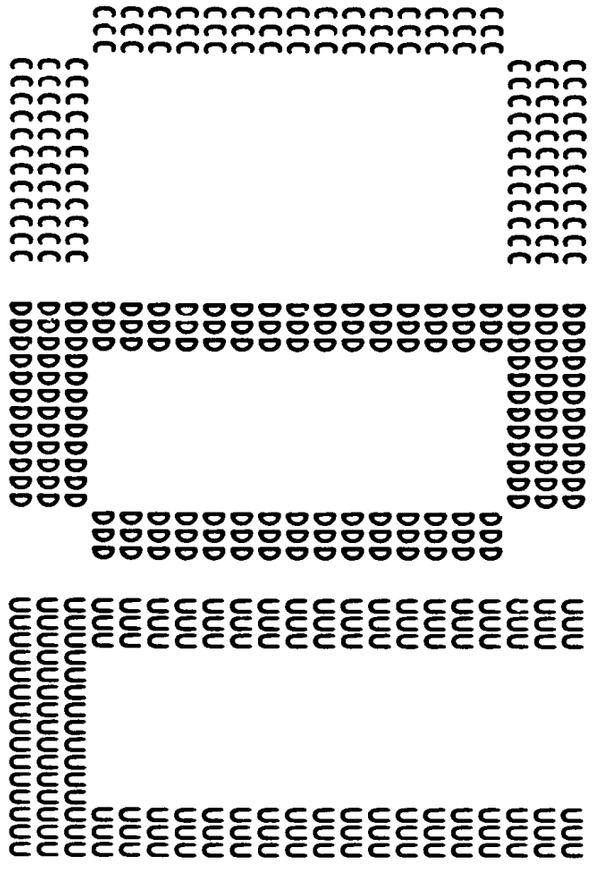

```
  GGGGGGGG  EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE  44    44
  GGGGGGGG  EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE  44    44
GG          EE          NN      NN  CC          OO      OO  DD      DD  EE          44    44
GG          EE          NN      NN  CC          OO      OO  DD      DD  EE          44    44
GG          EE          NNNN    NN  CC          OO      OO  DD      DD  EE          44    44
GG          EE          NNNN    NN  CC          OO      OO  DD      DD  EE          44    44
GG          EEEEEEEE    NN  NN  NN  CC          OO      OO  DD      DD  EEEEEEEE    4444444444
G           EEEEEEEE    NN  NN  NN  CC          OO      OO  DD      DD  EEEEEEEE    4444444444
GG  GGGGGG  EE          NN    NNNN  CC          OO      OO  DD      DD  EE                  44
GG  GGGGGG  EE          NN    NNNN  CC          OO      OO  DD      DD  EE                  44
GG      GG  EE          NN      NN  CC          OO      OO  DD      DD  EE                  44      ....
GG      GG  EE          NN      NN  CC          OO      OO  DD      DD  EE                  44      ....
  GGGGGG    EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE          44      ....
  GGGGGG    EEEEEEEEEE  NN      NN    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE          44      ....

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 gencode4          (IDENT='V04-000',
;     2          0002  0                          ADDRESSING_MODE(EXTERNAL=GENERAL))
;     3          0003  1 = BEGIN
;     4          0004  1
;     5          0005  1 !*****************************************************************************
;     6          0006  1 !*                                                                           *
;     7          0007  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     8          0008  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;     9          0009  1 !*  ALL RIGHTS RESERVED.                                                     *
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    12          0012  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    13          0013  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    14          0014  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    15          0015  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    16          0016  1 !*  TRANSFERRED.                                                             *
;    17          0017  1 !*                                                                           *
;    18          0018  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    19          0019  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    20          0020  1 !*  CORPORATION.                                                             *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    23          0023  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*****************************************************************************
;    27          0027  1
;    28          0028  1 !++
;    29          0029  1 ! Facility:     Command Definition Utility, Table Generator Module 4
;    30          0030  1 !
;    31          0031  1 ! Abstract:     This module is one of a few modules that is responsible
;    32          0032  1 !               for generating the blocks that make up a CLI table.
;    33          0033  1 !               The blocks are generated by traversing the intermediate
;    34          0034  1 !               representation of the CLD file created by the parsing
;    35          0035  1 !               modules.
;    36          0036  1 !
;    37          0037  1 ! Environment:  Standard CDU environment.
;    38          0038  1 !
;    39          0039  1 ! Author:       Paul C. Anagnostopoulos
;    40          0040  1 ! Creation:     28 February 2983
;    41          0041  1 !
;    42          0042  1 ! Modifications:
;    43          0043  1 !
;    44          0044  1 !       V04-001 BLS0338         Benn Schreiber          9-AUG-1984
;    45          0045  1 !               Count blocks created by generate_path in total table size
;    46          0046  1 !
;    47          0047  1 !--
;    48          0048  1
;    49          0049  1
;    50          0050  1 library 'sys$library:lib';
;    51          0051  1 require 'clitabdef';
;    52          0376  1 require 'cdureq';
```

```
;    54          0790  1 !       T A B L E   O F   C O N T E N T S
;    55          0791  1 !       ---------   ---   -----------------
;    56          0792  1
;    57          0793  1 forward routine
;    58          0794  1         cdu$generate_path: novalue,
;    59          0795  1         match_first_entity,
;    60          0796  1         match_remaining_entities,
;    61          0797  1         report_path_error: novalue;
;    62          0798  1
;    63          0799  1
;    64          0800  1 !       E X T E R N A L   R E F E R E N C E S
;    65          0801  1 !       -----------------   -------------------
;    66          0802  1
;    67          0803  1 external routine
;    68          0804  1         cdu$lookup_child,
;    69          0805  1         cdu$remember_reference,
;    70          0806  1         cdu$report_semantic_error,
;    71          0807  1         lib$get_vm;
;    72          0808  1
;    73          0809  1 external
;    74          0810  1         cdu$gl_root_node: ref node,
;    75          0811  1         cdu$gl_table: pointer;
```

```
;    77          0812  1 !       P A T H   S T A C K
;    78          0813  1 !       -------   ---------
;    79          0814  1
;    80          0815  1 ! The following items define the module-wide path stack, which is used to
;    81          0816  1 ! resolve an entity path into its complete entity hierarchy.
;    82          0817  1
;    83          0818  1 own
;    84          0819  1         path_stack_top: signed long,
;    85          0820  1         path_stack: vector[exp_k_max_path_entities,long];
;    86          0821  1
;    87          0822  1 ! These two macros are used to push and pop the path stack.
;    88          0823  1
;    89          0824  1 macro
;    90        M 0825  1         push_path_stack(item) =
;    91        M 0826  1                 (increment(path_stack_top);
;    92        M 0827  1                 if .path_stack_top gequ exp_k_max_path_entities then
;    93        M 0828  1                         signal(msg(cdu$_intpathstkov));
;    94        M 0829  1                 path_stack[.path_stack_top] = item;
;    95          0830  1                 ) %,
;    96          0831  1
;    97        M 0832  1         pop_path_stack =
;    98          0833  1                 (decrement(path_stack_top)) %;
;    99          0834  1
;   100          0835  1
;   101          0836  1 !       N O D E   M A N I P U L A T I O N   M A C R O S
;   102          0837  1 !       -------   -----------------------   -----------
;   103          0838  1
;   104          0839  1 ! The following macro is used to determine whether an arbitrary node
;   105          0840  1 ! represents an entity (i.e., a parameter, qualifier, or type keyword).
