture representation
;  8231          8331  3                 PICTVAL,                ! Pointer to language specific encoding
;  8232          8332  3                 PSCALE: VECTOR[2, BYTE];! Digits and Scale
;  8233          8333  3
;  8234          8334  3
;  8235          8335  3             IF .VALUE2[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_PICT
;  8236          8336  3             THEN
;  8237          8337  3                 $DBG_ERROR('DBGEVALOP\DBG$LANGUAGE_TYPE_CONV, fcode neq pict');
;  8238          8338  3
;  8239          8339  3             DBG$STA_TYP_PICT(.VALUE2[DBG$L_DHDR_TYPEID], LANGCODE, PICTPTR,
;  8240          8340  3                              PICTVAL, PSCALE);
;  8241          8341  3             CASE .LANGCODE FROM DBG$K_MIN_LANGUAGE TO DBG$K_MAX_LANGUAGE OF
;  8242          8342  3                 SET
;  8243          8343  3                 [DBG$K_COBOL]:
;  8244          8344  4                     BEGIN
;  8245          8345  4                     CHANGE_SCALE = .VALUE1[DBG$B_VALUE_SCALE] - .PSCALE[0];
;  8246          8346  4                     ASHP(CHANGE_SCALE, VALUE1[DBG$W_VALUE_LENGTH],
;  8247          8347  4                         .VALUE1[DBG$L_VALUE_POINTER], %REF (0), PSCALE[1],
```

```
;  8248          8348  4                        BUFFER);
;  8249          8349  4                    EDITPC(PSCALE[1], BUFFER, .PICTVAL, .VALUE2[DBG$L_VALUE_POINTER]);
;  8250          8350  3                    END;
;  8251          8351  3
;  8252          8352  3                [INRANGE, OUTRANGE]:
;  8253          8353  3                    SIGNAL(DBG$_UNIMPLENT);
;  8254          8354  3
;  8255          8355  3                TES;
;  8256          8356  3
;  8257          8357  2            END;
;  8258          8358  2
;  8259          8359  2        [INRANGE, OUTRANGE]:
;  8260          8360  2            $DBG_ERROR ('DBGEVALOP\DBG$LANGUAGE_TYPE_CONV unknown routine index');
;  8261          8361  2
;  8262          8362  2        TES;
;  8263          8363  2
;  8264          8364  2    RETURN .VALUE2;
;  8265          8365  1    END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 30  05EFC P.AME:   .ASCII  \0DBGEVALOP\<92>\DBG$LANGUAGE_TYPE_CONV,\      ;
43 5F 45 50 59 54 5F 45 47 41 55 47 4E 41 4C  05F0B                                                               ;
                              2C 56 4E 4F     05F1A                                                               ;
74 63 69 70 20 71 65 6E 20 65 64 6F 63 66 20  05F1E          .ASCII  \ fcode neg pict\                            ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 36  05F2D P.AMF:   .ASCII  \6DBGEVALOP\<92>\DBG$LANGUAGE_TYPE_CONV \     ;
43 5F 45 50 59 54 5F 45 47 41 55 47 4E 41 4C  05F3C                                                               ;
                              20 56 4E 4F     05F4B                                                               ;
65 6E 69 74 75 6F 72 20 6E 77 6F 6E 6B 6E 75  05F4F          .ASCII  \unknown routine index\                      ;
                        78 65 64 6E 69 20     05F5E                                                               ;


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

                                    007C 00000 DBG$LANGUAGE_TYPE_CONV:
                                                .WORD   Save R2,R3,R4,R5,R6                                     ; 8291
                 56 00000000G   00  9E 00002            MOVAB   LIB$SIGNAL, R6                                  ;
                 5E             20  C2 00009            SUBL2   #32, SP                                         ;
01               01       04    AC  CF 0000C            CASEL   CVT_ROUT_INDEX, #1, #1                          ; 8317
              0026          0017     00011 1$:          .WORD   2$-T$,-                                         ;
                                                                3$-1$                                           ;
                 00000000'      EF  9F 00015            PUSHAB  P.AMF                                           ; 8360
                                01  DD 0001B            PUSHL   #1                                              ;
                 00028362       8F  DD 0001D            PUSHL   #164706                                         ;
                 66             03  FB 00023            CALLS   #3, LIB$SIGNAL                                  ;
                                64  11 00026            BRB     7$                                              ;
                 7E       08    AC  7D 00028 2$:        MOVQ    VALUE1, -(SP)                                   ; 8321
           0000V CF             02  FB 0002C            CALLS   #2, PLI_TYPE_CONV                               ;
              0C AC             50  D0 00031            MOVL    R0, VALUE2                                      ;
                                75  11 00035            BRB     9$                                              ;
                 54       0C    AC  D0 00037 3$:        MOVL    VALUE2, R4                                      ; 8335
                 05       06    A4  91 0003B            CMPB    6(R4), #5                                       ;
                                11  13 0003F            BEQL    4$                                              ;
                 00000000'      EF  9F 00041            PUSHAB  P.AME                                           ; 8337
```

```
                                                01  DD 00047          PUSHL   #1                                              ;
                                      00028362  8F  DD 00049          PUSHL   #164706                                         ;
                                             66   03  FB 0004F          CALLS   #3, LIB$SIGNAL                                  ;
                                                  0C  AE  9F 00052 4$:    PUSHAB  PSCALE                                       ; 8339
                                                  04  AE  9F 00055       PUSHAB  PICTVAL                                      ;
                                                  0C  AE  9F 00058       PUSHAB  PICTPTR                                      ;
                                                  14  AE  9F 0005B       PUSHAB  LANGCODE                                     ;
                                                  08  A4  DD 0005E       PUSHL   8(R4)                                        ;
                                  00000000G       00  05  FB 00061       CALLS   #5, DBG$STA_TYP_PICT                         ;
                                          0A          00  08  AE  CF 00068    CASEL   LANGCODE, #0, #10                           ; 8341
                0021            0016            0016        0016  0006D 5$:    .WORD   6$-5$,-                                  ;
                0016            0016            0016        0016  00075           6$-5$,-                                  ;
                                0016            0016        0016  0007D           6$-5$,-                                  ;
                                                                                  8$-5$,-                                  ;
                                                                                  6$-5$,-                                  ;
                                                                                  6$-5$,-                                  ;
                                                                                  6$-5$,-                                  ;
                                                                                  6$-5$,-                                  ;
                                                                                  6$-5$,-                                  ;
                                                                                  6$-5$                                    ;
                                      00028800  8F  DD 00083 6$:    PUSHL   #165888                                         ; 8353
                                             66   01  FB 00089          CALLS   #1, LIB$SIGNAL                                  ;
                                                  1E  11 0008C 7$:       BRB     9$                                           ;
                                             50       08  AC  D0 0008E 8$:    MOVL    VALUE1, R0                               ; 8345
                                   51        1C      A0  0C  AE  83 00092       SUBB3   PSCALE, 28(R0), CHANGE_SCALE             ;
                00      18        B0        14      A0      51  F8 00098       ASHP    CHANGE_SCALE, 20(R0), @24(R0), #0, -     ; 8347
                                                  10  AE      0D  AE  0009F           PSCALE+1, BUFFER                         ;
                18      B4        00        BE      10  AE  0D  AE  38 000A3   EDITPC  PSCALE+1, BUFFER, @PICTVAL, @24(R4)      ; 8349
                                             50       0C  AC  D0 000AC 9$:    MOVL    VALUE2, R0                               ; 8364
                                                          04 000B0           RET                                              ; 8365

; Routine Size:  177 bytes,    Routine Base:  DBG$CODE + 1721
```

```
;  8267          8366  1 GLOBAL ROUTINE DBG$MAP_DTYPE_CLASS(IN_TYPE, FLAG) =
;  8268          8367  1 !
;  8269          8368  1 ! FUNCTION
;  8270          8369  1 !       This routine is used to obtain the class for the given dtype.
;  8271          8370  1 !       (When DEBUG override type qualifier is given, the class field
;  8272          8371  1 !       in the VMS value descriptor is zero, in order to call LIB$CVT_DX_DX,
;  8273          8372  1 !       class field must be supplied.)
;  8274          8373  1 !
;  8275          8374  1 ! INPUT
;  8276          8375  1 !       IN_TYPE         - a dtype code
;  8277          8376  1 !       FLAG            - if the flag is set that means we take SD as class
;  8278          8377  1 !                         instead of S.
;  8279          8378  1 !
;  8280          8379  1 ! OUTPUT
;  8281          8380  1 !       CLASS is returned.
;  8282          8381  1 !
;  8283          8382  2     BEGIN
;  8284          8383  2     RETURN
;  8285          8384  3         (CASE .IN_TYPE  FROM 0 TO DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_MAXIMUM OF
;  8286          8385  3         SET
;  8287          8386  3         [DSC$K_DTYPE_Z]:
;  8288          8387  3                 0;
;  8289          8388  3         [DSC$K_DTYPE_V,DSC$K_DTYPE_BU,DSC$K_DTYPE_WU,DSC$K_DTYPE_LU,
;  8290          8389  3          DSC$K_DTYPE_QU,DSC$K_DTYPE_B,DSC$K_DTYPE_W,DSC$K_DTYPE_L,
;  8291          8390  3          DSC$K_DTYPE_Q,DSC$K_DTYPE_F,DSC$K_DTYPE_D,DSC$K_DTYPE_FC,
;  8292          8391  3          DSC$K_DTYPE_DC,DSC$K_DTYPE_T,DSC$K_DTYPE_NU,DSC$K_DTYPE_NL,
;  8293          8392  3          DSC$K_DTYPE_NLO,DSC$K_DTYPE_NR,DSC$K_DTYPE_NRO,DSC$K_DTYPE_NZ,
;  8294          8393  3          DSC$K_DTYPE_ZI,DSC$K_DTYPE_ZEM,DSC$K_DTYPE_DSC,DSC$K_DTYPE_ADT,
;  8295          8394  3          DSC$K_DTYPE_OU,DSC$K_DTYPE_O,DSC$K_DTYPE_G,DSC$K_DTYPE_H,
;  8296          8395  3          DSC$K_DTYPE_GC,DSC$K_DTYPE_HC,DSC$K_DTYPE_CIT,DSC$K_DTYPE_BPV,
;  8297          8396  3          DSC$K_DTYPE_BLV,DSC$K_DTYPE_SV]:
;  8298          8397  3                 IF .FLAG THEN DSC$K_CLASS_SD ELSE DSC$K_CLASS_S;
;  8299          8398  3         [DSC$K_DTYPE_VU,DSC$K_DTYPE_TF,DSC$K_DTYPE_SVU]:
;  8300          8399  3                 DSC$K_CLASS_UBS;
;  8301          8400  3         [DSC$K_DTYPE_VT,DSC$K_DTYPE_AC,DSC$K_DTYPE_AZ]:
;  8302          8401  3                 DSC$K_CLASS_VS;
;  8303          8402  3         [DSC$K_DTYPE_P]:
;  8304          8403  3                 DSC$K_CLASS_SD;
;  8305          8404  3         [DSC$K_DTYPE_FIXED]:
;  8306          8405  3                 DSC$K_CLASS_SD;
;  8307          8406  3         [INRANGE]:
;  8308          8407  3                 DSC$K_CLASS_Z;
;  8309          8408  3         [OUTRANGE]:
;  8310          8409  3                 $DBG_ERROR ('DBGEVALOP\DBG$MAP_DTYPE_CLASS');
;  8311          8410  2         TES);
;  8312          8411  1     END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 1D  05F64 P.AMG:    .ASCII  <29>\DBGEVALOP\<92>\DBG$MAP_DTYPE_CLAS\     ;
   53 41 4C 43 5F 45 50 59 54 44 5F 50 41 4D  05F73                                                           ;
                                              53  05F81        .ASCII  \S\                                      ;
```

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

                                                  0000 00000        .ENTRY  DBG$MAP_DTYPE_CLASS, Save nothing        ; 8366
         00000041    8F        00       04    AC    CF 00002        CASEL   IN_TYPE, #0, #65                         ; 8384
    009A        009A      009A        00AE       0000B 1$:          .WORD   6$-1$,-                                  ;
    009A        009A      009A        009A       00013                      2$-1$,-                                  ;
    009A        009A      009A        009A       0001B                      2$-1$,-                                  ;
    009A        009A      009A        009A       00023                      2$-1$,-                                  ;
    009A        009A      009A        009A       0002B                      2$-1$,-                                  ;
    009A        009A      00AA        009A       00033                      2$-1$,-                                  ;
    009A        009A      009A        009A       0003B                      2$-1$,-                                  ;
    009A        009A      009A        009A       00043                      2$-1$,-                                  ;
    009A        00A2      009A        009A       0004B                      2$-1$,-                                  ;
    00A6        00A6      00A6        00AE       00053                      2$-1$,-                                  ;
    00AA        00A2      009A        00A2       0005B                      2$-1$,-                                  ;
    00AE        00AE      00AE        00AE       00063                      2$-1$,-                                  ;
    00AE        00AE      00AE        00AE       0006B                      2$-1$,-                                  ;
    00AE        00AE      00AE        00AE       00073                      2$-1$,-                                  ;
    00AE        00AE      00AE        00AE       0007B                      2$-1$,-                                  ;
    00AE        00AE      00AE        00AE       00083                      2$-1$,-                                  ;
                          00AE        00AE       0008B                      2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            5$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            3$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            4$-1$,-                                  ;
                                                                            4$-1$,-                                  ;
                                                                            4$-1$,-                                  ;
                                                                            3$-1$,-                                  ;
                                                                            2$-1$,-                                  ;
                                                                            3$-1$,-                                  ;
                                                                            5$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
                                                                            6$-1$,-                                  ;
```

```
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$,-
                                                                                6$-1$
                        00000000'  EF  9F 0008F          PUSHAB  P.AMG                  ;8409
                                   01  DD 00095          PUSHL   #1
                        00028362   8F  DD 00097          PUSHL   #164706
              00000000G  00        03  FB 0009D          CALLS   #3, LIB$SIGNAL
                                   04  000A4  RET
                     0C       08   AC  E8 000A5  2$:     BLBS    FLAG, 5$               ;8397
                     50            01  D0 000A9          MOVL    #1, R0
                                   04  000AC  RET
                     50            0D  D0 000AD  3$:     MOVL    #13, R0                ;8384
                                   04  000B0  RET
                     50            0B  D0 000B1  4$:     MOVL    #11, R0
                                   04  000B4  RET
                     50            09  D0 000B5  5$:     MOVL    #9, R0
                                   04  000B8  RET
                                   50  D4 000B9  6$:     CLRL    R0
                                   04  000BB  RET                                    ;8411

; Routine Size:  188 bytes,    Routine Base:  DBG$CODE + 17D2
```

```
;  8314          8412  1 GLOBAL ROUTINE DBG$NUM_BYTES(IN_TYPE) =
;  8315          8413  1 !
;  8316          8414  1 !  FUNCTION
;  8317          8415  1 !        This routine is used to obtain the number of bytes of storage that
;  8318          8416  1 !        are needed to hold data of a given dtype.
;  8319          8417  1 !
;  8320          8418  1 !  INPUT
;  8321          8419  1 !        IN_TYPE         - a dtype code
;  8322          8420  1 !
;  8323          8421  1 !  OUTPUT
;  8324          8422  1 !        The number of bytes of storage needed to hold data of type dtype
;  8325          8423  1 !        is returned.
;  8326          8424  1 !
;  8327          8425  2     BEGIN
;  8328          8426  2     RETURN
;  8329          8427  3         (CASE .IN_TYPE  FROM 0 TO DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_MAXIMUM OF
;  8330          8428  3         SET
;  8331          8429  3             [DSC$K_DTYPE_Z]         : 4;
;  8332          8430  3             [DSC$K_DTYPE_BU]        : 1;
;  8333          8431  3             [DSC$K_DTYPE_B]         : 1;
;  8334          8432  3             [DSC$K_DTYPE_WU]        : 2;
;  8335          8433  3             [DSC$K_DTYPE_W]         : 2;
;  8336          8434  3             [DSC$K_DTYPE_LU]        : 4;
;  8337          8435  3             [DSC$K_DTYPE_L]         : 4;
;  8338          8436  3             [DSC$K_DTYPE_QU]        : 4;
;  8339          8437  3             [DSC$K_DTYPE_Q]         : 8;
;  8340          8438  3             [DSC$K_DTYPE_OU]        : 16;
;  8341          8439  3             [DSC$K_DTYPE_O]         : 16;
;  8342          8440  3             [DSC$K_DTYPE_F]         : 4;
;  8343          8441  3             [DSC$K_DTYPE_D]         : 8;
;  8344          8442  3             [DSC$K_DTYPE_G]         : 8;
;  8345          8443  3             [DSC$K_DTYPE_H]         : 16;
;  8346          8444  3             [DSC$K_DTYPE_FC]        : 8;
;  8347          8445  3             [DSC$K_DTYPE_DC]        : 16;
;  8348          8446  3             [DSC$K_DTYPE_GC]        : 16;
;  8349          8447  3             [DSC$K_DTYPE_HC]        : 32;
;  8350          8448  3             [DSC$K_DTYPE_P]         : 31;
;  8351          8449  3             [DSC$K_DTYPE_TF]        : 1;
;  8352          8450  3             [DSC$K_DTYPE_FIXED]     : 4;
;  8353          8451  3             [DBG$K_DTYPE_ENUM]      : 4;
;  8354          8452  3             [DBG$K_DTYPE_TPTR]      : 4;
;  8355          8453  3             [DBG$K_DTYPE_PTR]       : 4;
;  8356          8454  3             [INRANGE]               : 64;   ! Default to 64 bytes
;  8357          8455  3             [OUTRANGE]              : $DBG_ERROR ('DBGEVALOP\DBG$NUM_BYTES');
;  8358          8456  2         TES);
;  8359          8457  1     END;


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

24  47  42  44  5C  50  4F  4C  41  56  45  47  42  44  17  05F82 P.AMH:  .ASCII  <23>\DBGEVALOP\<92>\DBG$NUM_BYTES\          ;
                    53  45  54  59  42  5F  4D  55  4E  05F91                                                               ;


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

```
                                                C000 00000          .ENTRY  DBG$NUM_BYTES, Save nothing                     ; 8412
        00000041    8F      00      04    AC    CF 00002          CASEL   IN_TYPE, #0, #65                                ; 8427
009A            00AE    00B6            00B2      0000B 1$:       .WORD   8$-1$,-
009A            00AE    00B2            00B2      00013                   9$-1$,-
009E            00B2    009E            00B2      0001B                   7$-1$,-
00B6            00B6    00A2            009E      00023                   2$-1$,-
00B6            00B6    00B6            00B6      0002B                   8$-1$,-
00B6            00B6    00AA            00B6      00033                   8$-1$,-
009E            00A2    00A2            00B6      0003B                   7$-1$,-
00B6            00A6    00A2            00A2      00043                   2$-1$,-
00B6            00B6    00B6            00B6      0004B                   8$-1$,-
00B6            00B6    00B6            00B6      00053                   3$-1$,-
00B2            00B6    00B6            00AE      0005B                   8$-1$,-
00B2            00B6    00B6            00B6      00063                   3$-1$,-
00B6            00B6    00B2            00B6      0006B                   3$-1$,-
00B6            00B6    00B6            00B6      00073                   4$-1$,-
00B2            00B6    00B6            00B6      0007B                   9$-1$,-
00B6            00B6    00B6            00B6      00083                   9$-1$,-
                        00B6            00B6      0008B                   9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          6$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          4$-1$,-
                                                                          9$-1$,-
                                                                          4$-1$,-
                                                                          4$-1$,-
                                                                          3$-1$,-
                                                                          4$-1$,-
                                                                          4$-1$,-
                                                                          5$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          7$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          8$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          8$-1$,-
                                                                          9$-1$,-
                                                                          8$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
                                                                          9$-1$,-
```

```
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              8$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$,-                          ;
                                                                              9$-1$                            ;
                              00000000'  EF  9F 0008F          PUSHAB  P.AMH                                   ; 8455
                                         01  DD 00095          PUSHL   #1                                      ;
                              00028362   8F  DD 00097          PUSHL   #164706                                 ;
               00000000G  00             03  FB 0009D          CALLS   #3, LIB$SIGNAL                          ;
                                             04 000A4          RET                                             ;
                          50             02  D0 000A5 2$:      MOVL    #2, R0                                  ; 8427
                                             04 000A8          RET                                             ;
                          50             08  D0 000A9 3$:      MOVL    #8, R0                                  ;
                                             04 000AC          RET                                             ;
                          50             10  D0 000AD 4$:      MOVL    #16, R0                                 ;
                                             04 000B0          RET                                             ;
                          50             20  D0 000B1 5$:      MOVL    #32, R0                                 ;
                                             04 000B4          RET                                             ;
                          50             1F  D0 000B5 6$:      MOVL    #31, R0                                 ;
                                             04 000B8          RET                                             ;
                          50             01  D0 000B9 7$:      MOVL    #1, R0                                  ;
                                             04 000BC          RET                                             ;
                          50             04  D0 000BD 8$:      MOVL    #4, R0                                  ;
                                             04 000C0          RET                                             ;
                          50       40    8F  9A 000C1 9$:      MOVZBL  #64, R0                                 ;
                                             04 000C5          RET                                             ; 8457

; Routine Size:  198 bytes,    Routine Base:  DBG$CODE + 188E
```

```
;  8361          8458  1 GLOBAL ROUTINE DBG$PERFORM_TYPEID_CHECK(TYPE_INDEX, LEFT_ARG, RIGHT_ARG, RESULT) =
;  8362          8459  1 !
;  8363          8460  1 ! FUNCTION
;  8364          8461  1 !       This routine performs type check on the arguments according to
;  8365          8462  1 !       the given type index.  This routine requires to have left argument,
;  8366          8463  1 !       and one of the right argument or result, or both.  For non-atomic data
;  8367          8464  1 !       items, left argument, right argument must have TYPEID.  For non-atomic
;  8368          8465  1 !       data item, if the result's typeid is zero, then TYPEID is taken from
;  8369          8466  1 !       left argument. (This only should occur after MAKE_VAL_DESC is called
;  8370          8467  1 !       to create the result value descriptor).  TYPEID check will be performed
;  8371          8468  1 !       on left argument and right argument for non-atomic data types.  And/Or
;  8372          8469  1 !       TYPEID check will be performed on left argument and result, and also
;  8373          8470  1 !       range check is done on the result.
;  8374          8471  1 !
;  8375          8472  1 !       This routine is called from DEPOSIT command with left_arg (src) and
;  8376          8473  1 !       right_arg (dst) to perform the typeid check before the DEPOSIT.
;  8377          8474  1 !       Then it is also called called from DEPOSIT command with left_arg
;  8378          8475  1 !       (src) and result (dst) to perform the range check after the DEPOSIT.
;  8379          8476  1 !
;  8380          8477  1 !       This routine is called from EV command to have standard left_arg,
;  8381          8478  1 !       or right_arg (operands) and result.
;  8382          8479  1 !
;  8383          8480  1 ! INPUTS
;  8384          8481  1 !       TYPE_INDEX      - Type check index.
;  8385          8482  1 !
;  8386          8483  1 !       LEFT_ARG        - Left argument value descriptor.  This argument
;  8387          8484  1 !                         must have typeid for non-atomic data types.
;  8388          8485  1 !
;  8389          8486  1 !       RIGHT_ARG       - Right argument value descriptor.  This argument
;  8390          8487  1 !                         must have typeid for non-atomic data types.  This
;  8391          8488  1 !                         argument may not present.
;  8392          8489  1 !
;  8393          8490  1 !       RESULT          - Result argument value descriptor.  This argument
;  8394          8491  1 !                         may not present.  Its typeid may be zero for non-
;  8395          8492  1 !                         atomic data types.
;  8396          8493  1 !
;  8397          8494  1 ! OUTPUTS
;  8398          8495  1 !       Returned status:
;  8399          8496  1 !
;  8400          8497  1 !       TRUE or FALSE
;  8401          8498  1 !
;  8402          8499  1
;  8403          8500  2     BEGIN
;  8404          8501  2
;  8405          8502  2     MAP
;  8406          8503  2         LEFT_ARG: REF DBG$VALDESC,      ! Pointer to left argument
;  8407          8504  2         RIGHT_ARG: REF DBG$VALDESC,     ! Pointer to right argument
;  8408          8505  2         RESULT: REF DBG$VALDESC;        ! Pointer to rusult argument
;  8409          8506  2
;  8410          8507  2     LOCAL
;  8411          8508  2         DUMMY1,
;  8412          8509  2         DUMMY2,
;  8413          8510  2         LEFT_DTYPE,                     ! Left arg's data type
;  8414          8511  2         LEFT_FCODE,                     ! Left arg's fcode
;  8415          8512  2         LEFT_TYPEID: REF RST$ENTRY,     ! Left arg's typeid
;  8416          8513  2         PARENT_TYPE: REF RST$ENTRY,     ! Parent typeid for subrange
;  8417          8514  2         RESULT_DTYPE,                   ! Result's data type
```

```
;  8418          8515  2          RESULT_FCODE,                   ! Result's fcode
;  8419          8516  2          RESULT_TYPEID: REF RST$ENTRY,   ! Result's typeid
;  8420          8517  2          RIGHT_DTYPE,                    ! Right arg's data type
;  8421          8518  2          RIGHT_FCODE,                    ! Right arg's fcode
;  8422          8519  2          RIGHT_TYPEID: REF RST$ENTRY,    ! Right arg's typeid
;  8423          8520  2          SIZE,                           ! The size of the elements
;  8424          8521  2          STATUS;                         ! TRUE or FALSE
;  8425          8522  2
;  8426          8523  2
;  8427          8524  2      ! Make sure left argument is supplied.
;  8428          8525  2      !
;  8429          8526  2      IF .LEFT_ARG EQL 0
;  8430          8527  2      THEN
;  8431          8528  2          $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK');
;  8432          8529  2
;  8433          8530  2
;  8434          8531  2      ! Make sure left arg has TYPEID for non-atomic data types.
;  8435          8532  2      !
;  8436          8533  3      IF NOT ((.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
;  8437          8534  3              (.LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR))
;  8438          8535  2      THEN
;  8439          8536  2          IF .LEFT_ARG[DBG$L_DHDR_TYPEID] EQL 0
;  8440          8537  2          THEN
;  8441          8538  2              $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, no typeid for non-atomic data');
;  8442          8539  2
;  8443          8540  2
;  8444          8541  2      ! If there is right argument, make sure right arg has TYPEID for non-atomic
;  8445          8542  2      ! data types.  (ie, for unary operand).
;  8446          8543  2      !
;  8447          8544  2      IF .RIGHT_ARG NEQ 0
;  8448          8545  2      THEN
;  8449          8546  3          BEGIN
;  8450          8547  4          IF NOT ((.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
;  8451          8548  4                  (.RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR))
;  8452          8549  3          THEN
;  8453          8550  3              IF .RIGHT_ARG[DBG$L_DHDR_TYPEID] EQL 0
;  8454          8551  3              THEN
;  8455          8552  3                  $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, no typeid for non-atomic data');
;  8456          8553  3
;  8457          8554  2          END;
;  8458          8555  2
;  8459          8556  2
;  8460          8557  2      ! One may only want to perform TYPEID check on the operands.  In this
;  8461          8558  2      ! case, there is no need to have result operand.
;  8462          8559  2      !
;  8463          8560  2      IF .RIGHT_ARG EQL 0 AND .RESULT EQL 0
;  8464          8561  2      THEN
;  8465          8562  2          $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK');
;  8466          8563  2
;  8467          8564  2
;  8468          8565  2      ! Perform type check.
;  8469          8566  2      !
;  8470          8567  2      STATUS = TRUE;
;  8471          8568  2      PARENT_TYPE = 0;
;  8472          8569  2      CASE .TYPE_INDEX FROM ORT$K_TYPEID_MIN_ROUT TO ORT$K_TYPEID_MAX_ROUT OF
;  8473          8570  2          SET
;  8474          8571  2          [ORT$K_TYPEID_ENUM_ENUM]:
```

```
;  8475          8572  3          BEGIN
;  8476          8573  3          IF .RIGHT_ARG NEQ 0
;  8477          8574  3          THEN
;  8478          8575  4              BEGIN
;  8479          8576  4
;  8480          8577  4
;  8481          8578  4              ! If this is called from DEPOSIT command, if RESULT is not
;  8482          8579  4              ! presented, this must be one of the sanity check to make
;  8483          8580  4              ! sure DEPOSIT is possible.  Note: DRINK = 1 is valid
;  8484          8581  4              ! case, so in here we allow this case to be legal.
;  8485          8582  4              ! (LEFT_ARG is the source, RIGHT_ARG is the target).
;  8486          8583  4              !
;  8487          8584  4              IF .LEFT_ARG[DBG$L_DHDR_TYPEID] NEQ 0
;  8488          8585  4              THEN
;  8489          8586  5                  BEGIN
;  8490          8587  5                  STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$L_DHDR_TYPEID],
;  8491          8588  5                                             .RIGHT_ARG[DBG$L_DHDR_TYPEID]);
;  8492          8589  5                  END
;  8493          8590  5
;  8494          8591  4              ELSE
;  8495          8592  5                  BEGIN
;  8496          8593  5                  IF .RESULT EQL 0
;  8497          8594  5                  THEN
;  8498          8595  6                      BEGIN
;  8499          8596  7                      IF (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
;  8500          8597  7                          .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
;  8501          8598  6                      THEN
;  8502          8599  6                          STATUS = TRUE
;  8503          8600  6                      ELSE
;  8504          8601  6                          STATUS = FALSE;
;  8505          8602  5                      END;
;  8506          8603  4                  END;
;  8507          8604  3              END;
;  8508          8605  3
;  8509          8606  3
;  8510          8607  3          ! If there is no result, or result is atomic data, or left and
;  8511          8608  3          ! right TYPEID check failed, returns.
;  8512          8609  3          !
;  8513          8610  3          IF (.RESULT EQL 0) OR
;  8514          8611  3             (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
;  8515          8612  3             (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR) OR
;  8516          8613  3             NOT .STATUS
;  8517          8614  3          THEN
;  8518          8615  3              RETURN .STATUS;
;  8519          8616  3
;  8520          8617  3
;  8521          8618  3          ! Perform result TYPEID check and range check.
;  8522          8619  3          !
;  8523          8620  3          CASE .RESULT[DBG$B_DHDR_FCODE] FROM RST$K_TYPE_MINIMUM
;  8524          8621  3                                         TO   RST$K_TYPE_MAXIMUM OF
;  8525          8622  3              SET
;  8526          8623  3              [RST$K_TYPE_ENUM]:
;  8527          8624  4                  BEGIN
;  8528          8625  4                  IF .RESULT[DBG$L_DHDR_TYPEID] EQL 0
;  8529          8626  4                  THEN
;  8530          8627  4                      RESULT[DBG$L_DHDR_TYPEID] = .LEFT_ARG[DBG$L_DHDR_TYPEID];
;  8531          8628  4
```

```
;  8532          8629  4                 IF .LEFT_ARG[DBG$L_DHDR_TYPEID] NEQ 0
;  8533          8630  4                 THEN
;  8534          8631  5                     BEGIN
;  8535          8632  5                     STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$L_DHDR_TYPEID],
;  8536          8633  5                                                .RESULT[DBG$L_DHDR_TYPEID]);
;  8537          8634  5                     END
;  8538          8635  5
;  8539          8636  4                 ELSE
;  8540          8637  5                     BEGIN
;  8541          8638  6                     IF NOT (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
;  8542          8639  6                             .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
;  8543          8640  5                     THEN
;  8544          8641  5                         STATUS = FALSE;
;  8545          8642  4                     END;
;  8546          8643  4
;  8547          8644  4                 IF NOT .STATUS THEN RETURN .STATUS;
;  8548          8645  4
;  8549          8646  4                 RETURN TYPEID_RANGE_CHECK_ENUM(.LEFT_ARG, .RESULT[DBG$L_DHDR_TYPEID]);
;  8550          8647  3                 END;
;  8551          8648  3
;  8552          8649  3             [INRANGE, OUTRANGE]:
;  8553          8650  3                 STATUS = FALSE;
;  8554          8651  3             TES;
;  8555          8652  3
;  8556          8653  2         END;
;  8557          8654  2
;  8558          8655  2     [ORT$K_TYPEID_SET_SET]:
;  8559          8656  3         BEGIN
;  8560          8657  3         LEFT_TYPEID = .LEFT_ARG[DBG$L_DHDR_TYPEID];
;  8561          8658  3         LEFT_FCODE = .LEFT_ARG[DBG$B_DHDR_FCODE];
;  8562          8659  3         LEFT_DTYPE = .LEFT_ARG[DBG$B_VALUE_DTYPE];
;  8563          8660  3         IF .LEFT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
;  8564          8661  3         THEN
;  8565          8662  4             BEGIN
;  8566          8663  4             LEFT_TYPEID = DBG$GET_SET_TYPEID(
;  8567          8664  4                 .LEFT_ARG[DBG$L_DHDR_TYPEID], PARENT_TYPE);
;  8568          8665  4             LEFT_FCODE = .LEFT_TYPEID[RST$B_FCODE];
;  8569          8666  4             IF .LEFT_FCODE EQL RST$K_TYPE_ATOMIC
;  8570          8667  4             THEN
;  8571          8668  5                 BEGIN
;  8572          8669  5                 DBG$STA_TYP_ATOMIC(.LEFT_TYPEID, LEFT_DTYPE, SIZE);
;  8573          8670  5                 IF .LEFT_DTYPE EQL DST$K_BOOL THEN LEFT_DTYPE = DSC$K_DTYPE_TF;
;  8574          8671  4                 END;
;  8575          8672  4
;  8576          8673  3             END;
;  8577          8674  3
;  8578          8675  3             IF .RIGHT_ARG NEQ 0
;  8579          8676  3             THEN
;  8580          8677  4                 BEGIN
;  8581          8678  4                 RIGHT_TYPEID = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
;  8582          8679  4                 RIGHT_FCODE = .RIGHT_ARG[DBG$B_DHDR_FCODE];
;  8583          8680  4                 RIGHT_DTYPE = .RIGHT_ARG[DBG$B_VALUE_DTYPE];
;  8584          8681  4                 IF .RIGHT_ARG[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
;  8585          8682  4                 THEN
;  8586          8683  5                     BEGIN
;  8587          8684  5                     RIGHT_TYPEID = DBG$GET_SET_TYPEID(
;  8588          8685  5                         .RIGHT_ARG[DBG$L_DHDR_TYPEID], PARENT_TYPE);
```

```
;  8589          8686  5                 RIGHT_FCODE = .RIGHT_TYPEID[RST$B_FCODE];
;  8590          8687  5                 IF .RIGHT_FCODE EQL RST$K_TYPE_ATOMIC
;  8591          8688  5                 THEN
;  8592          8689  6                     BEGIN
;  8593          8690  6                     DBG$STA_TYP_ATOMIC(.RIGHT_TYPEID, RIGHT_DTYPE, SIZE);
;  8594          8691  6                     IF .RIGHT_DTYPE EQL DST$K_BOOL THEN RIGHT_DTYPE = DSC$K_DTYPE_TF;
;  8595          8692  5                     END;
;  8596          8693  5
;  8597          8694  4                 END;
;  8598          8695  4
;  8599          8696  4             STATUS = TYPEID_CHECK_SET(.LEFT_TYPEID, .RIGHT_TYPEID,
;  8600          8697  4                                       .LEFT_FCODE, .RIGHT_FCODE,
;  8601          8698  4                                       .LEFT_DTYPE, .RIGHT_DTYPE);
;  8602          8699  3             END;
;  8603          8700  3
;  8604          8701  3         IF (.RESULT EQL 0) OR
;  8605          8702  3            (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_ATOMIC) OR
;  8606          8703  3            (.RESULT[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_DESCR) OR
;  8607          8704  3           NOT .STATUS
;  8608          8705  3         THEN
;  8609          8706  3             RETURN .STATUS;
;  8610          8707  3
;  8611          8708  3         CASE .RESULT[DBG$B_DHDR_FCODE] FROM RST$K_TYPE_MINIMUM
;  8612          8709  3                                        TO   RST$K_TYPE_MAXIMUM OF
;  8613          8710  3             SET
;  8614          8711  3             [RST$K_TYPE_SET]:
;  8615          8712  4                 BEGIN
;  8616          8713  4                 RESULT_FCODE = .RESULT[DBG$B_DHDR_FCODE];
;  8617          8714  4                 RESULT_DTYPE = .RESULT[DBG$B_VALUE_DTYPE];
;  8618          8715  4                 IF .RESULT[DBG$L_DHDR_TYPEID] EQL 0
;  8619          8716  4                 THEN
;  8620          8717  4                     RESULT[DBG$L_DHDR_TYPEID] = .LEFT_ARG[DBG$L_DHDR_TYPEID];
;  8621          8718  4
;  8622          8719  4                 RESULT_TYPEID = DBG$GET_SET_TYPEID(
;  8623          8720  4                     .RESULT[DBG$L_DHDR_TYPEID], PARENT_TYPE);
;  8624          8721  4                 RESULT_FCODE = .RESULT_TYPEID[RST$B_FCODE];
;  8625          8722  4                 IF .RESULT_FCODE EQL RST$K_TYPE_ATOMIC
;  8626          8723  4                 THEN
;  8627          8724  5                     BEGIN
;  8628          8725  5                     DBG$STA_TYP_ATOMIC(.RESULT_TYPEID, RESULT_DTYPE, SIZE);
;  8629          8726  5                     IF .RESULT_DTYPE EQL DST$K_BOOL THEN RESULT_DTYPE = DSC$K_DTYPE_TF;
;  8630          8727  4                     END;
;  8631          8728  4
;  8632          8729  4                 STATUS = TYPEID_CHECK_SET(.LEFT_TYPEID, .RESULT_TYPEID,
;  8633          8730  4                                           .LEFT_FCODE, .RESULT_FCODE,
;  8634          8731  4                                           .LEFT_DTYPE, .RESULT_DTYPE);
;  8635          8732  4
;  8636          8733  4                 IF NOT .STATUS THEN RETURN .STATUS;
;  8637          8734  4
;  8638          8735  4
;  8639          8736  4                 ! Set the typeid for subrange.  Typeid is set in routine
;  8640          8737  4                 ; DBG$GET_SET_TYPEID.
;  8641          8738  4                 !
;  8642          8739  4                 IF .PARENT_TYPE NEQ 0
;  8643          8740  4                 THEN
;  8644          8741  4                     IF .PARENT_TYPE[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG
;  8645          8742  4                     THEN
```

