Specification Number CR-CP-0059-C84
Code Ident: FSCM 80063
November 1983

DRAFT

ADA* LANGUAGE SYSTEM
COMPILER MACHINE-INDEPENDENT SECTION
C-5 SPECIFICATION

CR-CP-0059-C84

VOLUME II

Contract No. DAAK80-80-C-0507

CDRL Item B010

Submitted to

Headquarters, United States Army
Communications Electronics Command
Fort Monmouth, NJ 07703

Prepared by

SofTech, Inc.
460 Totten Pond Road
Waltham, MA 02154

1075-32.1

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

3.2.2.230.12 Algorithm. -

```
while AS_ID_S list of decl is not at end> loop
  set sm_obj_type of id to as_type_spec of decl;
  set sm_init_exp of id to as_object_def of decl;
  advance position in seq of ids;
end loop;
```

3.2.2.230.13 Diagnostic Messages Generated. - None

3.2.2.230.14 Examples of Data Structures. - None

3.2.2.231 Subprogram. - constant_var

3.2.2.231.1 Purpose. - Perform name resolution on and declare a constant, parameter, variable component, or object declaration.

3.2.2.231.2 Assumptions. - None

3.2.2.231.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.231.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.231.5 Nested Within. - objects

3.2.2.231.6 Host Dependencies. - None

3.2.2.231.7 Target Dependencies. - None

3.2.2.231.8 Subprogram Visibility. - Within Package Only.

3.2.2.231.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.231.10 Formal Parameters. -

```
variable :  in  boolean; -- set to true if a variable
                           declaration
decl :  in  cdm_types.c_node_ref; -- the decl to be processed
```

3.2.2.231.11 Side-Effects. - Sets current_id (global) by side effects.

3.2.2.231.12 Algorithm. -

```
invoke "decls.id_s" with as_id_s of decl;
invoke "decls.as_type_spec";
if ERROR> then
  mark decl as in error;
  return;
end if;
if PARAM_FLAG not set> then
     check that decl's type is contrained;
     if ERROR> then
        invoke "diagnose.mg" for error 14895;
                 return;
     end if;
end if;
invoke "decls.obj_init";
if ERROR> then
  mark decl as error;
  return;
end if;
if AS_TYPE_SPEC indicates a () private type> then
  invoke "diagnose.msg" with decl for error 14086;
  return;
end if;
if A non-deffered constant or variable id> then
  reset sm_obj_type and sm_init_exp attribute of redefined
     any deferred constants to that of the redefiner;
  invoke "decls.set_cv_ids";
else invoke "decls.set_dc_ids";
end if;
```

3.2.2.231.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14086	E	objects must not be declared of a limited private type before full decration of that type
14895	E	type must be contrained in this context

3.2.2.231.14 Examples of Data Structures. - None

3.2.2.232 Subprogram. - exception_dec

3.2.2.232.1 Purpose. - Check and declare an exception declaration.

3.2.2.232.2 Assumptions. - None

3.2.2.232.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.232.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.232.5 Nested Within. - objects

3.2.2.232.6 Host Dependencies. - None

3.2.2.232.7 Target Dependencies. - None

3.2.2.232.8 Subprogram Visibility. - Within Package Only.

3.2.2.232.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.232.10 Formal Parameters. -

```
decl :  in  cdm_types.c_node_ref; -- the exception decl. to be
                                     processed
```

3.2.2.232.11 Side-Effects. - Set global current_id.

3.2.2.232.12 Algorithm. -

```
invoke "decls.ident"
set sm_exception_def of decl's id to
  as_exception_def of decl;
if AS_EXCEPTION_DEF is not void> then
  set current_id to id;
  invoke "decls.rename" with decl reset current_id to void;
end if;
```

3.2.2.232.13 Diagnostic Messages Generated. - None

3.2.2.232.14 Examples of Data Structures. - None

3.2.2.233 Package. - match

3.2.2.233.1 Purpose. - Contains routines to match subprograms with each other for equivalence.

3.2.2.233.2 Number of Subprograms. - None

3.2.2.233.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.233.4 Additional Dependencies for Body. - list_seq, c_node_a, get_ix, get_as, get_sm, traversal

3.2.2.233.5 Package Specification. -

```
-- visible outside package
type class is (equivalence,redeclaration);   -- type of matches possible
```

3.2.2.233.6 Elaboration Code. - None

3.2.2.233.7 Examples of Data Structures. - None

3.2.2.234 Package. - reps

3.2.2.234.1 Purpose. - Define a set of routines and auxiliary data structures to perform name resolution upon representation specifications.