;   106          0841  1
;   107          0842  1 macro
;   108        M 0843  1         top_level_entity_node(a_node) =
;   109        M 0844  1                 (.a_node[node_w_type] eqlu node_k_parameter or
;   110        M 0845  1                  .a_node[node_w_type] eqlu node_k_qualifier or
;   111          0846  1                  .a_node[node_w_type] eqlu node_k_keyword) %;
;   112          0847  1
;   113          0848  1 ! The following macro is used to determine whether an arbitrary node is an
;   114          0849  1 ! entity node that matches a given name.  If there is a label node under
;   115          0850  1 ! it, then we try to match the label name.  Otherwise we try to match the
;   116          0851  1 ! entity name.
;   117          0852  1
;   118          0853  1 macro
;   119        M 0854  1         matching_entity(a_node,name_length,name_address) =
;   120        M 0855  1                 (bind
;   121        M 0856  1                         entity = a_node: node;
;   122        M 0857  1
;   123        M 0858  1                 if top_level_entity_node(entity) then (
;   124        M 0859  1                         local
;   125        M 0860  1                                 entitys_label: ref node;
;   126        M 0861  1
;   127        M 0862  1                         ! The node represents an entity.  Compare the
;   128        M 0863  1                         ! desired name to the label node, or entity node if
;   129        M 0864  1                         ! there is no label.
;   130        M 0865  1
;   131        M 0866  1                         if (entitys_label = cdu$lookup_child(entity,node_k_label)) neqa 0 then
;   132        M 0867  1                                 ch$eql(.entitys_label[node_b_text_length],entitys_label[node_t_text],
;   133        M 0868  1                                         name_length,name_address,%x'00')
```

```
;  134        M 0869  1                        else
;  135        M 0870  1                                ch$eql(.entity[node_b_text_length],entity[node_t_text],
;  136        M 0871  1                                       name_length,name_address,%x'00')
;  137        M 0872  1                    ) else
;  138        M 0873  1
;  139        M 0874  1                        ! The node does not represent an entity.
;  140        M 0875  1
;  141        M 0876  1                        false
;  142          0877  1                    ) %;
;  143          0878  1
;  144          0879  1 ! This macro is used to determine whether an arbitrary node represents an
;  145          0880  1 ! entity, and, if so, whether that entity references a type definition in
;  146          0881  1 ! its value clause.  The address of the node representing the type
;  147          0882  1 ! definition is returned.
;  148          0883  1
;  149          0884  1 macro
;  150        M 0885  1         type_reference(a_node) =
;  151        M 0886  1                    (bind
;  152        M 0887  1                        entity = a_node: node;
;  153        M 0888  1                    local
;  154        M 0889  1                        type: ref node;
;  155        M 0890  1
;  156        M 0891  1                    if top_level_entity_node(entity) then
;  157        M 0892  1                        if (type = cdu$lookup_child(entity,node_k_value)) neqa 0 then
;  158        M 0893  1                            if (type = cdu$lookup_child(.type,node_k_type_user)) neqa 0 then
;  159        M 0894  1                                cdu$lookup_child(.cdu$gl_root_node,node_k_define_type,
;  160        M 0895  1                                    .type[node_b_text_length],type[node_t_text])
;  161        M 0896  1                            else 0
;  162        M 0897  1                        else 0
;  163        M 0898  1                    else 0
;  164          0899  1                    ) %;
```

```
;  166          0900  1 !++
;  167          0901  1 ! Description:  This routine is called to generate an expression block
;  168          0902  1 !               that represents an entity path.  An entity path specifies a
;  169          0903  1 !               hierarcy of entities, and is used in a boolean expression to
;  170          0904  1 !               reference a specific entity (as in the DISALLOW clause).
;  171          0905  1 !               Because the entity path can be elided, this routine must
;  172          0906  1 !               resolve the path into a full entity hierarchy, beginning at
;  173          0907  1 !               a parameter or qualifier and ending at the final keyword in
;  174          0908  1 !               the path.
;  175          0909  1 !
;  176          0910  1 ! Parameters:   definition      By reference, the node representing the verb
;  177          0911  1 !                               or syntax change definition that includes
;  178          0912  1 !                               the path.
;  179          0913  1 !               path            By reference, the node representing the
;  180          0914  1 !                               path that we are to compile.  We will fill
;  181          0915  1 !                               in the TRO of the resulting block.
;  182          0916  1 !
;  183          0917  1 ! Returns:      Nothing.
;  184          0918  1 !
;  185          0919  1 ! Notes:
;  186          0920  1 !--
;  187          0921  1
;  188          0922  1 GLOBAL ROUTINE cdu$generate_path(definition: ref node,
;  189          0923  1                                 path: ref node)                : novalue
;  190          0924  2 = BEGIN
;  191          0925  2
;  192          0926  2 bind
;  193          0927  2         first_child = .path[node_l_child]: node;
;  194          0928  2
;  195          0929  2 local
;  196          0930  2         context: ref node,
;  197          0931  2         first_entity: ref node,
;  198          0932  2         path_stack_mark: signed long,
;  199          0933  2         ambiguous: boolean initial(false),
;  200          0934  2         expression: pointer;
;  201          0935  2 local
;  202          0936  2         saved_stack_top: signed long initial(-1),
;  203          0937  2         saved_stack: vector[exp_k_max_path_entities,long],
;  204          0938  2         saved_stack_mark: long;
;  205          0939  2
;  206          0940  2
;  207          0941  2 ! A path can include the name of a definition in which the entity hierarchy
;  208          0942  2 ! is to be resolved.  Or, if no such name is included, then it is to be
;  209          0943  2 ! resolved in the current definition.  See what the user wants to do.
;  210          0944  2
;  211          0945  3 if .first_child[node_w_type] eqlu node_k_path_definition then (
;  212          0946  3
;  213          0947  3         ! The user included an explicit definition name as the first element
;  214          0948  3         ! in the path.  Find the node representing the definition.
;  215          0949  3
;  216          0950  3         context = cdu$lookup_child(.cdu$gl_root_node,node_k_define_verb,
;  217          0951  3                                    .first_child[node_b_text_length],first_child[node_t_text]);
;  218          0952  3         if .context eqla 0 then
;  219          0953  3                 context = cdu$lookup_child(.cdu$gl_root_node,node_k_define_syntax,
;  220          0954  3                                            .first_child[node_b_text_length],first_child[node_t_text]);
;  221          0955  4         if .context eqla 0 then (
;  222          0956  4                 report_path_error(msg(cdu$_pathundef),.path);
```

```
;  223          0957  4                    return;
;  224          0958  3            );
;  225          0959  3
;  226          0960  3            ! Remember the address of the node following the path definition node,
;  227          0961  3            ! because this is the first entity we must match.
;  228          0962  3
;  229          0963  3            first_entity = .first_child[node_l_sister];
;  230          0964  3
;  231          0965  3        ) else (
;  232          0966  3
;  233          0967  3            ! No definition name was included, so use the current definition.
;  234          0968  3
;  235          0969  3            context = .definition;
;  236          0970  3
;  237          0971  3            ! Remember the address of this first node, because this is the
;  238          0972  3            ! first entity we must match.
;  239          0973  3
;  240          0974  3            first_entity = first_child;
;  241          0975  2        );
```

```
;  243          0976  2 ! Now we sit in a messy loop trying to resolve the entity path specified by
;  244          0977  2 ! the user.  It is resolved in the context of the definition we determined
;  245          0978  2 ! above.  The module-wide path stack will be used to search the definition's
;  246          0979  2 ! parameters, qualifiers, and type references.