```
;  8646          8743  4                        RESULT_TYPEID = .PARENT_TYPE;
;  8647          8744  4
;  8648          8745  4
;  8649          8746  4                    ! Perform range check.
;  8650          8747  4                    !
;  8651          8748  4                    SELECTONE .RESULT_TYPEID[RST$B_FCODE] OF
;  8652          8749  4                        SET
;  8653          8750  4                        [RST$K_TYPE_SUBRNG]:
;  8654          8751  4                            RETURN TYPEID_RANGE_CHECK_SUBRNG(.LEFT_ARG, .RESULT_TYPEID);
;  8655          8752  4
;  8656          8753  4                        TES;
;  8657          8754  3                    END;
;  8658          8755  3
;  8659          8756  3                [INRANGE, OUTRANGE]:
;  8660          8757  3                    STATUS = FALSE;
;  8661          8758  3                TES;
;  8662          8759  3
;  8663          8760  2            END;
;  8664          8761  2
;  8665          8762  2
;  8666          8763  2        ! Typeid pointer typeid check would be always called from DEPOSIT
;  8667          8764  2        ! command with Left_arg (src) and Right_arg (dst).  There is no
;  8668          8765  2        ! result. It turns out this is not supported.  For we have mapped
;  8669          8766  2        ! the TPTR into LU, so there is no typeid check required.
;  8670          8767  2        !
;  8671          8768  2        [ORT$K_TYPEID_TPTR_TPTR]:
;  8672          8769  3            BEGIN
;  8673          8770  3            0;
;  8674          8771  2            END;
;  8675          8772  2
;  8676          8773  2
;  8677          8774  2        ! Subrange typeid check would be always called from DEPOSIT
;  8678          8775  2        ! command with Left_arg (src) and Right_arg (dst) for typeid
;  8679          8776  2        ! check and also called with Left_arg (src) and Result (dst) for
;  8680          8777  2        ! range check.
;  8681          8778  2        !
;  8682          8779  2        [ORT$K_TYPEID_SUBRNG_SUBRNG]:
;  8683          8780  3            BEGIN
;  8684          8781  3            IF .RESULT EQL 0
;  8685          8782  3            THEN
;  8686          8783  4                BEGIN
;  8687          8784  4                PARENT_TYPE = .RIGHT_ARG[DBG$L_DHDR_TYPEID];
;  8688          8785  4                WHILE .PARENT_TYPE[RST$B_FCODE] EQL RST$K_TYPE_SUBRNG DO
;  8689          8786  4                    DBG$STA_TYP_SUBRNG(.PARENT_TYPE, PARENT_TYPE, DUMMY1, DUMMY2, SIZE);
;  8690          8787  4
;  8691          8788  4                SELECTONE .PARENT_TYPE[RST$B_FCODE] OF
;  8692          8789  4                    SET
;  8693          8790  4                    [RST$K_TYPE_ENUM]:
;  8694          8791  5                        BEGIN
;  8695          8792  5                        IF .LEFT_ARG[DBG$L_DHDR_TYPEID] NEQ 0
;  8696          8793  5                        THEN
;  8697          8794  6                            BEGIN
;  8698          8795  6                            STATUS = TYPEID_CHECK_ENUM(.LEFT_ARG[DBG$L_DHDR_TYPEID],
;  8699          8796  6                                                       .PARENT_TYPE);
;  8700          8797  6                            END
;  8701          8798  6
;  8702          8799  5                        ELSE
```

```
;  8703          8800  6                        BEGIN
;  8704          8801  7                        IF NOT (.LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_L OR
;  8705          8802  7                                .LEFT_ARG[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_LU)
;  8706          8803  6                        THEN
;  8707          8804  6                            STATUS = FALSE;
;  8708          8805  5                        END;
;  8709          8806  4                    END;
;  8710          8807  4
;  8711          8808  4                [RST$K_TYPE_ATOMIC]:
;  8712          8809  5                    BEGIN
;  8713          8810  5                    DBG$STA_TYP_ATOMIC(.PARENT_TYPE, RESULT_DTYPE, SIZE);
;  8714          8811  5                    IF .RESULT_DTYPE EQL DST$K_BOOL THEN RESULT_DTYPE = DSC$K_DTYPE_TF;
;  8715          8812  5                    IF .RESULT_DTYPE EQL .LEFT_ARG[DBG$B_VALUE_DTYPE]
;  8716          8813  5                    THEN
;  8717          8814  5                        STATUS = TRUE
;  8718          8815  5
;  8719          8816  5                    ELSE
;  8720          8817  5                        STATUS = FALSE;
;  8721          8818  4                    END;
;  8722          8819  4
;  8723          8820  4                [OTHERWISE]:
;  8724          8821  4                    STATUS = FALSE;
;  8725          8822  4                TES;
;  8726          8823  4
;  8727          8824  4                RETURN .STATUS;
;  8728          8825  4                END
;  8729          8826  4
;  8730          8827  3            ELSE
;  8731          8828  3                RETURN TYPEID_RANGE_CHECK_SUBRNG(.LEFT_ARG, .RESULT[DBG$L_DHDR_TYPEID]);
;  8732          8829  2            END;
;  8733          8830  2
;  8734          8831  2            [INRANGE, OUTRANGE]:
;  8735          8832  2                $DBG_ERROR('DBGEVALOP\DBG$PERFORM_TYPEID_CHECK, unknown type check index');
;  8736          8833  2            TES;
;  8737          8834  2
;  8738          8835  2        RETURN .STATUS;
;  8739          8836  1        END;


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

24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 22 05F9A P.AMI:   .ASCII  \''DBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHEC\   ;
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 05FA9                                                             ;
                           43 45 48 43 05FB8                                                                    ;
                                       4B 05FBC            .ASCII  \K\                                         ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 41 05FBD P.AMJ:   .ASCII  \ADBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHEC\   ;
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 05FCC                                                             ;
                           43 45 48 43 05FDB                                                                    ;
6F 66 20 64 69 65 70 79 74 20 6F 6E 20 2C 4B 05FDF          .ASCII  \K, no typeid for non-atomic data\          ;
61 64 20 63 69 6D 6F 74 61 2D 6E 6F 6E 20 72 05FEE                                                             ;
                                       61 74 05FFD                                                          ;
24 47 42 44 5C 50 4F 4C 41 56 45 47 42 44 41 05FFF P.AMK:   .ASCII  \ADBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHEC\   ;
5F 44 49 45 50 59 54 5F 4D 52 4F 46 52 45 50 0600E                                                             ;
                           43 45 48 43 0601D                                                                    ;
6F 66 20 64 69 65 70 79 74 20 6F 6E 20 2C 4B 06021          .ASCII  \K, no typeid for non-atomic data\          ;
61 64 20 63 69 6D 6F 74 61 2D 6E 6F 6E 20 72 06030                                                             ;
```

```
                                                                61  74  0603F
24  47  42  44  5C  50  4F  4C  41  56  45  47  42  44  22  06041 P.AML:  .ASCII  \''DBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHEC\
5F  44  49  45  50  59  54  5F  4D  52  4F  46  52  45  50  06050
                                    43  45  48  43  0605F
                                                4B  06063         .ASCII  \K\
24  47  42  44  5C  50  4F  4C  41  56  45  47  42  44  3C  06064 P.AMM:  .ASCII  \<DBGEVALOP\<92>\DBG$PERFORM_TYPEID_CHEC\
5F  44  49  45  50  59  54  5F  4D  52  4F  46  52  45  50  06073
                                    43  45  48  43  06082
65  70  79  74  20  6E  77  6F  6E  6B  6E  75  20  2C  4B  06086         .ASCII  \K, unknown type check index\
            78  65  64  6E  69  20  6B  63  65  68  63  20  06095


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

                                                    0FFC 00000          .ENTRY  DBG$PERFORM_TYPEID_CHECK, Save R2,R3,R4,R5,-  8458
                                                                                R6,R7,R8,R9,R10,R11
                                                5B 00000000G   00  9E 00002          MOVAB   DBG$STA_TYP_ATOMIC, R11
                                                5A 00000000'   EF  9E 00009          MOVAB   P.AMI, R10
                                                59 00000000G   00  9E 00010          MOVAB   LIB$SIGNAL, R9
                                                5E             1C  C2 00017          SUBL2   #28, SP
                                                52        08   AC  D0 0001A          MOVL    LEFT_ARG, R2                    8526
                                                               0D  12 0001E          BNEQ    1$
                                                               5A  DD 00020          PUSHL   R10                             8528
                                                               01  DD 00022          PUSHL   #1
                                                    00028362   8F  DD 00024          PUSHL   #164706
                                                69             03  FB 0002A          CALLS   #3, LIB$SIGNAL
                                                02        06   A2  91 0002D 1$:      CMPB    6(R2), #2                       8533
                                                               19  13 00031          BEQL    2$
                                                03        06   A2  91 00033          CMPB    6(R2), #3                       8534
                                                               13  13 00037          BEQL    2$
                                                          08   A2  D5 00039          TSTL    8(R2)                           8536
                                                               0E  12 0003C          BNEQ    2$
                                                          23   AA  9F 0003E          PUSHAB  P.AMJ                           8538
                                                               01  DD 00041          PUSHL   #1
                                                00028362       8F  DD 00043          PUSHL   #164706
                                                69             03  FB 00049          CALLS   #3, LIB$SIGNAL
                                                54        0C   AC  D0 0004C 2$:      MOVL    RIGHT_ARG, R4                   8544
                                                               56  D4 00050          CLRL    R6
                                                               54  D5 00052          TSTL    R4
                                                               21  13 00054          BEQL    3$
                                                               56  D6 00056          INCL    R6
                                                02        06   A4  91 00058          CMPB    6(R4), #2                       8547
                                                               19  13 0005C          BEQL    3$
                                                03        06   A4  91 0005E          CMPB    6(R4), #3                       8548
                                                               13  13 00062          BEQL    3$
                                                          08   A4  D5 00064          TSTL    8(R4)                           8550
                                                               0E  12 00067          BNEQ    3$
                                                          65   AA  9F 00069          PUSHAB  P.AMK                           8552
                                                               01  DD 0006C          PUSHL   #1
                                                00028362       8F  DD 0006E          PUSHL   #164706
                                                69             03  FB 00074          CALLS   #3, LIB$SIGNAL
                                                               54  D5 00077 3$:      TSTL    R4                              8560
                                                               14  12 00079          BNEQ    4$
                                                          10   AC  D5 0007B          TSTL    RESULT
                                                               0F  12 0007E          BNEQ    4$
                                                        00A7   CA  9F 00080          PUSHAB  P.AML                           8562
```

```
                                      01  DD 00084          PUSHL   #1                          ;
                            00028362  8F  DD 00086          PUSHL   #164706                     ;
                          69          03  FB 0008C          CALLS   #3, LIB$SIGNAL              ;
                          55          01  D0 0008F 4$:      MOVL    #1, STATUS                  ; 8567
                                  10  AE  D4 00092          CLRL    PARENT_TYPE                 ; 8568
                 03       01      04  AC  CF 00095          CASEL   TYPE_INDEX, #1, #3          ; 8569
    0231       02C7     00D1    001A         0009A 5$:      .WORD   7$-5$,-                     ;
                                                                    18$-5$,-                    ;
                                                                    39$-5$,-                    ;
                                                                    29$-5$                      ;
                                00CA  CA  9F 000A2          PUSHAB  P.AMM                       ; 8832
                                      01  DD 000A6          PUSHL   #1                          ;
                            00028362  8F  DD 000A8          PUSHL   #164706                     ;
                          69          03  FB 000AE          CALLS   #3, LIB$SIGNAL              ;
                                    02AD  31 000B1 6$:      BRW     39$                         ;
                          2D          56  E9 000B4 7$:      BLBC    R6, 11$                     ;
                                  08  A2  D5 000B7          TSTL    8(R2)                       ; 8584
                                      10  13 000BA          BEQL    8$                          ;
                                  08  A4  DD 000BC          PUSHL   8(R4)                       ; 8588
                                  08  A2  DD 000BF          PUSHL   8(R2)                       ; 8587
                    0000V CF          02  FB 000C2          CALLS   #2, TYPEID_CHECK_ENUM       ;
                          55          50  D0 000C7          MOVL    R0, STATUS                  ;
                                      18  11 000CA          BRB     11$                         ; 8584
                                  10  AC  D5 000CC 8$:      TSTL    RESULT                      ; 8593
                                      13  12 000CF          BNEQ    11$                         ;
                          08      16  A2  91 000D1          CMPB    22(R2), #8                  ; 8596
                                      06  13 000D5          BEQL    9$                          ;
                          04      16  A2  91 000D7          CMPB    22(R2), #4                  ; 8597
                                      05  12 000DB          BNEQ    10$                         ;
                          55          01  D0 000DD 9$:      MOVL    #1, STATUS                  ; 8599
                                      02  11 000E0          BRB     11$                         ;
                                      55  D4 000E2 10$:     CLRL    STATUS                      ; 8601
                          53      10  AC  D0 000E4 11$:     MOVL    RESULT, R3                  ; 8610
                                      C7  13 000E8          BEQL    6$                          ;
                          02      06  A3  91 000EA          CMPB    6(R3), #2                   ; 8611
                                      C1  13 000EE          BEQL    6$                          ;
                          03      06  A3  91 000F0          CMPB    6(R3), #3                   ; 8612
                                      BB  13 000F4          BEQL    6$                          ;
                          B8          55  E9 000F6          BLBC    STATUS, 6$                  ; 8613
                 15       01      06  A3  8F 000F9          CASEB   6(R3), #1, #21              ; 8620
    002F       014F     014F    014F         000FE 12$:     .WORD   23$-12$,-                   ;
    014F       014F     014F    014F         00106                  23$-12$,-                   ;
    014F       014F     014F    014F         0010E                  23$-12$,-                   ;
    014F       014F     014F    014F         00116                  13$-12$,-                   ;
    014F       014F     014F    014F         0011E                  23$-12$,-                   ;
                        014F    014F         00126                  23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
                                                                    23$-12$,-                   ;
```

```
                                                          23$-12$,-                              ;
                                                          23$-12$,-                              ;
                                                          23$-12$,-                              ;
                                                          23$-12$,-                              ;
                                                          23$-12$                                ;
                          0120   31 0012A        BRW      23$                                    ; 8650
                       08   A3   D5 0012D 13$:   TSTL     8(R3)                                  ; 8625
                            05   12 00130        BNEQ     14$                                    ;
          08    A3     08   A2   D0 00132        MOVL     8(R2), 8(R3)                           ; 8627
                       08   A2   D5 00137 14$:   TSTL     8(R2)                                  ; 8629
                            10   13 0013A        BEQL     15$                                    ;
                       08   A3   DD 0013C        PUSHL    8(R3)                                  ; 8633
                       08   A2   DD 0013F        PUSHL    8(R2)                                  ; 8632
       0000V    CF          02   FB 00142        CALLS    #2, TYPEID_CHECK_ENUM                  ;
                55          50   D0 00147        MOVL     R0, STATUS                             ;
                            0E   11 0014A        BRB      16$                                    ; 8629
                08     16   A2   91 0014C 15$:   CMPB     22(R2), #8                             ; 8638
                            08   13 00150        BEQL     16$                                    ;
                04     16   A2   91 00152        CMPB     22(R2), #4                             ; 8639
                            02   13 00156        BEQL     16$                                    ;
                            55   D4 00158        CLRL     STATUS                                 ; 8641
                03          55   E8 0015A 16$:   BLBS     STATUS, 17$                            ; 8644
                          0201   31 0015D        BRW      39$                                    ;
                       08   A3   DD 00160 17$:   PUSHL    8(R3)                                  ; 8646
                            52   DD 00163        PUSHL    R2                                     ;
       0000V    CF          02   FB 00165        CALLS    #2, TYPEID_RANGE_CHECK_ENUM            ;
                                 04 0016A        RET                                             ;
                57     08   A2   D0 0016B 18$:   MOVL     8(R2), LEFT_TYPEID                     ; 8657
                58     06   A2   9A 0016F        MOVZBL   6(R2), LEFT_FCODE                      ; 8658
                6E     16   A2   9A 00173        MOVZBL   22(R2), LEFT_DTYPE                     ; 8659
                08     06   A2   91 00177        CMPB     6(R2), #8                              ; 8660
                            2E   12 0017B        BNEQ     19$                                    ;
                       10   AE   9F 0017D        PUSHAB   PARENT_TYPE                            ; 8663
                       08   A2   DD 00180        PUSHL    8(R2)                                  ; 8664
       FB2B     CF          02   FB 00183        CALLS    #2, DBG$GET_SET_TYPEID                 ;
                57          50   D0 00188        MOVL     R0, LEFT_TYPEID                        ;
                58     18   A7   9A 0018B        MOVZBL   24(LEFT_TYPEID), LEFT_FCODE            ; 8665
                02          58   D1 0018F        CMPL     LEFT_FCODE, #2                         ; 8666
                            17   12 00192        BNEQ     19$                                    ;
                       14   AE   9F 00194        PUSHAB   SIZE                                   ; 8669
                       04   AE   9F 00197        PUSHAB   LEFT_DTYPE                             ;
                            57   DD 0019A        PUSHL    LEFT_TYPEID                            ;
                6B          03   FB 0019C        CALLS    #3, DBG$STA_TYP_ATOMIC                 ;
   0000009E     8F          6E   D1 0019F        CMPL     LEFT_DTYPE, #158                       ; 8670
                            03   12 001A6        BNEQ     19$                                    ;
                6E          28   D0 001A8        MOVL     #40, LEFT_DTYPE                        ;
                59          56   E9 001AB 19$:   BLBC     R6, 21$                                ; 8675
                53     08   A4   D0 001AE        MOVL     8(R4), RIGHT_TYPEID                    ; 8678
                56     06   A4   9A 001B2        MOVZBL   6(R4), RIGHT_FCODE                     ; 8679
          04    AE     16   A4   9A 001B6        MOVZBL   22(R4), RIGHT_DTYPE                    ; 8680
                08     06   A4   91 001BB        CMPB     6(R4), #8                              ; 8681
                            30   12 001BF        BNEQ     20$                                    ;
                       10   AE   9F 001C1        PUSHAB   PARENT_TYPE                            ; 8684
                       08   A4   DD 001C4        PUSHL    8(R4)                                  ; 8685
       FAE7     CF          02   FB 001C7        CALLS    #2, DBG$GET_SET_TYPEID                 ;
                53          50   D0 001CC        MOVL     R0, RIGHT_TYPEID                       ;
                56     18   A3   9A 001CF        MOVZBL   24(RIGHT_TYPEID), RIGHT_FCODE          ; 8686
```

```
                                02              56  D1 001D3          CMPL    RIGHT_FCODE, #2                                 ; 8687
                                                19  12 001D6          BNEQ    20$
                             14 AE              9F  9F 001D8          PUSHAB  SIZE                                            ; 8690
                             08 AE              9F  9F 001DB          PUSHAB  RIGHT_DTYPE
                                                53  DD 001DE          PUSHL   RIGHT_TYPEID
                                6B              03  FB 001E0          CALLS   #3, DBG$STA_TYP_ATOMIC
                     0000009E   8F              04  AE D1 001E3          CMPL    RIGHT_DTYPE, #158                           ; 8691
                                                04  12 001EB          BNEQ    20$
                           04   AE              28  D0 001ED          MOVL    #40, RIGHT_DTYPE
                                04 AE              DD 001F1 20$:      PUSHL   RIGHT_DTYPE                                     ; 8698
                                04 AE              DD 001F4           PUSHL   LEFT_DTYPE
                                                56  DD 001F7          PUSHL   RIGHT_FCODE                                     ; 8697
                                              0108  8F BB 001F9          PUSHR   #^M<R3,R8>                                      ; 8696
                                                57  DD 001FD          PUSHL   LEFT_TYPEID
                         0000V  CF                  06  FB 001FF          CALLS   #6, TYPEID_CHECK_SET
                                55                  50  D0 00204          MOVL    R0, STATUS
                                50              10  AC D0 00207 21$:      MOVL    RESULT, R0                                  ; 8701
                                                42  13 0020B          BEQL    24$
                                02              06  A0 91 0020D          CMPB    6(R0), #2                                   ; 8702
                                                3C  13 00211          BEQL    24$
                                03              06  A0 91 00213          CMPB    6(R0), #3                                   ; 8703
                                                36  13 00217          BEQL    24$
                                33              55  E9 00219          BLBC    STATUS, 24$                                 ; 8704
                             15 01              06  A0 8F 0021C          CASEB   6(R0), #1, #21                              ; 8708
                 002C        002C            002C        002C  00221 22$:      .WORD   23$-22$,-
                 0031        002C            002C        002C  00229           .WORD   23$-22$,-
                 002C        002C            002C        002C  00231           .WORD   23$-22$,-
                 002C        002C            002C        002C  00239           .WORD   23$-22$,-
                 002C        002C            002C        002C  00241           .WORD   23$-22$,-
                             002C            002C        002C  00249           .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   25$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$,-
                                                                               .WORD   23$-22$
                                                55  D4 0024D 23$:      CLRL    STATUS                                      ; 8757
                                              010F  31 0024F 24$:      BRW     39$
                                53              06  A0 9A 00252 25$:      MOVZBL  6(R0), RESULT_FCODE                         ; 8713
                           18   AE              16  A0 9A 00256           MOVZBL  22(R0), RESULT_DTYPE                        ; 8714
                                                08  A0 D5 0025B          TSTL    8(R0)                                       ; 8715
                                                05  12 0025E          BNEQ    26$
                           08   A0              08  A2 D0 00260          MOVL    8(R2), 8(R0)                                ; 8717
                                                10  AE 9F 00265 26$:      PUSHAB  PARENT_TYPE                                 ; 8719
                                                08  A0 DD 00268          PUSHL   8(R0)                                       ; 8720
                         FA43   CF                  02  FB 0026B          CALLS   #2, DBG$GET_SET_TYPEID
                                54                  50  D0 00270          MOVL    R0, RESULT_TYPEID
```

```
                53      18   A4   9A 00273          MOVZBL  24(RESULT_TYPEID), RESULT_FCODE                ; 8721
                02           53   D1 00277          CMPL    RESULT_FCODE, #2                            ; 8722
                             19   12 0027A          BNEQ    27$                                         ;
                          14 AE   9F 0027C          PUSHAB  SIZE                                        ; 8725
                          1C AE   9F 0027F          PUSHAB  RESULT_DTYPE                                ;
                             54   DD 00282          PUSHL   RESULT_TYPEID                               ;
                6B           03   FB 00284          CALLS   #3, DBG$STA_TYP_ATOMIC                      ;
0000009E        8F      18 AE   D1 00287          CMPL    RESULT_DTYPE, #TS8                          ; 8726
                             04   12 0028F          BNEQ    27$                                         ;
          18    AE           28   D0 00291          MOVL    #40, RESULT_DTYPE                           ;
                       18 AE   DD 00295 27$:        PUSHL   RESULT_DTYPE                                ; 8731
                       04 AE   DD 00298          PUSHL   LEFT_DTYPE                                  ;
                          53   DD 0029B          PUSHL   RESULT_FCODE                                ; 8730
                    0110   8F   BB 0029D          PUSHR   #^M<R4,R8>                                  ; 8729
                          57   DD 002A1          PUSHL   LEFT_TYPEID                                 ;
          0000V         CF     06   FB 002A3          CALLS   #6, TYPEID_CHECK_SET                        ;
                55           50   D0 002A8          MOVL    R0, STATUS                                  ;
                A1           55   E9 002AB          BLBC    STATUS, 24$                                 ; 8733
                50      10 AE   D0 002AE          MOVL    PARENT_TYPE, R0                             ; 8739
                             09   13 002B2          BEQL    28$                                         ;
                09      18   A0   91 002B4          CMPB    24(R0), #9                                  ; 8741
                             03   12 002B8          BNEQ    28$                                         ;
                54           50   D0 002BA          MOVL    R0, RESULT_TYPEID                           ; 8743
                50      18   A4   9A 002BD 28$:      MOVZBL  24(RESULT_TYPEID), R0                       ; 8748
                09           50   91 002C1          CMPB    R0, #9                                      ; 8750
                             89   12 002C4          BNEQ    24$                                         ;
                             54   DD 002C6          PUSHL   RESULT_TYPEID                               ; 8751
                          008E   31 002C8          BRW     38$                                         ;
                50      10   AC   D0 002CB 29$:      MOVL    RESULT, R0                                  ; 8781
                             03   13 002CF          BEQL    30$                                         ;
                          0082   31 002D1          BRW     37$                                         ;
          10    AE      08   A4   D0 002D4 30$:      MOVL    8(R4), PARENT_TYPE                          ; 8784
                50      10   AE   D0 002D9 31$:      MOVL    PARENT_TYPE, R0                             ; 8785
                09      18   A0   91 002DD          CMPB    24(R0), #9                                  ;
                             17   12 002E1          BNEQ    32$                                         ;
                          14 AE   9F 002E3          PUSHAB  SIZE                                        ; 8786
                          0C AE   9F 002E6          PUSHAB  DUMMY2                                      ;
                          14 AE   9F 002E9          PUSHAB  DUMMY1                                      ;
                          1C AE   9F 002EC          PUSHAB  PARENT_TYPE                                 ;
                             50   DD 002EF          PUSHL   R0                                          ;
00000000G       00           05   FB 002F1          CALLS   #5, DBG$STA_TYP_SUBRNG                      ;
                             DF   11 002F8          BRB     31$                                         ;
                50      10   AE   D0 002FA 32$:      MOVL    PARENT_TYPE, R0                             ; 8788
                51      18   A0   9A 002FE          MOVZBL  24(R0), R1                                  ;
                04           51   91 00302          CMPB    R1, #4                                      ; 8790
                             23   12 00305          BNEQ    35$                                         ;
                          08   A2   D5 00307          TSTL    8(R2)                                       ; 8792
                             0F   13 0030A          BEQL    33$                                         ;
                             50   DD 0030C          PUSHL   R0                                          ; 8796
                          08   A2   DD 0030E          PUSHL   8(R2)                                       ; 8795
          0000V         CF     02   FB 00311          CALLS   #2, TYPEID_CHECK_ENUM                       ;
                55           50   D0 00316          MOVL    R0, STATUS                                  ;
                             46   11 00319          BRB     39$                                         ; 8792
                08      16   A2   91 0031B 33$:      CMPB    22(R2), #8                                  ; 8801
                             40   13 0031F          BEQL    39$                                         ;
                04      16   A2   91 00321          CMPB    22(R2), #4                                  ; 8802
                             3A   13 00325          BEQL    39$                                         ;
```

```
                                                    FF23  31 00327 34$:   BRW     23$                                     ; 8804
                                          02          51  91 0032A 35$:   CMPB    R1, #2                                  ; 8808
                                                      F8  12 0032D         BNEQ    34$                                     ;
                                                  14  AE  9F 0032F         PUSHAB  SIZE                                    ; 8810
                                                  1C  AE  9F 00332         PUSHAB  RESULT_DTYPE                            ;
                                                      50  DD 00335         PUSHL   R0                                      ;
                                          6B          03  FB 00337         CALLS   #3, DBG$STA_TYP_ATOMIC                  ;
                                 0000009E 8F          18  AE  D1 0033A         CMPL    RESULT_DTYPE, #T58                  ; 8811
                                                      04  12 00342         BNEQ    36$                                     ;
                                       18 AE          28  D0 00344         MOVL    #40, RESULT_DTYPE                       ;
      18   AE        16   A2             08          00  ED 00348 36$:   CMPZV   #0, #8, 22(R2), RESULT_DTYPE            ; 8812
                                                      D6  12 0034F         BNEQ    34$                                     ;
                                          55          01  D0 00351         MOVL    #1, STATUS                              ; 8814
                                                      0B  11 00354         BRB     39$                                     ;
                                                  08  A0  DD 00356 37$:   PUSHL   8(R0)                                   ; 8828
                                                      52  DD 00359 38$:   PUSHL   R2                                      ;
                                   0000V  CF              02  FB 0035B         CALLS   #2, TYPEID_RANGE_CHECK_SUBRNG       ;
                                                          04  00360         RET                                             ;
                                          50          55  D0 00361 39$:   MOVL    STATUS, R0                              ; 8835
                                                          04  00364         RET                                             ; 8836

; Routine Size:  869 bytes,    Routine Base:  DBG$CODE + 1954
```

```
;  8741          8837  1 ROUTINE DBG$TYPE_CONV (VALUE1, VALUE2) =
;  8742          8838  1 !
;  8743          8839  1 ! FUNCTION
;  8744          8840  1 !       This routine is the top-level type converter for DEBUG. It takes a
;  8745          8841  1 !       source value descriptor and a target value descriptor. It first
;  8746          8842  1 !       checks whether there are any language-specific type conversion
;  8747          8843  1 !       rules to use. If not, it calls the language-independent routine
;  8748          8844  1 !       DBG$COVER_DX_DX.
;  8749          8845  1 !
;  8750          8846  1 ! INPUTS
;  8751          8847  1 !       VALUE1  - Pointer to a value descriptor to be type-converted.
;  8752          8848  1 !
;  8753          8849  1 !       VALUE2  - Pointer to the target value descriptor.
;  8754          8850  1 !
;  8755          8851  1 ! OUTPUTS
;  8756          8852  1 !       A pointer to a value descriptor is returned.  The target descriptor
;  8757          8853  1 !       is filled in with the result of the conversion.
;  8758          8854  1 !
;  8759          8855  2     BEGIN
;  8760          8856  2
;  8761          8857  2     MAP
;  8762          8858  2         VALUE1: REF DBG$VALDESC,
;  8763          8859  2         VALUE2: REF DBG$VALDESC;
;  8764          8860  2
;  8765          8861  2     LOCAL
;  8766          8862  2         CVT_TBL_ENTRY: TYPE_CVT$ENTRY,  ! An entry in the Type Conversion Table
;  8767          8863  2         CVT_TYPE_PAIR: TYPE$PAIR,       ! Data Type Pairs
;  8768          8864  2         STATUS;                         ! Return status from lib$cvt_dx_dx
;  8769          8865  2
;  8770          8866  2
;  8771          8867  2     ! Perform Type Conversion indicated by Language Dependent Type Conversion
;  8772          8868  2     ! Table.
;  8773          8869  2     ! (Left type in Type Pair = The type we want to convert into,
;  8774          8870  2     !  Right type             = From type)
;  8775          8871  2     !
;  8776          8872  2     IF .CVT_TBL NEQ TABLEBASE
;  8777          8873  2     THEN
;  8778          8874  3         BEGIN
;  8779          8875  3         CVT_TYPE_PAIR[B$_LEFT_TYPE] = .VALUE2[DBG$B_VALUE_DTYPE];
;  8780          8876  3         CVT_TYPE_PAIR[B$_RIGHT_TYPE] = .VALUE1[DBG$B_VALUE_DTYPE];
;  8781          8877  3         INCR I FROM 0 TO .CVT_TBL_SIZE - 1 DO
;  8782          8878  4             BEGIN
;  8783          8879  4             CVT_TBL_ENTRY = .CVT_TBL[.I];
;  8784          8880  4             IF .CVT_TBL_ENTRY EQL 0 THEN EXITLOOP;
;  8785          8881  5             IF (.CVT_TBL_ENTRY[TYPE_CVT$B_LOWER_TYPE] EQL DSC$K_DTYPE_ANY AND
;  8786          8882  4                 .CVT_TBL_ENTRY[TYPE_CVT$B_HIGHER_TYPE] EQL .CVT_TYPE_PAIR[B$_RIGHT_TYPE]) OR
;  8787          8883  5                (.CVT_TBL_ENTRY[TYPE_CVT$B_HIGHER_TYPE] EQL DSC$K_DTYPE_ANY AND
;  8788          8884  4                 .CVT_TBL_ENTRY[TYPE_CVT$B_LOWER_TYPE] EQL .CVT_TYPE_PAIR[B$_LEFT_TYPE]) OR
;  8789          8885  5                (.CVT_TBL_ENTRY[TYPE_CVT$W_MAP_PAIR] EQL  .CVT_TYPE_PAIR[W$_TYPE_PAIR])
;  8790          8886  4             THEN
;  8791          8887  5                 BEGIN
;  8792          8888  5                 IF .CVT_TBL_ENTRY[TYPE_CVT$W_ROUT] NEQ 0
;  8793          8889  5                 THEN
;  8794          8890  5                     DBG$LANGUAGE_TYPE_CONV (.CVT_TBL_ENTRY[TYPE_CVT$W_ROUT],
;  8795          8891  5                                             .VALUE1, .VALUE2);
;  8796          8892  5                 RETURN .VALUE2;
;  8797          8893  4                 END;
```

```
;  8798          8894  3            END;
;  8799          8895  2        END;
;  8800          8896  2
;  8801          8897  2
;  8802          8898  2        ! Just in case, the there was no entry found in CVT_TBL for depositing
;  8803          8899  2        ! a dtype into PICT type.  We treated PICT as 'T'.
;  8804          8900  2        !
;  8805          8901  2        IF .VALUE2[DBG$B_VALUE_DTYPE] EQL DBG$K_DTYPE_PICT AND
;  8806          8902  2           .VALUE2[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT
;  8807          8903  2        THEN
;  8808          8904  2            VALUE2[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_T;
;  8809          8905  2
;  8810          8906  2        VALUE2 = DBG$COVER_DX_DX (.VALUE1, .VALUE2, .CVT_ROUND_FLAG);
;  8811          8907  2
;  8812          8908  2    RETURN .VALUE2;
;  8813          8909  1    END;


                                01FC 00000 DBG$TYPE_CONV:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 8837
                    58 00000000'  EF  9E 00002     MOVAB   CVT_TBL, R8                          ;
                    50 00000000'  EF  9E 00009     MOVAB   TAB[EBASE, R0                        ; 8872
                    50            68  D1 00010     CMPL    CVT_TBL, R0                          ;
                                  69  13 00013     BEQL    7$                                   ;
                    54      08    AC  D0 00015     MOVL    VALUE2, R4                           ; 8875
          56        08            08  16 A4  F0 00019     INSV    22(R4), #8, #8, CVT_TYPE_PAIR
                    55            04  AC  D0 0001F     MOVL    VALUE1, R5                       ; 3876
                    56            16  A5  90 00023     MOVB    22(R5), CVT_TYPE_PAIR
                    57            04  A8  D0 00027     MOVL    CVT_TBL_SIZE, R7                 ; 8877
                    52                01  CE 0002B     MNEGL   #1, I                            ;
                                      4A  11 0002E     BRB     6$                               ;
                    53        00  B842  D0 00030 1$:    MOVL    @CVT_TBL[I], CVT_TBL_ENTRY       ; 8879
                                      47  13 00035     BEQL    7$                               ; 8880
                        FF        8F    53  91 00037     CMPB    CVT_TBL_ENTRY, #255              ; 8881
                                      0A  12 0003B     BNEQ    2$                               ;
                    50                56  9A 0003D     MOVZBL  CVT_TYPE_PAIR, R0                ; 8882
          50        53            08  08  ED 00040     CMPZV   #8, #8, CVT_TBL_ENTRY, R0        ;
                                      1A  13 00045     BEQL    4$                               ;
000000FF  8F        53            08  08  ED 00047 2$:    CMPZV   #8, #8, CVT_TBL_ENTRY, #255      ; 8883
                                      0A  12 00050     BNEQ    3$                               ;
                    50                53  9A 00052     MOVZBL  CVT_TBL_ENTRY, R0                ; 8884
          50        56            08  08  ED 00055     CMPZV   #8, #8, CVT_TYPE_PAIR, R0        ;
                                      05  13 0005A     BEQL    4$                               ;
                    56                53  B1 0005C 3$:    CMPW    CVT_TBL_ENTRY, CVT_TYPE_PAIR     ; 8885
                                      19  12 0005F     BNEQ    6$                               ;
          00        53            10  10  ED 00061 4$:    CMPZV   #16, #16, CVT_TBL_ENTRY, #0      ; 8888
                                      0E  13 00066     BEQL    5$                               ;
                                      54  DD 00068     PUSHL   R4                               ; 8891
                                      55  DD 0006A     PUSHL   R5                               ;
          7E        53            10  10  EF 0006C     EXTZV   #16, #16, CVT_TBL_ENTRY, -(SP)   ; 8890
                        F9F2      CF  03  FB 00071     CALLS   #3, DBG$LANGUAGE_TYPE_CONV       ;
                                  50  54  D0 00076 5$:    MOVL    R4, R0                           ; 8892
                                          04 00079     RET                                      ;
                        B2            52  57  F2 0007A 6$:    AOBLSS  R7, I, 1$                ; 8877
```