3.2.2.234.2 Number of Subprograms. - 5

3.2.2.234.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.234.4 Additional Dependencies for Body. - vistree, search, exps, stm_s, derived, diagnose, new_nodes, get_as, put_as, get_sm, put_sm, get_ix, put_ix, get_vs, put_vs, get_ft, put_ft, list_seq, array_seq, cdm_b_str, prag_avns;

3.2.2.234.5 Package Specification. -

```
-- visible outside package
procedure simple_rep ...;
procedure record_rep ...;
procedure address ...;
```

3.2.2.234.6 Elaboration Code. - None

3.2.2.234.7 Examples of Data Structures. - None

3.2.2.235 Subprogram. - simple_rep

3.2.2.235.1 Purpose. - Checks semantics of and performs name resolution upon a simple representation specification.

3.2.2.235.2 Assumptions. - None

3.2.2.235.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.235.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.235.5 Nested Within. - reps

3.2.2.235.6 Host Dependencies. - None

3.2.2.235.7 Target Dependencies. - None

3.2.2.235.8 Subprogram Visibility. - Within Package Only.

3.2.2.235.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.235.10 Formal Parameters. -

```
s_rep :  in  c_node_rep; -- the simple rep. spec. to
                            be processed
```

3.2.2.235.11 Side-Effects. - None

3.2.2.235.12 Algorithm. -

```
invoke "EXPS.EXP" with as_name of s_rep;
if ERROR> then
  mark s_rep as in error;
  return;
end if;
check that name is either a type name, (not a subtype), or a
  derived type, with an attribute or a (derived or not) enumeration type
  name;
if ERROR> then
  invoke "diagnose.msg" with s_rep for error 14139;
  return;
end if;
if AN enumeration type> then
  check that only a selected or identifier form of name is used;
  if ERROR> then
    invoke "diagnose.msg" with s_rep for error 14140;
    return;
  end if;
else invoke "decls.attribute";
  check that this is only one of the following attributes :
    actual_delta, storage_size, size;
  if ERROR> then
    invoke "diagnose.msg" with s_rep for error 14141;
    return;
  end if;
end if;
if TYPE in either case above has derived user defined subprograms
    (a global attribute if type decl tells this)> then
  invoke "diagnose.msg" with s_rep for error 14142;
  return;
end if;
check that this type is declared in the same decl part only as this rep;
```

```
if ERROR> then
  invoke "diagnose.msg" with s_rep for error 14143;
  return;
end if;
invoke "EXPS.EXP" with as_exp of rep;
if ERROR> then
  mark s_rep as in error;
  return;
end if;
if AN enumeration type rep> then
  check that as_exp of s_rep is only a single aggregate;
  if ERROR> then
    invoke "diagnose.msg" with s_rep for error 14144;
    return;
  end if;
end if;
check that this type does not already have a enumeration type length
  spec. already given for it (check temporary attribute of the type
  decl);
if ERROR> then
  invoke "diagnose.msg" with s_rep for error 14145;
end if;
```

3.2.2.235.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14139	E	illegal form of representation specification for this type
14140	E	illegal form of name in this context
14141	E	illegal attribute for this representation specification
14142	E	representation specification must not be given for types with derived user defined subprograms
14143	E	representation specifications must only be given for types in the same declaration part
14144	E	illegal representation expression for an enumeration type
14145	E	illegal repetition of representation specification

3.2.2.235.14 Examples of Data Structures. - None

3.2.2.236 Subprogram. - record_rep

3.2.2.236.1 Purpose. - Check semantic validity of and perform name resolution on record representation specifications.

3.2.2.236.2 Assumptions. - None

3.2.2.236.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.236.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.236.5 Nested Within. - reps

3.2.2.236.6 Host Dependencies. - None

3.2.2.236.7 Target Dependencies. - None

3.2.2.236.8 Subprogram Visibility. - Within Package Only.

3.2.2.236.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.236.10 Formal Parameters. -

```
rep :  in  c_node_rep; -- the record rep. spec to be
                          processed
```

3.2.2.236.11 <u>Side-Effects</u>. - None

3.2.2.236.12 <u>Algorithm</u>. -

```
invoke "EXPS.EXP" with as_name of rep
if ERROR> then
  mark rep as error;
  return;
end if;
check that name is only a selected form or an identifier,
  indicating a record definition;
if ERROR> then
  invoke "diagnose.msg" with rep for error 14145;
  return;
end if;
check that record type is defined in the same decl part only
  (a field of possible def);
if ERROR> then
  invoke "diagose.msg" with rep for error 14147;
  return;
end if;
check that the record type does not already have a record rep set
  for it (temp. attribute);
if ERROR> then
  invoke "diagnose.msg" with rep for error 14148;
  return;
end if;
mark record type as "record rep seen";
if AS_EXP is not void> then
  invoke EXPS.EXP" with as_exp;
  if ERROR> then
    mark rep as in error;
    return;
  end if;
end if;
if A derived record type that has user defined subprograms it
    has derived (a temp. attribute of type)> then
  invoke "diagnose.msg" with rep for error 14149;
  return;
end if;
else invoke "decls.as_comp_rep_s";
end if;
if ERROR> then
  mark rep as in error;
end if;
```