;  247          0980  2
;  248          0981  2 path_stack_top = -1;
;  249          0982  3 while match_first_entity(.context,.first_entity) do (
;  250          0983  3
;  251          0984  3         ! We have found a match on the first entity name and set up the
;  252          0985  3         ! path stack accordingly.  Mark the stack so we can restore it to
;  253          0986  3         ! this state later on.
;  254          0987  3
;  255          0988  3         path_stack_mark = .path_stack_top;
;  256          0989  3
;  257          0990  3         ! See if we can match the remaining entities in the path, beginning
;  258          0991  3         ! at the match we found.
;  259          0992  3
;  260          0993  4         if match_remaining_entities(.first_entity[node_l_sister]) then (
;  261          0994  4
;  262          0995  4                 ! Yup, we got a complete match and the path stack contains
;  263          0996  4                 ! the resolved entity hierarchy.  Decide if the match is
;  264          0997  4                 ! interesting.
;  265          0998  4
;  266          0999  5                 if .saved_stack_top eql -1 then (
;  267          1000  5
;  268          1001  5                         ! This is the first match, so it's useful.  Save
;  269          1002  5                         ! the state of the path stack so we can use it
;  270          1003  5                         ! later.
;  271          1004  5
;  272          1005  5                         saved_stack_top = .path_stack_top;
;  273          1006  5                         ch$move(%allocation(path_stack),path_stack, saved_stack);
;  274          1007  5                         saved_stack_mark = .path_stack_mark;
;  275          1008  5
;  276          1009  5                 ) else if .saved_stack_mark eqlu 0 then (
;  277          1010  5
;  278          1011  5                         ! We already have a match, and it's one where the
;  279          1012  5                         ! first entity in the path matched a parameter or
;  280          1013  5                         ! qualifier.  Such a match takes precedence over
;  281          1014  5                         ! one where the first entity matched a type keyword.
;  282          1015  5                         ! What kind of match have we got now?
;  283          1016  5
;  284          1017  5                         if .path_stack_mark eqlu 0 then
;  285          1018  5
;  286          1019  5                                 ! Oops, the new match is also on a
;  287          1020  5                                 ! parameter or qualifier.  That's ambiguous.
;  288          1021  5
;  289          1022  5                                 ambiguous = true;
;  290          1023  5
;  291          1024  5                         ! It appears that the new match is on a type
;  292          1025  5                         ! keyword.  Ignore it in favor of the saved one.
;  293          1026  5
;  294          1027  5                 ) else (
;  295          1028  5
;  296          1029  5                         ! The saved match is on a type keyword.  What does
;  297          1030  5                         ! the new one look like?
;  298          1031  5
;  299          1032  6                         if .path_stack_mark eqlu 0 then (
```

```
;  300          1033  6
;  301          1034  6                                ! The new one is on a parameter or
;  302          1035  6                                ! qualifier, so it takes precedence.  Save
;  303          1036  6                                ! it as an unambiguous match.
;  304          1037  6
;  305          1038  6                                saved_stack_top = .path_stack_top;
;  306          1039  6                                ch$move(%allocation(path_stack),path_stack, saved_stack);
;  307          1040  6                                saved_stack_mark = .path_stack_mark;
;  308          1041  6                                ambiguous = false;
;  309          1042  5                            ) else
;  310          1043  5
;  311          1044  5                                ! The new one also matches on a type
;  312          1045  5                                ! keyword, so it's ambiguous.
;  313          1046  5
;  314          1047  5                                ambiguous = true;
;  315          1048  4                        );
;  316          1049  3                );
;  317          1050  3
;  318          1051  3                ! Restore the path stack to its state when the first entity
;  319          1052  3                ! was matched, thus flushing the entity hierarchy.  Then we can
;  320          1053  3                ! loop for the next match (we need to look at all matches to
;  321          1054  3                ! detect ambiguity).
;  322          1055  3
;  323          1056  3                path_stack_top = .path_stack_mark;
;  324          1057  2 );
```

```
;  326          1058  2 ! If no match was found, tell the user and forget it.
;  327          1059  2
;  328          1060  3 if .saved_stack_top eql -1 then (
;  329          1061  3         report_path_error(msg(cdu$_pathunres),.path);
;  330          1062  3         return;
;  331          1063  2 );
;  332          1064  2
;  333          1065  2 ! If more than one possible match was found, tell the user and forget it.
;  334          1066  2
;  335          1067  3 if .ambiguous then (
;  336          1068  3         report_path_error(msg(cdu$_pathambig),.path);
;  337          1069  3         return;
;  338          1070  2 );
;  339          1071  2
;  340          1072  2 ! OK, we have an unambiguous match on the path.  Create an expression block
;  341          1073  2 ! to contain the entity hierarchy.
;  342          1074  2
;  343          1075  2 allocate_largest_table_block(ent_k_length + (.saved_stack_top+1)*4, expression);
;  344          1076  2
;  345          1077  2 ! Initialize the header of the expression block.
;  346          1078  2
;  347          1079  2 expression[exp_b_type] = block_k_expression;
;  348          1080  2 expression[exp_b_subtype] = exp_k_path;
;  349          1081  2 expression[exp_w_flags] = 0;
;  350          1082  2 expression[exp_w_tro_count] = .saved_stack_top+1;
;  351          1083  2
;  352          1084  2 ! Sit in a loop looking at each level in the path hierarchy.  Remember to
;  353          1085  2 ! resolve the final TRO of the associated entity block.
;  354          1086  2
;  355          1087  3 begin
;  356          1088  3 bind
;  357          1089  3         entity_list = expression[exp_l_operand_list]: vector[,long];
;  358          1090  3
;  359          1091  3 incru i from 0 to .saved_stack_top do
;  360          1092  3         cdu$remember_reference(entity_list[.i],.saved_stack[.i]);
;  361          1093  2 end;
;  362          1094  2
;  363          1095  2 ! Fill in the final length of the expression block and add it's
;  364          1096  2 ! size to the table being created.
;  365          1097  2
;  366          1098  2 expression[exp_w_size] = exp_k_length + (.saved_stack_top+1)*4;
;  367          1099  2 set_table_block_size(.expression[exp_w_size],expression);
;  368          1100  2
;  369          1101  2 ! Store the TRO of this new expression block in its representing node.