```
                                50      08   AC  D0 0007E 7$:     MOVL    VALUE2, R0                      ; 8901
                                30      16   A0  91 00082         CMPB    22(R0), #48                     ;
                                                 0A 12 00086         BNEQ    8$                              ;
                                05      06   A0  91 00088         CMPB    6(R0), #5                       ; 8902
                                                 04 12 0008C         BNEQ    8$                              ;
                             16 A0              0E 90 0008E         MOVB    #14, 22(R0)                     ; 8904
                                         FC      A8  DD 00092 8$:     PUSHL   CVT_ROUND_FLAG                  ; 8906
                                                 50  DD 00095         PUSHL   R0                              ;
                                         04      AC  DD 00097         PUSHL   VALUE1                          ;
                      00000000G  00              03  FB 0009A         CALLS   #3, DBG$COVER_DX_DX             ;
                             08  AC              50  D0 000A1         MOVL    R0, VALUE2                      ;
                                                 04 000A5            RET                                     ; 8909

; Routine Size:  166 bytes,    Routine Base:  DBG$CODE + 1CB9
```

```
;  8815          8910  1 ROUTINE CONV_TEXT_PACK_VALUE(VAL_DESC) =
;  8816          8911  1 !
;  8817          8912  1 ! FUNCTION
;  8818          8913  1 !       This routine is used to convert the text string to pack decimal.
;  8819          8914  1 !
;  8820          8915  1 ! INPUTS
;  8821          8916  1 !       VAL_DESC        - Unconverted constant.
;  8822          8917  1 !
;  8823          8918  1 ! OUTPUTS
;  8824          8919  1 !       A converted value descriptor is returned.
;  8825          8920  1 !
;  8826          8921  1
;  8827          8922  2     BEGIN
;  8828          8923  2
;  8829          8924  2     MAP
;  8830          8925  2         VAL_DESC: REF DBG$VALDESC;
;  8831          8926  2
;  8832          8927  2
;  8833          8928  2     LOCAL
;  8834          8929  2         DOT: REF VECTOR[,BYTE],
;  8835          8930  2         END_PTR: REF VECTOR[,BYTE],
;  8836          8931  2         P_VAL_DESC: REF DBG$VALDESC,
;  8837          8932  2         SCALE;
;  8838          8933  2
;  8839          8934  2
;  8840          8935  2     P_VAL_DESC = MAKE_VAL_DESC(DSC$K_DTYPE_P,
;  8841          8936  2                                DBG$NUM_BYTES(DSC$K_DTYPE_P),
;  8842          8937  2                                0,
;  8843          8938  2                                TRUE);
;  8844          8939  2
;  8845          8940  2     DOT = CH$FIND_CH(.VAL_DESC[DBG$W_VALUE_LENGTH],
;  8846          8941  2                      .VAL_DESC[DBG$L_VALUE_POINTER],
;  8847          8942  2                      %C'.');
;  8848          8943  2
;  8849          8944  2
;  8850          8945  2     ! A string of digits has no '.'.
;  8851          8946  2     !
;  8852          8947  2     IF .DOT EQL 0
;  8853          8948  2     THEN
;  8854          8949  3         BEGIN
;  8855          8950  3         SCALE = 0;
;  8856          8951  3         P_VAL_DESC[DBG$W_VALUE_LENGTH] = .VAL_DESC[DBG$W_VALUE_LENGTH];
;  8857          8952  3         P_VAL_DESC[DBG$B_VALUE_DIGITS] = .VAL_DESC[DBG$W_VALUE_LENGTH];
;  8858          8953  3         END
;  8859          8954  3
;  8860          8955  3
;  8861          8956  3     ! A string of digits has '.'.
;  8862          8957  3     !
;  8863          8958  2     ELSE
;  8864          8959  3         BEGIN
;  8865          8960  3         END_PTR = .VAL_DESC[DBG$L_VALUE_POINTER] + .VAL_DESC[DBG$W_VALUE_LENGTH] - 1;
;  8866          8961  3         SCALE = .END_PTR - .DOT;
;  8867          8962  3
;  8868          8963  3
;  8869          8964  3         ! Fill in the digits and fill in scaling factor after the
;  8870          8965  3         ! type conversion.
;  8871          8966  3         !
```

```
;  8872          8967  3          P_VAL_DESC[DBG$W_VALUE_LENGTH] = .VAL_DESC[DBG$W_VALUE_LENGTH] - 1;
;  8873          8968  3          P_VAL_DESC[DBG$B_VALUE_DIGITS] = .P_VAL_DESC[DBG$W_VALUE_LENGTH];
;  8874          8969  3
;  8875          8970  3
;  8876          8971  3          ! Fix up the original value descriptor.
;  8877          8972  3          !
;  8878          8973  3          IF .DOT LSS .END_PTR
;  8879          8974  3          THEN
;  8880          8975  3              CH$MOVE(.SCALE, .DOT+1, .DOT);
;  8881          8976  3
;  8882          8977  3          END_PTR = .END_PTR AND %X'00';
;  8883          8978  3          VAL_DESC[DBG$W_VALUE_LENGTH] = .P_VAL_DESC[DBG$W_VALUE_LENGTH];
;  8884          8979  2          END;
;  8885          8980  2
;  8886          8981  2
;  8887          8982  2      ! Convert the string to decimal.
;  8888          8983  2      !
;  8889          8984  2      VAL_DESC[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_T;
;  8890          8985  2      VAL_DESC[DBG$W_VALUE_TOKENCODE] = 0;
;  8891          8986  2      P_VAL_DESC = DBG$TYPE_CONV(.VAL_DESC, .P_VAL_DESC);
;  8892          8987  2      P_VAL_DESC[DBG$B_VALUE_SCALE] = -.SCALE;
;  8893          8988  2      RETURN .P_VAL_DESC;
;  8894          8989  1      END;


                                        03FC 00000 CONV_TEXT_PACK_VALUE:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                    ;  8910
                                        01  DD 00002     PUSHL   #1                                      ;  8935
                                    7E  15  7D 00004     MOVQ    #21, -(SP)                             ;  8936
                                FB23    CF  01  FB 00007     CALLS   #1, DBG$NUM_BYTES
                                        50  DD 0000C     PUSHL   R0
                                        15  DD 0000E     PUSHL   #21                                     ;  8935
                                0000V   CF  04  FB 00010     CALLS   #4, MAKE_VAL_DESC
                                        58  50  D0 00015     MOVL    R0, P_VAL_DESC
                                        57  04  AC  D0 00018     MOVL    VAL_DESC, R7                    ;  8940
                                        52  14  A7  3C 0001C     MOVZWL  20(R7), R2
                        18      B7          52  2E  3A 00020     LOCC    #46, R2, @24(R7)
                                            02  12 00025     BNEQ    1$
                                            51  D4 00027     CLRL    R1
                                            51  D5 00029 1$:  TSTL    DOT                             ;  8947
                                            0C  12 0002B     BNEQ    2$
                                            59  D4 0002D     CLRL    SCALE                           ;  8950
                                    14      A8  52  B0 0002F     MOVW    R2, 20(P_VAL_DESC)              ;  8951
                                    1D      A8  52  90 00033     MOVB    R2, 29(P_VAL_DESC)              ;  8952
                                            26  11 00037     BRB     4$                              ;  8947
                            56              52  18  A7  C1 00039 2$:  ADDL3   24(R7), R2, R6            ;  8960
                                            56  D7 0003E     DECL    END_PTR
                            59              56  51  C3 00040     SUBL3   DOT, END_PTR, SCALE             ;  8961
                        14  A8              52  01  A3 00044     SUBW3   #1, R2, 20(P_VAL_DESC)          ;  8967
                                    1D      A8  14  A8  90 00049     MOVB    20(P_VAL_DESC), 29(P_VAL_DESC)  ;  8968
                                            56  51  D1 0004E     CMPL    DOT, END_PTR                    ;  8973
                                            05  18 00051     BGEQ    3$
                            61      01      A1  59  28 00053     MOVC3   SCALE, 1(DOT), (DOT)            ;  8975
                                            56  D4 00058 3$:  CLRL    END_PTR                         ;  8977
```

```
                    14   A7        14   A8  B0 0005A          MOVW    20(P_VAL_DESC), 20(R7)          ; 8978
                    16   A7              0E  90 0005F 4$:     MOVB    #14, 22(R7)                     ; 8984
                                    10   A7  B4 00063          CLRW    16(R7)                          ; 8985
                         7E              57  7D 00066          MOVQ    R7, -(SP)                       ; 8986
                  FEEC   CF              02  FB 00069          CALLS   #2, DBG$TYPE_CONV               ;
                         58              50  D0 0006E          MOVL    R0, P_VAL_DESC                  ;
                    1C   A8              59  8E 00071          MNEGB   SCALE, 28(P_VAL_DESC)           ; 8987
                         50              58  D0 00075          MOVL    P_VAL_DESC, R0                  ; 8988
                                             04 00078          RET                                     ; 8989

; Routine Size:  121 bytes,    Routine Base:  DBG$CODE + 1D5F

; 8895          8990  1
```

```
;  8897          8991  1 ROUTINE GET_DATA_LENGTH(FROM_DTYPE, TO_DTYPE, FROM_LENGTH) =
;  8898          8992  1 !
;  8899          8993  1 ! FUNCTION
;  8900          8994  1 !       This routine trys to guess the length of the TO_DTYPE given
;  8901          8995  1 !       FROM_DTYPE.  Sometimes, it is not obivious what is the length
;  8902          8996  1 !       for the TO_DTYPE, for example, L --> V.  This routine first
;  8903          8997  1 !       will fill in the normal length, then will correct the length
;  8904          8998  1 !       by the language rules.  Note: language code in here is not
;  8905          8999  1 !       tested for most of the cases are isolated.
;  8906          9000  1 !
;  8907          9001  1 ! INPUTS
;  8908          9002  1 !       FROM_DTYPE      - Source dtype.
;  8909          9003  1 !
;  8910          9004  1 !       TO_DTYPE        - Target dtype.
;  8911          9005  1 !
;  8912          9006  1 !       FROM_LENGTH     - Source dtype length.
;  8913          9007  1 !
;  8914          9008  1 ! OUTPUTS
;  8915          9009  1 !       TO_LENGTH (target dtype length) is returned.
;  8916          9010  1 !
;  8917          9011  1
;  8918          9012  2     BEGIN
;  8919          9013  2
;  8920          9014  2     LOCAL
;  8921          9015  2         TO_LENGTH;                      ! Target dtype length
;  8922          9016  2
;  8923          9017  2
;  8924          9018  2     ! Fill in the length by normal way.  Just in case, we did not update
;  8925          9019  2     ! the length correctly later on, we still have a length.
;  8926          9020  2     !
;  8927          9021  2     TO_LENGTH = DBG$NUM_BYTES(.TO_DTYPE);
;  8928          9022  2
;  8929          9023  2
;  8930          9024  2     ! Update the length by specified rules.
;  8931          9025  2     !
;  8932          9026  2     CASE .TO_DTYPE FROM DBG$K_MINIMUM_DTYPE TO DBG$K_MAXIMUM_DTYPE OF
;  8933          9027  2         SET
;  8934          9028  2
;  8935          9029  2
;  8936          9030  2         ! If we are converting to FIXED then we use the length from
;  8937          9031  2         ! the source descriptor.
;  8938          9032  2         !
;  8939          9033  2         [DSC$K_DTYPE_FIXED]:
;  8940          9034  2             TO_LENGTH = .FROM_LENGTH;
;  8941          9035  2
;  8942          9036  2
;  8943          9037  2         [DSC$K_DTYPE_V, DSC$K_DTYPE_VU]:
;  8944          9038  3             BEGIN
;  8945          9039  3             CASE .FROM_DTYPE FROM DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST OF
;  8946          9040  3                 SET
;  8947          9041  3                 [DSC$K_DTYPE_V, DSC$K_DTYPE_VU]:
;  8948          9042  3                     TO_LENGTH = .FROM_LENGTH;
;  8949          9043  3                 [DSC$K_DTYPE_B]:
;  8950          9044  3                     TO_LENGTH = 7;
;  8951          9045  3                 [DSC$K_DTYPE_W]:
;  8952          9046  3                     TO_LENGTH = 15;
;  8953          9047  3                 [DSC$K_DTYPE_P]:
```

```
;  8954          9048  4                 BEGIN
;  8955          9049  4                 TO_LENGTH = ((.FROM_LENGTH * 332) + 99) / 100;
;  8956          9050  4                 TO_LENGTH = MIN(31, .TO_LENGTH);
;  8957          9051  3                 END;
;  8958          9052  3             [INRANGE]:
;  8959          9053  3                 TO_LENGTH = 31;
;  8960          9054  3             [OUTRANGE]:
;  8961          9055  4                 BEGIN
;  8962          9056  4                 IF .FROM_DTYPE EQL DBG$K_DTYPE_PICT
;  8963          9057  4                 THEN
;  8964          9058  4                     TO_LENGTH = 31
;  8965          9059  4                 ELSE
;  8966          9060  4                     $DBG_ERROR('DBGEVALOP\GET_DATA_LENGTH');
;  8967          9061  3                 END;
;  8968          9062  3             TES;
8 ;  8969          9063  2         END;
;  8970          9064  2
;  8971          9065  2     [DSC$K_DTYPE_T, DSC$K_DTYPE_VT]:
;  8972          9066  3         BEGIN
;  8973          9067  3         CASE .FROM_DTYPE FROM DSC$K_DTYPE_LOWEST TO DSC$K_DTYPE_HIGHEST OF
;  8974          9068  3             SET
6 ;  8975          9069  3             [DSC$K_DTYPE_V, DSC$K_DTYPE_VU, DSC$K_DTYPE_T, DSC$K_DTYPE_VT]:
;  8976          9070  3                 TO_LENGTH = .FROM_LENGTH;
8 ;  8977          9071  3             [DSC$K_DTYPE_B]:
;  8978          9072  3                 TO_LENGTH = 7;
;  8979          9073  3             [DSC$K_DTYPE_W]:
;  8980          9074  3                 TO_LENGTH = 7;
3 ;  8981          9075  3             [DSC$K_DTYPE_L]:
;  8982          9076  3                 TO_LENGTH = 15;
4 ;  8983          9077  3             [DSC$K_DTYPE_F]:
;  8984          9078  3                 TO_LENGTH = 15;
6 ;  8985          9079  3             [DSC$K_DTYPE_D, DSC$K_DTYPE_G]:
;  8986          9080  3                 TO_LENGTH = 25;
8 ;  8987          9081  3             [DSC$K_DTYPE_H]:
;  8988          9082  3                 TO_LENGTH = 42;
;  8989          9083  3             [DSC$K_DTYPE_P]:
;  8990          9084  3                 TO_LENGTH = .FROM_LENGTH + 5;
4 ;  8991          9085  3             [INRANGE, OUTRANGE]:
;  8992          9086  4                 BEGIN
;  8993          9087  4                 IF .FROM_DTYPE EQL DBG$K_DTYPE_PICT
;  8994          9088  4                 THEN
;  8995          9089  4                     TO_LENGTH = .FROM_LENGTH
7 ;  8996          9090  4                 ELSE
;  8997          9091  4                     $DBG_ERROR('DBGEVALOP\GET_DATA_LENGTH');
8 ;  8998          9092  3                 END;
;  8999          9093  3             TES;
0 ;  9000          9094  2         END;
;  9001          9095  2
2 ;  9002          9096  2         [INRANGE, OUTRANGE]:
;  9003          9097  3             BEGIN
;  9004          9098  3             0;
;  9005          9099  2             END;
0 ;  9006          9100  2
;  9007          9101  2         TES;
;  9008          9102  2
;  9009          9103  2     RETURN .TO_LENGTH;
2 ;  9010          9104  1     END;
```

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

5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 19  060A1 P.AMN:   .ASCII  <25>\DBGEVALOP\<92>\GET_DATA_LENGTH\
                48 54 47 4E 45 4C 5F 41 54 41 44  060B0
5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 19  060BB P.AMO:   .ASCII  <25>\DBGEVALOP\<92>\GET_DATA_LENGTH\
                48 54 47 4E 45 4C 5F 41 54 41 44  060CA


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

                                                  0004 00000 GET_DATA_LENGTH:
                                                             .WORD    Save R2                                    ; 8991
                                              08   AC  DD 00002        PUSHL    TO_DTYPE                         ; 9021
                                        FAAC    CF      01  FB 00005    CALLS    #1, DBG$NUM_BYTES
                                          52            50  D0 0000A    MOVL     R0, TO_LENGTH
                                          2B      00    08  AC  CF 0000D    CASEL    TO_DTYPE, #0, #43           ; 9026
                                016E    016E    005B        016E      00012 1$:      .WORD    19$-1$,-
                                016E    016E    016E        016E      0001A                   2$-1$,-
                                016E    016E    016E        016E      00022                   19$-1$,-
                                016E    00E3    016E        016E      0002A                   19$-1$,-
                                016E    016E    016E        016E      00032                   19$-1$,-
                                016E    016E    016E        016E      0003A                   19$-1$,-
                                016E    016E    016E        016E      00042                   19$-1$,-
                                016E    016E    016E        016E      0004A                   19$-1$,-
                                016E    005B    016E        016E      00052                   19$-1$,-
                                016E    016E    00E3        016E      0005A                   19$-1$,-
                                014F    016E    016E        016E      00062                   19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              7$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
                                                                                              2$-1$,-
                                                                                              19$-1$,-
                                                                                              19$-1$,-
```

```
                                                                  7$-1$,-                    ;
                                                                  19$-1$,-                   ;
                                                                  19$-1$,-                   ;
                                                                  19$-1$,-                   ;
                                                                  19$-1$,-                   ;
                                                                  19$-1$,-                   ;
                                                                  11$-1$                     ;
                                     0113  31 0006A       BRW     19$                        ;
                      24      01  04   AC  CF 0006D 2$:   CASEL   FROM_DTYPE, #1, #36        ;  9039
    007E      007E      007E         00EF     00072 3$:   .WORD   11$-3$,-                   ;
    007E      00FA      00F5         007E     0007A               6$-3$,-                    ;
    007E      007E      007E         007E     00082               6$-3$,-                    ;
    007E      007E      007E         007E     0008A               6$-3$,-                    ;
    007E      007E      007E         007E     00092               6$-3$,-                    ;
    007E      007E      007E         0059     0009A               12$-3$,-                   ;
    007E      007E      007E         007E     000A2               13$-3$,-                   ;
    007E      007E      007E         007E     000AA               6$-3$,-                    ;
    007E      007E      00EF         007E     000B2               6$-3$,-                    ;
                                     007E     000BA               6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  4$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  11$-3$,-                   ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$,-                    ;
                                                                  6$-3$                      ;
                              30      04   AC  D1 000BC           CMPL    FROM_DTYPE, #48    ;  9056
                                           2E  13 000C0           BEQL    6$                 ;
                                 00000000'  EF  9F 000C2          PUSHAB  P.AMN              ;  9060
                                       0085  31 000C8             BRW     10$                ;
                   50       0C  AC 0000014C  8F  C5 000CB 4$:     MULL3   #332, FROM_LENGTH, R0  ;  9049
                                50       63  A0  9E 000D4         MOVAB   99(R0), R0         ;
                   52           50 00000064  8F  C7 000D8         DIVL3   #100, R0, TO_LENGTH    ;
                                50           52  D0 000E0         MOVL    TO_LENGTH, R0      ;  9050
                                1F           50  D1 000E3         CMPL    R0, #31            ;
                                             03  15 000E6         BLEQ    5$                 ;
                                50           1F  D0 000E8         MOVL    #31, R0            ;
```

```
                            52          50  D0 000EB 5$:      MOVL    R0, TO_LENGTH                                  ;
                                        7F  11 000EE          BRB     14$                                            ; 9039
                            52          1F  D0 000F0 6$:      MOVL    #31, TO_LENGTH                                 ; 9053
                                        7F  11 000F3          BRB     16$                                            ; 9026
                  24        01      04  AC  CF 000F5 7$:      CASEL   FROM_DTYPE, #1, #36                            ; 9067
      004A      004A      004A        0067     000FA 8$:      .WORD   11$-8$,-                                       ;
      0072      006D      006D        004A     00102                  9$-8$,-                                        ;
      004A      0077      0072        004A     0010A                  9$-8$,-                                        ;
      004A      004A      0067        004A     00112                  9$-8$,-                                        ;
      004A      004A      004A        004A     0011A                  9$-8$,-                                        ;
      004A      004A      004A        0081     00122                  12$-8$,-                                       ;
      007C      0077      004A        004A     0012A                  12$-8$,-                                       ;
      004A      004A      004A        004A     00132                  13$-8$,-                                       ;
      004A      004A      0067        004A     0013A                  9$-8$,-                                        ;
                                      0067     00142                  13$-8$,-                                       ;
                                                                      15$-8$,-                                       ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      11$-8$,-                                       ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      18$-8$,-                                       ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      15$-8$,-                                       ;
                                                                      17$-8$,-                                       ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      11$-8$,-                                       ;
                                                                      9$-8$,-                                        ;
                                                                      9$-8$,-                                        ;
                                                                      11$-8$                                         ;
                            30      04  AC  D1 00144 9$:      CMPL    FROM_DTYPE, #48                                ; 9087
                                        17  13 00148          BEQL    11$                                            ;
                              00000000' EF  9F 0014A          PUSHAB  P.AMO                                          ; 9091
                                        01  DD 00150 10$:     PUSHL   #1                                             ;
                              00028362  8F  DD 00152          PUSHL   #164706                                        ;
                 00000000G  00          03  FB 00158          CALLS   #3, LIB$SIGNAL                                 ;
                                        1F  11 0015F          BRB     19$                                            ; 9067
                            52      0C  AC  D0 00161 11$:     MOVL    FROM_LENGTH, TO_LENGTH                         ; 9070
                                        19  11 00165          BRB     19$                                            ;
                            52          07  D0 00167 12$:     MOVL    #7, TO_LENGTH                                  ; 9074
                                        14  11 0016A          BRB     19$                                            ;
                            52          0F  D0 0016C 13$:     MOVL    #15, TO_LENGTH                                 ; 9078
                                        0F  11 0016F 14$:     BRB     19$                                            ;
                            52          19  D0 00171 15$:     MOVL    #25, TO_LENGTH                                 ; 9080
                                        0A  11 00174 16$:     BRB     19$                                            ;
```

```
                                52              2A  D0 00176 17$:    MOVL    #42, TO_LENGTH                          ; 9082
                                                05  11 00179         BRB     19$                                     ;
                          52    0C    AC        05  C1 0017B 18$:    ADDL3   #5, FROM_LENGTH, TO_LENGTH              ; 9084
                                50              52  D0 00180 19$:    MOVL    TO_LENGTH, R0                           ; 9103
                                                    04 00183         RET                                             ; 9104

; Routine Size:  388 bytes,    Routine Base:  DBG$CODE + 1DD8

; 9011          9105  1
```

```
; 9013          9106  1 ROUTINE GET_SCALE(VAL_DESC, DIGITS) =
; 9014          9107  1 !
; 9015          9108  1 ! FUNCTION
; 9016          9109  1 !       This routine takes Floating-Point value descriptor data, convert its
; 9017          9110  1 !       value to text string, get the exponent and number of digits information
; 9018          9111  1 !       from the text string.  Then we get the scaling factor from these
; 9019          9112  1 !       two numbers.
; 9020          9113  1 !
; 9021          9114  1 ! INPUT
; 9022          9115  1 !       VAL_DESC        - Floating-Point value descriptor.
; 9023          9116  1 !
; 9024          9117  1 !       DIGITS          - Address of the number of the digits.
; 9025          9118  1 !
; 9026          9119  1 ! OUTPUT
; 9027          9120  1 !       The scaling factor is returned.
; 9028          9121  1 !
; 9029          9122  1
; 9030          9123  2     BEGIN
; 9031          9124  2
; 9032          9125  2     MAP
; 9033          9126  2         DIGITS: REF VECTOR[1],          ! Address of the number of the
; 9034          9127  2                                         !    digits.
; 9035          9128  2         VAL_DESC: REF DBG$VALDESC;      ! Pointer to value descriptor.
; 9036          9129  2
; 9037          9130  2     LOCAL
; 9038          9131  2         BUFFER: VECTOR[50, BYTE],       ! Text string buffer
; 9039          9132  2         DESCRIPTOR: BLOCK[8, BYTE],     ! Vax standard descriptor.
; 9040          9133  2         DIGITS_IN_FRACT,                ! The number of digits in the fraction
; 9041          9134  2         D_PTR: REF VECTOR[,BYTE],       ! Pointer to text string at the '.'
; 9042          9135  2         E_FLAG,                         ! Flag set to indicate the E-format
; 9043          9136  2                                         !       text string has 'E'
; 9044          9137  2         E_PTR: REF VECTOR[,BYTE],       ! Pointer to text string at E position
; 9045          9138  2         EXPONENT,                       ! The exponent of the floating-point
; 9046          9139  2         SCALE,                          ! The scaling factor
; 9047          9140  2         STATUS;                         ! Return status from library routine
; 9048          9141  2
; 9049          9142  2
; 9050          9143  2     ! Set up the descriptor.
; 9051          9144  2     !
; 9052          9145  2     DESCRIPTOR[DSC$B_CLASS] = DSC$K_CLASS_S;
; 9053          9146  2     DESCRIPTOR[DSC$B_DTYPE] = DSC$K_DTYPE_T;
; 9054          9147  2     DESCRIPTOR[DSC$W_LENGTH] = 50;
; 9055          9148  2     DESCRIPTOR[DSC$A_POINTER] = BUFFER;
; 9056          9149  2
; 9057          9150  2
; 9058          9151  2     ! Convert Floating-Point data to E formatted text string.
; 9059          9152  2     !
; 9060          9153  2     CASE .VAL_DESC[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_H OF
; 9061          9154  2         SET
; 9062          9155  2         [DSC$K_DTYPE_F, DSC$K_DTYPE_D]:
; 9063          9156  3             BEGIN
; 9064          9157  3             CASE .VAL_DESC[DBG$B_VALUE_DTYPE] FROM DSC$K_DTYPE_F TO DSC$K_DTYPE_D OF
; 9065          9158  3                 SET
; 9066          9159  3                 [DSC$K_DTYPE_F]:
; 9067          9160  3                     DIGITS_IN_FRACT = 7;
; 9068          9161  3                 [DSC$K_DTYPE_D]:
; 9069          9162  3                     DIGITS_IN_FRACT = 16;
```

```
;  9070          9163  3                TES;
;  9071          9164  3
;  9072          9165  3
;  9073          9166  3            ! In here, the exponent digits is default to 2.
;  9074          9167  3            !
;  9075          9168  3            STATUS = FOR$CVT_D_TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
;  9076          9169  3                .DIGITS_IN_FRACT);
;  9077          9170  2            END;
;  9078          9171  2
;  9079          9172  2        [DSC$K_DTYPE_G]:
;  9080          9173  3            BEGIN
;  9081          9174  3            DIGITS_IN_FRACT = 15;
;  9082          9175  3            STATUS = FOR$CVT_G_TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
;  9083          9176  3                .DIGITS_IN_FRACT, 0, 0, 3);
;  9084          9177  2            END;
;  9085          9178  2
;  9086          9179  2        [DSC$K_DTYPE_H]:
;  9087          9180  3            BEGIN
;  9088          9181  3            DIGITS_IN_FRACT = 33;
;  9089          9182  3            STATUS = FOR$CVT_H_TE(.VAL_DESC[DBG$L_VALUE_POINTER], DESCRIPTOR,
;  9090          9183  3                .DIGITS_IN_FRACT, 0, 0, 4);
;  9091          9184  2            END;
;  9092          9185  2
;  9093          9186  2        [INRANGE, OUTRANGE]:
;  9094          9187  2            $DBG_ERROR('DBGEVALOP\GET_SCALE, not floating-point data');
;  9095          9188  2
;  9096          9189  2        TES;
;  9097          9190  2
;  9098          9191  2    IF NOT .STATUS THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, FOR$CVT_x_yE error');
;  9099          9192  2
;  9100          9193  2
;  9101          9194  2    ! Locate "+" or "-".
;  9102          9195  2    ! (E+nn or E-nn for exp. <= 99), (+nnn or -nnn for exp. <=999)
;  9103          9196  2    ! is the standard fortran E-format.
;  9104          9197  2    !
;  9105          9198  2    E_FLAG = CH$FIND_CH(50, BUFFER, %C'E');
;  9106          9199  2    E_PTR = CH$FIND_CH(50, BUFFER, %C'+');
;  9107          9200  2    IF .E_PTR EQL 0
;  9108          9201  2    THEN
;  9109          9202  2        E_PTR = CH$FIND_CH(50, BUFFER, %C'-');
;  9110          9203  2
;  9111          9204  2    IF .E_PTR EQL 0 THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, not E-format expected');
;  9112          9205  2
;  9113          9206  2
;  9114          9207  2    ! Locate the '.'.  For example: 0.48E+06.
;  9115          9208  2    !
;  9116          9209  2    D_PTR = CH$FIND_CH(50, BUFFER, %C'.');
;  9117          9210  2
;  9118          9211  2
;  9119          9212  2    ! Convert the next few characters into integer (+/- and digits).
;  9120          9213  2    !
;  9121          9214  2    DESCRIPTOR[DSC$W_LENGTH] = BUFFER[49] - .E_PTR + 1;
;  9122          9215  2    DESCRIPTOR[DSC$A_POINTER] = .E_PTR;
;  9123          9216  2    STATUS = OTS$CVT_TI_L(DESCRIPTOR, EXPONENT);
;  9124          9217  2    IF NOT .STATUS THEN $DBG_ERROR('DBGEVALOP\GET_SCALE, OTS$CVT_TI_L error');
;  9125          9218  2
;  9126          9219  2    DIGITS[0] = .E_PTR - .D_PTR - 1;
```

```
: 9127          9220  2      IF .E_FLAG NEQ 0 THEN DIGITS[0] = .DIGITS[0] - 1;
: 9128          9221  2      SCALE = .EXPONENT - .DIGITS[0];
: 9129          9222  2      RETURN .SCALE;
: 9130          9223  2
: 9131          9224  1      END;



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

5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 2C 060D5 P.AMP:  .ASCII  \,DBGEVALOP\<92>\GET_SCALE, not floating\  ;
61 6F 6C 66 20 74 6F 6E 20 2C 45 4C 41 43 53 060E4                                                          ;
                                    67 6E 69 74 060F3                                                          ;
            61 74 61 64 20 74 6E 69 6F 70 2D 060F7          .ASCII  \-point data\                              ;
5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 27 06102 P.AMQ:  .ASCII  \'DBGEVALOP\<92>\GET_SCALE, FOR$CVT_x_yE\  ;
5F 54 56 43 24 52 4F 46 20 2C 45 4C 41 43 53 06111                                                          ;
                                    45 79 5F 78 06120                                                          ;
                              72 6F 72 72 65 20 06124          .ASCII  \ error\                                   ;
5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 2A 0612A P.AMR:  .ASCII  \*DBGEVALOP\<92>\GET_SCALE, not E-format\  ;
6F 66 2D 45 20 74 6F 6E 20 2C 45 4C 41 43 53 06139                                                          ;
                                    74 61 6D 72 06148                                                          ;
                  64 65 74 63 65 70 78 65 20 0614C          .ASCII  \ expected\                                ;
5F 54 45 47 5C 50 4F 4C 41 56 45 47 42 44 27 06155 P.AMS:  .ASCII  \'DBGEVALOP\<92>\GET_SCALE, OTS$CVT_TI_L\  ;
5F 54 56 43 24 53 54 4F 20 2C 45 4C 41 43 53 06164                                                          ;
                                    4C 5F 49 54 06173                                                          ;
                              72 6F 72 72 65 20 06177          .ASCII  \ error\                                   ;



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

                                    00FC 00000 GET_SCALE:
                                                            .WORD   Save R2,R3,R4,R5,R6,R7                      ; 9106
                  57 00000000'  EF  9E 00002                MOVAB   P.AMP, R7                                   ;
                  56 00000000G  00  9E 00009                MOVAB   LIB$SIGNAL, R6                              ;
                  5E        C0  AE  9E 00010                MOVAB   -64(SP), SP                                 ;
            04    AE  010E0032  8F  D0 00014                MOVL    #17694770, DESCRIPTOR                       ; 9147
            08    AE        0C  AE  9E 0001C                MOVAB   BUFFER, DESCRIPTOR+4                        ; 9148
                  52        04  AC  D0 00021                MOVL    VAL_DESC, R2                                ; 9153
      12          0A        16  A2  8F 00025                CASEB   22(R2), #10, #18                            ;
0026  0026      0035      0035         0002A 1$:            .WORD   3$-1$,-                                     ;
0026  0026      0026      0026         00032                        3$-1$,-                                     ;
0026  0026      0026      0026         0003A                        2$-1$,-                                     ;
0026  0026      0026      0026         00042                        2$-1$,-                                     ;
      006F      0057      0026         0004A                        2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
                                                                    2$-1$,-                                     ;
```

```
                                                                8$-1$,-                                                 ;
                                                                9$-1$                                                   ;
                                 57  DD 00050 2$:    PUSHL   R7                                                      ; 9187
                                 01  DD 00052        PUSHL   #1                                                      ;
                       00028362  8F  DD 00054        PUSHL   #164706                                                 ;
                              66   03  FB 00059        CALLS   #3, LIB$SIGNAL                                          ;
                                 53  11  0005D        BRB     11$                                                     ;
                              0A  16  A2  8F 0005F 3$:  CASEB   22(R2), #10, #1                                        ; 9157
                            0009      0004     00064 4$:  .WORD   5$-4$,-                                              ;
                                                                6$-4$                                                   ;
                              50        07  D0 00068 5$:  MOVL    #7, DIGITS_IN_FRACT                                   ; 9160
                                 03  11  0006B        BRB     7$                                                      ;
                              50        10  D0 0006D 6$:  MOVL    #16, DIGITS_IN_FRACT                                  ; 9162
                                 50  DD  00070 7$:  PUSHL   DIGITS_IN_FRACT                                         ; 9169
                           08    AE  9F  00072        PUSHAB  DESCRIPTOR                                              ; 9168
                           18    A2  DD  00075        PUSHL   24(R2)                                                  ;
                 00000000G   00        03  FB 00078        CALLS   #3, FOR$CVT_D_TE                                  ;
                                 2E  11  0007F        BRB     10$                                                     ;
                              50        0F  D0 00081 8$:  MOVL    #15, DIGITS_IN_FRACT                                  ; 9174
                                 03  DD  00084        PUSHL   #3                                                      ; 9175
                                 7E  7C  00086        CLRQ    -(SP)                                                   ;
                                 50  DD  00088        PUSHL   DIGITS_IN_FRACT                                         ; 9176
                           14    AE  9F  0008A        PUSHAB  DESCRIPTOR                                              ; 9175
                           18    A2  DD  0008D        PUSHL   24(R2)                                                  ;
                 00000000G   00        06  FB 00090        CALLS   #6, FOR$CVT_G_TE                                  ;
                                 16  11  00097        BRB     10$                                                     ;
                              50        21  D0 00099 9$:  MOVL    #33, DIGITS_IN_FRACT                                  ; 9181
                                 04  DD  0009C        PUSHL   #4                                                      ; 9182
                                 7E  7C  0009E        CLRQ    -(SP)                                                   ;
                                 50  DD  000A0        PUSHL   DIGITS_IN_FRACT                                         ; 9183
                           14    AE  9F  000A2        PUSHAB  DESCRIPTOR                                              ; 9182
                           18    A2  DD  000A5        PUSHL   24(R2)                                                  ;
                 00000000G   00        06  FB 000A8        CALLS   #6, FOR$CVT_H_TE                                  ;
                              53        50  D0 000AF 10$:  MOVL    R0, STATUS                                        ;
                              0E        53  E8 000B2 11$:  BLBS    STATUS, 12$                                       ; 9191
                           2D    A7  9F  000B5        PUSHAB  P.AMQ                                                   ;
                                 01  DD  000B8        PUSHL   #1                                                      ;
                       00028362  8F  DD  000BA        PUSHL   #164706                                                 ;
                              66   03  FB 000C0        CALLS   #3, LIB$SIGNAL                                          ;
                 0C    AE           32        8F  3A 000C3 12$:  LOCC    #69, #50, BUFFER                            ; 9198
                                 02  12  000C9        BNEQ    13$                                                     ;
                                 51  D4  000CB        CLRL    R1                                                      ;
                              55        51  D0 000CD 13$:  MOVL    R1, E_FLAG                                        ;
                 0C    AE           32        2B  3A 000D0        LOCC    #43, #50, BUFFER                            ; 9199
                                 02  12  000D5        BNEQ    14$                                                     ;
                                 51  D4  000D7        CLRL    R1                                                      ;
                              52        51  D0 000D9 14$:  MOVL    R1, E_PTR                                         ;
                                 1C  12  000DC        BNEQ    16$                                                     ; 9200
                 0C    AE           32        2D  3A 000DE        LOCC    #45, #50, BUFFER                            ; 9202
                                 02  12  000E3        BNEQ    15$                                                     ;
                                 51  D4  000E5        CLRL    R1                                                      ;
                              52        51  D0 000E7 15$:  MOVL    R1, E_PTR                                         ;
                                 0E  12  000EA        BNEQ    16$                                                     ; 9204
                           55    A7  9F  000EC        PUSHAB  P.AMR                                                   ;
                                 01  DD  000EF        PUSHL   #1                                                      ;
                       00028362  8F  DD  000F1        PUSHL   #164706                                                 ;
                              66   03  FB 000F7        CALLS   #3, LIB$SIGNAL                                          ;
```

```
                 0C   AE          32          2E  3A 000FA 16$:   LOCC    #46, #50, BUFFER                ; 9209
                                              02  12 000FF        BNEQ    17$                             ;
                                              51  D4 00101        CLRL    R1                              ;
                             54               51  D0 00103 17$:   MOVL    R1, D_PTR                       ;
                             50        3D     AE  9E 00106        MOVAB   BUFFER+49, R0                   ; 9214
                             50               52  C2 0010A        SUBL2   E_PTR, R0                       ;
                 04   AE     50               01  A1 0010D        ADDW3   #T, R0, DESCRIPTOR              ;
                       08    AE               52  D0 00112        MOVL    E_PTR, DESCRIPTOR+4             ; 9215
                                              5E  DD 00116        PUSHL   SP                              ; 9216
                                  08          AE  9F 00118        PUSHAB  DESCRIPTOR                      ;
                    00000000G     00          CF  FB 0011B        CALLS   #2, OTS$CVT_TI_L                ;
                             53               50  D0 00122        MOVL    R0, STATUS                      ;
                             0F               53  E8 00125        BLBS    STATUS, 18$                     ; 9217
                                  0080        C7  9F 00128        PUSHAB  P.AMS                           ;
                                              01  DD 0012C        PUSHL   #1                              ;
                                00028362      8F  DD 0012E        PUSHL   #164706                         ;
                             66               03  FB 00134        CALLS   #3, LIB$SIGNAL                  ;
                 51          52               54  C3 00137 18$:   SUBL3   D_PTR, E_PTR, R1                ; 9219
                       08    BC        FF     A1  9E 0013B        MOVAB   -T(R1), @DIGITS                 ;
                                              55  D5 00140        TSTL    E_FLAG                          ; 9220
                                              03  13 00142        BEQL    19$                             ;
                                  08          BC  D7 00144        DECL    @DIGITS                         ;
                 50               6E        08  BC  C3 00147 19$:   SUBL3   @DIGITS, EXPONENT, SCALE      ; 9221
                                              04  0014C        RET                                     ; 9224

; Routine Size:  333 bytes,    Routine Base:  DBG$CODE + 1F5C

; 9132          9225  1
```

```
;  9134          9226  1 ROUTINE FIND_JOIN (LEFT_TYPE, RIGHT_TYPE,
;  9135          9227  1                    NEW_LEFT_TYPE, NEW_RIGHT_TYPE, ROUT_INDEX,
;  9136          9228  1                    DEPTH, MIN,
;  9137          9229  1                    HIER_TBL, HIER_TBL_SIZE,
;  9138          9230  1                    INCOMP_TBL, INCOMP_TBL_SIZE,
;  9139          9231  1                    ROUT_TBL, ROUT_TBL_SIZE) =
;  9140          9232  1
;  9141          9233  1 !
;  9142          9234  1 ! FUNCTION
;  9143          9235  1 !       Given a pair of types and a Type Hierarchy Graph, this routine attempts
;  9144          9236  1 !       to determine what conversions should be done on the types. It does
;  9145          9237  1 !       this by finding the "join" of the two nodes in the graph. Usually,
;  9146          9238  1 !       this will just be a path from the lower type to the higher type; e.g.,
;  9147          9239  1 !       if integer is added to float, float is the higher type, and a path
;  9148          9240  1 !       from integer to float is found, so the integer is converted to float.
;  9149          9241  1 !       There may be cases where both are converted to a higher type; e.g.,
;  9150          9242  1 !       in FORTRAN, if FLOAT COMPLEX is added to D_FLOAT, both are promoted
;  9151          9243  1 !       to D_FLOAT COMPLEX.
;  9152          9244  1 !
;  9153          9245  1 ! INPUTS
;  9154          9246  1 !       HIER_TBL         - Hier. Table
;  9155          9247  1 !       HIER_TBL_SIZE    - Hier. Table Size
;  9156          9248  1 !       INCOMP_TBL       - Incomp. Table
;  9157          9249  1 !       INCOMP_TBL_SIZE  - Incomp. Table Size
;  9158          9250  1 !       ROUT_TBL         - Rout. Table
;  9159          9251  1 !       ROUT_TBL_SIZE    - Rout. Table Size
;  9160          9252  1 !       LEFT_TYPE        - The type of the left operand.
;  9161          9253  1 !       RIGHT_TYPE       - The type of the right operand.
;  9162          9254  1 !       NEW_LEFT_TYPE    - The address in which to place the new type of the
;  9163          9255  1 !                          left operand.
;  9164          9256  1 !       NEW_RIGHT_TYPE   - The address in which to place the new type of the
;  9165          9257  1 !                          right operand.
;  9166          9258  1 !       ROUT_INDEX       - An address in which to fill in a routine index from
;  9167          9259  1 !                        -       the Operator Routine Table
;  9168          9260  1 !       DEPTH            - Depth of recursion. DBG$EVAL_LANG_OPERATOR passes
;  9169          9261  1 !                          in zero, and this routine passes 1+.DEPTH when
;  9170          9262  1 !                          it calls itself, so it can keep track of the
;  9171          9263  1 !                          recursion depth.
;  9172          9264  1 !       MIN              - The minimum length conversion path so far.
;  9173          9265  1 !
;  9174          9266  1 ! OUTPUTS
;  9175          9267  1 !       Routine Value: If a set of conversions to legal types was found,
;  9176          9268  1 !       the length of the paths through the conversion graph is returned.
;  9177          9269  1 !       Otherwise, MAX_DEPTH is returned. (I.e., a set of conversions were
;  9178          9270  1 !       found if the return value is less than MAX_DEPTH. The reason for
;  9179          9271  1 !       doing it this way (instead of just returning TRUE/FALSE) is so that
;  9180          9272  1 !       a recursive method can be used to find the minimal path).
;  9181          9273  1 !
;  9182          9274  1 !       The following output parameters are filled in:
;  9183          9275  1 !       NEW_LEFT_TYPE  - The type to convert the left operand to.
;  9184          9276  1 !       NEW_RIGHT_TYPE - The type to convert the right operand to.
;  9185          9277  1 !       ROUT_INDEX     - An index into the Operator Routine Table, pointing
;  9186          9278  1 !                        to a legal operator on the new types
;  9187          9279  1 !
;  9188          9280  1
;  9189          9281  2     BEGIN
;  9190          9282  2
```

```
;  9191          9283  2     MAP
;  9192          9284  2         HIER_TBL: REF VECTOR [,WORD],   ! Pointer to a Type Hierarchy Table
;  9193          9285  2         INCOMP_TBL: REF VECTOR [,WORD], ! Pointer to a Type Incompatibility Table
;  9194          9286  2         ROUT_TBL: REF ORT$TABLE;        ! Pointer to an Operator Routine Table
;  9195          9287  2
;  9196          9288  2     LOCAL
;  9197          9289  2         HIER_TBL_ENTRY : TYPE_GRAPH$ENTRY,      ! An entry in the Type Hierarchy
;  9198          9290  2                                                 !     Table.
;  9199          9291  2         INCOMP_TBL_ENTRY: TYPE_GRAPH$ENTRY,     ! An entry in the Type
;  9200          9292  2                                                 !     Incompatibility Table
;  9201          9293  2         LEFT_DONE: BYTE,                        ! Flag saying we have searched
;  9202          9294  2                                                 !     recursively from left sone
;  9203          9295  2         LEFT_FOUND: BYTE,                       ! A flag saying whether we found
;  9204          9296  2                                                 !     LEFT_TYPE in the Type
;  9205          9297  2                                                 !     Hierarchy Table.
;  9206          9298  2         LEFT_INDEX,                             ! Index of the first occurence
;  9207          9299  2                                                 !     of LEFT_TYPE in the
;  9208          9300  2                                                 !     Type Hierarchy Table
;  9209          9301  2         LEFT_POTENTIAL_TYPE,                    ! Potential new left type.
;  9210          9302  2         POTENTIAL_DEPTH,                        ! Possible minimum length path
;  9211          9303  2         POTENTIAL_INDEX,                        ! A candidate for an index into
;  9212          9304  2                                                 !     the Operator Routine Table
;  9213          9305  2         REVERSE_TYPES: TYPE$PAIR,               ! A type pair
;  9214          9306  2         RIGHT_DONE: BYTE,                       ! Flag saying we have searched
;  9215          9307  2                                                 !     recursively from the
;  9216          9308  2                                                 !     right son.
;  9217          9309  2         RIGHT_FOUND: BYTE,                      ! A flag saying whether we found
;  9218          9310  2                                                 !     RIGHT_TYPE in the Type
;  9219          9311  2                                                 !     Hierarchy Table.
;  9220          9312  2         RIGHT_INDEX,                            ! Index of the first occurence
;  9221          9313  2                                                 !     of RIGHT_TYPE in the
;  9222          9314  2                                                 !     Type Hierarchy Table
;  9223          9315  2         RIGHT_POTENTIAL_TYPE,                   ! Potential new right type
;  9224          9316  2         TYPES: TYPE$PAIR;                       ! A type pair.
;  9225          9317  2
;  9226          9318  2
;  9227          9319  2     ! First see if the given type pair is legal, by searching the Operator
;  9228          9320  2     ! Routine Table. If it is, then the length of the path to a legal
;  9229          9321  2     ! type pair is zero and this is what we return.
   9230          9322  2     !
;  9231          9323  2     TYPES [BS_LEFT_TYPE] = .LEFT_TYPE;
;  9232          9324  2     TYPES [BS_RIGHT_TYPE] = .RIGHT_TYPE;
;  9233          9325  2     INCR I FROM 0 TO .ROUT_TBL_SIZE - 1 DO
;  9234          9326  2         IF .TYPES EQL .ROUT_TBL [.I, ORT$W_TYPES]
;  9235          9327  2         THEN
;  9236          9328  3             BEGIN
;  9237          9329  3             .NEW_LEFT_TYPE = .LEFT_TYPE;
;  9238          9330  3             .NEW_RIGHT_TYPE = .RIGHT_TYPE;
;  9239          9331  3             .ROUT_INDEX = .I;
;  9240          9332  3             RETURN .DEPTH;
;  9241          9333  2             END;
;  9242          9334  2
;  9243          9335  2     ! Terminate further searching if minimum so far is already smaller
;  9244          9336  2     ! than potential paths.
;  9245          9337  2     !
;  9246          9338  2     IF .DEPTH + 1 GEQ .MIN
;  9247          9339  2     THEN
```

```
;  9248          9340  2          RETURN .MAX_DEPTH;
;  9249          9341  2
;  9250          9342  2      ! Check for incompatibility of the given types.
;  9251          9343  2      !
;  9252          9344  2      REVERSE_TYPES [BS_LEFT_TYPE] = .RIGHT_TYPE;
;  9253          9345  2      REVERSE_TYPES [BS_RIGHT_TYPE] = .LEFT_TYPE;
;  9254          9346  2      IF .INCOMP_TBL NEQ TABLEBASE
;  9255          9347  2      THEN
;  9256          9348  2          INCR I FROM 0 TO .INCOMP_TBL_SIZE - 1 DO
;  9257          9349  3              BEGIN
;  9258          9350  3              INCOMP_TBL_ENTRY = .INCOMP_TBL [.I];
;  9259          9351  3              IF .INCOMP_TBL_ENTRY EQL 0 THEN EXITLOOP;
;  9260          9352  3              IF .TYPES EQL .INCOMP_TBL_ENTRY [TYPE_GRAPH$W_BOTH_TYPES]
;  9261          9353  3              THEN
;  9262          9354  3                  RETURN .MAX_DEPTH;
;  9263          9355  3              IF .REVERSE_TYPES EQL .INCOMP_TBL_ENTRY [TYPE_GRAPH$W_BOTH_TYPES]
;  9264          9356  3              THEN
;  9265          9357  3                  RETURN .MAX_DEPTH;
;  9266          9358  2              END;
;  9267          9359  2
;  9268          9360  2      ! Check for no Hierarchy Table being present.
;  9269          9361  2      !
;  9270          9362  2      IF .HIER_TBL EQL TABLEBASE
;  9271          9363  2      THEN
;  9272          9364  2          RETURN .MAX_DEPTH;
;  9273          9365  2
;  9274          9366  2      ! Find the first edges emanating from LEFT_TYPE and RIGHT_TYPE
;  9275          9367  2      ! in the Type Hierarchy Table.
;  9276          9368  2      !
;  9277          9369  2      LEFT_FOUND = FALSE;
;  9278          9370  2      RIGHT_FOUND = FALSE;
;  9279          9371  2      INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
;  9280          9372  3          BEGIN
;  9281          9373  3          HIER_TBL_ENTRY = .HIER_TBL [.I];
;  9282          9374  3          IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
;  9283          9375  3          IF NOT .LEFT_FOUND
;  9284          9376  3          THEN
;  9285          9377  3              IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE
;  9286          9378  3              THEN
;  9287          9379  4                  BEGIN
;  9288          9380  4                  LEFT_FOUND = TRUE;
;  9289          9381  4                  LEFT_INDEX = .I;
;  9290          9382  3                  END;
;  9291          9383  3          IF NOT .RIGHT_FOUND
;  9292          9384  3          THEN
;  9293          9385  3              IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .RIGHT_TYPE
;  9294          9386  3              THEN
;  9295          9387  4                  BEGIN
;  9296          9388  4                  RIGHT_FOUND = TRUE;
;  9297          9389  4                  RIGHT_INDEX = .I;
;  9298          9390  3                  END;
;  9299          9391  3          IF .LEFT_FOUND AND .RIGHT_FOUND THEN EXITLOOP;
;  9300          9392  2          END;
;  9301          9393  2
;  9302          9394  2      ! Now do the recursive calls on the edges emanating out of the given
;  9303          9395  2      ! types. As a heuristic to speed up the search, start with the lower
;  9304          9396  2      ! index.
```

```
; 9305          9397  2     !
; 9306          9398  2     LEFT_DONE = FALSE;
; 9307          9399  2     RIGHT_DONE = FALSE;
; 9308          9400  2     WHILE TRUE DO
; 9309          9401  3         BEGIN
; 9310          9402  3         IF .LEFT_FOUND
; 9311          9403  3         THEN
; 9312          9404  4             BEGIN
; 9313          9405  4
; 9314          9406  4             ! Only do the recursive call from here if either we have already
; 9315          9407  4             ! searched from the right son, or if the left index comes before
; 9316          9408  4             ! the right index.
; 9317          9409  4             !
; 9318          9410  4             IF .RIGHT_DONE OR .LEFT_INDEX LEQ .RIGHT_INDEX
; 9319          9411  4             THEN
; 9320          9412  5                 BEGIN
; 9321          9413  5                 LEFT_DONE = TRUE;
; 9322          9414  5
; 9323          9415  5                 ! Loop through all the successors to LEFT_TYPE.
; 9324          9416  5                 !
; 9325          9417  5                 HIER_TBL_ENTRY = .HIER_TBL [.LEFT_INDEX];
; 9326          9418  5                 WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .LEFT_TYPE DO
; 9327          9419  6                     BEGIN
; 9328          9420  6                     POTENTIAL_DEPTH = FIND_JOIN (
; 9329          9421  6                                 .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE],
; 9330          9422  6                                 .RIGHT_TYPE,
; 9331          9423  6                                 LEFT_POTENTIAL_TYPE,
; 9332          9424  6                                 RIGHT_POTENTIAL_TYPE,
; 9333          9425  6                                 POTENTIAL_INDEX,
; 9334          9426  6                                 1+.DEPTH,
; 9335          9427  6                                 .MIN,
; 9336          9428  6                                 .HIER_TBL, .HIER_TBL_SIZE,
; 9337          9429  6                                 .INCOMP_TBL, .INCOMP_TBL_SIZE,
; 9338          9430  6                                 .ROUT_TBL, .ROUT_TBL_SIZE);
; 9339          9431  6
; 9340          9432  6                     ! If we have a new minimum length path, then record
; 9341          9433  6                     ! all the relevant information.
; 9342          9434  6                     !
; 9343          9435  6                     IF .POTENTIAL_DEPTH LSS .MIN
; 9344          9436  6                     THEN
; 9345          9437  7                         BEGIN
; 9346          9438  7                         MIN = .POTENTIAL_DEPTH;
; 9347          9439  7                         .NEW_LEFT_TYPE = .LEFT_POTENTIAL_TYPE;
; 9348          9440  7                         .NEW_RIGHT_TYPE = .RIGHT_POTENTIAL_TYPE;
; 9349          9441  7                         .ROUT_INDEX = .POTENTIAL_INDEX;
; 9350          9442  6                         END;
; 9351          9443  6
; 9352          9444  6                     ! Set up for next time around loop.
; 9353          9445  6                     !
; 9354          9446  6                     LEFT_INDEX = .LEFT_INDEX + 1;
; 9355          9447  6                     HIER_TBL_ENTRY = .HIER_TBL [.LEFT_INDEX];
; 9356          9448  5                     END;
; 9357          9449  4                 END;
; 9358          9450  3             END;
; 9359          9451  3
; 9360          9452  3         IF .RIGHT_FOUND
; 9361          9453  3         THEN
```

```
;  9362          9454  4          BEGIN
;  9363          9455  4
;  9364          9456  4          ! Only do the recursive call from here if either we have already
;  9365          9457  4          ! searched from the left son, or if the right index comes before
;  9366          9458  4          ! the right index.
;  9367          9459  4          !
;  9368          9460  4          IF .LEFT_DONE OR .RIGHT_INDEX LEQ .LEFT_INDEX
;  9369          9461  4          THEN
;  9370          9462  5              BEGIN
;  9371          9463  5              RIGHT_DONE = TRUE;
;  9372          9464  5
;  9373          9465  5              ! Loop through all the successors to RIGHT_TYPE.
;  9374          9466  5              !
;  9375          9467  5              HIER_TBL_ENTRY = .HIER_TBL [.RIGHT_INDEX];
;  9376          9468  5              WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .RIGHT_TYPE DO
;  9377          9469  6                  BEGIN
;  9378          9470  6                  POTENTIAL_DEPTH = FIND_JOIN (
;  9379          9471  6                                      .LEFT_TYPE,
;  9380          9472  6                                      .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE],
;  9381          9473  6                                      LEFT_POTENTIAL_TYPE,
;  9382          9474  6                                      RIGHT_POTENTIAL_TYPE,
;  9383          9475  6                                      POTENTIAL_INDEX,
;  9384          9476  6                                      1+.DEPTH,
;  9385          9477  6                                      .MIN,
;  9386          9478  6                                      .HIER_TBL, .HIER_TBL_SIZE,
;  9387          9479  6                                      .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  9388          9     6                                      .ROUT_TBL, .ROUT_TBL_SIZE);
;  9389          '     6
;  9390          '  2  6
;  9391          9483  6                  ! If we have a new minimum length path, then record
;  9392          9484  6                  ! all the relevant information.
;  9393          9485  6                  !
;  9394          9486  6                  IF .POTENTIAL_DEPTH LSS .MIN
;  9395          9487  6                  THEN
;  9396          9488  7                      BEGIN
;  9397          9489  7                      MIN = .POTENTIAL_DEPTH;
;  9398          9490  7                      .NEW_LEFT_TYPE = .LEFT_POTENTIAL_TYPE;
;  9399          9491  7                      .NEW_RIGHT_TYPE = .RIGHT_POTENTIAL_TYPE;
;  9400          9492  7                      .ROUT_INDEX = .POTENTIAL_INDEX;
;  9401          9493  6                      END;
;  9402          9494  6
;  940*          9495  6
;  9404          9496  6                  ! Set up for next time around loop.
;  9405          9497  6                  !
;  9406          9498  6                  RIGHT_INDEX = .RIGHT_INDEX + 1;
;  9407          9499  6                  HIER_TBL_ENTRY = .HIER_TBL [.RIGHT_INDEX];
;  9408          9500  5                  END;
;  9409          9501  4              END;
;  941u          9502  3          END;
;  9411          9503  3
;  9412          9504  3      ! If we have searched both paths, we are done.
;  9413          9505  3      !
;  9414          9506  4      IF (.RIGHT_DONE OR NOT .RIGHT_FOUND) AND (.LEFT_DONE OR NOT .LEFT_FOUND)
;  9415          9507  3      THEN
;  9416          9508  3          EXITLOOP;
;  9417          9509  2      END;
;  9418          9510  2
```

```
;  9419          9511  2     RETURN .MIN;
;  9420          9512  2
;  9421          9513  1     END;


                                                OFFC 00000 FIND_JOIN:
                                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 9226
                                                5E          0C  C2 00002    SUBL2   #12, SP                             ;
        52              08                      08      04  AC  F0 00005    INSV    LEFT_TYPE, #8, #8, TYPES            ; 9323
                                                5B      08  AC  D0 0000B    MOVL    RIGHT_TYPE, R11                     ; 9324
                                                52          5B  90 0000F    MOVB    R11, TYPES                          ;
                                                50          01  CE 00012    MNEGL   #1, I                               ; 9326
                                                        1B  11  00015       BRB     2$                                  ;
                                                    30 BC40 7F  00017 1$:   PUSHAQ  @ROUT_TBL[I]                        ;
                                                9E          52  B1 0001B    CMPW    TYPES, @(SP)+                       ;
                                                        12  12  0001E       BNEQ    2$                                  ;
                                        0C  BC      04  AC  D0 00020        MOVL    LEFT_TYPE, @NEW_LEFT_TYPE           ; 9329
                                        10  BC          5B  D0 00025        MOVL    R11, @NEW_RIGHT_TYPE                ; 9330
                                        14  BC          50  D0 00029        MOVL    I, @ROUT_INDEX                      ; 9331
                                                50      18  AC  D0 0002D    MOVL    DEPTH, R0                           ; 9332
                                                            04  00031        RET                                         ;
                    E0                  50      34  AC  F2 00032 2$:       AOBLSS  ROUT_TBL_SIZE, I, 1$                ; 9326
                    56          18      AC              01  C1 00037        ADDL3   #1, DEPTH, R6                       ; 9338
                                1C      AC              56  D1 0003C        CMPL    R6, MIN                             ;
                                                        41  18  00040       BGEQ    6$                                  ;
        53              08                      08          5B  F0 00042    INSV    R11, #8, #8, REVERSE_TYPES          ; 9344
                                                53      04  AC  90 00047    MOVB    LEFT_TYPE, REVERSE_TYPES            ; 9345
                                                50 00000000'    EF  9E 0004B    MOVAB   TABLEBASE, R0                       ; 9346
                                                50          28  AC  D1 00052    CMPL    INCOMP_TBL, R0                      ;
                                                        1B  13  00056       BEQL    5$                                  ;
                                                50          01  CE 00058    MNEGL   #1, I                               ; 9355
                                                        11  11  0005B       BRB     4$                                  ;
                                                51          28 BC40 B0  0005D 3$:   MOVW    @INCOMP_TBL[I], INCOMP_TBL_ENTRY    ; 9350
                                                        0F  13  00062       BEQL    5$                                  ; 9351
                                                51          52  B1 00064    CMPW    TYPES, INCOMP_TBL_ENTRY             ; 9352
                                                        1A  13  00067       BEQL    6$                                  ;
                                                51          53  B1 00069    CMPW    REVERSE_TYPES, INCOMP_TBL_ENTRY     ; 9355
                                                        15  13  0006C       BEQL    6$                                  ;
                    EA                  50      2C  AC  F2 0006E 4$:       AOBLSS  INCOMP_TBL_SIZE, I, 3$              ; 9348
                                        55      20  AC  D0 00073 5$:       MOVL    HIER_TBL, R5                        ; 9362
                                                50 00000000'    EF  9E 00077    MOVAB   TABLEBASE, R0                       ;
                                                50          55  D1 0007E    CMPL    R5, R0                              ;