;  370          1102  2
;  371          1103  2 path[node_l_code] = .expression - .cdu$gl_table;
;  372          1104  2 return;
;  373          1105  2
;  374          1106  1 END;
; INFO#250                 L1:1009
; Referenced LOCAL symbol SAVED_STACK_MARK is probably not initialized

                                                        TITLE  GENCODE4
                                                      .IDENT  \V04-000\

                                                      .PSECT  $OWN$,NOEXE,2
```

```
                                00000 PATH_STACK_TOP:
                                                .BLKB   4
                                00004 PATH_STACK:
                                                .BLKB   32

                                                .EXTRN  CDU$LOOKUP_CHILD
                                                .EXTRN  CDU$REMEMBER_REFERENCE
                                                .EXTRN  CDU$REPORT_SEMANTIC_ERROR
                                                .EXTRN  LIB$GET_VM, CDU$GL_ROOT_NODE
                                                .EXTRN  CDU$GL_TABLE, CDU$_PATHUNDEF
                                                .EXTRN  CDU$_PATHUNRES, CDU$_PATHAMBIG

                                                .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000          .ENTRY  CDU$GENERATE_PATH, Save R2,R3,R4,R5,R6,R7,- ; 0922
                                                                R8,R9,R10,R11
                          5E            28  C2 00002          SUBL2   #40, SP
                          57      08    AC  D0 00005          MOVL    PATH, R7                                ; 0927
                          52      08    A7  D0 00009          MOVL    8(R7), R2
                                        5A  94 0000D          CLRB    AMBIGUOUS
                          56            01  CE 0000F          MNEGL   #1, SAVED_STACK_TOP
                          30            62  B1 00012          CMPW    (R2), #48                               ; 0945
                                        47  12 00015          BNEQ    2$
                                  11    A2  9F 00017          PUSHAB  17(R2)                                  ; 0951
                          7E      10    A2  9A 0001A          MOVZBL  16(R2), -(SP)
                                        04  DD 0001E          PUSHL   #4
                              00000000G 00  DD 00020          PUSHL   CDU$GL_ROOT_NODE
                00000000G     00        04  FB 00026          CALLS   #4, CDU$LOOKUP_CHILD
                          6E            50  D0 0002D          MOVL    R0, CONTEXT
                                        26  12 00030          BNEQ    1$                                      ; 0952
                                  11    A2  9F 00032          PUSHAB  17(R2)                                  ; 0954
                          7E      10    A2  9A 00035          MOVZBL  16(R2), -(SP)
                                        05  DD 00039          PUSHL   #5
                              00000000G 00  DD 0003B          PUSHL   CDU$GL_ROOT_NODE
                00000000G     00        04  FB 00041          CALLS   #4, CDU$LOOKUP_CHILD
                          6E            50  D0 00048          MOVL    R0, CONTEXT
                                        0B  12 0004B          BNEQ    1$                                      ; 0955
                                        57  DD 0004D          PUSHL   R7                                      ; 0956
                              00000000G 8F  DD 0004F          PUSHL   #CDU$_PATHUNDEF
                                    0092 31  00055          BRW     11$
                          59      04    A2  D0 00058 1$:      MOVL    4(R2), FIRST_ENTITY                     ; 0963
                                        07  11 0005C          BRB     3$                                      ; 0945
                          6E      04    AC  D0 0005E 2$:      MOVL    DEFINITION, CONTEXT                     ; 0969
                          59            52  D0 00062          MOVL    R2, FIRST_ENTITY                        ; 0974
                    0000'       CF        01  CE 00065 3$:      MNEGL   #1, PATH_STACK_TOP                      ; 0981
                                        59  DD 0006A 4$:      PUSHL   FIRST_ENTITY                            ; 0982
                                  04    AE  DD 0006C          PUSHL   CONTEXT
                    0000V       CF        02  FB 0006F          CALLS   #2, MATCH_FIRST_ENTITY
                          55            50  E9 00074          BLBC    R0, 9$
                          58      0000' CF  D0 00077          MOVL    PATH_STACK_TOP, PATH_STACK_MARK          ; 0988
                                  04    A9  DD 0007C          PUSHL   4(FIRST_ENTITY)                          ; 0993
                    0000V       CF        01  FB 0007F          CALLS   #1, MATCH_REMAINING_ENTITIES
                          3E            50  E9 00084          BLBC    R0, 8$
                FFFFFFFF  8F            56  D1 00087          CMPL    SAVED_STACK_TOP, #-1                    ; 0999
                                        11  12 0008E          BNEQ    5$
                          56      0000' CF  D0 00090          MOVL    PATH_STACK_TOP, SAVED_STACK_TOP         ; 1005
```

```
                08   AE      0000'  CF              20  28 00095          MOVC3   #32, PATH_STACK, SAVED_STACK                ; 1006
                                    5B              58  D0 0009C          MOVL     ATH_STACK_MARK, SAVED_STACK_MARK           ; 1007
                                                    24  11 0009F          BRB     8$                                         ; 0999
                                                    5B  D5 000A1 5$:       TSTL    SAVED_STACK_MARK                           ; 1009
                                                    06  12 000A3          BNEQ    6$                                         ;
                                                    58  D5 000A5          TSTL    PATH_STACK_MARK                            ; 1017
                                                    1C  12 000A7          BNEQ    8$                                         ;
                                                    17  11 000A9          BRB     7$                                         ; 1022
                                                    58  D5 000AB 6$:       TSTL    PATH_STACK_MARK                            ; 1032
                                                    13  12 000AD          BNEQ    7$                                         ;
                                56      0000'   CF  D0 000AF              MOVL    PATH_STACK_TOP, SAVED_STACK_TOP            ; 1038
                08   AE      0000'   CF          20  28 000B4              MOVC3   #32, PATH_STACK, SAVED_STACK               ; 1039
                                    5B              58  D0 000BB          MOVL    PATH_STACK_MARK, SAVED_STACK_MARK          ; 1040
                                                    5A  94 000BE          CLRB    AMBIGUOUS                                  ; 1041
                                                    03  11 000C0          BRB     8$                                         ; 1032
                                    5A              01  90 000C2 7$:       MOVB    #1, AMBIGUOUS                              ; 1047
                                0000'   CF          58  D0 000C5 8$:       MOVL    PATH_STACK_MARK, PATH_STACK_TOP            ; 1056
                                                    9E  11 000CA          BRB     4$                                         ; 0982
                            FFFFFFFF    8F          56  D1 000CC 9$:       CMPL    SAVED_STACK_TOP, #-1                       ; 1060
                                                    0A  12 000D3          BNEQ    10$                                        ;
                                                    57  DD 000D5          PUSHL   R7                                         ; 1061
                                    00000000G   8F      DD 000D7          PUSHL   #CDU$_PATHUNRES                            ;
                                                    0B  11 000DD          BRB     11$                                        ;
                                    0E              5A  E9 000DF 10$:      BLBC    AMBIGUOUS, 12$                             ; 1067
                                                    57  DD 000E2          PUSHL   R7                                         ; 1068
                                    00000000G   8F      DD 000E4          PUSHL   #CDU$_PATHAMBIG                            ;
                                0000V   CF          02  FB 000EA 11$:      CALLS   #2, REPORT_PATH_ERROR                      ;
                                                        04 000EF          RET                                                ; 1067
                                        04      AE  9F  00 000F0 12$:      PUSHAB  EXPRESSION                                 ; 1075
                04   AE                 56          02  78 000F3          ASHL    #2, SAVED_STACK_TOP, 4(SP)                 ;
                                    04      AE      22  C0 000F8          ADDL2   #34, 4(SP)                                 ;
                                        04      AE  9F  00 000FC          PUSHAB  4(SP)                                      ;
                            00000000G   00          02  FB 000FF          CALLS   #2, LIB$GET_VM                             ;
                                                    09  50 E8 00106       BLBS    STATUS, 13$                                ;
                                                    50  DD 00109          PUSHL   STATUS                                     ;
                            00000000G   00          01  FB 0010B          CALLS   #1, LIB$SIGNAL                             ;
                                                    52      04  AE  D0 00112 13$:  MOVL    EXPRESSION, R2                     ; 1079
                                    02      A2      0105    8F  3C 00116      MOVZWL  #261, 2(R2)                            ;
                06   A2                 56          01  A1 0011C          ADDW3   #1, SAVED_STACK_TOP, 6(R2)                 ; 1082
                                                    53  D4 00121          CLRL    I                                          ; 1091
                                                    11  11 00123          BRB     15$                                        ;
                                                        08 AE43 DD 00125 14$:  PUSHL   SAVED_STACK[I]                     ; 1092
                                                        08 A243 DF 00129      PUSHAL  8(R2)[I]                               ;
                            00000000G   00          02  FB 0012D          CALLS   #2, CDU$REMEMBER_REFERENCE                 ;
                                                    53  D6 00134          INCL    I                                          ;
                                                    56  53  D1 00136 15$:  CMPL    I, SAVED_STACK_TOP                         ;
                                                    EA  1B 00139          BLEQU   14$                                        ;
                50                  56          02  78 0013B          ASHL    #2, SAVED_STACK_TOP, R0                    ; 1098
                62                  50          0C  A1 0013F          ADDW3   #12, R0, (R2)                              ;
                                    50          62  3C 00143          MOVZWL  (R2), R0                                   ; 1099
                                    50          03  C0 00146          ADDL2   #3, R0                                     ;
                                    50          04  C6 00149          DIVL2   #4, R0                                     ;
                62                  50          04  A5 0014C          MULW3   #4, R0, (R2)                               ;
                                    50  00000000G  00  D0 00150          MOVL    CDU$GL_TABLE, R0                           ;
                                    51          62  3C 00157          MOVZWL  (R2), R1                                   ;
                                10  A0          51  C0 0015A          ADDL2   R1, 16(R0)                                 ;
                0C   A7             52          50  C3 0015E          SUBL3   R0, R2, 12(R7)                             ; 1103
```

```
                                04 00163        RET                                     ; 1106

; Routine Size:  356 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  376          1107  1 !++
;  377          1108  1 ! Description:  This routine is called to find a match on the first entity in
;  378          1109  1 !               a path specified by the user.  We perform a depth-first
;  379          1110  1 !               search of all the entities under a given verb or syntax
;  380          1111  1 !               definition, looking for an entity which matches that
;  381          1112  1 !               specified by the user.  As we go, we maintain a path stack
;  382          1113  1 !               that shows how we got down to the entity.