                                                        08  12  00081       BNEQ    7$                                  ;
                                                50 00000000'    EF  D0 00083 6$:   MOVL    MAX_DEPTH, R0                   ; 9364
                                                            04  0008A       RET                                         ;
                                                        59  94  0008B 7$:   CLRB    LEFT_FOUND                          ; 9369
                                                        57  94  0008D       CLRB    RIGHT_FOUND                         ; 9370
                                                50          01  CE 0008F    MNEGL   #1, I                               ; 9371
                                                        2D  11  00092       BRB     11$                                 ;
                                                54              6540 B0  00094 8$:   MOVW    (R5)[I], HIER_TBL_ENTRY             ; 9373
                                                        2C  13  00098       BEQL    12$                                 ; 9374
                                                0E          59  E8 0009A    BLBS    LEFT_FOUND, 9$                      ; 9375
    04  AC              54              08          00  ED 0009D    CMPZV   #0, #8, HIER_TBL_ENTRY, LEFT_TYPE           ; 9377
                                                        06  12  000A3       BNEQ    9$                                  ;
```

```
                59              01  90 000A5          MOVB    #1, LEFT_FOUND                          ; 9380
                53              50  D0 000A8          MOVL    I, LEFT_INDEX                           ; 9381
                0D              57  E8 000AB 9$:      BLBS    RIGHT_FOUND, 10$                        ; 9383
        5B      54              08  00 ED 000AE      CMPZV   #0, #8, HIER_TBL_ENTRY, R11             ; 9385
                                    06 12 000B3      BNEQ    10$                                     ;
                57              01  90 000B5          MOVB    #1, RIGHT_FOUND                         ; 9388
                52              50  D0 000B8          MOVL    I, RIGHT_INDEX                          ; 9389
                03              59  E9 000BB 10$:     BLBC    LEFT_FOUND, 11$                         ; 9391
                05              57  E8 000BE          BLBS    RIGHT_FOUND, 12$                        ;
                CE      50      24  AC F2 000C1 11$:  AOBLSS  HIER_TBL_SIZE, I, 8$                   ; 9371
                                    5A 94 000C6 12$:  CLRB    LEFT_DONE                               ; 9398
                                    58 94 000C8       CLRB    RIGHT_DONE                              ; 9399
                5A              59  E9 000CA 13$:     BLBC    LEFT_FOUND, 18$                         ; 9452
                05              58  E8 000CD 14$:     BLBS    RIGHT_DONE, 15$                         ; 9410
                52              53  D1 000D0          CMPL    LEFT_INDEX, RIGHT_INDEX                 ;
                                    52 14 000D3       BGTR    18$                                     ;
                5A              01  90 000D5 15$:     MOVB    #1, LEFT_DONE                           ; 9413
                54          6543    B0 000D8 16$:     MOVW    (R5)[LEFT_INDEX], HIER_TBL_ENTRY        ; 9417
    04  AC      54              08  00 ED 000DC      CMPZV   #0, #8, HIER_TBL_ENTRY, LEFT_TYPE       ; 9418
                                    43 12 000E2      BNEQ    18$                                     ;
                7E      30      AC  7D 000E4          MOVQ    ROUT_TBL, -(SP)                         ; 9430
                7E      28      AC  7D 000E8          MOVQ    INCOMP_TBL, -(SP)                       ; 9429
                        24      AC  DD 000EC          PUSHL   HIER_TBL_SIZE                           ; 9428
                                55  DD 000EF          PUSHL   R5                                      ;
                        1C      AC  DD 000F1          PUSHL   MIN                                     ; 9427
                                56  DD 000F4          PUSHL   R6                                      ; 9426
                        20      AE  9F 000F6          PUSHAB  POTENTIAL_INDEX                         ; 9420
                        28      AE  9F 000F9          PUSHAB  RIGHT_POTENTIAL_TYPE                    ;
                        30      AE  9F 000FC          PUSHAB  LEFT_POTENTIAL_TYPE                     ;
                                5B  DD 000FF          PUSHL   R11                                     ; 9422
        7E      54              08  08 EF 00101        EXTZV   #8, #8, HIER_TBL_ENTRY, -(SP)           ; 9421
                    FEF5        CF  0D FB 00106       CALLS   #13, FIND_JOIN                          ;
                    1C      AC      50 D1 0010B       CMPL    POTENTIAL_DEPTH, MIN                    ; 9435
                                    12 18 0010F       BGEQ    17$                                     ;
                    1C      AC      50 D0 00111       MOVL    POTENTIAL_DEPTH, MIN                    ; 9438
                    0C      BC  08  AE D0 00115       MOVL    LEFT_POTENTIAL_TYPE, @NEW_LEFT_TYPE     ; 9439
                    10      BC  04  AE D0 0011A       MOVL    RIGHT_POTENTIAL_TYPE, @NEW_RIGHT_TYPE   ; 9440
                    14      BC      6E D0 0011F       MOVL    POTENTIAL_INDEX, @ROUT_INDEX            ; 9441
                                    53 D6 00123 17$:  INCL    LEFT_INDEX                              ; 9446
                                    B1 11 00125       BRB     16$                                     ; 9447
                5A              57  E9 00127 18$:     BLBC    RIGHT_FOUND, 22$                        ; 9452
                05              5A  E8 0012A          BLBS    LEFT_DONE, 19$                          ; 9460
                53              52  D1 0012D          CMPL    RIGHT_INDEX, LEFT_INDEX                 ;
                                    52 14 00130       BGTR    22$                                     ;
                5B              01  90 00132 19$:     MOVB    #1, RIGHT_DONE                          ; 9463
                54          6542    B0 00135 20$:     MOVW    (R5)[RIGHT_INDEX], HIER_TBL_ENTRY       ; 9467
    5B  54              08          00 ED 00139       CMPZV   #0, #8, HIER_TBL_ENTRY, R11             ; 9468
                                    44 12 0013E       BNEQ    22$                                     ;
                7E      30      AC  7D 00140          MOVQ    ROUT_TBL, -(SP)                         ; 9480
                7E      28      AC  7D 00144          MOVQ    INCOMP_TBL, -(SP)                       ; 9479
                        24      AC  DD 00148          PUSHL   HIER_TBL_SIZE                           ; 9478
                                55  DD 0014B          PUSHL   R5                                      ;
                        1C      AC  DD 0014D          PUSHL   MIN                                     ; 9477
                                56  DD 00150          PUSHL   R6                                      ; 9476
                        20      AE  9F 00152          PUSHAB  POTENTIAL_INDEX                         ; 9470
                        28      AE  9F 00155          PUSHAB  RIGHT_POTENTIAL_TYPE                    ;
                        30      AE  9F 00158          PUSHAB  LEFT_POTENTIAL_TYPE                     ;
```

```
                7E      54          08          08  EF 0015B          EXTZV   #8, #8, HIER_TBL_ENTRY, -(SP)          ; 9472
                                        04      AC  DD 00160          PUSHL   LEFT_TYPE                              ; 9471
                            FE98        CF          0D  FB 00163      CALLS   #13, FIND_JOIN                         ;
                              1C        AC          50  D1 00168      CMPL    POTENTIAL_DEPTH, MIN                   ; 9486
                                                    12  18 0016C      BGEQ    21$                                    ;
                              1C        AC          50  D0 0016E      MOVL    POTENTIAL_DEPTH, MIN                   ; 9489
                              0C        BC      08  AE  D0 00172      MOVL    LEFT_POTENTIAL_TYPE, @NEW_LEFT_TYPE    ; 9490
                              10        BC      04  AE  D0 00177      MOVL    RIGHT_POTENTIAL_TYPE, @NEW_RIGHT_TYPE  ; 9491
                              14        BC          6E  D0 0017C      MOVL    POTENTIAL_INDEX, @ROUT_INDEX           ; 9492
                                                    52  D6 00180 21$:  INCL    RIGHT_INDEX                            ; 9498
                                                    B1  11 00182      BRB     20$                                    ; 9499
                                        06          58  E8 00184 22$:  BLBS    RIGHT_DONE, 23$                        ; 9506
                                        03          57  E9 00187      BLBC    RIGHT_FOUND, 23$                       ;
                                            FF3D    31  0018A         BRW     13$                                    ;
                                        06          5A  E8 0018D 23$:  BLBS    LEFT_DONE, 24$                         ;
                                        03          59  E9 00190      BLBC    LEFT_FOUND, 24$                        ;
                                            FF37    31  00193         BRW     14$                                    ;
                                        50      1C  AC  D0 00196 24$:  MOVL    MIN, R0                                ; 9511
                                                    04  0019A         RET                                            ; 9513

; Routine Size:  411 bytes,    Routine Base:  DBG$CODE + 20A9
```

```
;  9423          9514  1 ROUTINE FIND_PATH (OLD_TYPE, NEW_TYPE, ROUT_INDEX,
;  9424          9515  1                   HIER_TBL, HIER_TBL_SIZE,
;  9425          9516  1                   INCOMP_TBL, INCOMP_TBL_SIZE,
;  9426          9517  1                   ROUT_TBL, ROUT_TBL_SIZE) =
;  9427          9518  1 !
;  9428          9519  1 ! FUNCTION
;  9429          9520  1 !       This routine handles implicit type conversions on unary operators.
;  9430          9521  1 !       It determines whether the given OLD_TYPE can be converted to
;  9431          9522  1 !       a type that is legal for the current operator. The legal types
;  9432          9523  1 !       can be determined using ORT_TBL. HIER_TBL gives the legal conversion
;  9433          9524  1 !       paths. Thus, what this routine is doing is finding a path in
;  9434          9525  1 !       HIER_TBL from the OLD_TYPE to a type that is legal for the
;  9435          9526  1 !       current operator.
;  9436          9527  1 !
;  9437          9528  1 !       The value returned by the routine is the length of the shortest
;  9438          9529  1 !       path to a legal type. If there are no paths, the value MAX_DEPTH
;  9439          9530  1 !       is returned.
;  9440          9531  1 !
;  9441          9532  1 !       The search for a path uses the following recursive method:
;  9442          9533  1 !       1) If the input type is legal, return 0
;  9443          9534  1 !       2) If there are no edges out of the type, return MAX_DEPTH.
;  9444          9535  1 !       3) For each edge out of the type, do a recursive call to obtain
;  9445          9536  1 !          MIN(MAX_DEPTH,1+FIND_PATH(edge)). Return the minimum of these.
;  9446          9537  1 !
;  9447          9538  1 !       *** Note - This routine may also require additional input to
;  9448          9539  1 !           disambiguate cases where there are two or more paths of
;  9449          9540  1 !           equal length.
;  9450          9541  1 !
;  9451          9542  1 ! INPUTS
;  9452          9543  1 !       HIER_TBL        - Hier. Table
;  9453          9544  1 !       HIER_TBL_SIZE   - Hier. Table Size
;  9454          9545  1 !       INCOMP_TBL      - Incomp. Table
;  9455          9546  1 !       INCOMP_TBL_SIZE - Incomp. Table Size
;  9456          9547  1 !       ROUT_TBL        - Rout. Table
;  9457          9548  1 !       ROUT_TBL_SIZE   - Rout. Table Size
;  9458          9549  1 !       OLD_TYPE        - The type of the operand (A VAX standard type code.)
;  9459          9550  1 !       NEW_TYPE        - The address in which to leave the result type.
;  9460          9551  1 !       ROUT_INDEX      - The address in which to leave an index into the
;  9461          9552  1 !                             Operator Routine Table, pointing to a routine
;  9462          9553  1 !                             that handles the new type.
;  9463          9554  1 !
;  9464          9555  1 ! OUTPUTS
;  9465          9556  1 !       NEW_TYPE        - This output parameter is filled in with the type
;  9466          9557  1 !                             to convert to.
;  9467          9558  1 !       ROUT_INDEX      - Filled in with an index into the Operator Routine
;  9468          9559  1 !                             Table, pointing to a routine that handles the
;  9469          9560  1 !                             new type.
;  9470          9561  1 !
;  9471          9562  1 ! ROUTINE VALUE
;  9472          9563  1 !       The length of the shortest path is returned. If this is equal to
;  9473          9564  1 !       MAX_DEPTH then no path was found.
;  9474          9565  1 !
;  9475          9566  1
;  9476          9567  2     BEGIN
;  9477          9568  2
;  9478          9569  2     MAP
;  9479          9570  2         HIER_TBL: REF VECTOR [,WORD],   ! Pointer to a Type Hierarchy Table
```

```
;  9480          9571  2          INCOMP_TBL: REF VECTOR [,WORD], . Pointer to a Type Incompatibility Table
;  9481          9572  2          ROUT_TBL: REF ORT$TABLE;         ! Pointer to an Operator Routine Table
;  9482          9573  2
;  9483          9574  2      LOCAL
;  9484          9575  2          FOUND,                                   ! A flag saying whether
;  9485          9576  2                                                   !    we have found a new type
;  9486          9577  2          HIER_TBL_ENTRY: TYPE_GRAPH$ENTRY,        ! An entry in the Type Hierarchy Table
;  9487          9578  2          II,                                      ! A loop counter
;  9488          9579  2          MIN,                                     ! The minimum depth so far
;  9489          9580  2          POTENTIAL_DEPTH,                         ! A candidate for shortest depth
;  9490          9581  2          POTENTIAL_ROUT_INDEX,                    ! A candidate for routine index
;  9491          9582  2          POTENTIAL_TYPE;                          ! A candidate for the new type
;  9492          9583  2
;  9493          9584  2
;  9494          9585  2      ! First see if the given type is legal, by searching the Operator
;  9495          9586  2      ! Routine Table. If it is, then the length of the path to a legal
;  9496          9587  2      ! type is zero and this is what we return.
;  9497          9588  2      !
;  9498          9589  2      INCR I FROM 0 TO .ROUT_TBL_SIZE - 1 DO
;  9499          9590  2          IF .OLD_TYPE EQL .ROUT_TBL [.I, ORT$B_LEFT_TYPE]
;  9500          9591  2          THEN
;  9501          9592  3              BEGIN
;  9502          9593  3              .NEW_TYPE = .OLD_TYPE;
;  9503          9594  3              .ROUT_INDEX = .I;
;  9504          9595  3              RETURN 0;
;  9505          9596  2              END;
;  9506          9597  2
;  9507          9598  2      ! Check for no Hierarchy Table being present.
;  9508          9599  2      !
;  9509          9600  2      IF .HIER_TBL EQL TABLEBASE
;  9510          9601  2      THEN
;  9511          9602  2          RETURN .MAX_DEPTH;
;  9512          9603  2
;  9513          9604  2      ! Find the first edge emanating from OLD_TYPE in the Type Hierarchy Table.
;  9514          9605  2      !
;  9515          9606  2      FOUND = FALSE;
;  9516          9607  2      INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
;  9517          9608  3          BEGIN
;  9518          9609  3          II = .I;
;  9519          9610  3          HIER_TBL_ENTRY = .HIER_TBL [.I];
;  9520          9611  3          IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
;  9521          9612  3          IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE
;  9522          9613  3          THEN
;  9523          9614  4              BEGIN
;  9524          9615  4              FOUND = TRUE;
;  9525          9616  4              EXITLOOP;
;  9526          9617  3              END;
;  9527          9618  2          END;
;  9528          9619  2
;  9529          9620  2      ! If there were no edges emanating from OLD_TYPE, then no type conversion
;  9530          9621  2      ! can be done. Indicate this by returning MAX_DEPTH.
;  9531          9622  2      !
;  9532          9623  2      IF NOT .FOUND THEN RETURN .MAX_DEPTH;
;  9533          9624  2
;  9534          9625  2      ! For each edge out of OLD_TYPE, chase down the target of that edge.
;  9535          9626  2      ! Do a recursive call on that target to find a path from the target
;  9536          9627  2      ! to a legal type. Keep track of the minimum length path in the
```

```
;  9537          9628  2      ! variable MIN.
;  9538          9629  2      !
;  9539          9630  2      MIN = .MAX_DEPTH;
;  9540          9631  2      WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE DO
;  9541          9632  3          BEGIN
;  9542          9633  3
;  9543          9634  3          POTENTIAL_DEPTH = FIND_PATH (
;  9544          9635  3                  .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE], POTENTIAL_TYPE,
;  9545          9636  3                  POTENTIAL_ROUT_INDEX,
;  9546          9637  3                  .HIER_TBL, .HIER_TBL_SIZE,
;  9547          9638  3                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
;  9548          9639  3                  .ROUT_TBL, .ROUT_TBL_SIZE);
;  9549          9640  3
;  9550          9641  3          IF .POTENTIAL_DEPTH LSS .MIN
;  9551          9642  3          THEN
;  9552          9643  4              BEGIN
;  9553          9644  4              MIN = .POTENTIAL_DEPTH;
;  9554          9645  4              .NEW_TYPE = .POTENTIAL_TYPE;
;  9555          9646  4              .ROUT_INDEX = .POTENTIAL_ROUT_INDEX;
;  9556          9647  3              END;
;  9557          9648  3
;  9558          9649  3          ! Set up for next time around loop.
;  9559          9650  3          II = .II + 1;
;  9560          9651  3          HIER_TBL_ENTRY = .HIER_TBL [.II];
;  9561          9652  2          END;
;  9562          9653  2
;  9563          9654  2      IF .MIN EQL .MAX_DEPTH
;  9564          9655  2      THEN
;  9565          9656  2          RETURN .MAX_DEPTH
;  9566          9657  2      ELSE
;  9567          9658  2          RETURN .MIN + 1;
;  9568          9659  2
;  9569          9660  1      END; ! FIND_PATH


                                                003C 00000 FIND_PATH:
                                                           .WORD   Save R2,R3,R4,R5                    ;  9514
                                    55 00000000'   EF  9E 00002    MOVAB   MAX_DEPTH, R5               ;
                                    5E             08  C2 00009    SUBL2   #8, SP                      ;
                                    50             01  CE 0000C    MNEGL   #1, I                       ;  9590
                                                   18  11 0000F    BRB     2$                          ;
                                          20 BC40  7F  00011 1$:    PUSHAQ  @ROUT_TBL[I]                ;
04    AC              9E              08           08  ED 00015    CMPZV   #8, #8, @(SP)+, OLD_TYPE    ;
                                                   0C  12 0001B    BNEQ    2$                          ;
                                08  BC       04    AC  D0 0001D    MOVL    OLD_TYPE, @NEW_TYPE         ;  9593
                                0C  BC             50  D0 00022    MOVL    I, @ROUT_INDEX              ;  9594
                                              008D 31  00026    BRW     13$                         ;  9595
                                E3             50      24    AC  F2 00029 2$:  AOBLSS  ROUT_TBL_SIZE, I, 1$ ;  9590
                                               50 00000000'   EF  9E 0002E    MOVAB   TABLEBASE, R0       ;  9600
                                               50      10     AC  D1 00035    CMPL    HIER_TBL, R0        ;
                                                              26  13 00039    BEQL    6$                  ;
                                                              51  D4 0003B    CLRL    FOUND               ;  9606
                                               50             01  CE 0003D    MNEGL   #1, I               ;  9607
                                                              17  11 00040    BRB     4$                  ;
```

```
                                53           50   D0 00042 3$:     MOVL    I, II                                   ; 9609
                                54     10 BC40    B0 00045         MOVW    @HIER_TBL[I], HIER_TBL_ENTRY            ; 9610
                                           12    13 0004A         BEQL    5$                                      ; 9611
        04   AC          54             08       00   ED 0004C    CMPZV   #0, #8, HIER_TBL_ENTRY, OLD_TYPE        ; 9612
                                           05    12 00052         BNEQ    4$                                      ;
                                51             01   D0 00054    MOVL    #1, FOUND                               ; 9615
                                           05    11 00057         BRB     5$                                      ; 9614
                                E4             50   14   AC F2 00059 4$:  AOBLSS  HIER_TBL_SIZE, I, 3$           ; 9607
                                04             51   E8 0005E 5$:  BLBS    FOUND, 7$                               ; 9623
                                50             65   D0 00061 6$:  MOVL    MAX_DEPTH, R0                           ;
                                               04   04 00064     RET                                             ;
                                52             65   D0 00065 7$:  MOVL    MAX_DEPTH, MIN                          ; 9630
        04   AC          54             08       00   ED 00068 8$:  CMPZV   #0, #8, HIER_TBL_ENTRY, OLD_TYPE      ; 9631
                                           36    12 0006E         BNEQ    10$                                     ;
                                7E     20   AC 7D 00070         MOVQ    ROUT_TBL, -(SP)                         ; 9639
                                7E     18   AC 7D 00074         MOVQ    INCOMP_TBL, -(SP)                       ; 9638
                                7E     10   AC 7D 00078         MOVQ    HIER_TBL, -(SP)                         ; 9637
                                       18   AE 9F 0007C         PUSHAB  POTENTIAL_ROUT_INDEX                    ; 9634
                                       20   AE 9F 0007F         PUSHAB  POTENTIAL_TYPE                          ;
        7E   54                        08       08   EF 00082    EXTZV   #8, #8, HIER_TBL_ENTRY, -(SP)           ; 9635
                                FF74   CF       09   FB 00087    CALLS   #9, FIND_PATH                           ;
                                52             50   D1 0008C    CMPL    POTENTIAL_DEPTH, MIN                    ; 9641
                                           0C    18 0008F         BGEQ    9$                                      ;
                                52             50   D0 00091    MOVL    POTENTIAL_DEPTH, MIN                    ; 9644
                          08    BC       04   AE D0 00094         MOVL    POTENTIAL_TYPE, @NEW_TYPE               ; 9645
                          0C    BC             6E   D0 00099    MOVL    POTENTIAL_ROUT_INDEX, @ROUT_INDEX       ; 9646
                                               53   D6 0009D 9$:  INCL    II                                      ; 9650
                                54     10 BC43    B0 0009F         MOVW    @HIER_TBL[II], HIER_TBL_ENTRY           ; 9651
                                           C2    11 000A4         BRB     8$                                      ; 9631
                                65             52   D1 000A6 10$: CMPL    MIN, MAX_DEPTH                          ; 9654
                                           05    12 000A9         BNEQ    11$                                     ;
                                52             65   D0 000AB    MOVL    MAX_DEPTH, R2                           ; 9658
                                           02    11 000AE         BRB     12$                                     ;
                                               52   D6 000B0 11$: INCL    R2                                      ;
                                50             52   D0 000B2 12$: MOVL    R2, R0                                  ;
                                               04   04 000B5     RET                                             ;
                                               50   D4 000B6 13$: CLRL    R0                                      ; 9660
                                               04   04 000B8     RET                                             ;

; Routine Size:  185 bytes,    Routine Base:  DBG$CODE + 2244
```

```
;  9571          9661  1 ROUTINE FIND_PATH_DEPOSIT (OLD_TYPE, NEW_TYPE, DEPTH,
;  9572          9662  1                  HIER_TBL, HIER_TBL_SIZE,
;  9573          9663  1                  INCOMP_TBL, INCOMP_TBL_SIZE,
;  9574          9664  1                  ROUT_TBL, ROUT_TBL_SIZE) =
;  9575          9665  1 !
;  9576          9666  1 ! FUNCTION
;  9577          9667  1 !       This routine handles explicit type conversions on unary operators.
;  9578          9668  1 !       It is used to determine whether the type given in OLD_TYPE
;  9579          9669  1 !       can be converted to the type given in NEW_TYPE. The main
;  9580          9670  1 !       application of this is when the user is doing a DEPOSIT of a
;  9581          9671  1 !       value of type OLD_TYPE into a primary of type NEW_TYPE.
;  9582          9672  1 !       The routine may also get used to implement the "cast" operator
;  9583          9673  1 !       or to implement type conversions of subscript values.
;  9584          9674  1 !
;  9585          9675  1 !       This routine uses HIER_TBL and attempts to find a path from
;  9586          9676  1 !       OLD_TYPE to NEW_TYPE. The value TRUE is returned if a path
;  9587          9677  1 !       is found and FALSE is returned if a path is not found.
;  9588          9678  1 !
;  9589          9679  1 !       The search for a path uses the following recursive method:
;  9590          9680  1 !       1) If the two types are the same, return TRUE
;  9591          9681  1 !       2) If there are no edges out of OLD_TYPE, return FALSE.
;  9592          9682  1 !       3) For each edge out of OLD_TYPE, do a recursive call. If
;  9593          9683  1 !          any of these return TRUE, then return TRUE.
;  9594          9684  1 !
;  9595          9685  1 !
;  9596          9686  1 ! INPUTS
;  9597          9687  1 !       HIER_TBL        - Hier. Table
;  9598          9688  1 !       HIER_TBL_SIZE   - Hier. Table Size
;  9599          9689  1 !       INCOMP_TBL      - Incomp. Table
;  9600          9690  1 !       INCOMP_TBL_SIZE - Incomp. Table Size
;  9601          9691  1 !       ROUT_TBL        - Rout. Table
;  9602          9692  1 !       ROUT_TBL_SIZE   - Rout. Table Size
;  9603          9693  1 !       OLD_TYPE        - The type of the source operand (A VAX standard type code.)
;  9604          9694  1 !       NEW_TYPE        - The target type (A VAX standard type code).
;  9605          9695  1 !       DEPTH           - The current recursion depth
;  9606          9696  1 !
;  9607          9697  1 ! OUTPUTS
;  9608          9698  1 !       The routine value is one of:
;  9609          9699  1 !       TRUE    - The source can be converted to the target type
;  9610          9700  1 !       FALSE   - The source cannot be converted to the target type.
;  9611          9701  1 !
;  9612          9702  1
;  9613          9703  2     BEGIN
;  9614          9704  2
;  9615          9705  2     MAP
;  9616          9706  2         HIER_TBL: REF VECTOR [,WORD],   ! Pointer to a Type Hierarchy Table
;  9617          9707  2         INCOMP_TBL: REF VECTOR [,WORD], ! Pointer to a Type Incompatibility Table
;  9618          9708  2         ROUT_TBL: REF ORT$TABLE;        ! Pointer to an Operator Routine Table
;  9619          9709  2
;  9620          9710  2     LOCAL
;  9621          9711  2         FOUND,                                  ! A flag saying whether
;  9622          9712  2                                                 !    we have found a new type
;  9623          9713  2         HIER_TBL_ENTRY: TYPE_GRAPH$ENTRY,       ! An entry in the Type Hierarchy Table
;  9624          9714  2         II;                                     ! A loop counter
;  9625          9715  2
;  9626          9716  2
;  9627          9717  2     ! First see if the source and target are already of the same type.
```

```
; 9628          9718  2      !
; 9629          9719  2      IF .NEW_TYPE EQL .OLD_TYPE
; 9630          9720  2      THEN
; 9631          9721  2          RETURN TRUE;
; 9632          9722  2
; 9633          9723  2      ! Then check whether we have already surpassed the maximum depth
; 9634          9724  2      ! to which we have to search.
; 9635          9725  2      !
; 9636          9726  2      IF .DEPTH GTR .MAX_DEPTH
; 9637          9727  2      THEN
; 9638          9728  2          RETURN FALSE;
; 9639          9729  2
; 9640          9730  2      ! Check for no Hierarchy Table being present.
; 9641          9731  2      !
; 9642          9732  2      IF .HIER_TBL EQL TABLEBASE
; 9643          9733  2      THEN
; 9644          9734  2          RETURN FALSE;
; 9645          9735  2
; 9646          9736  2      ! Find the first edge emanating from OLD_TYPE in the Type Hierarchy Table.
; 9647          9737  2      !
; 9648          9738  2      FOUND = FALSE;
; 9649          9739  2      INCR I FROM 0 TO .HIER_TBL_SIZE - 1 DO
; 9650          9740  3          BEGIN
; 9651          9741  3          II = .I;
; 9652          9742  3          HIER_TBL_ENTRY = .HIER_TBL [.I];
; 9653          9743  3          IF .HIER_TBL_ENTRY EQL 0 THEN EXITLOOP;
; 9654          9744  3          IF .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE
; 9655          9745  3          THEN
; 9656          9746  4              BEGIN
; 9657          9747  4              FOUND = TRUE;
; 9658          9748  4              EXITLOOP;
; 9659          9749  3              END;
; 9660          9750  2          END;
; 9661          9751  2
; 9662          9752  2      ! If there were no edges emanating from OLD_TYPE, then no type conversion
; 9663          9753  2      ! can be done. Indicate this by returning FALSE.
; 9664          9754  2      !
; 9665          9755  2      IF NOT .FOUND
; 9666          9756  2      THEN
; 9667          9757  2          RETURN FALSE;
; 9668          9758  2
; 9669          9759  2      ! For each edge out of OLD_TYPE, chase down the target of that edge.
; 9670          9760  2      ! Do a recursive call on that target to find out whether there is
; 9671          9761  2      ! a path from the edge.
; 9672          9762  2      !
; 9673          9763  2      WHILE .HIER_TBL_ENTRY [TYPE_GRAPH$B_LOWER_TYPE] EQL .OLD_TYPE DO
; 9674          9764  3          BEGIN
; 9675          9765  3
; 9676          9766  3          IF FIND_PATH_DEPOSIT (
; 9677          9767  3                  .HIER_TBL_ENTRY [TYPE_GRAPH$B_HIGHER_TYPE], .NEW_TYPE,
; 9678          9768  3                  .DEPTH+1,
; 9679          9769  3                  .HIER_TBL, .HIER_TBL_SIZE,
; 9680          9770  3                  .INCOMP_TBL, .INCOMP_TBL_SIZE,
; 9681          9771  3                  .ROUT_TBL, .ROUT_TBL_SIZE)
; 9682          9772  3          THEN
; 9683          9773  3              RETURN TRUE;
; 9684          9774  3
```

```
;  9685          9775  3
;  9686          9776  3          ! Set up for next time around loop.
;  9687          9777  3          II = .II + 1;
;  9688          9778  3          HIER_TBL_ENTRY = .HIER_TBL [.II];
;  9689          9779  2          END;
;  9690          9780  2
;  9691          9781  2      ! If we didn't return true from within the loop, then we failed to
;  9692          9782  2      ! find a path. Thus, return false here.
;  9693          9783  2      !
;  9694          9784  2      RETURN FALSE;
;  9695          9785  2
;  9696          9786  1      END; ! FIND_PATH_DEPOSIT


                                                001C 00000 FIND_PATH_DEPOSIT:
                                                           .WORD   Save R2,R3,R4                                   ; 9661
      04   AC        08    AC  D1 00002                     CMPL    NEW_TYPE, OLD_TYPE                              ; 9719
                           67  13 00007                     BEQL    5$                                              ;
00000000'  EF        0C    AC  D1 00009                     CMPL    DEPTH, MAX_DEPTH                                ; 9726
                           6A  14 00011                     BGTR    7$                                              ;
           50 00000000'    EF  9E 00013                     MOVAB   TABLEBASE, R0                                   ; 9732
           50        10    AC  D1 0001A                     CMPL    HIER_TBL, R0                                    ;
                           5D  13 0001E                     BEQL    7$                                              ;
                           51  D4 00020                     CLRL    FOUND                                           ; 9738
           50              01  CE 00022                     MNEGL   #1, I                                           ; 9739
                           17  11 00025                     BRB     2$                                              ;
           52              50  D0 00027 1$:                MOVL    I, II                                           ; 9741
           53        10 BC40   B0 0002A                     MOVW    @HIER_TBL[I], HIER_TBL_ENTRY                    ; 9742
                           12  13 0002F                     BEQL    3$                                              ; 9743
04   AC        53        08    00  ED 00031                CMPZV   #0, #8, HIER_TBL_ENTRY, OLD_TYPE                ; 9744
                           05  12 00037                     BNEQ    2$                                              ;
           51              01  D0 00039                     MOVL    #1, FOUND                                       ; 9747
                           05  11 0003C                     BRB     3$                                              ; 9746
      E4   50        14    AC  F2 0003E 2$:                AOBLSS  HIER_TBL_SIZE, I, 1$                            ; 9739
      37   51              E9  00043 3$:                   BLBC    FOUND, 7$                                       ; 9755
      54   0C              AC  01  C1 00046                ADDL3   #1, DEPTH, R4                                   ; 9768
04   AC        53        08    00  ED 0004B 4$:           CMPZV   #0, #8, HIER_TBL_ENTRY, OLD_TYPE                ; 9763
                           2A  12 00051                     BNEQ    7$                                              ;
      7E   20   AC            7D  00053                     MOVQ    ROUT_TBL, -(SP)                                 ; 9771
      7E   18   AC            7D  00057                     MOVQ    INCOMP_TBL, -(SP)                               ; 9770
      7E   10   AC            7D  0005B                     MOVQ    HIER_TBL, -(SP)                                 ; 9769
                           54  DD  0005F                     PUSHL   R4                                              ; 9768
                      08   AC  DD  00061                     PUSHL   NEW_TYPE                                        ; 9767
   7E        53        08        08  EF 00064                EXTZV   #8, #8, HIER_TBL_ENTRY, -(SP)                   ;
         93  AF              09  FB 00069                     CALLS   #9, FIND_PATH_DEPOSIT                           ;
             04              50  E9 0006D                     BLBC    R0, 6$                                          ;
             50              01  D0 00070 5$:                MOVL    #1, R0                                          ; 9773
                                 04  00073                   RET                                                     ;
                           52    D6  00074 6$:                INCL    II                                              ; 9777
           53        10 BC42   B0  00076                     MOVW    @HIER_TBL[II], HIER_TBL_ENTRY                   ; 9778
                           CE  11  0007B                     BRB     4$                                              ; 9763
                           50  D4  0007D 7$:                CLRL    R0                                              ; 9786
                               04  0007F                   RET                                                     ;
```

```
; Routine Size:  128 bytes,    Routine Base:  DBG$CODE + 22FD

; 9697          9787  1
```

```
;  9699          9788  1 ROUTINE FIXUP_EMPTY_SET(EMPTY_SET, VAL_DESC) =
;  9700          9789  1 !
;  9701          9790  1 ! FUNCTION
;  9702          9791  1 !       This routine sets up the typeid in empty set value descriptor
;  9703          9792  1 !       either by taking it from val_desc or creating a new one by
;  9704          9793  1 !       calling DBG$TYPEID_FOR_SET.
;  9705          9794  1 !
;  9706          9795  1 ! INPUTS
;  9707          9796  1 !       EMPTY_SET       - [] value descriptor.
;  9708          9797  1 !
;  9709          9798  1 !       VAL_DESC        - Value descriptor.  This may be 0 for unary operation,
;  9710          9799  1 !                         This may be another [], or standard value descriptor.
;  9711          9800  1 !
;  9712          9801  1 ! OUTPUTS
;  9713          9802  1 !       Fixed Empty Set Value Descriptor is returned.
;  9714          9803  1 !
;  9715          9804  1
;  9716          9805  2     BEGIN
;  9717          9806  2
;  9718          9807  2     MAP
;  9719          9808  2         EMPTY_SET: REF DBG$VALDESC,
;  9720          9809  2         VAL_DESC: REF DBG$VALDESC;
;  9721          9810  2
;  9722          9811  2
;  9723          9812  2     EMPTY_SET[DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
;  9724          9813  2     EMPTY_SET[DBG$B_VALUE_CLASS] = 0;
;  9725          9814  2     EMPTY_SET[DBG$B_VALUE_DTYPE] = 0;
;  9726          9815  2     IF .VAL_DESC EQL 0 OR
;  9727          9816  2        .VAL_DESC[DBG$B_DHDR_FCODE] NEQ RST$K_TYPE_SET OR
;  9728          9817  3        (.VAL_DESC[DBG$B_DHDR_LANG] EQL %X'FF' AND
;  9729          9818  3         .VAL_DESC[DBG$B_VALUE_CLASS] EQL %X'FF' AND
;  9730          9819  3         .VAL_DESC[DBG$B_VALUE_DTYPE] EQL %X'FF')
;  9731          9820  2     THEN
;  9732          9821  2         EMPTY_SET[DBG$L_DHDR_TYPEID] = DBG$TYPEID_FOR_SET(
;  9733          9822  2             DSC$K_DTYPE_L, RST$K_TYPE_SET, 256, TRUE)
;  9734          9823  2
;  9735          9824  2     ELSE
;  9736          9825  3         BEGIN
;  9737          9826  3         EMPTY_SET[DBG$L_DHDR_TYPEID] = .VAL_DESC[DBG$L_DHDR_TYPEID];
;  9738          9827  3         EMPTY_SET[DBG$W_VALUE_LENGTH] = .VAL_DESC[DBG$W_VALUE_LENGTH];
;  9739          9828  2         END;
;  9740          9829  2
;  9741          9830  2     RETURN .EMPTY_SET;
;  9742          9831  1     END;


                                000C 00000 FIXUP_EMPTY_SET:
                                           .WORD   Save R2,R3                              ; 9788
                    53      04   AC  D0 00002   MOVL   EMPTY_SET, R3                      ; 9812
              03    A3 00000000G 00  90 00006   MOVB   DBG$GB_LANGUAGE, 3(R3)
                               16  A3  B4 0000E   CLRW   22(R3)                           ; 9814
                    52      08   AC  D0 00011   MOVL   VAL_DESC, R2                      ; 9815
                               1B  13 00015   BEQL   1$                                  ;
                    08      06   A2  91 00017   CMPB   6(R2), #8                         ; 9816
```

```
                                 15  12 0001B         BNEQ    1$                              ;
         FF    8F        03      A2  91 0001D         CMPB    3(R2), #255                     ; 9817
                                 26  12 00022         BNEQ    2$                              ;
         FF    8F        17      A2  91 00024         CMPB    23(R2), #255                    ; 9818
                                 1F  12 00029         BNEQ    2$                              ;
         FF    8F        16      A2  91 0002B         CMPB    22(R2), #255                    ; 9819
                                 18  12 00030         BNEQ    2$                              ;
                                 01  DD 00032 1$:     PUSHL   #1                              ; 9821
               7E      0100      8F  3C 00034         MOVZWL  #256, -(SP)                     ;
                                 08  DD 00039         PUSHL   #8                              ;
                                 08  DD 0003B         PUSHL   #8                              ;
  00000000G    00                04  FB 0003D         CALLS   #4, DBG$TYPEID_FOR_SET          ;
         08    A3                50  D0 00044         MOVL    R0, 8(R3)                       ;
                                 0A  11 00048         BRB     3$                              ;
         08    A3        08      A2  D0 0004A 2$:     MOVL    8(R2), 8(R3)                    ; 9826
         14    A3        14      A2  B0 0004F         MOVW    20(R2), 20(R3)                  ; 9827
               50                53  D0 00054 3$:     MOVL    R3, R0                          ; 9830
                                     04 00057         RET                                     ; 9831

; Routine Size:  88 bytes,    Routine Base:  DBG$CODE + 237D


; 9743          9832  1
```

```
; 9745     9833  1 ROUTINE INTMED_DATA_FOR_DEP(FROM_DESC, TO_DATA_TYPE, FLAG) =
; 9746     9834  1 !
; 9747     9835  1 ! FUNCTION
; 9748     9836  1 !       There are times an intermediate data is needed before the deposit.
; 9749     9837  1 !       This routine takes the from descriptor converts to intermediate
; 9750     9838  1 !       data, return the intermediate data back to the caller.  The caller
; 9751     9839  1 !       takes intermediate data then performs the deposit into TO_DESC.
; 9752     9840  1 !
; 9753     9841  1 ! INPUTS
; 9754     9842  1 !       FROM_DESC       - From value descriptor.
; 9755     9843  1 !
; 9756     9844  1 !       TO_DATA_TYPE    - Intermediate data type.
; 9757     9845  1 !
; 9758     9846  1 !       FLAG            - Flag set to TRUE to indidate left hand side
; 9759     9847  1 !                         of the deposit.
; 9760     9848  1 !
; 9761     9849  1 ! OUTPUTS
; 9762     9850  1 !       Intermediate data type is returned.
; 9763     9851  1 !
; 9764     9852  1
; 9765     9853  2     BEGIN
; 9766     9854  2
; 9767     9855  2     MAP
; 9768     9856  2         FROM_DESC: REF DBG$VALDESC;     ! Pointer to value descriptor.
; 9769     9857  2
; 9770     9858  2     LOCAL
; 9771     9859  2         DIGITS,                         ! The number of digits.
; 9772     9860  2         INTMED_DESC: REF DBG$VALDESC;   ! Intermediate data value descriptor.
; 9773     9861  2
; 9774     9862  2
; 9775     9863  2     CASE .FROM_DESC[DBG$B_VALUE_DTYPE] FROM 0 TO DBG$K_MAXIMUM_DTYPE +
; 9776     9864  2                                                 RST$K_TYPE_MAXIMUM    OF
; 9777     9865  2         SET
; 9778     9866  2
; 9779     9867  2
; 9780     9868  2         [DSC$K_DTYPE_NL, DSC$K_DTYPE_NLO, DSC$K_DTYPE_NR, DSC$K_DTYPE_NRO,
; 9781     9869  2          DSC$K_DTYPE_NZ]:
; 9782     9870  3             BEGIN
; 9783     9871  3             IF .FLAG THEN RETURN .FROM_DESC;
; 9784     9872  3             INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,
; 9785     9873  3                                         DBG$NUM_BYTES(.TO_DATA_TYPE),
; 9786     9874  3                                         0,
; 9787     9875  3                                         TRUE);
; 9788     9876  3             INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
; 9789     9877  3             INTMED_DESC[DBG$B_VALUE_SCALE] = .FROM_DESC[DBG$B_VALUE_SCALE];
; 9790     9878  3             INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$W_VALUE_LENGTH];
; 9791     9879  3             INTMED_DESC = DBG$TYPE_CONV (.FROM_DESC, .INTMED_DESC);
; 9792     9880  2             END;
; 9793     9881  2
; 9794     9882  2
; 9795     9883  2         ! If right hand side is one of the floating-point data type, we need to
; 9796     9884  2         ! get the exponent value and that is the scaling factor.  For
; 9797     9885  2         ! example F --> P, F = 0.1234567E+04  --> P = 1234.567.  This should
; 9798     9886  2         ! not appear on the left hand side.
; 9799     9887  2         !
; 9800     9888  2         [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H]:
; 9801     9889  3             BEGIN
```

```
;  9802     9890  3            INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,
;  9803     9891  3                                       DBG$NUM_BYTES(.TO_DATA_TYPE),
;  9804     9892  3                                       0,
;  9805     9893  3                                       TRUE);
;  9806     9894  3            .INTMED_DESC[DBG$B_VALUE_SCALE] = GET_SCALE(.FROM_DESC, DIGITS);
;  9807     9895  3            INTMED_DESC[DBG$B_VALUE_DIGITS] = .DIGITS;
;  9808     9896  3            INTMED_DESC[DBG$W_VALUE_LENGTH] = .DIGITS;
;  9809     9897  3            INTMED_DESC = DBG$TYPE_CONV (.FROM_DESC, .INTMED_DESC);
;  9810     9898  2            END;
;  9811     9899  2
;  9812     9900  2
;  9813     9901  2        ! If we deposit a interger data type into a scaled data type,
;  9814     9902  2        ! change its class to be scaled.  This can be on the left
;  9815     9903  2        ! hand side of the deposit (may be voliatile) or on the
;  9816     9904  2        ! right hand side.
;  9817     9905  2        !
;  9818     9906  2        [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_Q,
;  9819     9907  2         DSC$K_DTYPE_BU, DSC$K_DTYPE_WU, DSC$K_DTYPE_LU, DSC$K_DTYPE_QU]:
;  9820     9908  3            BEGIN
;  9821     9909  3            INTMED_DESC = MAKE_VAL_DESC(.TO_DATA_TYPE,
;  9822     9910  3                                       DBG$NUM_BYTES(.TO_DATA_TYPE),
;  9823     9911  3                                       0,
;  9824     9912  3                                       TRUE);
;  9825     9913  3
;  9826     9914  3
;  9827     9915  3            ! Put in the scale information.
;  9828     9916  3            !
;  9829     9917  3            IF .FROM_DESC[DBG$B_VALUE_CLASS] EQL DSC$K_CLASS_SD
;  9830     9918  3            THEN
;  9831     9919  4                BEGIN
;  9832     9920  4                INTMED_DESC[DBG$B_VALUE_SCALE] = .FROM_DESC[DBG$B_VALUE_SCALE];
;  9833     9921  4
;  9834     9922  4
;  9835     9923  4                ! The number of digits is assigned according to the COBOL rule.
;  9836     9924  4                ! This should be adequate for all the languages.
;  9837     9925  4                !
;  9838     9926  4                SELECTONE .FROM_DESC[DBG$B_VALUE_DTYPE] OF
;  9839     9927  4                    SET
;  9840     9928  4                    [DSC$K_DTYPE_B, DSC$K_DTYPE_BU]:
;  9841     9929  5                        BEGIN
;  9842     9930  5                        IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
;  9843     9931  5                        THEN
;  9844     9932  6                            BEGIN
;  9845     9933  6                            INTMED_DESC[DBG$W_VALUE_LENGTH] = 3;
;  9846     9934  6                            INTMED_DESC[DBG$B_VALUE_DIGITS] = 3;
;  9847     9935  6                            END
;  9848     9936  5                        ELSE
;  9849     9937  6                            BEGIN
;  9850     9938  6                            INTMED_DESC[DBG$W_VALUE_LENGTH] - .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9851     9939  6                            INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9852     9940  5                            END;
;  9853     9941  4                        END;
;  9854     9942  4
;  9855     9943  4                    [DSC$K_DTYPE_W, DSC$K_DTYPE_WU]:
;  9856     9944  5                        BEGIN
;  9857     9945  5                        IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
;  9858     9946  5                        THEN
```

```
;  9859          9947  6                    BEGIN
;  9860          9948  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = 5;
;  9861          9949  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = 5;
;  9862          9950  6                    END
;  9863          9951  5                ELSE
;  9864          9952  6                    BEGIN
;  9865          9953  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9866          9954  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9867          9955  5                    END;
;  9868          9956  5
;  9869          9957  4                ;
;  9870          9958  4
;  9871          9959  4            [DSC$K_DTYPE_L, DSC$K_DTYPE_LU]:
;  9872          9960  5                BEGIN
;  9873          9961  5                IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
;  9874          9962  5                THEN
;  9875          9963  6                    BEGIN
;  9876          9964  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = 10;
;  9877          9965  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = 10;
;  9878          9966  6                    END
;  9879          9967  5                ELSE
;  9880          9968  6                    BEGIN
;  9881          9969  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9882          9970  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9883          9971  5                    END;
;  9884          9972  5
;  9885          9973  4                ;
;  9886          9974  4
;  9887          9975  4            [DSC$K_DTYPE_Q, DSC$K_DTYPE_QU]:
;  9888          9976  5                BEGIN
;  9889          9977  5                IF .FROM_DESC[DBG$B_VALUE_DIGITS] EQL 0
;  9890          9978  5                THEN
;  9891          9979  6                    BEGIN
;  9892          9980  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = 20;
;  9893          9981  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = 20;
;  9894          9982  6                    END
;  9895          9983  5                ELSE
;  9896          9984  6                    BEGIN
;  9897          9985  6                    INTMED_DESC[DBG$W_VALUE_LENGTH] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9898          9986  6                    INTMED_DESC[DBG$B_VALUE_DIGITS] = .FROM_DESC[DBG$B_VALUE_DIGITS];
;  9899          9987  5                    END;
;  9900          9988  5
;  9901          9989  4                ;
;  9902          9990  4
;  9903          9991  4                TES;
;  9904          9992  4
;  9905          9993  3            END;
;  9906          9994  3
;  9907          9995  3
;  9908          9996  3        ! There is no real conversion needed for left hand side
;  9909          9997  3        ! of the deposit, we only want to make a place holder.
;  9910          9998  3        !
;  9911          9999  3        IF NOT .FLAG
;  9912         10000  3            THEN
;  9913         10001  3                INTMED_DESC = DBG$TYPE_CONV(.FROM_DESC, .INTMED_DESC);
;  9914         10002  3
;  9915         10003  3            RETURN .INTMED_DESC;
```

```
; 9916        10004  2              END;
; 9917        10005  2
; 9918        10006  2
; 9919        10007  2          ! Does nothing, return the from_desc.
; 9920        10008  2          !
; 9921        10009  2          [INRANGE, OUTRANGE]:
; 9922        10010  2              RETURN .FROM_DESC;
; 9923        10011  2
; 9924        10012  2          TES;
; 9925        10013  2
; 9926        10014  2      RETURN .INTMED_DESC;
; 9927        10015  1      END;
```

```
                                              007C 00000 INTMED_DATA_FOR_DEP:
                                                                .WORD   Save R2,R3,R4,R5,R6                      ; 9833
                              56    0000V  CF  9E 00002        MOVAB   MAKE_VAL_DESC, R6                        ;
                              55    F4AE   CF  9E 00007        MOVAB   DBG$NUM_BYTES, R5                        ;
                              5E           04  C2 0000C        SUBL2   #4, SP                                   ;
                              53      04   AC  D0 0000F        MOVL    FROM_DESC, R3                            ; 9863
                              54      16   A3  9A 00013        MOVZBL  22(R3), R4                               ;
               41      8F     00           54  8F 00017        CASEB   R4, #0, #65                              ;
      00DC           00DC     008A       008A     0001C 1$:    .WORD   3$-1$,-                                  ;
      00DC           00DC     00DC       00DC     00024                3$-1$,-                                  ;
      00AF           00AF     00DC       00DC     0002C                7$-1$,-                                  ;
      008A           008A     008A       008A     00034                7$-1$,-                                  ;
      0086           0086     0086       0086     0003C                7$-1$,-                                  ;
      008A           008A     008A       0086     00044                7$-1$,-                                  ;
      00AF           008A     008A       008A     0004C                7$-1$,-                                  ;
      008A           008A     008A       00AF     00054                7$-1$,-                                  ;
      008A           008A     008A       008A     0005C                7$-1$,-                                  ;
      008A           008A     008A       008A     00064                7$-1$,-                                  ;
      008A           008A     008A       008A     0006C                5$-1$,-                                  ;
      008A           008A     008A       008A     00074                5$-1$,-                                  ;
      008A           008A     008A       008A     0007C                3$-1$,-                                  ;
      008A           008A     008A       008A     00084                3$-1$,-                                  ;
      008A           008A     008A       008A     0008C                3$-1$,-                                  ;
      008A           008A     008A       008A     00094                3$-1$,-                                  ;
                              008A       008A     0009C                2$-1$,-                                  ;
                                                                       2$-1$,-                                  ;
                                                                       2$-1$,-                                  ;
                                                                       2$-1$,-                                  ;
                                                                       2$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       5$-1$,-                                  ;
                                                                       5$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
                                                                       3$-1$,-                                  ;
```

```
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$,-                                  ;
                                                                3$-1$                                    ;
                        04  11 000A0          BRB     3$                                       ; 1001
     04          0C     AC  E9 000A2 2$:      BLBC    FLAG, 4$                                 ; 9871
     50                 53  D0 000A6 3$:      MOVL    R3, R0                                   ;
                            04 000A9          RET                                              ;
                        01  DD 000AA 4$:      PUSHL   #1                                       ; 9872
                        7E  D4 000AC          CLRL    -(SP)                                    ;
                 08     AC  DD 000AE          PUSHL   TO_DATA_TYPE                             ; 9873
     65                 01  FB 000B1          CALLS   #1, DBG$NUM_BYTES                        ;
                        50  DD 000B4          PUSHL   R0                                       ;
                 08     AC  DD 000B6          PUSHL   TO_DATA_TYPE                             ; 9872
     66                 04  FB 000B9          CALLS   #4, MAKE_VAL_DESC                        ;
     52                 50  D0 000BC          MOVL    R0, INTMED_DESC                          ;
1C   A2          1C     A3  B0 000BF          MOVW    28(R3), 28(INTMED_DESC)                  ; 9877
14   A2          14     A3  B0 000C4          MOVW    20(R3), 20(INTMED_DESC)                  ; 9878
                        2A  11 000C9          BRB     6$                                       ; 9879
                        01  DD 000CB 5$:      PUSHL   #1                                       ; 9890
                        7E  D4 000CD          CLRL    -(SP)                                    ;
                 08     AC  DD 000CF          PUSHL   TO_DATA_TYPE                             ; 9891
     65                 01  FB 000D2          CALLS   #1, DBG$NUM_BYTES                        ;
                        50  DD 000D5          PUSHL   R0                                       ;
                 08     AC  DD 000D7          PUSHL   TO_DATA_TYPE                             ; 9890
     66                 04  FB 000DA          CALLS   #4, MAKE_VAL_DESC                        ;
     52                 50  D0 000DD          MOVL    R0, INTMED_DESC                          ;
```

```
                 4008   8F   BB 000E0          PUSHR   #^M<R3,SP>                              ; 9894
06CE   C5               02   FB 000E4          CALLS   #2, GET_SCALE
  1C   A2               50   90 000E9          MOVB    R0, 28(INTMED_DESC)
  1D   A2               6E   90 000ED          MOVB    DIGITS, 29(INTMED_DESC)                 ; 9895
  14   A2               6E   B0 000F1          MOVW    DIGITS, 20(INTMED_DESC)                 ; 9896
                   0098 31   00F5 6$:          BRW     17$                                     ; 9897
                        01   DD 000F8 7$:      PUSHL   #1                                      ; 9909
                        7E   D4 000FA          CLRL    -(SP)
                     08 AC   DD 000FC          PUSHL   TO_DATA_TYPE                            ; 9910
                   65        01   FB 000FF    CALLS   #1, DBG$NUM_BYTES
                             50   DD 00102    PUSHL   R0
                     08 AC   DD 00104          PUSHL   TO_DATA_TYPE                            ; 9909
                   66        04   FB 00107    CALLS   #4, MAKE_VAL_DESC
                   52        50   D0 0010A    MOVL    R0, INTMED_DESC
                   09   17   A3   91 0010D    CMPB    23(R3), #9                              ; 9917
                             79   12 00111    BNEQ    16$
                   51   1C   A2   9E 00113    MOVAB   28(INTMED_DESC), R1                     ; 9920
                   50   1C   A3   9E 00117    MOVAB   28(R3), R0
                   61        60   90 0011B    MOVB    (R0), (R1)
                   02        54   91 0011E    CMPB    R4, #2                                  ; 9928
                             05   13 00121    BEQL    8$
                   06        54   91 00123    CMPB    R4, #6
                             0F   12 00126    BNEQ    9$
                     01 A0   95   00128 8$:    TSTB    1(R0)                                   ; 9930
                             55   12 0012B    BNEQ    15$
  14   A2               03   B0 0012D          MOVW    #3, 20(INTMED_DESC)                     ; 9933
  01   A1               03   90 00131          MOVB    #3, 1(R1)                               ; 9934
                        55   11 00135          BRB     16$                                     ; 9930
                   03        54   91 00137 9$:    CMPB    R4, #3                              ; 9943
                             05   13 0013A    BEQL    10$
                   07        54   91 0013C    CMPB    R4, #7
                             0F   12 0013F    BNEQ    11$
                     01 A0   95   00141 10$:   TSTB    1(R0)                                   ; 9945
                             3C   12 00144    BNEQ    15$
  14   A2               05   B0 00146          MOVW    #5, 20(INTMED_DESC)                     ; 9948
  01   A1               05   90 0014A          MOVB    #5, 1(R1)                               ; 9949
                        3C   11 0014E          BRB     16$                                     ; 9945
                   04        54   91 00150 11$:   CMPB    R4, #4                              ; 9959
                             05   13 00153    BEQL    12$
                   08        54   91 00155    CMPB    R4, #8
                             0F   12 00158    BNEQ    13$
                     01 A0   95   0015A 12$:   TSTB    1(R0)                                   ; 9961
                             23   12 0015D    BNEQ    15$
  14   A2               0A   B0 0015F          MOVW    #10, 20(INTMED_DESC)                    ; 9964
  01   A1               0A   90 00163          MOVB    #10, 1(R1)                              ; 9965
                        23   11 00167          BRB     16$                                     ; 9961
                   05        54   91 00169 13$:   CMPB    R4, #5                              ; 9975
                             05   13 0016C    BEQL    14$
                   09        54   91 0016E    CMPB    R4, #9
                             19   12 00171    BNEQ    16$
                     01 A0   95   00173 14$:   TSTB    1(R0)                                   ; 9977
                             0A   12 00176    BNEQ    15$
  14   A2               14   B0 00178          MOVW    #20, 20(INTMED_DESC)                    ; 9980
  01   A1               14   90 0017C          MOVB    #20, 1(R1)                              ; 9981
                        0A   11 00180          BRB     16$                                     ; 9977
  14   A2   01   A0   9B   00182 15$:   MOVZBW  1(R0), 20(INTMED_DESC)                         ; 9985
  01   A1   01   A0   90   00187          MOVB    1(R0), 1(R1)                                ; 9986
```

```
                         0C      0C    AC  E8 0018C 16$:    BLBS    FLAG, 18$                       ; 9999
                                       52  DD 00190 17$:    PUSHL   INTMED_DESC                     ; 1000
                                       53  DD 00192         PUSHL   R3                              ;
                         042B    C5    02  FB 00194         CALLS   #2, DBG$TYPE_CONV               ;
                                 52    50  D0 00199         MOVL    R0, INTMED_DESC                 ;
                                 50    52  D0 0019C 18$:     MOVL    INTMED_DESC, R0                 ; 1001
                                           04 0019F         RET                                     ; 1001

; Routine Size:  416 bytes,    Routine Base:  DBG$CODE + 23D5

; 9928          10016  1
```

```
; 9930          10017  1        ROUTINE MAKE_VAL_DESC (TYPE, LENGTH, DESC, FLAG) =
; 9931          10018  1        !
; 9932          10019  1        ! FUNCTION
; 9933          10020  1        !       Allocates space for a value descriptor of the given type, and
; 9934          10021  1        !       fills in the fields.
; 9935          10022  1        !
; 9936          10023  1        ! INPUTS
; 9937          10024  1        !       TYPE  -         A VAX standard dtype code.
; 9938          10025  1        !       LENGTH -        Length of the value descriptor
; 9939          10026  1        !       DESC -          Points to a Value descriptor which is
; 9940          10027  1        !                       used in some cases to determine the fields
; 9941          10028  1        !                       of this new descriptor. This is described
; 9942          10029  1        !                       in more detail in the code.
; 9943          10030  1        !       FLAG  -         Flag set to indicate when call dbg$map_dtype_class
; 9944          10031  1        !                       the returned class should be SD instead of S for
; 9945          10032  1        !                       returning S case.
; 9946          10033  1        !
; 9947          10034  1        ! OUTPUTS
; 9948          10035  1        !       Returns the address of a value descriptor allocated out of temporary
; 9949          10036  1        !       memory.
; 9950          10037  1        !
; 9951          10038  1
; 9952          10039  2            BEGIN
; 9953          10040  2            MAP
; 9954          10041  2                DESC: REF DBG$VALDESC;
; 9955          10042  2            LOCAL
; 9956          10043  2                ALLOC_LENGTH,                   ! Allocated length of descriptor
; 9957          10044  2                                                !    in bytes
; 9958          10045  2                DESCR_LENGTH,                   ! Length of the descriptor
; 9959          10046  2                RESULT: REF DBG$VALDESC;        ! Address of the result descriptor
; 9960          10047  2
; 9961          10048  2
; 9962          10049  2            DESCR_LENGTH = .LENGTH;
; 9963          10050  2            IF .TYPE EQL DSC$K_DTYPE_VT
; 9964          10051  2            THEN
; 9965          10052  3                BEGIN
; 9966          10053  3                IF .DESCR_LENGTH LSS 14
; 9967          10054  3                THEN
; 9968          10055  3                    DESCR_LENGTH = 16
; 9969          10056  3                ELSE
; 9970          10057  3                    DESCR_LENGTH = .DESCR_LENGTH + 2;
; 9971          10058  3                RESULT = DBG$GET_TEMPMEM (ALLOC_LENGTH = DBG$K_VALDESC_BASE_SIZE +
; 9972          10059  3                        (3 + MIN (.DESCR_LENGTH, 256 + 2)) / 4);
; 9973          10060  3                END
; 9974          10061  3
; 9975          10062  2            ELSE
; 9976          10063  3                BEGIN
; 9977          10064  3                IF .TYPE EQL DSC$K_DTYPE_P
; 9978          10065  3                THEN
; 9979          10066  3                    DESCR_LENGTH = MAX (16, (.DESCR_LENGTH + 1) / 4)
; 9980          10067  3                ELSE
; 9981          10068  3                    DESCR_LENGTH = MAX (16, .DESCR_LENGTH);
; 9982          10069  3
; 9983          10070  3                RESULT = DBG$GET_TEMPMEM (ALLOC_LENGTH = DBG$K_VALDESC_BASE_SIZE +
; 9984          10071  3                        (3 + MIN (.DESCR_LENGTH, 256)) / 4);
; 9985          10072  2                END;
; 9986          10073  2
```

```
;  9987          10074  2
;  9988          10075  2          ! Fill in the fields of the new value descriptor.
;  9989          10076  2          !
;  9990          10077  2          RESULT [DBG$B_DHDR_LANG] = .DBG$GB_LANGUAGE;
;  9991          10078  2          RESULT [DBG$B_DHDR_TYPE] = DBG$K_VALUE_DESC;
;  9992          10079  2          RESULT [DBG$W_DHDR_LENGTH] = 4*.ALLOC_LENGTH;
;  9993          10080  2          RESULT [DBG$B_DHDR_KIND] = RST$K_DATA;
;  9994          10081  2          IF .TYPE GTR DBG$K_MAXIMUM_DTYPE
;  9995          10082  2          THEN
;  9996          10083  3              BEGIN
;  9997          10084  3              RESULT [DBG$B_DHDR_FCODE] = .TYPE - DBG$K_MAXIMUM_DTYPE;
;  9998          10085  3              RESULT [DBG$B_VALUE_CLASS] = 0;
;  9999          10086  3              RESULT [DBG$B_VALUE_DTYPE] = 0;
; 10000          10087  3              END
; 10001          10088  3
; 10002          10089  2          ELSE
; 10003          10090  3              BEGIN
; 10004          10091  3              IF .FLAG
; 10005          10092  3              THEN
; 10006          10093  3                  RESULT [DBG$B_DHDR_FCODE] = RST$K_TYPE_DESCR
; 10007          10094  3
; 10008          10095  3              ELSE
; 10009          10096  3                  RESULT [DBG$B_DHDR_FCODE] = RST$K_TYPE_ATOMIC;
; 10010          10097  3              RESULT [DBG$B_VALUE_CLASS] = DBG$MAP_DTYPE_CLASS(.TYPE, .FLAG);
; 10011          10098  3              RESULT [DBG$B_VALUE_DTYPE] = .TYPE;
; 10012          10099  2              END;
; 10013          10100  2
; 10014          10101  2          RESULT [DBG$W_VALUE_LENGTH] = MIN(.LENGTH, 256);
; 10015          10102  2          RESULT [DBG$L_VALUE_POINTER] = RESULT[DBG$A_VALUE_ADDRESS];
; 10016          10103  2
; 10017          10104  2          ! If we are converting to "FIXED" then we need to fill in the
; 10018          10105  2          ! dtype field. We also need to light the binscale flag.
; 10019          10106  2          !
; 10020          10107  2          IF .TYPE EQL DSC$K_DTYPE_FIXED
; 10021          10108  2          THEN
; 10022          10109  3              BEGIN
; 10023          10110  3              RESULT[DBG$V_VALUE_FL_BINSCALE] = 1;
; 10024          10111  3              RESULT[DBG$B_VALUE_DTYPE] = DSC$K_DTYPE_L;
; 10025          10112  2              END;
; 10026          10113  2
; 10027          10114  2          RETURN .RESULT;
; 10028          10115  1          END;    ! MAKE_VAL_DESC


                                001C 00000 MAKE_VAL_DESC:
                                           .WORD   Save R2,R3,R4                          ; 1001
   50        08    AC   D0 00002           MOVL    LENGTH, DESCR_LENGTH                   ; 1004
   54        04    AC   D0 00006           MOVL    TYPE, R4                               ; 1005
   25              54   D1 0000A           CMPL    R4, #37                                ;
                   2A   12 0000D           BNEQ    4$                                     ;
   0E              50   D1 0000F           CMPL    DESCR_LENGTH, #14                      ; 1005
                   05   18 00012           BGEQ    1$                                     ;
   50              10   D0 00014           MOVL    #16, DESCR_LENGTH                      ; 1005
                   03   11 00017           BRB     2$                                     ;
```

```
                50              02  C0 00019 1$:    ADDL2   #2, DESCR_LENGTH                          ; 1005
                51              50  D0 0001C 2$:    MOVL    DESCR_LENGTH, R1                        ; 1005
00000102        8F              51  D1 0001F         CMPL    R1, #258                                ;
                                05  15 00026         BLEQ    3$                                      ;
                51      0102    8F  3C 00028         MOVZWL  #258, R1                                ;
                51              03  C0 0002D 3$:    ADDL2   #3, R1                                  ;
                51              04  C6 00030         DIVL2   #4, R1                                  ;
                52        08    A1  9E 00033         MOVAB   8(R1), ALLOC_LENGTH                     ; 1005
                                39  11 00037         BRB     9$                                      ;
                15              54  D1 00039 4$:    CMPL    R4, #21                                 ; 1006
                                0E  12 0003C         BNEQ    5$                                      ;
                51        01    A0  9E 0003E         MOVAB   1(R0), R1                               ; 1006
                51              04  C6 00042         DIVL2   #4, R1                                  ;
                10              51  D1 00045         CMPL    R1, #16                                 ;
                                0A  19 00048         BLSS    6$                                      ;
                                0B  11 0004A         BRB     7$                                      ;
                51              50  D0 0004C 5$:    MOVL    DESCR_LENGTH, R1                        ; 1006
                10              51  D1 0004F         CMPL    R1, #16                                 ;
                                03  18 00052         BGEQ    7$                                      ;
                51              10  D0 00054 6$:    MOVL    #16, R1                                 ;
                50              51  D0 00057 7$:    MOVL    R1, DESCR_LENGTH                        ;
00000100        8F              50  D1 0005A         CMPL    R0, #256                                ; 1007
                                05  15 00061         BLEQ    8$                                      ;
                50      0100    8F  3C 00063         MOVZWL  #256, R0                                ;
                50              03  C0 00068 8$:    ADDL2   #3, R0                                  ;
                50              04  C6 0006B         DIVL2   #4, R0                                  ;
                52        08    A0  9E 0006E         MOVAB   8(R0), ALLOC_LENGTH                     ; 1007
                                52  DD 00072 9$:    PUSHL   ALLOC_LENGTH                            ;
00000000G       00              01  FB 00074         CALLS   #1, DBG$GET_TEMPMEM                     ;
                53              50  D0 0007B         MOVL    R0, RESULT                              ;
          03    A3  00000000G   00  90 0007E         MOVB    DBG$GB_LANGUAGE, 3(RESULT)              ; 1007
          02    A3        7A    8F  90 00086         MOVB    #122, 2(RESULT)                         ; 1007
    63            52              04  A5 0008B         MULW3   #4, ALLOC_LENGTH, (RESULT)              ; 1007
          07    A3              06  90 0008F         MOVB    #6, 7(RESULT)                           ; 1008
                52        14    A3  9E 00093         MOVAB   20(RESULT), R2                          ; 1008
                2B              54  D1 00097         CMPL    R4, #43                                 ; 1008
                                0A  15 0009A         BLEQ    10$                                     ;
06  A3            54              2B  83 0009C         SUBB3   #43, R4, 6(RESULT)                      ; 1008
                          02    A2  B4 000A1         CLRW    2(R2)                                   ; 1008
                                20  11 000A4         BRB     13$                                     ; 1008
                06        10    AC  E9 000A6 10$:    BLBC    FLAG, 11$                               ; 1009
          06    A3              03  90 000AA         MOVB    #3, 6(RESULT)                           ; 1009
                                04  11 000AE         BRB     12$                                     ;
          06    A3              02  90 000B0 11$:    MOVB    #2, 6(RESULT)                           ; 1009
                          10    AC  DD 000B4 12$:    PUSHL   FLAG                                    ; 1009
                                54  DD 000B7         PUSHL   R4                                      ;
              F19F      CF          02  FB 000B9         CALLS   #2, DBG$MAP_DTYPE_CLASS                 ;
          03    A2              50  90 000BE         MOVB    R0, 3(R2)                               ;
          02    A2              54  90 000C2         MOVB    R4, 2(R2)                               ; 1009
                50        08    AC  D0 000C6 13$:    MOVL    LENGTH, R0                              ; 1010
        00000100    8F          50  D1 000CA         CMPL    R0, #256                                ;
                                05  15 000D1         BLEQ    14$                                     ;
                50      0100    8F  3C 000D3         MOVZWL  #256, R0                                ;
                62              50  B0 000D8 14$:    MOVW    R0, (R2)                                ;
          18    A3        20    A3  9E 000DB         MOVAB   32(R3), 24(RESULT)                      ; 1010
                2B              54  D1 000E0         CMPL    R4, #43                                 ; 1010
                                08  12 000E3         BNEQ    15$                                     ;
```

```
                1E   A3          08  88 000E5          BISB2   #8, 30(RESULT)                  ; 1011
                02   A2          08  90 000E9          MOVB    #8, 2(R2)                       ; 1011
                     50          53  D0 000ED 15$:     MOVL    RESULT, R0                      ; 1011
                                     04 000F0          RET                                     ; 1011

; Routine Size:  241 bytes,    Routine Base:  DBG$CODE + 2575
```

```
;10030          10116  1        ROUTINE MAP_NRO_DTYPE_IN_RPG(VAL_DESC) =
;10031          10117  1        !
;10032          10118  1        ! FUNCTION
;10033          10119  1        !       This routine maps the representation of least significant digit and
;10034          10120  1        !       sign into RPG standard output format acceptting both normal or
;10035          10121  1        !       alternate as inputs.
;10036          10122  1        !
;10037          10123  1        ! INPUTS
;10038          10124  1        !       VAL_DESC        - Pointer to NRO data type value descriptor.
;10039          10125  1        !
;10040          10126  1        ! OUTPUTS
;10041          10127  1        !       VAL_DESC is returned.
;10042          10128  1        !
;10043          10129  1
;10044          10130  2            BEGIN
;10045          10131  2
;10046          10132  2            MAP
;10047          10133  2                VAL_DESC: REF DBG$VALDESC;      ! Pointer to value descriptor
;10048          10134  2
;10049          10135  2            LOCAL
;10050          10136  2                LEAST_SIG_DIGIT: REF VECTOR[,BYTE];! Pointer to the least significant
;10051          10137  2                                                  !    digit
;10052          10138  2
;10053          10139  2
;10054          10140  2            ! In RPG, representation of least significant digit and sign
;10055          10141  2            ! both normal/alternate ASCII char. can be accepted in RPG.
;10056          10142  2            ! but RPG choose one format to be the standard output format.
;10057          10143  2            ! In here we simply map it into the standard output format.
;10058          10144  2            !
;10059          10145  2            LEAST_SIG_DIGIT = .VAL_DESC[DBG$L_VALUE_POINTER] +
;10060          10146  2                              .VAL_DESC[DBG$W_VALUE_LENGTH] - 1;
;10061          10147  2
;10062          10148  2
;10063          10149  2            ! We perform the following mapping:
;10064          10150  2            !
;10065          10151  2            ! Input                  Output
;10066          10152  2            ! norm.  alt.
;10067          10153  2            !  {     0 [ ?            0
;10068          10154  2            !  A     1                1
;10069          10155  2            !  B     2                2
;10070          10156  2            !  C     3                3
;10071          10157  2            !  D     4                4
;10072          10158  2            !  E     5                5
;10073          10159  2            !  F     6                6
;10074          10160  2            !  G     7                7
;10075          10161  2            !  H     8                8
;10076          10162  2            !  I     9                9
;10077          10163  2            !  }     ] : !            }
;10078          10164  2            !
;10079          10165  2            IF .LEAST_SIG_DIGIT[0] EQL '{' OR
;10080          10166  2               .LEAST_SIG_DIGIT[0] EQL '0' OR
;10081          10167  2               .LEAST_SIG_DIGIT[0] EQL '[' OR
;10082          10168  2               .LEAST_SIG_DIGIT[0] EQL '?'
;10083          10169  2            THEN
;10084          10170  2                LEAST_SIG_DIGIT[0] = '0';
;10085          10171  2
;10086          10172  2            IF .LEAST_SIG_DIGIT[0] GEQ 'A' AND
```

```
;10087          10173  2              .LEAST_SIG_DIGIT[0] LEQ 'I'
;10088          10174  2          THEN
;10089          10175  2              LEAST_SIG_DIGIT[0] = .LEAST_SIG_DIGIT[0] - %X'10';
;10090          10176  2
;10091          10177  2          IF .LEAST_SIG_DIGIT[0] EQL '}' OR
;10092          10178  2             .LEAST_SIG_DIGIT[0] EQL ']' OR
;10093          10179  2             .LEAST_SIG_DIGIT[0] EQL ':' OR
;10094          10180  2             .LEAST_SIG_DIGIT[0] EQL '!'
;10095          10181  2          THEN
;10096          10182  2              LEAST_SIG_DIGIT[0] = '}';
;10097          10183  2
;10098          10184  2          RETURN .VAL_DESC;
;10099          10185  1          END;


                                0000 00000 MAP_NRO_DTYPE_IN_RPG:
                                           .WORD   Save nothing                                    ; 1011
     51        04   AC  D0 00002          MOVL    VAL_DESC, R1                                    ; 1014
     50        14   A1  3C 00006          MOVZWL  20(R1), R0                                      ; 1014
     50        18   A1  C0 0000A          ADDL2   24(R1), R0                                      ;
7B   8F             70  91 0000E          CMPB    -(LEAST_SIG_DIGIT), #123                        ; 1016
                    10  13 00012          BEQL    1$                                              ;
     30             60  91 00014          CMPB    (LEAST_SIG_DIGIT), #48                          ; 1016
                    0B  13 00017          BEQL    1$                                              ;
5B   8F             60  91 00019          CMPB    (LEAST_SIG_DIGIT), #91                          ; 1016
                    05  13 0001D          BEQL    1$                                              ;
     3F             60  91 0001F          CMPB    (LEAST_SIG_DIGIT), #63                          ; 1016
                    03  12 00022          BNEQ    2$                                              ;
     60             30  90 00024 1$:      MOVB    #48, (LEAST_SIG_DIGIT)                          ; 1017
41   8F             60  91 00027 2$:      CMPB    (LEAST_SIG_DIGIT), #65                          ; 1017
                    09  1F 0002B          BLSSU   3$                                              ;
49   8F             60  91 0002D          CMPB    (LEAST_SIG_DIGIT), #73                          ; 1017
                    03  1A 00031          BGTRU   3$                                              ;
     60             10  82 00033          SUBB2   #16, (LEAST_SIG_DIGIT)                          ; 1017
7D   8F             60  91 00036 3$:      CMPB    (LEAST_SIG_DIGIT), #125                         ; 1017
                    10  13 0003A          BEQL    4$                                              ;
5D   8F             60  91 0003C          CMPB    (LEAST_SIG_DIGIT), #93                          ; 1017
                    0A  13 00040          BEQL    4$                                              ;
     3A             60  91 00042          CMPB    (LEAST_SIG_DIGIT), #58                          ; 1017
                    05  13 00045          BEQL    4$                                              ;
     21             60  91 00047          CMPB    (LEAST_SIG_DIGIT), #33                          ; 1018
                    04  12 0004A          BNEQ    5$                                              ;
     60        7D   8F  90 0004C 4$:      MOVB    #125, (LEAST_SIG_DIGIT)                         ; 1018
     50             51  D0 00050 5$:      MOVL    R1, R0                                          ; 1018
                        04 00053          RET                                                     ; 1018

; Routine Size:  84 bytes,    Routine Base:  DBG$CODE + 2666

;10100          10186  1
```

```
:10102          10187  1        ROUTINE MAP_PACKED (NEW_TYPE, DIGITS, OTHER_TYPE):  NOVALUE =
:10103          10188  1        !
:10104          10189  1        ! FUNCTION
:10105          10190  1        !       This routine maps a packed decimal type to another type.  This cannot
:10106          10191  1        !       always be done in the hierarchy tables; in the case of converting
:10107          10192  1        !       packed to floating point, it is necessary to know how many digits are
:10108          10193  1        !       involved before mapping the type.
:10109          10194  1        !
:10110          10195  1        ! INPUTS
:10111          10196  1        !       NEW_TYPE        - The address in which to place the new type.  On
:10112          10197  1        !                         entering the routine, it should already point to
:10113          10198  1        !                         type DSC$K_DTYPE_P.
:10114          10199  1        !       DIGITS          - The number of digits in the packed decimal number.
:10115          10200  1        !       OTHER_TYPE      - In certain instances, the other operand helps in
:10116          10201  1        !                         determining how to map the packed decimal number.
:10117          10202  1        !                         For example, if there are 15 digits in the packed
:10118          10203  1        !                         decimal number:  it is mapped to gfloat if the other
:10119          10204  1        !                         operand is gfloat, hfloat if the other operand is
:10120          10205  1        !                         hfloat, otherwise double.
:10121          10206  1        !
:10122          10207  1        ! OUTPUTS
:10123          10208  1        !       A new type may be returned in NEW_TYPE.
:10124          10209  1        !
:10125          10210  2            BEGIN
:10126          10211  2
:10127          10212  2
:10128          10213  2            ! Expect NEW_TYPE to already point to a packed decimal type.  This assures
:10129          10214  2            ! that we have at least one packed decimal operand, and can case on the
:10130          10215  2            ! other.
:10131          10216  2            !
:10132          10217  2            IF ..NEW_TYPE NEQ DSC$K_DTYPE_P
:10133          10218  2            THEN
:10134          10219  2                RETURN;
:10135          10220  2
:10136          10221  2            CASE .OTHER_TYPE FROM 0 TO DBG$K_MAXIMUM_DTYPE + RST$K_TYPE_MAXIMUM OF
:10137          10222  2                SET
:10138          10223  2
:10139          10224  2
:10140          10225  2                ! If the other operand is floating, then the conversion is packed ->
:10141          10226  2                ! float.  The type of floating point conversion (whether to single or
:10142          10227  2                ! double precision, or to gfloat or hfloat) depends on both the number
:10143          10228  2                ! of digits in the packed decimal number, and in some cases, on the
:10144          10229  2                ! type of the other operand.
:10145          10230  2                !
:10146          10231  2                [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H]:
:10147          10232  3                    BEGIN
:10148          10233  3                    CASE .DIGITS FROM 1 TO DBG$K_LARGEST_PACKED OF
:10149          10234  3                        SET
:10150          10235  3
:10151          10236  3                        [1 TO 6]:
:10152          10237  3                            .NEW_TYPE = DSC$K_DTYPE_F;
:10153          10238  3
:10154          10239  3                        [7 TO 15]:
:10155          10240  4                            BEGIN
:10156          10241  4                            IF .OTHER_TYPE EQL DSC$K_DTYPE_G OR
:10157          10242  4                                .OTHER_TYPE EQL DSC$K_DTYPE_H
:  .  58        10243  4                            THEN
```

```
;10159          10244  4                        .NEW_TYPE = .OTHER_TYPE
;10160          10245  4                    ELSE
;10161          10246  4                        .NEW_TYPE = DSC$K_DTYPE_D;
;10162          10247  3                    END,
;10163          10248  3
;10164          10249  3                [16]:
;10165          10250  3                    .NEW_TYPE = DSC$K_DTYPE_D;
;10166          10251  3
;10167          10252  3                [17 TO 31]:
;10168          10253  3                    .NEW_TYPE = DSC$K_DTYPE_H;
;10169          10254  3
;10170          10255  3                [OUTRANGE]:
;10171          10256  3                    SIGNAL (DBG$_ILLTYPE);
;10172          10257  3                TES;
;10173          10258  3            RETURN;
;10174          10259  2            END;
;10175          10260  2
;10176          10261  2
;10177          10262  2            ! Presently, the other type conversions involving packed decimal types
;10178          10263  2            ! can be handled through the hierarchy tables.
;10179          10264  2            !
;10180          10265  2            [INRANGE]:
;10181          10266  2                RETURN;
;10182          10267  2
;10183          10268  2            [OUTRANGE]:
;10184          10269  2                $DBG_ERROR ('DBGEVALOP\MAP_PACKED');
;10185          10270  2            TES;
;10186          10271  1        END;


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

5F 50 41 4D 5C 50 4F 4C 41 56 45 47 42 44 14 0617D P.AMT:   .ASCII  <20>\DBGEVALOP\<92>\MAP_PACKED\                ;
                        44 45 4B 43 41 50 0618C                                                                 ;


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

                                        000C 00000 MAP_PACKED:
                                                .WORD   Save R2,R3                                              ; 1018
                                  53 00000000G   00  9E 00002           MOVAB   LIB$SIGNAL, R3                  ;
                                  52         04  AC  D0 00009           MOVL    NEW_TYPE, R2                    ; 1021
                                  15             62  D1 0000D           CMPL    (R2), #21                       ;
                                                 01  13 00010           BEQL    1$                              ;
                                                     04 00012           RET                                     ;
                        00000041   8F       00    0C  AC  CF 00013 1$:  CASEL   OTHER_TYPE, #0, #65             ; 1022
                0FFF        0FFF        0FFF        0FFF    0001C 2$:   .WORD   10$-2$,-                        ;
                0FFF        0FFF        0FFF        0FFF    00024       10$-2$,-                                ;
                0096        0096        0FFF        0FFF    0002C       10$-2$,-                                ;
                0FFF        0FFF        0FFF        0FFF    00034       10$-2$,-                                ;
                0FFF        0FFF        0FFF        0FFF    0003C       10$-2$,-                                ;
                0FFF        0FFF        0FFF        0FFF    00044       10$-2$,-                                ;
                0096        0FFF        0FFF        0FFF    0004C       10$-2$,-                                ;
                0FFF        0FFF        0FFF        0096    00054       10$-2$,-                                ;
                0FFF        0FFF        0FFF        0FFF    0005C       10$-2$,-                                ;
```

```
        00FF      00FF      00FF      00FF   00064          10$-2$,-          ;
        00FF      00FF      00FF      00FF   0006C          3$-2$,-           ;
        00FF      00FF      00FF      00FF   00074          3$-2$,-           ;
        00FF      00FF      00FF      00FF   0007C          10$-2$,-          ;
        00FF      00FF      00FF      00FF   00084          10$-2$,-          ;
        00FF      00FF      00FF      00FF   0008C          10$-2$,-          ;
        00FF      00FF      00FF      00FF   00094          10$-2$,-          ;
                            00FF      00FF   0009C          10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            3$-2$,-           ;
                                                            3$-2$,-           ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$,-          ;
                                                            10$-2$            ;
```

```
                              00000000'  EF  9F 000A0          PUSHAB  P.AMT                    ; 1026
                                         ^1  DD 000A6          PUSHL   #1                       ;
                              00028362   8F  DC 000A8          PUSHL   #164706                  ;
                         63              03  FB 000AE          CALLS   #3, LIB$SIGNAL           ;
                                         04     000B1          RET                              ;
                 1E      01       08     AC  CF 000B2 3$:      CASEL   DIGITS, #1, #30          ; 1023
      0048     0048    0048     0048             000B7 4$:      .WORD   5$-4$,-                  ;
      004C     004C    0048     0048             000BF                  5$-4$,-                  ;
      004C     004C    004C     004C             000C7                  5$-4$,-                  ;
      005D     004C    004C     004C             000CF                  5$-4$,-                  ;
      0061     0061    0061     0061             000D7                  5$-4$,-                  ;
      0061     0061    0061     0061             000DF                  5$-4$,-                  ;
      0061     0061    0061     0061             000E7                  6$-4$,-                  ;
               0061    0061     0061             000EF                  6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        6$-4$,-                  ;
                                                                        8$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$,-                  ;
                                                                        9$-4$                    ;
                              000287D8   8F  DD 000F5          PUSHL   #165848                  ; 1025
                         63              01  FB 000FB          CALLS   #1, LIB$SIGNAL           ;
                                         04     000FE          RET                              ;
                         62              0A  D0 000FF 5$:      MOVL    #10, (R2)                ; 1023
                                         04     00102          RET                              ;
                         1B       0C     AC  D1 00103 6$:      CMPL    OTHER_TYPE, #27          ; 1024
                                         06  13 00107          BEQL    7$                       ;
                         1C       0C     AC  D1 00109          CMPL    OTHER_TYPE, #28          ; 1024
                                         05  12 0010D          BNEQ    8$                       ;
                         62       0C     AC  D0 0010F 7$:      MOVL    OTHER_TYPE, (R2)         ; 1024
                                         04     00113          RET                              ;
                         62              0B  D0 00114 8$:      MOVL    #11, (R2)                ; 1025
                                         04     00117          RET                              ;
                         62              1C  D0 00118 9$:      MOVL    #28, (R2)                ; 1025
                                         04     0011B 10$:     RET                              ; 1027

; Routine Size:  284 bytes,    Routine Base:  DBG$CODE + 26BA
```

```
;10188      10272  1     ROUTINE MAP_PLI_TYPE_SIZE (VALDESC, TYPE, SIZE, SRC_FLAG):  NOVALUE =
;10189      10273  1     !
;10190      10274  1     ! FUNCTION
;10191      10275  1     !       PL/I run-time routines do not follow the VAX calling standard.  They
;10192      10276  1     !       also do not understand VMS descriptors.  Consequently, this routine
;10193      10277  1     !       was developed to map dtypes to PL/I specific types, and to determine
;10194      10278  1     !       the correct size for that specific type.  This information is
;10195      10279  1     !       required in the interface to a PL/I routine.
;10196      10280  1     !
;10197      10281  1     ! INPUTS
;10198      10282  1     !       VALDESC - Pointer to a value descriptor.
;10199      10283  1     !
;10200      10284  1     !       TYPE    - Address where to place the PL/I specific type.
;10201      10285  1     !
;10202      10286  1     !       SIZE    - Address where to place the size of the type.
;10203      10287  1     !
;10204      10288  1     !       SRC_FLAG - Set to TRUE to indicate this is the source,
;10205      10289  1     !                  set to FALSE to indicate this is the target.
;10206      10290  1     !
;10207      10291  1     ! OUTPUTS
;10208      10292  1     !       The PL/I specific data type and size are returned in TYPE and SIZE.
;10209      10293  1     !
;10210      10294  2       BEGIN
;10211      10295  2
;10212      10296  2       MAP
;10213      10297  2           TYPE: REF VECTOR[1],
;10214      10298  2           SIZE: REF VECTOR[1],
;10215      10299  2           VALDESC:  REF DBG$VALDESC;
;10216      10300  2
;10217      10301  2       LOCAL
;10218      10302  2           SIZE_BYTE: REF VECTOR[,BYTE],    ! Size in byte vector
;10219      10303  2           LANGCODE,                        ! Language code
;10220      10304  2           PICTPTR: REF VECTOR[,BYTE],      ! Pointer to picture representation
;10221      10305  2           PICTVAL,                         ! Pointer to language specific encoding
;10222      10306  2           PSCALE: VECTOR[2, BYTE];         ! Digits and Scale
;10223      10307  2
;10224      10308  2
;10225      10309  2       SIZE_BYTE = .SIZE;
;10226      10310  2       SIZE[0] = 0;
;10227      10311  2       CASE .VALDESC[DBG$B_VALUE_DTYPE] FROM 0 TO DBG$K_MAXIMUM_DTYPE OF
;10228      10312  2           SET
;10229      10313  2           [DSC$K_DTYPE_B]:
;10230      10314  3               BEGIN
;10231      10315  3               TYPE[0] = DBG$K_PLI_FIX_BIN;
;10232      10316  3               SIZE[0] = 7;
;10233      10317  2               END;
;10234      10318  2           [DSC$K_DTYPE_W]:
;10235      10319  3               BEGIN
;10236      10320  3               TYPE[0] = DBG$K_PLI_FIX_BIN;
;10237      10321  3               SIZE[0] = 15;
;10238      10322  2               END;
;10239      10323  2           [DSC$K_DTYPE_L]:
;10240      10324  3               BEGIN
;10241      10325  3               TYPE[0] = DBG$K_PLI_FIX_BIN;
;10242      10326  3               SIZE[0] = 31;
;10243      10327  2               END;
;10244      10328  2           [DSC$K_DTYPE_F]:
```

```
;10245      10329  3                BEGIN
;10246      10330  3                TYPE[0] = DBG$K_PLI_FLO_DEC;
;10247      10331  3                SIZE[0] = 7;
;10248      10332  2                END;
;10249      10333  2            [DSC$K_DTYPE_D, DSC$K_DTYPE_G]:
;10250      10334  3                BEGIN
;10251      10335  3                TYPE[0] = DBG$K_PLI_FLO_DEC;
;10252      10336  3                SIZE[0] = 15;
;10253      10337  2                END;
;10254      10338  2            [DSC$K_DTYPE_H]:
;10255      10339  3                BEGIN
;10256      10340  3                TYPE[0] = DBG$K_PLI_FLO_DEC;
;10257      10341  3                SIZE[0] = 34;
;10258      10342  2                END;
;10259      10343  2            [DSC$K_DTYPE_T]:
;10260      10344  3                BEGIN
;10261      10345  3                IF .VALDESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT
;10262      10346  3                THEN
;10263      10347  4                    BEGIN
;10264      10348  4                    TYPE[0] = DBG$K_PLI_PIC;
;10265      10349  4                    DBG$STA_TYP_PICT(.VALDESC[DBG$L_DHDR_TYPEID], LANGCODE,
;10266      10350  4                            PICTPTR, PICTVAL, PSCALE);
;10267      10351  4                    SIZE[0] = .PICTVAL;
;10268      10352  4                    END
;10269      10353  3                ELSE
;10270      10354  4                    BEGIN
;10271      10355  4                    TYPE[0] = DBG$K_PLI_CHAR;
;10272      10356  4                    SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
;10273      10357  3                    END;
;10274      10358  2                END;
;10275      10359  2            [DSC$K_DTYPE_VT]:
;10276      10360  3                BEGIN
;10277      10361  3                LOCAL
;10278      10362  3                    VALPTR: REF VECTOR[,WORD];
;10279      10363  3
;10280      10364  3                TYPE[0] = DBG$K_PLI_CHAR_VAR;
;10281      10365  3                IF .SRC_FLAG
;10282      10366  3                THEN
;10283      10367  4                    BEGIN
;10284      10368  4                    VALPTR = .VALDESC[DBG$L_VALUE_POINTER];
;10285      10369  4                    SIZE[0] = .VALPTR[0];
;10286      10370  4                    END
;10287      10371  3                ELSE
;10288      10372  3                    SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
;10289      10373  3
;10290      10374  2                END;
;10291      10375  2            [DSC$K_DTYPE_P]:
;10292      10376  3                BEGIN
;10293      10377  3                TYPE[0] = DBG$K_PLI_FIX_DEC;
;10294      10378  3
;10295      10379  3
;10296      10380  3                ! PLI runtime library routine expect to see 1234.1234 (8 digits
;10297      10381  3                ! and 4 as scale factor, normally, would be -4 as scale factor).
;10298      10382  3                !
;10299      10383  3                SIZE_BYTE[1] = - .VALDESC[DBG$B_VALUE_SCALE];
;10300      10384  3                SIZE_BYTE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
;10301      10385  2                END;
```

```
;10302          10386  2              [DSC$K_DTYPE_V]:
;10303          10387  3                  BEGIN
;10304          10388  3                  TYPE[0] = DBG$K_PLI_ABIT;
;10305          10389  3                  SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
;10306          10390  2                  END;
;10307          10391  2              [DSC$K_DTYPE_VU]:
;10308          10392  3                  BEGIN
;10309          10393  3                  TYPE[0] = DBG$K_PLI_UBIT;
;10310          10394  3                  SIZE[0] = .VALDESC[DBG$W_VALUE_LENGTH];
;10311          10395  2                  END;
;10312          10396  2              [INRANGE, OUTRANGE]:
;10313          10397  3                  BEGIN
;10314          10398  3                  $DBG_ERROR ('DBGEVALOP\MAP_PLI_TYPE_SIZE:  invalid dtype');
;10315          10399  2                  END;
;10316          10400  2              TES;
;10317          10401  2
;10318          10402  1          END;


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

5F 50 41 4D 5C 50 4F 4C 41 56 45 47 42 44 2B  06192 P.AMU:  .ASCII  \+DBGEVALOP\<92>\MAP_PLI_TYPE_SIZE:  inv\  ;
20 3A 45 5A 49 53 5F 45 50 59 54 5F 49 4C 50  061A1                                                          ;
                                76 6E 69 20   061B0                                                          ;
                  65 70 79 74 64 20 64 69 6C 61 061B4        .ASCII  \alid dtype\                            ;


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

                                          003C 00000 MAP_PLI_TYPE_SIZE:
                                                        .WORD   Save R2,R3,R4,R5                        ; 1027
                                5E          10  C2 00002               SUBL2   #16, SP                 ;
                                53      0C  AC  D0 00005               MOVL    SIZE, R3                ; 1030
                                54          53  D0 00009               MOVL    R3, SIZE_BYTE           ;
                                            63  D4 0000C               CLRL    (R3)                    ; 1031
                                52      04  AC  D0 0000E               MOVL    VALDESC, R2             ; 1031
                                55      14  A2  9E 00012               MOVAB   20(R2), R5              ;
                          2B    00      02  A5  8F 00016               CASEB   2(R5), #0, #43          ;
                0058      0058  00E1        0058      0001B 1$:         .WORD   2$-1$,-                 ;
                0074      006E  0058        0058      00023             15$-1$,-                        ;
                008A      0082  0058        007A      0002B             2$-1$,-                         ;
                0058      009A  0058        0058      00033             2$-1$,-                         ;
                0058      0058  0058        0058      0003B             2$-1$,-                         ;
                0058      0058  00D4        0058      00043             2$-1$,-                         ;
                008A      0058  0058        0058      0004B             3$-1$,-                         ;
                0058      0058  0058        0092      00053             4$-1$,-                         ;
                0058      00E7  0058        0058      0005B             5$-1$,-                         ;
                0058      0058  00C4        0058      00063             2$-1$,-                         ;
                0058      0058  0058        0058      0006B             6$-1$,-                         ;
                                                                        8$-1$,-                         ;
                                                                        2$-1$,-                         ;
                                                                        2$-1$,-                         ;
                                                                        11$-1$,-                        ;
                                                                        2$-1$,-                         ;
                                                                        2$-1$,-                         ;
```

```
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              14$-1$,-                         ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              8$-1$,-                          ;
                                                              10$-1$,-                         ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              16$-1$,-                         ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              13$-1$,-                         ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$,-                          ;
                                                              2$-1$                            ;
                00000000'  EF  9F 00073 2$:     PUSHAB  P.AMU                                  ; 1039
                           01  DD 00079         PUSHL   #1                                     ;
                00028362   8F  DD 0007B         PUSHL   #164706                                ;
00000000G  00              03  FB 00081         CALLS   #3, LIB$SIGNAL                         ;
                               04 00088         RET                                            ; 1031
      08   BC              02  D0 00089 3$:     MOVL    #2, @TYPE                              ; 1031
                           12  11 0008D         BRB     7$                                     ; 1031
      08   BC              02  D0 0008F 4$:     MOVL    #2, @TYPE                              ; 1032
                           14  11 00093         BRB     9$                                     ; 1032
      08   BC              02  D0 00095 5$:     MOVL    #2, @TYPE                              ; 1032
           63              1F  D0 00099         MOVL    #31, (R3)                              ; 1032
                               04 0009C         RET                                            ; 1031
      08   BC              05  D0 0009D 6$:     MOVL    #5, @TYPE                              ; 1033
           63              07  D0 000A1 7$:     MOVL    #7, (R3)                               ; 1033
                               04 000A4         RET                                            ; 1031
      08   BC              05  D0 000A5 8$:     MOVL    #5, @TYPE                              ; 1033
           63              0F  D0 000A9 9$:     MOVL    #15, (R3)                              ; 1033
                               04 000AC         RET                                            ; 1031
      08   BC              05  D0 000AD 10$:    MOVL    #5, @TYPE                              ; 1034
           63              22  D0 000B1         MOVL    #34, (R3)                              ; 1034
                               04 000B4         RET                                            ; 1031
           05        06    A2  91 000B5 11$:    CMPB    6(R2), #5                              ; 1034
                           1E  12 000B9         BNEQ    12$                                    ;
      08   BC              01  D0 000BB         MOVL    #1, @TYPE                              ; 1034
                           5E  DD 000BF         PUSHL   SP                                     ; 1034
                     08    AE  9F 000C1         PUSHAB  PICTVAL                                ;
                     10    AE  9F 000C4         PUSHAB  PICTPTR                                ;
                     18    AE  9F 000C7         PUSHAB  LANGCODE                               ;
                     08    A2  DD 000CA         PUSHL   8(R2)                                  ;
00000000G  00              05  FB 000CD         CALLS   #5, DBG$STA_TYP_PICT                   ;
```

```
                63        04    AE  D0 000D4          MOVL    PICTVAL, (R3)                   ; 1035
                                    04 000D8          RET                                     ; 1034
             08 BC              0A  D0 000D9 12$:     MOVL    #10, @TYPE                      ; 1035
                                27  11 000DD          BRB     17$                             ; 1035
             08 BC              0B  D0 000DF 13$:     MOVL    #11, @TYPE                      ; 1036
                1F        10    AC  E9 000E3          BLBC    SRC_FLAG, 17$                   ; 1036
                50        18    A2  D0 000E7          MOVL    24(R2), VALPTR                  ; 1036
                63              60  3C 000EB          MOVZWL  (VALPTR), (R3)                  ; 1036
                                    04 000EE          RET                                     ; 1036
             08 BC              04  D0 000EF 14$:     MOVL    #4, @TYPE                       ; 1037
             01 A4        1C    A2  8E 000F3          MNEGB   28(R2), 1(SIZE_BYTE)            ; 1038
                64              65  90 000F8          MOVB    (R5), (SIZE_BYTE)               ; 1038
                                    04 000FB          RET                                     ; 1031
             08 BC              0E  D0 000FC 15$:     MOVL    #14, @TYPE                      ; 1038
                                04  11 00100          BRB     17$                             ; 1038
             08 BC              0C  D0 00102 16$:     MOVL    #12, @TYPE                      ; 1039
                63              65  3C 00106 17$:     MOVZWL  (R5), (R3)                      ; 1039
                                    04 00109          RET                                     ; 1040

; Routine Size:  266 bytes,    Routine Base:  DBG$CODE + 27D6
```

```
;10320      10403  1        ROUTINE MODIFY_PLI_TARGET_TYPE(OPERATOR, LEFT_TYPE, RIGHT_TYPE,
;10321      10404  1                NEW_LEFT_TYPE, NEW_RIGHT_TYPE, NEW_ROOT_INDEX,
;10322      10405  1                HIER_TBL, HIER_TBL_SIZE,
;10323      10406  1                INCOMP_TBL, INCOMP_TBL_SIZE,
;10324      10407  1                ROUT_TBL, ROUT_TBL_SIZE) : NOVALUE =
;10325      10408  1
;10326      10409  1        !
;10327      10410  1        ! FUNCTION
;10328      10411  1        !       This routine is used to adjust the target data type based on
;10329      10412  1        !       the type of the other operand for arithmetic and relational
;10330      10413  1        !       operators in PLI.
;10331      10414  1        !
;10332      10415  1        !       PLI has very unique type conversion rule, which differs from
;10333      10416  1        !       the other language, graph hierarchy scheme stand alone can
;10334      10417  1        !       not satisfy the needs without modification.  This routine
;10335      10418  1        !       is a separate routine to serve the needs.
;10336      10419  1        !
;10337      10420  1        !       After the walking the graph from previous FIND_JOIN, we know
;10338      10421  1        !       the data types we work with are valid.  First we check the
;10339      10422  1        !       operator, if the operator is not one of the arithmetic/relational
;10340      10423  1        !       operators, we know the result from FIND_JOIN is ok.  Second,
;10341      10424  1        !       we check to see if one of the operands has data type V, VU, T, or VT,
;10342      10425  1        !       if not, we know the result from FIND_JOIN is ok.  Third, we modify
;10343      10426  1        !       the target based on the following tables:
;10344      10427  1        !
;10345      10428  1        !       +---------+---------+---------+---------+---------+---------+
;10346      10429  1        !       !         ! B,W,L   !    P    ! F,D,G,H ! V,VU    ! T,VT    !