;  383          1114  1 !
;  384          1115  1 ! Parameters:   path_definition By reference, the node representing the verb
;  385          1116  1 !                               or syntax definition which is to be
;  386          1117  1 !                               searched.
;  387          1118  1 !               path_entity     By reference, the node representing the
;  388          1119  1 !                               first entity in the path.
;  389          1120  1 !
;  390          1121  1 ! Returns:      By value, a boolean which is true if we found a match.
;  391          1122  1 !
;  392          1123  1 ! Notes:
;  393          1124  1 !--
;  394          1125  1
;  395          1126  1 GLOBAL ROUTINE match_first_entity(path_definition: ref node,
;  396          1127  1                                  path_entity: ref node)
;  397          1128  2 = BEGIN
;  398          1129  2
;  399          1130  2 local
;  400          1131  2         type: ref node;
;  401          1132  2
;  402          1133  2
;  403          1134  2 ! The path stack is either empty, or it contains the entity hierarchy
;  404          1135  2 ! leading to the previous match.  Sit in a loop to find the next match.
;  405          1136  2
;  406          1137  3 loop (
;  407          1138  3
;  408          1139  3         ! If the stack is empty, begin searching the children of the
;  409          1140  3         ! definition.  Otherwise, move on to the sister of the top entity
;  410          1141  3         ! on the stack.
;  411          1142  3
;  412          1143  3         if .path_stack_top eql -1 then
;  413          1144  4                 push_path_stack(.path_definition[node_l_child])
;  414          1145  4         else (
;  415          1146  4                 bind
;  416          1147  4                         entity = .path_stack[.path_stack_top]: node;
;  417          1148  4
;  418          1149  4                 path_stack[.path_stack_top] = .entity[node_l_sister];
;  419          1150  3         );
;  420          1151  3
;  421          1152  3         if .path_stack[.path_stack_top] neqa 0 then
;  422          1153  3
;  423          1154  3                 ! If there is another node at this level, then we want to
;  424          1155  3                 ! search all of the type definitions below it.  Push the
;  425          1156  3                 ! first keyword node of each lower-level type definition
;  426          1157  3                 ! on the stack.
;  427          1158  3
;  428          1159  3                 while (type = type_reference(.path_stack[.path_stack_top])) neqa 0 do
;  429          1160  4                         push_path_stack(.type[node_l_child])
;  430          1161  4
;  431          1162  3         else
;  432          1163  3
```

```
;  433          1164  3                ! There isn't another node at this level, so pop the stack.
;  434          1165  3
;  435          1166  3                pop_path_stack;
;  436          1167  3
;  437          1168  3            ! If the stack is empty, we're done searching.
;  438          1169  3
;  439          1170  3            if .path_stack_top eql -1 then exitloop;
;  440          1171  3
;  441          1172  3            ! If the entity at the top of the stack matches the one we are
;  442          1173  3            ! looking for, then we got a match.
;  443          1174  3
;  444        P 1175  3            if matching_entity(.path_stack[.path_stack_top],
;  445          1176  3                               .path_entity[node_b_text_length],path_entity[node_t_text]) then
;  446          1177  3                return true;
;  447          1178  2    );
;  448          1179  2
;  449          1180  2    ! We searched the entire definition without finding a match.
;  450          1181  2
;  451          1182  2    return false;
;  452          1183  2
;  453          1184  1    END;


                                                        .EXTRN  CDU$_INTPATHSTKOV

                                        03FC 00000      .ENTRY  MATCH_FIRST_ENTITY, Save R2,R3,R4,R5,R6,R7,-; 1126
                                                                R8,R9                                         ;
                              59 00000000G   00  9E 00002       MOVAB   LIB$SIGNAL, R9                        ;
                              58 00000000G   8F  D0 00009       MOVL    #CDU$_INTPATHSTKOV, R8                ;
                              57 00000000G   00  9E 00010       MOVAB   CDU$LOOKUP_CHILD, R7                  ;
                              56     0000'   CF  9E 00017       MOVAB   PATH_STACK_TOP, R6                    ;
                              50             66  D0 0001C 1$:   MOVL    PATH_STACK_TOP, R0                    ; 1143
                FFFFFFFF      8F             50  D1 0001F       CMPL    R0, #-1                               ;
                                             1E  12 00026       BNEQ    3$                                    ;
                                             66  D6 00028       INCL    PATH_STACK_TOP                        ; 1144
                              08             66  D1 0002A       CMPL    PATH_STACK_TOP, #8                    ;
                                             05  1F 0002D       BLSSU   2$                                    ;
                                             58  DD 0002F       PUSHL   R8                                    ;
                              69             01  FB 00031       CALLS   #1, LIB$SIGNAL                        ;
                              50             66  D0 00034 2$:   MOVL    PATH_STACK_TOP, R0                    ;
                              51         04 A640  DE 00037       MOVAL   PATH_STACK[R0], R1                    ;
                              50         04   AC  D0 0003C       MOVL    PATH_DEFINITION, R0                   ;
                              61         08   A0  D0 00040       MOVL    8(R0), (R1)                           ;
                                            0C  11 00044       BRB     4$                                    ; 1143
                              51         04 A640  DE 00046 3$:   MOVAL   PATH_STACK[R0], R1                    ; 1147
                              50             61  D0 0004B       MOVL    (R1), R0                              ;
                              61         04   A0  D0 0004E       MOVL    4(R0), (R1)                           ; 1149
                                            69  13 00052 4$:   BEQL    12$                                   ; 1152
                              50             66  D0 00054       MOVL    PATH_STACK_TOP, R0                    ; 1159
                              52         04 A640  DE 00057       MOVAL   PATH_STACK[R0], R2                    ;
                              0D         00   B2  B1 0005C 5$:   CMPW    @0(R2), #13                           ;
                                            0C  13 00060       BEQL    6$                                    ;
                              10         00   B2  B1 00062       CMPW    @0(R2), #16                           ;
                                            06  13 00066       BEQL    6$                                    ;
                              18         00   B2  B1 00068       CMPW    @0(R2), #24                           ;
                                            31  12 0006C       BNEQ    9$                                    ;
```

```
                        1C  DD 0006E 6$:      PUSHL   #28                                                             ;
                        62  DD 00070         PUSHL   (R2)                                                            ;
67                      02  FB 00072         CALLS   #2, CDU$LOOKUP_CHILD                                            ;
                        50  D5 00075         TSTL    TYPE                                                            ;
                        1F  13 00077         BEQL    7$                                                              ;
                        21  DD 00079         PUSHL   #33                                                             ;
                        50  DD 0007B         PUSHL   TYPE                                                            ;
67                      02  FB 0007D         CALLS   #2, CDU$LOOKUP_CHILD                                            ;
                        50  D5 00080         TSTL    TYPE                                                            ;
                        14  13 00082         BEQL    7$                                                              ;
                11      A0  9F 00084         PUSHAB  17(TYPE)                                                        ;
7E              10      A0  9A 00087         MOVZBL  16(TYPE), -(SP)                                                 ;
                        06  DD 0008B         PUSHL   #6                                                              ;
            00000000G   00  DD 0008D         PUSHL   CDU$GL_ROOT_NODE                                                ;
67                      04  FB 00093         CALLS   #4, CDU$LOOKUP_CHILD                                            ;
                        02  11 00096         BRB     8$                                                              ;
                        50  D4 00098 7$:      CLRL    R0                                                              ;
55                      50  D0 0009A 8$:      MOVL    R0, TYPE                                                        ;
                        02  11 0009D         BRB     10$                                                             ;
                        55  D4 0009F 9$:      CLRL    TYPE                                                            ;
                        1C  13 000A1 10$:     BEQL    13$                                                             ;
                        66  D6 000A3         INCL    PATH_STACK_TOP                                                  ; 1160
08                      66  D1 000A5         CMPL    PATH_STACK_TOP, #8                                              ;
                        05  1F 000A8         BLSSU   11$                                                             ;
                        58  DD 000AA         PUSHL   R8                                                              ;
69                      01  FB 000AC         CALLS   #1, LIB$SIGNAL                                                  ;
50                      66  D0 000AF 11$:     MOVL    PATH_STACK_TOP, R0                                              ;
52              04  A640  DE 000B2         MOVAL   PATH_STACK[R0], R2                                              ;
62              08      A5  D0 000B7         MOVL    8(TYPE), (R2)                                                   ;
                        9F  11 000BB         BRB     5$                                                              ; 1159
                        66  D7 000BD 12$:     DECL    PATH_STACK_TOP                                                  ; 1162
                50      66  D0 000BF 13$:     MOVL    PATH_STACK_TOP, R0                                              ; 1170
                FFFFFFFF  8F        50  D1 000C2         CMPL    R0, #-1                                                 ;
                        52  13 000C9         BEQL    18$                                                             ;
                52      04  A640  D0 000CB         MOVL    PATH_STACK[R0], R2                                      ; 1176
                0D          62  B1 000D0         CMPW    (R2), #13                                                       ;
                        0A  13 000D3         BEQL    14$                                                             ;
                10          62  B1 000D5         CMPW    (R2), #16                                                       ;
                        05  13 000D8         BEQL    14$                                                             ;
                18          62  B1 000DA         CMPW    (R2), #24                                                       ;
                        35  12 000DD         BNEQ    16$                                                             ;
                54      08      AC  D0 000DF 14$:     MOVL    PATH_ENTITY, R4                                         ;
                53      08      AC  D0 000E3         MOVL    PATH_ENTITY, R3                                         ;
                            1A  DD 000E7         PUSHL   #26                                                             ;
                            52  DD 000E9         PUSHL   R2                                                              ;
                67          02  FB 000EB         CALLS   #2, CDU$LOOKUP_CHILD                                            ;
                            50  D5 000EE         TSTL    ENTITYS_LABEL                                                   ;
                            12  13 000F0         BEQL    15$                                                             ;
                51      10      A0  9A 000F2         MOVZBL  16(ENTITYS_LABEL), R1                                   ;
                54      10      A4  9A 000F6         MOVZBL  16(R4), R4                                              ;
54          00      11  A0              51  2D 000FA         CMPC5   R1, 17(ENTITYS_LABEL), #0, R4, 17(R3)           ;
                                    11  A3      00100                                                               ;
                            10  11 00102         BRB     16$                                                             ;
                51      10      A2  9A 00104 15$:     MOVZBL  16(R2), R1                                              ;
                50      10      A4  9A 00108         MOVZBL  16(R4), R0                                              ;
50          00      11  A2              51  2D 0010C         CMPC5   R1, 17(R2), #0, R0, 17(R3)                      ;
                                    11  A3      00112                                                               ;
```

```
                          03  13 00114 16$:    BEQL    17$                                     ;
                        FF03  31 00116          BRW     1$                                      ;
                  50      01  D0 00119 17$:    MOVL    #1, R0                                  ; 1177
                              04 0011C          RET                                             ;
                          50  D4 0011D 18$:    CLRL    R0                                      ; 1182
                              04 0011F          RET                                             ; 1184

; Routine Size:  288 bytes,    Routine Base:  $CODE$ + 0164
```

```
;  455          1185  1 !++
;  456          1186  1 ! Description:  This routine is called to match all but the first entity in
;  457          1187  1 !                a multilevel entity path, after the first entity has been
;  458          1188  1 !                matched.  To constitute a match, the remaining entities
;  459          1189  1 !                must match successively deeper levels of nested type
;  460          1190  1 !                definitions.  As we go, we maintain a path stack that shows
;  461          1191  1 !                how we got down to the final entity.
;  462          1192  1 !
;  463          1193  1 ! Parameters:   path_entity     By reference, the node representing the
;  464          1194  1 !                               second entity in the path.  Its sisters
;  465          1195  1 !                               are the rest of the entities.
;  466          1196  1 !
;  467          1197  1 ! Returns:      By value, a boolean which is true if a match is detected.
;  468          1198  1 !
;  469          1199  1 ! Notes:
;  470          1200  1 !--
;  471          1201  1
;  472          1202  1 GLOBAL ROUTINE match_remaining_entities(path_entity: ref node)
;  473          1203  2 = BEGIN
;  474          1204  2
;  475          1205  2 local
;  476          1206  2         type: ref node,
;  477          1207  2         keyword: ref node;
;  478          1208  2
;  479          1209  2
;  480          1210  2 ! Sit in a loop looking at each of the remaining entities in the path.
;  481          1211  2
;  482          1212  3 while .path_entity neqa 0 do (
;  483          1213  3
;  484          1214  3         ! The entity on the top of the path stack must reference a type
;  485          1215  3         ! definition in its value clause, or we can't possible match this
;  486          1216  3         ! next entity.  Find that referencing node.