;10347      10430  1        !       +---------+---------+---------+---------+---------+---------+
;10348      10431  1        !       ! B,W,L   ! B,W,L   ! B,W,L   ! F,D,G,H !    L    !    L    !
;10349      10432  1        !       +---------+---------+---------+---------+---------+---------+
;10350      10433  1        !       !    P    !    L    !    P    ! F,D,G,H !    L    !    P    !
;10351      10434  1        !       +---------+---------+---------+---------+---------+---------+
;10352      10435  1        !       ! F,D,G,H ! F,D,G,H ! F,D,G,H ! F,D,G,H ! F,D,G,H ! F,D,G,H !
;10353      10436  1        !       +---------+---------+---------+---------+---------+---------+
;10354      10437  1        !       ! V,VU    !    L    !    L    ! F,D,G,H !   *1    !    L    !
;10355      10438  1        !       +---------+---------+---------+---------+---------+---------+
;10356      10439  1        !       ! T,VT    !    L    !    P    ! F,D,G,H !    L    !   *2    !
;10357      10440  1        !       +---------+---------+---------+---------+---------+---------+
;10358      10441  1        !
;10359      10442  1        !       *1 - Relational operator: no change for data types.
;10360      10443  1        !            Arithmetic operator: L
;10361      10444  1        !
;10362      10445  1        !       *2 - Relational operator: no change for data types.
;10363      10446  1        !            Arithmetic operator: P
;10364      10447  1        !
;10365      10448  1        !       Note: Unary operator does not have the problem, so this routine
;10366      10449  1        !       only works with binary operator.
;10367      10450  1        !
;10368      10451  1        !       Note: FIXED_BIN, FIXED_DEC, FLOAT_DEC, CHAR, CHAR_VAR, BIT, and
;10369      10452  1        !       BIT_ALGNED are the data types we know in PLI.  (there are others
;10370      10453  1        !       ie., FLOAT_BIN, BIT_VAR.).
;10371      10454  1        !
;10372      10455  1        ! INPUTS
;10373      10456  1        !       OPERATOR        - Token Operator.
;10374      10457  1        !       LEFT_TYPE       - Original left data type.
;10375      10458  1        !       RIGHT_TYPE      - Original right data type.
;10376      10459  1        !       NEW_LEFT_TYPE   - The address of the new left data type.
```

```
;10377      10460  1          !         EW_RIGHT_TYPE  - The address of the new right data type.
;10378      10461  1          !         EW_ROUT_INDEX  - The address of the new corresponding routine index.
;10379      10462  1          !         IER_TBL        - Hier. table.
;10380      10463  1          !          R_TBL_SIZE    - Hier. table size.
;10381      10464  1          !           MP_TBL       - Incomp. table.
;10382      10465  1          !           MP_TBL_SIZE  - Incomp. table size.
;10383      10466  1          !           T_TBL        - Rout. table.
;10384      10467  1          !           T_TBL_SIZE   - Rout. table size.
;10385      10468  1          !
;10386      10469  1          ! OUTPUTS
;10387      10470  1          !       If there is modification NEW targer data type is returned along
;10388      10471  1          !       with corresponding routine index.
;10389      10472  1          !
;10390      10473  1
;10391      10474  2              BEGIN
;10392      10475  2
;10393      10476  2              MAP
;10394      10477  2                  OPERATOR: REF TOKEN$ENTRY,
;10395      10478  2                  NEW_LEFT_TYPE: REF VECTOR[,LONG],
;10396      10479  2                  NEW_RIGHT_TYPE: REF VECTOR[,LONG],
;10397      10480  2                  NEW_ROUT_INDEX: REF VECTOR[,LONG];
;10398      10481  2
;10399      10482  2              LOCAL
;10400      10483  2                  LEFT,
;10401      10484  2                  RIGHT,
;10402      10485  2                  GROUP1,
;10403      10486  2                  GROUP2;
;10404      10487  2
;10405      10488  2
;10406      10489  2              ! We only worry about this set of tokens.  If the token is not in the set
;10407      10490  2              ! simply returns.
;10408      10491  2              !
;10409      10492  2              GROUP1 = FALSE;
;10410      10493  2              GROUP2 = FALSE;
;10411      10494  2              IF  .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_ADD OR
;10412      10495  2                  .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_SUBTRACT OR
;10413      10496  2                  .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_MULTIPLY OR
;10414      10497  2                  .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_DIVIDE
;10415      10498  2              THEN
;10416      10499  2                  GROUP1 = TRUE
;10417      10500  2
;10418      10501  2              ELSE
;10419      10502  3                  BEGIN
;10420      10503  3                  IF  .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_EQUAL OR
;10421      10504  3                      .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_NOT_EQUAL OR
;10422      10505  3                      .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_GTR_THAN OR
;10423      10506  3                      .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_GTR_EQUAL OR
;10424      10507  3                      .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_LSS_THAN OR
;10425      10508  3                      .OPERATOR[TOKEN$W_CODE] EQL TOKEN$K_LSS_EQUAL
;10426      10509  3                  THEN
;10427      10510  3                      GROUP2 = TRUE
;10428      10511  3
;10429      10512  3                  ELSE
;10430      10513  3                      RETURN 0;
;10431      10514  2                  END;
;10432      10515  2
;10433      10516  2
```

```
;10434      10517  2          ! Smash VT to T, VU to V.  As far as this routine's concern, they are
;10435      10518  2          ! treated the same.  In this way, we reduce some data types to handle.
;10436      10519  2          !
;10437      10520  2          LEFT = .LEFT_TYPE;
;10438      10521  2          RIGHT = .RIGHT_TYPE;
;10439      10522  2          IF .LEFT EQL DSC$K_DTYPE_VT THEN LEFT = DSC$K_DTYPE_T;
;10440      10523  2          IF .LEFT EQL DSC$K_DTYPE_VU THEN LEFT = DSC$K_DTYPE_V;
;10441      10524  2          IF .RIGHT EQL DSC$K_DTYPE_VT THEN RIGHT = DSC$K_DTYPE_T;
;10442      10525  2          IF .RIGHT EQ  DSC$K_DTYPE_VU THEN RIGHT = DSC$K_DTYPE_V;
;10443      10526  2
;10444      10527  2
;10445      10528  2          ! We only worry about this set of data types.  If one of the operand
;10446      10529  2          ! is V or T then we need to modify the target.  Otherwise, simply
;10447      10530  2          ! returns.
;10448      10531  2          !
;10449      10532  2          IF .LEFT_TYPE NEQ DSC$K_DTYPE_V AND
;10450      10533  2             .LEFT_TYPE NEQ DSC$K_DTYPE_T AND
;10451      10534  2             .RIGHT_TYPE NEQ DSC$K_DTYPE_V AND
;10452      10535  2             .RIGHT_TYPE NEQ DSC$K_DTYPE_T
;10453      10536  2          THEN
;10454      10537  2              RETURN 0;
;10455      10538  2
;10456      10539  2
;10457      10540  2          ! Filter out V op V, T op T, V op T, T op V cases.
;10458      10541  2          !
;10459      10542  2          IF (.LEFT EQL DSC$K_DTYPE_V AND .RIGHT EQL DSC$K_DTYPE_V) OR
;10460      10543  2             (.LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_T) OR
;10461      10544  2             (.LEFT EQL DSC$K_DTYPE_V AND .RIGHT EQL DSC$K_DTYPE_T) OR
;10462      10545  3             (.LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_V)
;10463      10546  2          THEN
;10464      10547  3              BEGIN
;10465      10548  3
;10466      10549  3
;10467      10550  3              ! If this is relational operator, then we all set.
;10468      10551  3              !
;10469      10552  3              IF .GROUP2
;10470      10553  3              THEN
;10471      10554  3                  RETURN 0
;10472      10555  3
;10473      10556  3
;10474      10557  3              ! If this is arithmetic operator, modify the target.
;10475      10558  3              !
;10476      10559  3              ELSE
;10477      10560  4                  BEGIN
;10478      10561  4                  IF .LEFT EQL DSC$K_DTYPE_T AND .RIGHT EQL DSC$K_DTYPE_T
;10479      10562  4                  THEN
;10480      10563  5                      BEGIN
;10481      10564  5                      NEW_LEFT_TYPE[0] = DSC$K_DTYPE_P;
;10482      10565  5                      NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_P;
;10483      10566  5                      END
;10484      10567  5
;10485      10568  4                  ELSE
;10486      10569  5                      BEGIN
;10487      10570  5                      NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
;10488      10571  5                      NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_L;
;10489      10572  4                      END;
;10490      10573  4
```

```
;10491          10574  4
;10492          10575  4                    ! Get the corresponding routine table index.
;10493          10576  4                    !
;10494          10577  4                    IF .MAX_DEPTH EQL FIND_JOIN(
;10495          10578  4                            .NEW_LEFT_TYPE[0], .NEW_RIGHT_TYPE[0],
;10496          10579  4                            NEW_LEFT_TYPE[0], NEW_RIGHT_TYPE[0],
;10497          10580  4                            NEW_ROUT_INDEX[0],
;10498          10581  4                            0, .MAX_DEPTH,
;10499          10582  4                            .HIER_TBL, .HIER_TBL_SIZE,
;10500          10583  4                            .INCOMP_TBL, .INCOMP_TBL_SIZE,
;10501          10584  4                            .ROUT_TBL, .ROUT_TBL_SIZE)
;10502          10585  4                    THEN
;10503          10586  4                        SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;10504          10587  4
;10505          10588  4                    RETURN 0;
;10506          10589  3                    END;
;10507          10590  2                END;
;10508          10591  2
;10509          10592  2
;10510          10593  2            ! Now we have V op (B, W, L, P, F, D, G, H) or T op (B, W, ...) cases.
;10511          10594  2            !
;10512          10595  2            INCR I FROM 0 TO 1 DO
;10513          10596  3                BEGIN
;10514          10597  3                LOCAL
;10515          10598  3                    TYPE, OTHER_TYPE;
;10516          10599  3
;10517          10600  3                IF .I EQL 0
;10518          10601  3                THEN
;10519          10602  4                    BEGIN
;10520          10603  4                    TYPE = .LEFT;
;10521          10604  4                    OTHER_TYPE = .RIGHT;
;10522          10605  4                    END
;10523          10606  4
;10524          10607  3                ELSE
;10525          10608  4                    BEGIN
;10526          10609  4                    TYPE = .RIGHT;
;10527          10610  4                    OTHER_TYPE = .LEFT;
;10528          10611  3                    END;
;10529          10612  3
;10530          10613  3                SELECTONE .TYPE OF
;10531          10614  3                    SET
;10532          10615  3                    [DSC$K_DTYPE_L]:
;10533          10616  4                        BEGIN
;10534          10617  4                        NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
;10535          10618  4                        NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_L;
;10536          10619  3                        END;
;10537          10620  3
;10538          10621  3                    [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_P]:
;10539          10622  4                        BEGIN
;10540          10623  4
;10541          10624  4            ! Apparently, PLI documentation and PLI compiler have a different point
;10542          10625  4            ! of view to decide which target to assign.  This is the reason we
;10543          10626  4            ! commented out this piece.  We take PLI compiler's result.
;10544          10627  4            !                   IF .OTHER_TYPE EQL DSC$K_DTYPE_V
;10545          10628  4            !                   THEN
;10546          10629  4            !                       BEGIN
;10547          10630  4            !                       NEW_LEFT_TYPE[0] = DSC$K_DTYPE_L;
```

```
;10548      10631  4          !                   NEW_RIGHT_TYPE[0] = DSC$K_DTYPE_L;
;10549      10632  4          !                   END
;10550      10633  4          !
;10551      10634  4          !               ELSE
;10552      10635  4          !                   BEGIN
;10553      10636  4                              NEW_LEFT_TYPE[0] = DSC$K_DTYPE_P;
;10554      10637  4                              NEW_RIGHT_TYPE[0] = DSC$R_DTYPE_P;
;10555      10638  4          !                   END;
;10556      10639  3                          END;
;10557      10640  3
;10558      10641  3
;10559      10642  3                          ! This is not really needed.  Because the graph can take care
;10560      10643  3                          ! of these nicely.  I left this in for documentation purpose.
;10561      10644  3                          !
;10562      10645  3                          [DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H]:
;10563      10646  4                              BEGIN
;10564      10647  4                              NEW_LEFT_TYPE[0] = .TYPE;
;10565      10648  4                              NEW_RIGHT_TYPE[0] = .TYPE;
;10566      10649  3                              END;
;10567      10650  3
;10568      10651  3                          [DSC$K_DTYPE_V, DSC$K_DTYPE_T]:
;10569      10652  3                              0;
;10570      10653  3
;10571      10654  3                          TES;
;10572      10655  3
;10573      10656  2                      END;
;10574      10657  2
;10575      10658  2                  IF .MAX_DEPTH EQL FIND_JOIN(
;10576      10659  2                            .NEW_LEFT_TYPE[0], .NEW_RIGHT_TYPE[0],
;10577      10660  2                            NEW_LEFT_TYPE[0], NEW_RIGHT_TYPE[0],
;10578      10661  2                            NEW_ROUT_INDEX[0],
;10579      10662  2                            0, .MAX_DEPTH,
;10580      10663  2                            .HIER_TBL, .HIER_TBL_SIZE,
;10581      10664  2                            .INCOMP_TBL, .INCOMP_TBL_SIZE,
;10582      10665  2                            .ROUT_TBL, .ROUT_TBL_SIZE)
;10583      10666  2                  THEN
;10584      10667  2                      SIGNAL(DBG$_OPNOTALLOW, 1, OPERATOR[TOKEN$B_OPLEN]);
;10585      10668  2
;10586      10669  2                  RETURN 0;
;10587      10670  1                  END;


                                                003C 00000 MODIFY_PLI_TARGET_TYPE:
                                                           .WORD   Save R2,R3,R4,R5                    ; 1040
                                    55 00000000'  EF  9E 00002     MOVAB   MAX_DEPTH, R5               ;
                                                  51  7C 00009     CLRQ    GROUP1                      ; 1049
                50        04   BC                10  EF 0000B     EXTZV   #16, #16, @OPERATOR, R0     ; 1049
                                                  06  50 B1 00011 CMPW    R0, #6                      ;
                                                      0F 13 00014 BEQL    1$                          ;
                                                  07  50 B1 00016 CMPW    R0, #7                      ; 1049
                                                      0A 13 00019 BEQL    1$                          ;
                                                  08  50 B1 0001B CMPW    R0, #8                      ; 1049
                                                      05 13 0001E BEQL    1$                          ;
                                                  09  50 B1 00020 CMPW    R0, #9                      ; 1049
```

```
                05  12 00023          BNEQ    2$
51              01  D0 00025 1$:      MOVL    #1, GROUP1                      ; 1049
                21  11 00028          BRB     4$
0D              50  B1 0002A 2$:      CMPW    R0, #13                         ; 1050
                19  13 0002D          BEQL    3$
0E              50  B1 0002F          CMPW    R0, #14                         ; 1050
                14  13 00032          BEQL    3$
0F              50  B1 00034          CMPW    R0, #15                         ; 1050
                0F  13 00037          BEQL    3$
11              50  B1 00039          CMPW    R0, #17                         ; 1050
                0A  13 0003C          BEQL    3$
13              50  B1 0003E          CMPW    R0, #19                         ; 1050
                05  13 00041          BEQL    3$
15              50  B1 00043          CMPW    R0, #21                         ; 1050
                41  12 00046          BNEQ    9$
52              01  D0 00048 3$:      MOVL    #1, GROUP2                      ; 1051
54      08      AC  D0 0004B 4$:      MOVL    LEFT_TYPE, LEFT                 ; 1052
53      0C      AC  D0 0004F          MOVL    RIGHT_TYPE, RIGHT               ; 1052
25              54  D1 00053          CMPL    LEFT, #37                       ; 1052
                03  12 00056          BNEQ    5$
54              0F  D0 00058          MOVL    #14, LEFT
22              54  D1 0005B 5$:      CMPL    LEFT, #34                       ; 1052
                03  12 0005E          BNEQ    6$
54              01  D0 00060          MOVL    #1, LEFT
25              53  D1 00063 6$:      CMPL    RIGHT, #37                      ; 1052
                03  12 00066          BNEQ    7$
53              0E  D0 00068          MOVL    #14, RIGHT
22              53  D1 0006B 7$:      CMPL    RIGHT, #34                      ; 1052
                03  12 0006E          BNEQ    8$
53              01  D0 00070          MOVL    #1, RIGHT
01      08      AC  D1 00073 8$:      CMPL    LEFT_TYPE, #1                   ; 1053
                13  13 00077          BEQL    10$
0E      08      AC  D1 00079          CMPL    LEFT_TYPE, #14                  ; 1053
                0D  13 0007D          BEQL    10$
01      0C      AC  D1 0007F          CMPL    RIGHT_TYPE, #1                  ; 1053
                07  13 00083          BEQL    10$
0E      0C      AC  D1 00085          CMPL    RIGHT_TYPE, #14                 ; 1053
                01  13 00089 9$:      BEQL    10$
                    04 0008B          RET
                50  D4 0008C 10$:     CLRL    R0                              ; 1054
01              54  D1 0008E          CMPL    LEFT, #1
                07  12 00091          BNEQ    11$
                50  D6 00093          INCL    R0
01              53  D1 00095          CMPL    RIGHT, #1
                1C  13 00098          BEQL    14$
0E              54  D1 0009A 11$:     CMPL    LEFT, #14                       ; 1054
                05  12 0009D          BNEQ    12$
0E              53  D1 0009F          CMPL    RIGHT, #14
                12  13 000A2          BEQL    14$
05              50  E9 000A4 12$:     BLBC    R0, 13$                         ; 1054
0E              53  D1 000A7          CMPL    RIGHT, #14
                0A  13 000AA          BEQL    14$
0E              54  D1 000AC 13$:     CMPL    LEFT, #14                       ; 1054
                27  12 000AF          BNEQ    17$
01              53  D1 000B1          CMPL    RIGHT, #1
                22  12 000B4          BNEQ    17$
01              52  E9 000B6 14$:     BLBC    GROUP2, 15$                     ; 1055
```

```
                    04 000B9                RET                                             ;
         0E      54 D1 000BA 15$:           CMPL    LEFT, #14                               ; 1056
                 0F 12 000BD                BNEQ    16$                                     ;
         0E      53 D1 000BF                CMPL    RIGHT, #14                              ;
                 0A 12 000C2                BNEQ    16$                                     ;
   10    BC      15 D0 000C4                MOVL    #21, @NEW_LEFT_TYPE                     ; 1056
   14    BC      15 D0 000C8                MOVL    #21, @NEW_RIGHT_TYPE                    ; 1056
                 61 11 000CC                BRB     28$                                     ; 1056
   10    BC      08 D0 000CE 16$:           MOVL    #8, @NEW_LEFT_TYPE                      ; 1057
   14    BC      08 D0 000D2                MOVL    #8, @NEW_RIGHT_TYPE                     ; 1057
                 57 11 000D6                BRB     28$                                     ; 1058
                 52 D4 000D8 17$:           CLRL    I                                       ; 1060
                 08 12 000DA 18$:           BNEQ    19$                                     ; 1060
         50      54 D0 000DC                MOVL    LEFT, TYPE                              ; 1060
         51      53 D0 000DF                MOVL    RIGHT, OTHER_TYPE                       ; 1060
                 03 11 000E2                BRB     20$                                     ; 1060
         50      53 7D 000E4 19$:           MOVQ    RIGHT, TYPE                             ; 1060
         08      50 D1 000E7 20$:           CMPL    TYPE, #8                                ; 1061
                 0A 12 000EA                BNEQ    21$                                     ;
   10    BC      08 D0 000EC                MOVL    #8, @NEW_LEFT_TYPE                      ; 1061
   14    BC      08 D0 000F0                MOVL    #8, @NEW_RIGHT_TYPE                     ; 1061
                 35 11 000F4                BRB     27$                                     ; 1061
         06      50 D1 000F6 21$:           CMPL    TYPE, #6                                ; 1062
                 05 19 000F9                BLSS    22$                                     ;
         07      50 D1 000FB                CMPL    TYPE, #7                                ;
                 05 15 000FE                BLEQ    23$                                     ;
         15      50 D1 00100 22$:           CMPL    TYPE, #21                               ;
                 0A 12 00103                BNEQ    24$                                     ;
   10    BC      15 D0 00105 23$:           MOVL    #21, @NEW_LEFT_TYPE                     ; 1063
   14    BC      15 D0 00109                MOVL    #21, @NEW_RIGHT_TYPE                    ; 1063
                 1C 11 0010D                BRB     27$                                     ; 1061
         0A      50 D1 0010F 24$:           CMPL    TYPE, #10                               ; 1064
                 05 19 00112                BLSS    25$                                     ;
         0B      50 D1 00114                CMPL    TYPE, #11                               ;
                 0A 15 00117                BLEQ    26$                                     ;
         1B      50 D1 00119 25$:           CMPL    TYPE, #27                               ;
                 0D 19 0011C                BLSS    27$                                     ;
         1C      50 D1 0011E                CMPL    TYPE, #28                               ;
                 08 14 00121                BGTR    27$                                     ;
      10 BC      50 D0 00123 26$:           MOVL    TYPE, @NEW_LEFT_TYPE                    ; 1064
      14 BC      50 D0 00127                MOVL    TYPE, @NEW_RIGHT_TYPE                   ; 1064
AB               52 01 F3 0012B 27$:        AOBLEQ  #1, I, 18$                              ; 1059
         7E      2C AC 7D 0012F 28$:        MOVQ    ROUT_TBL, -(SP)                         ; 1066
         7E      24 AC 7D 00133             MOVQ    INCOMP_TBL, -(SP)                       ; 1066
         7E      1C AC 7D 00137             MOVQ    HIER_TBL, -(SP)                         ; 1066
                    65 DD 0013B             PUSHL   MAX_DEPTH                               ; 1066
                    7E D4 0013D             CLRL    -(SP)                                   ; 1066
         7E      14 AC 7D 0013F             MOVQ    NEW_RIGHT_TYPE, -(SP)                   ;
                 10 AC DD 00143             PUSHL   NEW_LEFT_TYPE                           ;
                 14 BC DD 00146             PUSHL   @NEW_RIGHT_TYPE                         ;
                 10 BC DD 00149             PUSHL   @NEW_LEFT_TYPE                          ;
      F678       CF    0D FB 0014C         CALLS   #13, FIND_JOIN                          ;
                 50    65 D1 00151         CMPL    MAX_DEPTH, R0                           ;
                       14 12 00154         BNEQ    29$                                     ;
7E       04 AC          0C C1 00156         ADDL3   #12, OPERATOR, -(SP)                    ; 1066
                       01 DD 0015B         PUSHL   #1                                      ;
                000289CA 8F DD 0015D       PUSHL   #166346                                 ;
```

```
                          00000000G  00          03  FB 00163          CALLS   #3, LIB$SIGNAL                          ;
                                                    04 0016A 29$:     RET                                            ; 1067

; Routine Size:  363 bytes,    Routine Base:  DBG$CODE + 28E0

;10588          10671  1
```

```
;10590      10672  1     ROUTINE PLI_TYPE_CONV (VALUE1, VALUE2) =
;10591      10673  1     !
;10592      10674  1     ! FUNCTION
;10593      10675  1     !       Performs PLI language-specific type conversion on the given
;10594      10676  1     !       descriptors.
;10595      10677  1     !
;10596      10678  1     ! INPUTS
;10597      10679  1     !       VALUE1          - DEBUG value descriptor for the source
;10598      10680  1     !       VALUE2          - DEBUG value descriptor for the target
;10599      10681  1     !
;10600      10682  1     ! OUTPUTS
;10601      10683  1     !       A pointer to the result descriptor is returned.
;10602      10684  1     !
;10603      10685  2         BEGIN
;10604      10686  2
;10605      10687  2         ENABLE
;10606      10688  2             PLI_HANDLER;
;10607      10689  2
;10608      10690  2         MAP
;10609      10691  2             VALUE1 : REF DBG$VALDESC,
;10610      10692  2             VALUE2 : REF DBG$VALDESC;
;10611      10693  2
;10612      10694  2         LOCAL
;10613      10695  2             SRC_ADDR,   DST_ADDR,
;10614      10696  2             SRC_TYPE,   DST_TYPE,
;10615      10697  2             SRC_SIZE,   DST_SIZE,
;10616      10698  2             SRC_OFFSET, DST_OFFSET;
;10617      10699  2
;10618      10700  2
;10619      10701  2         ! In PL/I, special conversions are required when going to/from
;10620      10702  2         ! bit-strings.  PL/I bit-strings are stored in reverse order in
;10621      10703  2         ! memory; although they are mapped to type DSC$K_DTYPE_TF, this
;10622      10704  2         ! means that they cannot be converted by DBG$CVT_DX_DX, which
;10623      10705  2         ! treats bit-strings like integers.
;10624      10706  2         !
;10625      10707  2         ! Get the source and destination pointers.
;10626      10708  2         !
;10627      10709  2         SRC_ADDR = .VALUE1[DBG$L_VALUE_POINTER];
;10628      10710  2         DST_ADDR = .VALUE2[DBG$L_VALUE_POINTER];
;10629      10711  2
;10630      10712  2
;10631      10713  2         ! Map the current dtype to PL/I specific types; calculate the
;10632      10714  2         ! size for the type.
;10633      10715  2         !
;10634      10716  2         MAP_PLI_TYPE_SIZE(.VALUE1, SRC_TYPE, SRC_SIZE, TRUE);
;10635      10717  2
;10636      10718  2
;10637      10719  2         ! Get the digit and scale info from picture data type.
;10638      10720  2         !
;10639      10721  2         IF .VALUE1[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_PICT AND
;10640      10722  2            .VALUE2[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_P
;10641      10723  2         THEN
;10642      10724  3             BEGIN
;10643      10725  3             LOCAL
;10644      10726  3                 PICTVAL: REF VECTOR[,BYTE]·
;10645      10727  3
;10646      10728  3             PICTVAL = .SRC_SIZE;
```

```
;10647      10729  3          VALUE2[DBG$B_VALUE_DIGITS] = .PICTVAL[0];
;10648      10730  3
;10649      10731  3
;10650      10732  3          ! note: PLIRTL expects to see positive scale 2 for (12.34),
;10651      10733  3          ! normally we'll have -2 as the scale.
;10652      10734  3          !
;10653      10735  3          VALUE2[DBG$B_VALUE_SCALE] = - .PICTVAL[1];
;10654      10736  2          END;
;10655      10737  2
;10656      10738  2      MAP_PLI_TYPE_SIZE(.VALUE2, DST_TYPE, DST_SIZE, FALSE);
;10657      10739  2
;10658      10740  2
;10659      10741  2      ! Determine the bit offset.  This should be zero for aligned types.
;10660      10742  2      ! The [DBG$L_VALUE_POS] field of the value descriptor could be
;10661      10743  2      ! garbage for packed decimal types, but PLI$CVRT_ANY ignores the
;10662      10744  2      ! offset for any types except those that might be unaligned.
;10663      10745  2      !
;10664      10746  2      SRC_OFFSET = 0;
;10665      10747  2      DST_OFFSET = 0;
;10666      10748  2      IF .VALUE1[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU
;10667      10749  2      THEN
;10668      10750  2          SRC_OFFSET = .VALUE1[DBG$L_VALUE_POS];
;10669      10751  2
;10670      10752  2      IF .VALUE2[DBG$B_VALUE_DTYPE] EQL DSC$K_DTYPE_VU
;10671      10753  2      THEN
;10672      10754  2          DST_OFFSET = .VALUE2[DBG$L_VALUE_POS];
;10673      10755  2
;10674      10756  2
;10675      10757  2      ! Call a PL/I run-time routine to do the conversion.
;10676      10758  2      !
;10677      10759  2      PLI$CVRT_ANY (SRC_ADDR, .SRC_TYPE, .SRC_SIZE, .SRC_OFFSET,
;10678      10760  2                    DST_ADDR, .DST_TYPE, .DST_SIZE, .DST_OFFSET);
;10679      10761  2
;10680      10762  2      RETURN .VALUE2;
;10681      10763  1      END;


                                    000C 00000 PLI_TYPE_CONV:
                                                       .WORD   Save R2,R3                      ;  1067
                5E                      18  C2 00002          SUBL2   #24, SP                 ;
                6D          007C        CF  DE 00005          MOVAL   4$, (FP)                ;  1068
                52            04        AC  7D 0000A          MOVQ    VALUE1, R2              ;  1070
          14    AE            18        A2  D0 0000E          MOVL    24(R2), SRC_ADDR        ;
          10    AE            18        A3  D0 00013          MOVL    24(R3), DST_ADDR        ;  1071
                                        01  DD 00018          PUSHL   #1                      ;  1071
                              04        AE  9F 0001A          PUSHAB  SRC_SIZE                ;
                              0C        AE  9F 0001D          PUSHAB  SRC_TYPE                ;
                                        52  DD 00020          PUSHL   R2                      ;
              FD64      CF              04  FB 00022          CALLS   #4, MAP_PLI_TYPE_SIZE   ;
                        05            06  A2  91 00027          CMPB    6(R2), #5               ;  1072
                                        12  12 0002B          BNEQ    1$                      ;
                        15            16  A3  91 0002D          CMPB    22(R3), #21             ;  1072
                                        0C  12 00031          BNEQ    1$                      ;
                        50                  6E  D0 00033          MOVL    SRC_SIZE, PICTVAL       ;  1072
```

```
                1D   A3              60  90 00036          MOVB    (PICTVAL), 29(R3)                       ; 1072
                1C   A3        01    A0  8E 0003A          MNEGB   1(PICTVAL), 28(R3)                      ; 1073
                                     7E  D4 0003F 1$:      CLRL    -(SP)                                   ; 1073
                                  0C AE  9F 00041          PUSHAB  DST_SIZE
                                  14 AE  9F 00044          PUSHAB  DST_TYPE
                                     53  DD 00047          PUSHL   R3
              FD3D  CF                 04  FB 00049          CALLS   #4, MAP_PLI_TYPE_SIZE
                                     50  7C 0004E          CLRQ    DST_OFFSET                              ; 1074
                      22       16    A2  91 00050          CMPB    22(R2), #34                             ; 1074
                                     04  12 00054          BNEQ    2$
                      51       1C    A2  D0 00056          MOVL    28(R2), SRC_OFFSET                      ; 1075
                      22       16    A3  91 0005A 2$:      CMPB    22(R3), #34                             ; 1075
                                     04  12 0005E          BNEQ    3$
                      50       1C    A3  D0 00060          MOVL    28(R3), DST_OFFSET                      ; 1075
                                     50  DD 00064 3$:      PUSHL   DST_OFFSET                              ; 1076
                                  0C AE  DD 00066          PUSHL   DST_SIZE
                                  14 AE  DD 00069          PUSHL   DST_TYPE
                                  1C AE  9F 0006C          PUSHAB  DST_ADDR                                ; 1075
                                     51  DD 0006F          PUSHL   SRC_OFFSET
                                  14 AE  DD 00071          PUSHL   SRC_SIZE
                                  1C AE  DD 00074          PUSHL   SRC_TYPE
                                  30 AE  9F 00077          PUSHAB  SRC_ADDR
          00000000G   00              08  FB 0007A          CALLS   #8, PLI$CVRT_ANY
                      50              53  D0 00081          MOVL    R3, R0                                  ; 1076
                                         04 00084          RET                                             ; 1076
                                   0000  00085 4$:          .WORD   Save nothing                            ; 1068
                                     7E  D4 00087          CLRL    -(SP)
                                     5E  DD 00089          PUSHL   SP
                      7E       04    AC  7D 0008B          MOVQ    4(AP), -(SP)
                0000V CF              03  FB 0008F          CALLS   #3, PLI_HANDLER
                                         04 00094          RET                                             ;

; Routine Size:  149 bytes,    Routine Base:  DBG$CODE + 2A4B
```

```
;10683          10764  1        ROUTINE PLI_HANDLER (SIG, MECH) =
;10684          10765  1        !
;10685          10766  1        ! FUNCTION
;10686          10767  1        !       This handler catches PL/I specific data type conversion errors, and
;10687          10768  1        !       signals them as Debug errors.  It resignals all other conditions.
;10688          10769  1        !
;10689          10770  1        ! INPUTS
;10690          10771  1        !       SIG  - A counted vector of parameters describing the condition.
;10691          10772  1        !
;10692          10773  1        !       MECH - A counted vector of parameters from CHF.
;10693          10774  1        !
;10694          10775  1        ! OUTPUTS
;10695          10776  1        !
;10696          10777  1        !
;10697          10778  2            BEGIN
;10698          10779  2
;10699          10780  2            MAP
;10700          10781  2                SIG: REF VECTOR[,LONG];
;10701          10782  2
;10702          10783  2            !Translate all numeric exceptions to debug's facility code.
;10703          10784  2            !Also, translate SS$_ROPRAND to DBG$_ROPRANDF.
;10704          10785  2            !
;10705          10786  2            SELECTONE .SIG[1] OF
;10706          10787  2                SET
;10707          10788  2                [SS$_INTOVF]:
;10708          10789  2                    SIGNAL (DBG$_IINTOVF, 1, .DBG$GL_OPCODE_NAME);
;10709          10790  2                [SS$_DECOVF]:
;10710          10791  2                    SIGNAL (DBG$_DECOVF, 1, .DBG$GL_OPCODE_NAME);
;10711          10792  2                [SS$_FLTOVF, SS$_FLTOVF_F]:
;10712          10793  2                    SIGNAL (DBG$_FLTOVF, 1, .DBG$GL_OPCODE_NAME);
;10713          10794  2                [SS$_FLTUND, SS$_FLTUND_F]:
;10714          10795  2                    SIGNAL (DBG$_IFLTUND, 1, .DBG$GL_OPCODE_NAME);
;10715          10796  2                [SS$_ROPRAND]:
;10716          10797  2                    SIGNAL (DBG$_ROPRANDF, 1, .DBG$GL_OPCODE_NAME);
;10717          10798  2                [SS$_UNWIND]:
;10718          10799  2                    RETURN (SS$_RESIGNAL);
;10719          10800  2                [OTHERWISE]:
;10720          10801  2                    SIGNAL (DBG$_PLICVTERR, 1, .DBG$GL_OPCODE_NAME);
;10721          10802  2                TES;
;10722          10803  2
;10723          10804  2            SETUNWIND();
;10724          10805  2            RETURN 0;
;10725          10806  1            END;                                    ! End of PLI_HANDLER


                                000C 00000 PLI_HANDLER:
                                           .WORD   Save R2,R3                                  ; 1076
                53 00000000'    EF  9E 00002       MOVAB   DBG$GL_OPCODE_NAME, R3           ;
                50        04    AC  D0 00009       MOVL    SIG, R0                          ; 1078
                50              04  C0 0000D       ADDL2   #4, R0                           ;
                52              60  D0 00010       MOVL    (R0), R2                         ;
          0000047C    8F        52  D1 00013       CMPL    R2, #1148                        ; 1078
                                0C  12 0001A       BNEQ    1$                               ;
                                63  DD 0001C       PUSHL   DBG$GL_OPCODE_NAME               ; 1078
```

```
                                 01  DD 0001E          PUSHL   #1                                          ;
                      000286A3   8F  DD 00020          PUSHL   #165539                                     ;
                                 7F  11 00026          BRB     9$                                          ;
000004A4   8F                    52  D1 00028 1$:      CMPL    R2, #1188                                   ; 1079
                                 0C  12 0002F          BNEQ    2$                                          ;
                                 63  DD 00031          PUSHL   DBG$GL_OPCODE_NAME                          ; 1079
                                 01  DD 00033          PUSHL   #1                                          ;
                      00028A3A   8F  DD 00035          PUSHL   #166458                                     ;
                                 6A  11 0003B          BRB     9$                                          ;
0000048C   8F                    52  D1 0003D 2$:      CMPL    R2, #1164                                   ; 1079
                                 09  13 00044          BEQL    3$                                          ;
000004B4   8F                    52  D1 00046          CMPL    R2, #1204                                   ;
                                 0C  12 0004D          BNEQ    4$                                          ;
                                 63  DD 0004F 3$:      PUSHL   DBG$GL_OPCODE_NAME                          ; 1079
                                 01  DD 00051          PUSHL   #1                                          ;
                      00028A02   8F  DD 00053          PUSHL   #166402                                     ;
                                 4C  11 00059          BRB     9$                                          ;
0000049C   8F                    52  D1 0005B 4$:      CMPL    R2, #1180                                   ; 1079
                                 09  13 00062          BEQL    5$                                          ;
000004C4   8F                    52  D1 00064          CMPL    R2, #1220                                   ;
                                 0C  12 0006B          BNEQ    6$                                          ;
                                 63  DD 0006D 5$:      PUSHL   DBG$GL_OPCODE_NAME                          ; 1079
                                 01  DD 0006F          PUSHL   #1                                          ;
                      0002869B   8F  DD 00071          PUSHL   #165531                                     ;
                                 2E  11 00077          BRB     9$                                          ;
00000454   8F                    52  D1 00079 6$:      CMPL    R2, #1108                                   ; 1079
                                 0C  12 00080          BNEQ    7$                                          ;
                                 63  DD 00082          PUSHL   DBG$GL_OPCODE_NAME                          ; 1079
                                 01  DD 00084          PUSHL   #1                                          ;
                      00028A0A   8F  DD 00086          PUSHL   #166410                                     ;
                                 19  11 0008C          BRB     9$                                          ;
00000920   8F                    52  D1 0008E 7$:      CMPL    R2, #2336                                   ; 1079
                                 06  12 00095          BNEQ    8$                                          ;
                      50   0918   8F  3C 00097          MOVZWL  #2328, R0                                   ; 1079
                                     04 0009C          RET                                                 ;
                                 63  DD 0009D 8$:      PUSHL   DBG$GL_OPCODE_NAME                          ; 1080
                                 01  DD 0009F          PUSHL   #1                                          ;
                      00028260   8F  DD 000A1          PUSHL   #164448                                     ;
00000000G  00                    03  FB 000A7 9$:      CALLS   #3, LIB$SIGNAL                              ;
                                 7E  7C 000AE          CLRQ    -(SP)                                       ; 1080
00000000G  00                    02  FB 000B0          CALLS   #2, SYS$UNWIND                              ;
                                 50  D4 000B7          CLRL    R0                                          ; 1080
                                     04 000B9          RET                                                 ; 1080

; Routine Size:  186 bytes,    Routine Base:  DBG$CODE + 2AE0
```

```
;10727          10807  1        ROUTINE TYPEID_CHECK_ENUM(TYPEID1, TYPEID2) =
;10728          10808  1        !
;10729          10809  1        ! FUNCTION
;10730          10810  1        !       This routine performs typeid check on the given TYPEIDs for enumeration
;10731          10811  1        !       data types.
;10732          10812  1        !
;10733          10813  1        ! INPUTS
;10734          10814  1        !       TYPEID1 - TYPEID RST entry pointer.
;10735          10815  1        !
;10736          10816  1        !       TYPEID2 - TYPEID RST entry pointer.
;10737          10817  1        !
;10738          10818  1        ! OUTPUTS
;10739          10819  1        !       Returned Status: TRUE or FALSE.
;10740          10820  1        !
;10741          10821  1
;10742          10822  2            BEGIN
;10743          10823  2
;10744          10824  2            MAP
;10745          10825  2                TYPEID1: REF RST$ENTRY,         ! Typeid pointer to RST entry
;10746          10826  2                TYPEID2: REF RST$ENTRY;         ! Typeid pointer to RST entry
;10747          10827  2
;10748          10828  2            LOCAL
;10749          10829  2                STATUS;                         ! Return status
;10750          10830  2
;10751          10831  2
;10752          10832  2            IF .TYPEID1 EQL 0 OR .TYPEID2 EQL 0
;10753          10833  2            THEN
;10754          10834  2                RETURN FALSE;
;10755          10835  2
;10756          10836  2            IF .TYPEID1 EQL .TYPEID2
;10757          10837  2            THEN
;10758          10838  2                STATUS = TRUE
;10759          10839  2
;10760          10840  2            ELSE
;10761          10841  3                BEGIN
;10762          10842  3                IF .TYPEID1[RST$B_FCODE] EQL .TYPEID2[RST$B_FCODE]
;10763          10843  3                THEN
;10764          10844  4                    BEGIN
;10765          10845  4                    IF .TYPEID1[RST$B_FCODE] EQL RST$K_TYPE_ENUM
;10766          10846  4                    THEN
;10767          10847  5                        BEGIN
;10768          10848  5                        IF .TYPEID1[RST$L_DSTPTR] EQL .TYPEID2[RST$L_DSTPTR]
;10769          10849  5                        THEN
;10770          10850  5                            STATUS = TRUE
;10771          10851  5                        ELSE
;10772          10852  5                            STATUS = FALSE;
;10773          10853  5
;10774          10854  5                        END
;10775          10855  5
;10776          10856  4                    ELSE
;10777          10857  4                        STATUS = FALSE;
;10778          10858  4
;10779          10859  4                    END
;10780          10860  4
;10781          10861  3                ELSE
;10782          10862  3                    STATUS = FALSE;
;10783          10863  3
```

```
;10784          10864  2          END;
;10785          10865  2
;10786          10866  2        RETURN .STATUS;
;10787          10867  2
;10788          10868  1        END;


                                        0004 00000 TYPEID_CHECK_ENUM:
                                                   .WORD   Save R2                                  ; 1080
                    51      04     AC   D0 00002          MOVL    TYPEID1, R1                       ; 1083
                                  2D   13 00006          BEQL    4$                                ;
                            08    AC   D5 00008          TSTL    TYPEID2                           ;
                                  28   13 0000B          BEQL    4$                                ;
                    50      08    AC   D0 0000D          MOVL    TYPEID2, R0                       ; 1083
                    50            51   D1 00011          CMPL    R1, R0                            ;
                                  14   13 00014          BEQL    1$                                ;
                18  A0      18    A1   91 00016          CMPB    24(R1), 24(R0)                    ; 1084
                                  12   12 00018          BNEQ    2$                                ;
                    04      18    A1   91 0001D          CMPB    24(R1), #4                        ; 1084
                                  0C   12 00021          BNEQ    2$                                ;
                0C  A0      0C    A1   D1 00023          CMPL    12(R1), 12(R0)                    ; 1084
                                  05   12 00028          BNEQ    2$                                ;
                    52            01   D0 0002A 1$:      MOVL    #1, STATUS                        ; 1085
                                  02   11 0002D          BRB     3$                                ;
                                  52   D4 0002F 2$:      CLRL    STATUS                            ; 1086
                    50            52   D0 00031 3$:      MOVL    STATUS, R0                        ; 1086
                                       04 00034          RET                                       ;
                                  50   D4 00035 4$:      CLRL    R0                                ; 1086
                                       04 00037          RET                                       ;

; Routine Size:  56 bytes,    Routine Base:  DBG$CODE + 2B9A
```

```
;10790      10869  1     ROUTINE TYPEID_CHECK_SET(TYPEID1, TYPEID2, FCODE1, FCODE2, DTYPE1, DTYPE2) =
;10791      10870  1     !
;10792      10871  1     ! FUNCTION
;10793      10872  1     !       This routine performs typeid check on the given TYPEIDs for set
;10794      10873  1     !       data types.
;10795      10874  1     !
;10796      10875  1     ! INPUTS
;10797      10876  1     !       TYPEID1 - TYPEID RST entry pointer.
;10798      10877  1     !
;10799      10878  1     !       TYPEID2 - TYPEID RST entry pointer.
;10800      10879  1     !
;10801      10880  1     !       FCODE1  - Fcode
;10802      10881  1     !
;10803      10882  1     !       FCODE2  - Fcode
;10804      10883  1     !
;10805      10884  1     !       DTYPE1  - Data type
;10806      10885  1     !
;10807      10886  1     !       DTYPE2  - Data type
;10808      10887  1     !
;10809      10888  1     ! OUTPUTS
;10810      10889  1     !       Returned Status: TRUE or FALSE.
;10811      10890  1     !
;10812      10891  1
;10813      10892  2         BEGIN
;10814      10893  2
;10815      10894  2         MAP
;10816      10895  2             TYPEID1: REF RST$ENTRY,          ! Typeid pointer to RST entry
;10817      10896  2             TYPEID2: REF RST$ENTRY;          ! Typeid pointer to RST entry
;10818      10897  2
;10819      10898  2         LOCAL
;10820      10899  2             STATUS;                          ! Return status
;10821      10900  2
;10822      10901  2
;10823      10902  2         IF .TYPEID1 EQL .TYPEID2
;10824      10903  2         THEN
;10825      10904  2             STATUS = TRUE
;10826      10905  2         ELSE
;10827      10906  3             BEGIN
;10828      10907  3             IF .FCODE1 EQL .FCODE2
;10829      10908  3             THEN
;10830      10909  4                 BEGIN
;10831      10910  4                 IF .FCODE1 EQL RST$K_TYPE_ATOMIC
;10832      10911  4                 THEN
;10833      10912  5                     BEGIN
;10834      10913  5                     IF .DTYPE1 EQL .DTYPE2
;10835      10914  5                     THEN
;10836      10915  6                         BEGIN
;10837      10916  6                         IF .DTYPE1 EQL DSC$K_DTYPE_TF OR
;10838      10917  6                            .DTYPE1 EQL DSC$K_DTYPE_L OR
;10839      10918  6                            .DTYPE1 EQL DSC$K_DTYPE_LU OR
;10840      10919  6                            .DTYPE1 EQL DSC$K_DTYPE_T
;10841      10920  6                         THEN
;10842      10921  6                             STATUS = TRUE
;10843      10922  6                         ELSE
;10844      10923  6                             STATUS = FALSE;
;10845      10924  6
;10846      10925  6                         END
```

```
6
5                   ELSE
5                       STATUS = FALSE;
5
5                   END
5
4               ELSE
5                   BEGIN
5                   IF .FCODE1 EQL RST$K_TYPE_ENUM
5                   THEN
6                       BEGIN
6                       IF .TYPEID1[RST$L_DS PTR] EQL .TYPEID2[RST$L_DSTPTR]
6                       THEN
6                           STATUS = TRUE
6                       ELSE
6                           STATUS = FALSE;
6
6                       END
6
5                   ELSE
5                       STATUS = FALSE;
5
4                   END;
4
4               END
4
3           ELSE
3               STATUS = FALSE;
3
2           END;
2
2       RETURN .STATUS;
1       END;
```

```
                               000C 00000 TYPEID_CHECK_SET:
                                                     .WORD   Save R2,R3           ; 1086
       52       04   AC   D0 00002        MOVL    TYPEID1, R2          ; 1090
       51       08   AC   D0 00006        MOVL    TYPEID2, R1          ;
       51            52   D1 0000A        CMPL    R2, R1               ;
                     38   13 0000D        BEQL    3$                   ;
  10   AC       0C   AC   D1 0000F        CMPL    FCODE1, FCODE2       ; 1090
                     36   12 00014        BNEQ    4$                   ;
       02       0C   AC   D1 00016        CMPL    FCODE1, #2           ; 1091
                     1E   12 0001A        BNEQ    1$                   ;
       50       14   AC   D0 0001C        MOVL    DTYPE1, R0           ; 1091
  18   AC            50   D1 00020        CMPL    R0, DTYPE2           ;
                     26   12 00024        BNEQ    4$                   ;
       28            50   D1 00026        CMPL    R0, #40              ; 1091
                     1C   13 00029        BEQL    3$                   ;
       08            50   D1 0002B        CMPL    R0, #8               ; 1091
                     17   13 0002E        BEQL    3$                   ;
       04            50   D1 00030        CMPL    R0, #4               ; 1091
                     12   13 00033        BEQL    3$                   ;
```

```
                        0E              50  D1 00035          CMPL    R0, #14                         ; 1091
                                        0B  11 00038          BRB     2$                              ;
                        04      0C      AC  D1 0003A 1$:      CMPL    FCODE1, #4                      ; 1093
                                        0C  12 0003E          BNEQ    4$                              ;
                    0C  A1      0C      A2  D1 00040          CMPL    12(R2), 12(R1)                  ; 1093
                                        05  12 00045 2$:      BNEQ    4$                              ;
                        53              01  D0 00047 3$:      MOVL    #1, STATUS                      ; 1093
                                        02  11 0004A          BRB     5$                              ;
                                        53  D4 0004C 4$:      CLRL    STATUS                          ; 1095
                        50              53  D0 0004E 5$:      MOVL    STATUS, R0                      ; 1095
                                            04 00051          RET                                     ; 1095

; Routine Size:  82 bytes,    Routine Base:  DBG$CODE + 2BD2

;10880          10959  1
```

```
;10882          10960  1        ROUTINE TYPEID_RANGE_CHECK_ENUM(VAL_DESC, TYPEID) =
;10883          10961  1        !
;10884          10962  1        ! FUNCTION
;10885          10963  1        !       This routine takes given typeid in the value descriptor and
;10886          10964  1        !       performs the value range check for enumeration type.
;10887          10965  1        !
;10888          10966  1        ! INPUTS
;10889          10967  1        !       VAL_DESC        - Pointer to value descriptor.
;10890          10968  1        !
;10891          10969  1        !       TYPEID          - Typeid of the data type.
;10892          10970  1        !
;10893          10971  1        ! OUTPUTS
;10894          10972  1        !       Return True to indicate the value is in the range else
;10895          10973  1        !       return false.
;10896          10974  1        !
;10897          10975  1
;10898          10976  2            BEGIN
;10899          10977  2
;10900          10978  2            MAP
;10901          10979  2                TYPEID: REF RST$ENTRY,
;10902          10980  2                VAL_DESC: REF DBG$VALDESC;
;10903          10981  2
;10904          10982  2            LOCAL
;10905          10983  2                ADR_KIND,                       ! Address kind
;10906          10984  2                ADR_PTRS: VECTOR[3,LONG],       ! Address value
;10907          10985  2                NELTS,                          ! Number of elements
;10908          10986  2                NLTVECPTR: REF VECTOR[,LONG],   ! A vector of RST entries for the elements
;10909          10987  2                SIZE;
;10910          10988  2
;10911          10989  2
;10912          10990  2            IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_ENUM
;10913          10991  2            THEN
;10914          10992  2                $DBG_ERROR('DBGEVALOP\TYPEID_RANGE_CHECK_ENUM');
;10915          10993  2
;10916          10994  2            DBG$STA_TYP_ENUM(.TYPEID, NELTS, NLTVECPTR, SIZE);
;10917          10995  2            INCR I FROM 0 TO .NELTS - 1 DO
;10918          10996  3                BEGIN
;10919          10997  3                DBG$STA_SYMVALUE(.NLTVECPTR[.I], ADR_PTRS, ADR_KIND);
;10920          10998  3                IF .(.ADR_PTRS[0])<.ADR_PTRS[1], .SIZE, 0> EQL
;10921          10999  4                    .(.VAL_DESC[DBG$L_VALUE_POINTER])
;10922          11000  3                THEN
;10923          11001  3                    RETURN TRUE;
;10924          11002  2                END;
;10925          11003  2
;10926          11004  2            RETURN FALSE;
;10927          11005  2
;10928          11006  1            END;


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

45  50  59  54  5C  50  4F  4C  41  56  45  47  42  44  21  061BE P.AMV:  .ASCII  \!DBGEVALOP\<92>\TYPEID_RANGE_CHECK_ENUM\   ;
5F  4B  43  45  48  43  5F  45  47  4E  41  52  5F  44  49  061CD                                                            ;
                                    4D  55  4E  45  061DC                                                                ;
```