;  487          1217  3
;  488          1218  3         type = type_reference(.path_stack[.path_stack_top]);
;  489          1219  3         if .type eqla 0 then
;  490          1220  3                 return false;
;  491          1221  3
;  492          1222  3         ! We have a type definition referenced by the top entity on the
;  493          1223  3         ! stack.  It must include a keyword that matches this next entity.
;  494          1224  3         ! Find it.
;  495          1225  3
;  496        P 1226  3         scan_children(type,keyword,
;  497        P 1227  3                 if matching_entity(.keyword,.path_entity[node_b_text_length],
;  498        P 1228  3                                    path_entity[node_t_text]) then exitloop;
;  499          1229  3         );
;  500          1230  3         if .keyword eqla 0 then
;  501          1231  3                 return false;
;  502          1232  3
;  503          1233  3         ! The keyword represents the next level in the entity hierarchy for
;  504          1234  3         ! this path, so push it on the path stack.
;  505          1235  3
;  506          1236  3         push_path_stack(.keyword);
;  507          1237  3
;  508          1238  3         ! Go on to the next entity in the specified path.
;  509          1239  3
;  510          1240  3         path_entity = .path_entity[node_l_sister];
;  511          1241  2 );
```

```
;  512          1242  2
;  513          1243  2 ! If we dropped out of that loop, then we got a match.
;  514          1244  2
;  515          1245  2 return true;
;  516          1246  2
;  517          1247  1 END;


                                03FC 00000          .ENTRY  MATCH_REMAINING_ENTITIES, Save R2,R3,R4,R5,-; 1202
                                                            R6,R7,R8,R9                                    ;
59      0000'    CF  9E 00002          MOVAB   PATH_STACK_TOP, R9                                         ;
58 00000000G     00  9E 00007          MOVAB   CDU$LOOKUP_CHILD, R8                                       ;
50               69  D0 0000E          MOVL    PATH_STACK_TOP, R0                                         ; 1218
56          04 A940  DE 00011          MOVAL   PATH_STACK[R0], R6                                         ;
54          04   AC  D0 00016 1$:      MOVL    PATH_ENTITY, R7                                            ; 1212
                 03  12 0001A          BNEQ    2$                                                         ;
              00BC  31 0001C          BRW     15$                                                        ;
0D          00   B6  B1 0001F 2$:      CMPW    @0(R6), #13                                                ; 1218
                 0C  13 00023          BEQL    3$                                                         ;
10          00   B6  B1 00025          CMPW    @0(R6), #16                                                ;
                 06  13 00029          BEQL    3$                                                         ;
18          00   B6  B1 0002B          CMPW    @0(R6), #24                                                ;
                 31  12 0002F          BNEQ    6$                                                         ;
                 1C  DD 00031 3$:      PUSHL   #28                                                        ;
                 66  DD 00033          PUSHL   (R6)                                                       ;
68               02  FB 00035          CALLS   #2, CDU$LOOKUP_CHILD                                       ;
                 50  D5 00038          TSTL    TYPE                                                       ;
                 1F  13 0003A          BEQL    4$                                                         ;
                 21  DD 0003C          PUSHL   #33                                                        ;
                 50  DD 0003E          PUSHL   TYPE                                                       ;
68               02  FB 00040          CALLS   #2, CDU$LOOKUP_CHILD                                       ;
                 50  D5 00043          TSTL    TYPE                                                       ;
                 14  13 00045          BEQL    4$                                                         ;
            11   A0  9F 00047          PUSHAB  17(TYPE)                                                   ;
7E          10   A0  9A 0004A          MOVZBL  16(TYPE), -(SP)                                            ;
                 C6  DD 0004E          PUSHL   #6                                                         ;
   00000000G     00  DD 00050          PUSHL   CDU$GL_ROOT_NODE                                           ;
68               04  FB 00056          CALLS   #4, CDU$LOOKUP_CHILD                                       ;
                 02  11 00059          BRB     5$                                                         ;
                 50  D4 0005B 4$:      CLRL    R0                                                         ;
57               50  D0 0005D 5$:      MOVL    R0, TYPE                                                   ;
                 02  11 00060          BRB     7$                                                         ;
                 57  D4 00062 6$:      CLRL    TYPE                                                       ;
                 79  13 00064 7$:      BEQL    16$                                                        ; 1219
55          08   A7  D0 00066          MOVL    8(TYPE), KEYWORD                                           ; 1229
                 44  13 0006A 8$:      BEQL    13$                                                        ;
0D               65  B1 0006C          CMPW    (KEYWORD), #13                                             ;
                 0A  13 0006F          BEQL    9$                                                         ;
10               65  B1 00071          CMPW    (KEYWORD), #16                                             ;
                 05  13 00074          BEQL    9$                                                         ;
18               65  B1 00076          CMPW    (KEYWORD), #24                                             ;
                 2F  12 00079          BNEQ    12$                                                        ;
                 1A  DD 0007B 9$:      PUSHL   #26                                                        ;
                 55  DD 0007D          PUSHL   KEYWORD                                                    ;
```

```
                                68              02  FB 0007F           CALLS   #2, CDU$LOOKUP_CHILD                                ;
                                                50  D5 00082           TSTL    ENTITYS_LABEL                                       ;
                                                12  13 00084           BEQL    10$                                                 ;
                                52      10      A0  9A 00086           MOVZBL  16(ENTITYS_LABEL), R2                               ;
                                51      10      A4  9A 0008A           MOVZBL  16(R4), R1                                          ;
        51              00      11      AC          52  2D 0008E       CMPC5   R2, 17(ENTITYS_LABEL), #0, R1, 17(R4)               ;
                                                11  A4      00094                                                                   ;
                                                10  11 00096           BRB     11$                                                 ;
                                51      10      A5  9A 00098 10$:      MOVZBL  16(KEYWORD), R1                                     ;
                                50      10      A4  9A 0009C           MOVZBL  16(R4), R0                                          ;
        50              00      11      A5          51  2D 000A0       CMPC5   R1, 17(KEYWORD), #0, R0, 17(R4)                     ;
                                                11  A4      000A6                                                                   ;
                                                06  13 000A8 11$:      BEQL    13$                                                 ;
                                55      04      A5  D0 000AA 12$:      MOVL    4(KEYWORD), KEYWORD                                 ;
                                                BA  11 000AE           BRB     8$                                                  ;
                                                55  D5 000B0 13$:      TSTL    KEYWORD                                             ; 1230
                                                2B  13 000B2           BEQL    16$                                                 ;
                                                69  D6 000B4           INCL    PATH_STACK_TOP                                      ; 1236
                                08              69  D1 000B6           CMPL    PATH_STACK_TOP, #8                                  ;
                                                0D  1F 000B9           BLSSU   14$                                                 ;
                            00000000G   8F      DD 000BB               PUSHL   #CDU$_INTPATHSTKOV                                  ;
                    00000000G   00              01  FB 000C1           CALLS   #1, LIB$SIGNAL                                      ;
                                50              69  D0 000C8 14$:      MOVL    PATH_STACK_TOP, R0                                  ;
                                56      04  A940  DE 000CB           MOVAL   PATH_STACK[R0], R6                                  ;
                                66              55  D0 000D0           MOVL    KEYWORD, (R6)                                       ;
                        04      AC      04      A4  D0 000D3           MOVL    4(R4), PATH_ENTITY                                  ; 1240
                                            FF3B    31 000D8           BRW     1$                                                  ; 1212
                                50              01  D0 000DB 15$:      MOVL    #1, R0                                              ; 1245
                                                    04 000DE           RET                                                         ;
                                                50  D4 000DF 16$:      CLRL    R0                                                  ; 1247
                                                    04 000E1           RET                                                         ;

; Routine Size:  226 bytes,    Routine Base:  $CODE$ + 0284
```

```
;  519          1248  1 !++
;  520          1249  1 ! Description:  This routine is called to report an error concerning a path.
;  521          1250  1 !               The routine is needed because the path string must be
;  522          1251  1 !               reconstructed from the intermediate representation.
;  523          1252  1 !
;  524          1253  1 ! Parameters:   message         By value, a message status code used for the
;  525          1254  1 !                               message.  It is assumed to take a two $FAO
;  526          1255  1 !                               arguments, a !UL for the line number and a
;  527          1256  1 !                               !AS for the path string.
;  528          1257  1 !               path            By reference, the top-level node
;  529          1258  1 !                               representing the path in error.
;  530          1259  1 !
;  531          1260  1 ! Returns:      Nothing.
;  532          1261  1 !
;  533          1262  1 ! Notes:
;  534          1263  1 !--
;  535          1264  1
;  536          1265  1 GLOBAL ROUTINE report_path_error(message: long,
;  537          1266  1                                  path: ref node)        : novalue
;  538          1267  2 = BEGIN
;  539          1268  2
;  540          1269  2 local
;  541          1270  2         path_string: vector[tkn_k_max_length,byte],
;  542          1271  2         path_string_dsc: descriptor,
;  543          1272  2         index: long initial(0),
;  544          1273  2         child: ref node;
;  545          1274  2
;  546          1275  2
;  547          1276  2 ! Build the path string from the intermediate representation.
;  548          1277  2 ! If the first child of the path node is a path definition node,
;  549          1278  2 ! then the user specified an explicit definition.  Place it at the
;  550          1279  2 ! front of the string in angle brackets.
;  551          1280  2
;  552          1281  2 child = .path[node_l_child];
;  553          1282  3 if .child[node_w_type] eqlu node_k_path_definition then (
;  554          1283  3         path_string[0] = '<';
;  555          1284  3         ch$move(.child[node_b_text_length],child[node_t_text], path_string[1]);
;  556          1285  3         index = 1 + .child[node_b_text_length];
;  557          1286  3         path_string[.index] = '>';
;  558          1287  3         increment(index);
;  559          1288  3         child = .child[node_l_sister];
;  560          1289  2 );
;  561          1290  2
;  562          1291  2 ! Now sit in a loop and concatenate all of the entity names onto the path
;  563          1292  2 ! string, with periods in between.
;  564          1293  2
;  565          1294  3 while .child nega 0 do (
;  566          1295  3         ch$move(.child[node_b_text_length],child[node_t_text], path_string[.index]);
;  567          1296  3         index = .index + .child[node_b_text_length];
;  568          1297  3         path_string[.index] = '.';
;  569          1298  3         increment(index);
;  570          1299  3         child = .child[node_l_sister];
;  571          1300  2 );
;  572          1301  2
;  573          1302  2 ! Build a descriptor of the node string a report the error.
;  574          1303  2
;  575          1304  2 build_descriptor(path_string_dsc,.index-1,path_string);
```

```
;  576          1305  2 cdu$report_semantic_error(.message,2,.path[node_w_line],path_string_dsc);
;  577          1306  2
;  578          1307  2 return;
;  579          1308  2
;  580          1309  1 END;


                                        01FC 00000          .ENTRY  REPORT_PATH_ERROR, Save R2,R3,R4,R5,R6,R7,- ; 1265
                                                                    R8
                                5E      FEF8    CE  9E 00002        MOVAB   -264(SP), SP
                                                57  D4 00007        CLRL    INDEX                                ; 1267
                                58        08    AC  D0 00009        MOVL    PATH, R8                             ; 1281
                                56        08    A8  D0 0000D        MOVL    8(R8), CHILD
                                30              66  B1 00011        CMPW    (CHILD), #48                         ; 1282
                                                1F  12 00014        BNEQ    2$
                          08    AE              3C  90 00016        MOVB    #60, PATH_STRING                     ; 1283
                                50        10    A6  9A 0001A        MOVZBL  16(CHILD), R0                        ; 1284
                09    AE        11    A6      50  28 0001E        MOVC3   R0, 17(CHILD), PATH_STRING+1
                                57        10    A6  9A 00024        MOVZBL  16(CHILD), INDEX                     ; 1285
                                                57  D6 00028        INCL    INDEX
                          08 AE47               3E  90 0002A        MOVB    #62, PATH_STRING[INDEX]              ; 1286
                                                57  D6 0002F 1$:     INCL    INDEX                                ; 1287
                                56        04    A6  D0 00031        MOVL    4(CHILD), CHILD                      ; 1288
                                                56  D5 00035 2$:     TSTL    CHILD                                ; 1294
                                                19  13 00037        BEQL    3$
                                50        10    A6  9A 00039        MOVZBL  16(CHILD), R0                        ; 1295
                       08 AE47        11    A6      50  28 0003D        MOVC3   R0, 17(CHILD), PATH_STRING[INDEX]
                                50        10    A6  9A 00044        MOVZBL  16(CHILD), R0                        ; 1296
                                57              50  C0 00048        ADDL2   R0, INDEX
                          08 AE47               2E  90 0004B        MOVB    #46, PATH_STRING[INDEX]              ; 1297
                                                DD  11 00050        BRB     1$                                   ; 1298
                     6E                 57      01  A3 00052 3$:     SUBW3   #1, INDEX, PATH_STRING_DSC           ; 1304
                                              02    AE  B4 00056        CLRW    PATH_STRING_DSC+2
                                04    AE        08  AE  9E 00059        MOVAB   PATH_STRING, PATH_STRING_DSC+4
                                                5E  DD 0005E        PUSHL   SP                                   ; 1305
                                7E              02  A8  3C 00060        MOVZWL  2(R8), -(SP)
                                                02  DD 00064        PUSHL   #2
                                              04    AC  DD 00066        PUSHL   MESSAGE
                    00000000G   00                  04  FB 00069        CALLS   #4, CDU$REPORT_SEMANTIC_ERROR
                                                    04 00070        RET                                          ; 1309

; Routine Size:  113 bytes,    Routine Base:  $CODE$ + 0366

;  581          1310  1
;  582          1311  1 END
;  583          1312  0 ELUDOM


                                                                    .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
```

```
;
;       Name                    Bytes                   Attributes
;
;  $OWN$                         36  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       983  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;
;
;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619         4         0      1000          00:01.9
;
;
; Information:  1
; Warnings:     0
; Errors:       0
;
;                                   COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:GENCODE4/OBJ=OBJ$:GENCODE4 MSRC$:GENCODE4/UPDATE=(ENH$:GENCODE4)
;
; Size:         983 code + 36 data bytes
; Run Time:         00:23.9
; Elapsed Time:     01:11.4
; Lines/CPU Min:     3289
; Lexemes/CPU-Min: 22463
; Memory Used:  187 pages
; Compilation Complete
```

GENRALREQ
R32

LEXICAL
LIS

GENCODE4
LIS

CLI5DEF
R32

CDUMSGS
LIS

CDU

CDU
MAP

GENCODE1
LIS

CDUREQ
R02

GENCODE3
LIS

GENCODE2
LIS

CDDTYPDEF
LIS