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

                                                000C 00000 TYPEID_RANGE_CHECK_ENUM:
                                                           .WORD   Save R2,R3                       ; 1096
                                     5E            1C   C2 00002                SUBL2   #28, SP
                                     52       08   AC   D0 00005                MOVL    TYPEID, R2  ; 1099
                                     04       18   A2   91 00009                CMPB    24(R2), #4
                                                   15   13 0000D                BEQL    1$
                                 00000000'         EF   9F 0000F                PUSHAB  P.AMV       ; 1099
                                                   01   DD 00015                PUSHL   #1
                                 00028362          8F   DD 00017                PUSHL   #164706
                        00000000G   00             03   FB 0001D                CALLS   #3, LIB$SIGNAL
                                                   5E   DD 00024 1$:            PUSHL   SP          ; 1099
                                           08      AE   9F 00026                PUSHAB  NLTVECPTR
                                           10      AE   9F 00029                PUSHAB  NELTS
                                                   52   DD 0002C                PUSHL   R2
                        00000000G   00             04   FB 0002E                CALLS   #4, DBG$STA_TYP_ENUM
                                     53       04   AC   D0 00035                MOVL    VAL_DESC, R3; 1099
                                     52            01   CE 00039                MNEGL   #1, I
                                                   22   11 0003C                BRB     3$
                                           0C      AE   9F 0003E 2$:            PUSHAB  ADR_KIND    ; 1099
                                           14      AE   9F 00041                PUSHAB  ADR_PTRS
                                           0C BE42 DD   00044                   PUSHL   @NLTVECPTR[I]
                        00000000G   00             03   FB 00048                CALLS   #3, DBG$STA_SYMVALUE
          50       10   BE          6E       14   AE   EF 0004F                EXTZV   ADR_PTRS+4, SIZE, @ADR_PTRS, R0  ; 1099
                  18   B3                        50   D1 00056                CMPL    R0, @24(R3) ; 1099
                                                   04   12 0005A                BNEQ    3$
                                     50            01   D0 0005C                MOVL    #1, R0      ; 1100
                                                   04   0005F                   RET
                        D9           52       08   AE   F2 00060 3$:           AOBLSS  NELTS, I, 2$; 1099
                                                   50   D4 00065                CLRL    R0          ; 1100
                                                   04   00067                   RET                 ; 1100

; Routine Size:  104 bytes,    Routine Base:  DBG$CODE + 2C24
```

```
;10930      11007  1  ROUTINE TYPEID_RANGE_CHECK_SUBRNG(VAL_DESC, TYPEID) =
;10931      11008  1  !
;10932      11009  1  ! FUNCTION
;10933      11010  1  !       This routine takes given typeid in the value descriptor and
;10934      11011  1  !       performs the value range check for subrange type.
;10935      11012  1  !
;10936      11013  1  ! INPUTS
;10937      11014  1  !       VAL_DESC        - Pointer to value descriptor.
;10938      11015  1  !
;10939      11016  1  !       TYPEID          - Typeid of the data type.  This is not
;10940      11017  1  !                         necessary from the VAL_DESC directly.
;10941      11018  1  !
;10942      11019  1  ! OUTPUTS
;10943      11020  1  !       Return True to indicate the value is in the range else
;10944      11021  1  !       return false.
;10945      11022  1  !
;10946      11023  1
;10947      11024  2      BEGIN
;10948      11025  2
;10949      11026  2      MAP
;10950      11027  2          VAL_DESC: REF DBG$VALDESC,
;10951      11028  2          TYPEID: REF RST$ENTRY;
;10952      11029  2
;10953      11030  2      LOCAL
;10954      11031  2          LENGTH,                         ! Length
;10955      11032  2          HIGHPTR,                        ! High range value
;10956      11033  2          LOWPTR,                         ! Low range value
;10957      11034  2          PARENT_TYPE,                    ! Parent typeid
;10958      11035  2          SETVAL: VECTOR[8, LONG],
;10959      11036  2          SETVALUE: REF BITVECTOR[],
;10960      11037  2          SET_VAL_DESC: REF VECTOR[, LONG],
;10961      11038  2          SIZE;
;10962      11039  2
;10963      11040  2      IF .TYPEID[RST$B_FCODE] NEQ RST$K_TYPE_SUBRNG
;10964      11041  2      THEN
;10965      11042  2          $DBG_ERROR('DBGEVALOP\TYPEID_RANGE_CHECK_SUBRNG');
;10966      11043  2
;10967      11044  2      DBG$STA_TYP_SUBRNG(.TYPEID, PARENT_TYPE, LOWPTR, HIGHPTR, SIZE);
;10968      11045  2      IF .VAL_DESC[DBG$B_DHDR_FCODE] EQL RST$K_TYPE_SET
;10969      11046  2      THEN
;10970      11047  3          BEGIN
;10971      11048  3          LENGTH = (.VAL_DESC[DBG$W_VALUE_LENGTH] - 1) / 4 + 1;
;10972      11049  3          INCR I FROM 0 TO .LENGTH - 1 DO
;10973      11050  3              SETVAL[.I] = %X'FFFFFFFF';
;10974      11051  3          SETVALUE = SETVAL[0];
;10975      11052  3          INCR I FROM ..LOWPTR TO ..HIGHPTR DO
;10976      11053  3              SETVALUE[.I] = 0;
;10977      11054  3          SET_VAL_DESC = .VAL_DESC[DBG$L_VALUE_POINTER];
;10978      11055  3          INCR I FROM 0 TO .LENGTH - 1 DO
;10979      11056  4              BEGIN
;10980      11057  4              SETVAL[.I] = .SET_VAL_DESC[.I] AND .SETVAL[.I];
;10981      11058  4              IF .SETVAL[.I] NEQ 0 THEN RETURN FALSE;
;10982      11059  3              END;
;10983      11060  3
;10984      11061  3          RETURN TRUE;
;10985      11062  3          END
;10986      11063  3
```

```
;10987          11064  2          ELSE
;10988          11065  3              BEGIN
;10989          11066  3              IF (..HIGHPTR GEQ ..VAL_DESC[DBG$L_VALUE_POINTER]) AND
;10990          11067  4                 (..LOWPTR LEQ ..VAL_DESC[DBG$L_VALUE_POINTER])
;10991          11068  3              THEN
;10992          11069  3                  RETURN TRUE
;10993          11070  3              ELSE
;10994          11071  3                  RETURN FALSE;
;10995          11072  2              END;
;10996          11073  2
;10997          11074  1          END;


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

45 50 59 54 5C 50 4F 4C 41 56 45 47 42 44 23  061E0 P.AMW:  .ASCII  \#DBGEVALOP\<92>\TYPEID_RANGE_CHECK_SUBR\   ;
5F 4B 43 45 48 43 5F 45 47 4E 41 52 5F 44 49  061EF
                           52 42 55 53  061FE
                              47 4E  06202              .ASCII  \NG\                                            ;


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

                                    000C 00000 TYPEID_RANGE_CHECK_SUBRNG:
                                               .WORD   Save R2,R3                                               ; 1100
                                5E      30  C2 00002       SUBL2   #48, SP                                      ;
                                52  08  AC  D0 00005       MOVL    TYPEID, R2                                   ; 1104
                                09  18  A2  91 00009       CMPB    24(R2), #9                                   ;
                                          15  13 0000D       BEQL    1$                                       ;
                            00000000'     EF  9F 00010       PUSHAB  P.AMW                                        ; 1104
                                          01  DD 00015       PUSHL   #1                                           ;
                                00028362  8F  DD 00017       PUSHL   #164706                                      ;
                        00000000G   00        03  FB 0001D       CALLS   #3, LIB$SIGNAL                           ;
                                          5E  DD 00024 1$:     PUSHL   SP                                       ; 1104
                                      08  AE  9F 00026       PUSHAB  HIGHPTR                                      ;
                                      10  AE  9F 00029       PUSHAB  LOWPTR                                       ;
                                      18  AE  9F 0002C       PUSHAB  PARENT_TYPE                                  ;
                                          52  DD 0002F       PUSHL   R2                                           ;
                        00000000G   00        05  FB 00031       CALLS   #5, DBG$STA_TYP_SUBRNG                   ;
                                52        04  AC  D0 00038       MOVL    VAL_DESC, R2                             ; 1104
                                08        06  A2  91 0003C       CMPB    6(R2), #8                                ;
                                          47  12 00040       BNEQ    8$                                       ;
                                50        14  A2  3C 00042       MOVZWL  20(R2), R0                               ; 1104
                                          50  D7 00046       DECL    R0                                       ;
                                50            04  C6 00048       DIVL2   #4, R0                                   ;
                                          50  D6 0004B       INCL    LENGTH                                   ;
                                51            01  CE 0004D       MNEGL   #1, I                                    ; 1104
                                          05  11 00050       BRB     3$                                       ;
                            10 AE41           01  CE 00052 2$:     MNEGL   #1, SETVAL[I]                            ; 1105
                        F7      51            50  F2 00057 3$:     AOBLSS  LENGTH, I, 2$                            ;
                                53        10  AE  9E 0005B       MOVAB   SETVAL, SETVALUE                         ; 1105
                        51  08  BE            01  C3 0005F       SUBL3   #1, @LOWPTR, I                           ; 1105
                                          04  11 00064       BRB     5$                                       ;
                        00      63            51  E5 00066 4$:     BBCC    I, (SETVALUE), 5$                        ;
                        F7      51        04  BE  F3 0006A 5$:     AOBLEQ  @HIGHPTR, I, 4$                          ;
```

```
                        52        18   A2  D0 0006F          MOVL    24(R2), SET_VAL_DESC                    ; 1105
                        51             01  CE 00073          MNEGL   #1, I                                   ; 1105
                                       0B  11 00076          BRB     7$                                      ;
                        53           6241  D2 00078 6$:      MCOML   (SET_VAL_DESC)[I], R3                   ;
                    10 AE41             53  CA 0007C          BICL2   R3, SETVAL[I]                           ;
                                        18  12 00081          BNEQ    10$                                     ; 1105
                F1      51              50  F2 00083 7$:      AOBLSS  LENGTH, I, 6$                           ; 1105
                                        0E  11 00087          BRB     9$                                      ; 1106
                    18  B2      04     BE  D1 00089 8$:      CMPL    @HIGHPTR, @24(R2)                       ; 1106
                                        0B  19 0008E          BLSS    10$                                     ;
                    18  B2      08     BE  D1 00090          CMPL    @LOWPTR, @24(R2)                        ; 1106
                                        04  14 00095          BGTR    10$                                     ;
                        50              01  D0 00097 9$:      MOVL    #1, R0                                  ; 1107
                                            04 0009A          RET                                             ;
                                        50  D4 0009B 10$:     CLRL    R0                                      ; 1107
                                            04 0009D          RET                                             ;

; Routine Size:  158 bytes,    Routine Base:  DBG$CODE + 2C8C

;10998          11075  1
;10999          11076  0        END ELUDOM



                                                              .EXTRN  LIB$SIGNAL, SYS$UNWIND

;                                   PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  DBG$PLIT                      25092  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  DBG$GLOBAL                        4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$OWN                          56  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  DBG$CODE                      11562  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        56         0        1000      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1     32         3         9           7      00:00.1
;  _$255$DUA28:[DEBUG.OBJ]DBGLIB.L32;1     1545       518        33          97      00:01.9
;  _$255$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1
;                                           418        24         5          31      00:00.3
;  _$255$DUA28:[DEBUG.OBJ]DBGMSG.L32;1      386        17         4          22      00:00.3


; Information:  1
; Warnings:     0
```

```
; Errors:       0


;                                       COMMAND QUALIFIERS

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

; Size:         11562 code + 25152 data bytes
; Run Time:       08:19.9
; Elapsed Time:   26:29.6
; Lines/CPU Min:   1329
; Lexemes/CPU-Min: 53361
; Memory Used:  1211 pages
; Compilation Complete
```

0080 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGEVALOP
LIS

DBGENCDEC
LIS

0081 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBGEVENT
.DIS