3.2.2.236.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14146	E	illegal form of name in this context
14147	E	representation specifications must only be given for types in the same declaration part
14148	E	illegal repetition of representation specification
14149	E	Types with user defined subprograms must not be given representation specifications

3.2.2.236.14 Examples of Data Structures. - None

3.2.2.237 Subprogram. - as_comp_rep_s

3.2.2.237.1 Purpose. - Check semantic validity of and perform name resolution upon component representation specifications.

3.2.2.237.2 Assumptions. - None

3.2.2.237.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.237.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.237.5 Nested Within. - reps

3.2.2.237.6 Host Dependencies. - None

3.2.2.237.7 Target Dependencies. - None

3.2.2.237.8 Subprogram Visibility. - Within Package Only.

3.2.2.237.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.237.10 Formal Parameters. -

```
rep :  in  c_node_rep; -- the record rep containing the
                          component reps to be checked
```

3.2.2.237.11 Side-Effects. - None

3.2.2.237.12 Algorithm. -

```
while SEQUENCE of component representations not at end> loop
  invoke "EXPS.EXP" with as_name of component rep;
  if ERROR> then
    mark sequence as error;
    return;
  end if;
  check that as_name indicates a component of the record in
    selected component or identifier form only;
  if ERROR> then
    invoke "diagnose.msg" with sequence for error 14150;
    return;
  end if;
  if COMPONENT not already given a rep> then
    mark component as given a rep;
  else invoke "diagnose.msg" with sequence for error 14151;
    return;
  end if;
  invoke "EXPS.EXP" with as_exp of component rep;
  if ERROR> then
    mark sequence as error;
    return;
  end if;
  invoke "decls.as_range" with as_range of component rep;
  if ERROR> then
    mark sequence as error;
    return;
  end if;
  advance position in sequence;
```

```
end loop;
```

3.2.2.237.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14150	E	name must indicate a record component only
14151	E	illegal repetition of representation specification

3.2.2.237.14 Examples of Data Structures. - None

3.2.2.238 Subprogram. - reps

3.2.2.238.1 Purpose. - The particular representation specification is checked and a routine is called to perform name resolution on this particular rep spec.

3.2.2.238.2 Assumptions. - None

3.2.2.238.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.238.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.238.5 Nested Within. - reps

3.2.2.238.6 Host Dependencies. - None

3.2.2.238.7 Target Dependencies. - None

3.2.2.238.8 Subprogram Visibility. - Within Package Only.

3.2.2.238.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.238.10 Formal Parameters. -

```
rep :  in cdm_types.c_node_ref; -- the representation spec to be processed
```

3.2.2.238.11 Side-Effects. - None

3.2.2.238.12 Algorithm. -

```
CASE ,rep type> of
SIMPLE_REP>: invoke "decls.simple_rep";
RECORD_REP>: invoke "decls.record_rep";
ADDRESS>: invoke "decls.address";
end case;
```

3.2.2.238.13 Diagnostic Messages Generated. - None

3.2.2.238.14 Examples of Data Structures. - None

3.2.2.239 Subprogram. - address

3.2.2.239.1 Purpose. - This routine performs name resolution and checks semantics of address representation specifications.

3.2.2.239.2 Assumptions. - None

3.2.2.239.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.239.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.239.5 Nested within. - reps

3.2.2.239.6 Host Dependencies. - None

3.2.2.239.7 Target Dependencies. - None

3.2.2.239.8 Subprogram Visibility. - Within Package Only.

3.2.2.239.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.239.10 Formal Parameters. -

```
rep :  in  cdm_types.c_node_ref; -- the address repres. spec. to be
                                  processed
```

3.2.2.239.11 Side-Effects. - None

3.2.2.239.12 Algorithm. -

```
invoke "EXPS.EXP" with as_name of rep;
if ERROR> then
  mark rep as in error;
  return;
end if;
check that as_name is declared in the same decl part;
if ERROR> then
  invoke "diagnose.msg" with rep for error 14125;
  return;
end if;
check that name indicates a subprogram entry, package, task,
  or object only;
if ERROR> then
  invoke "diagnose.msg" with rep for error 14126;
  return;
end if;
mark the object as having had an address rep spec given for it;
invoke "EXPS.EXP" with as_exp of rep;
if ERROR> then
  mark rep as in error;
  return;
end if;
if A derived type of object that has derived
    user defined subprograms> then
  invoke "diagnose.msg" with rep for error 14127;
end if;
```

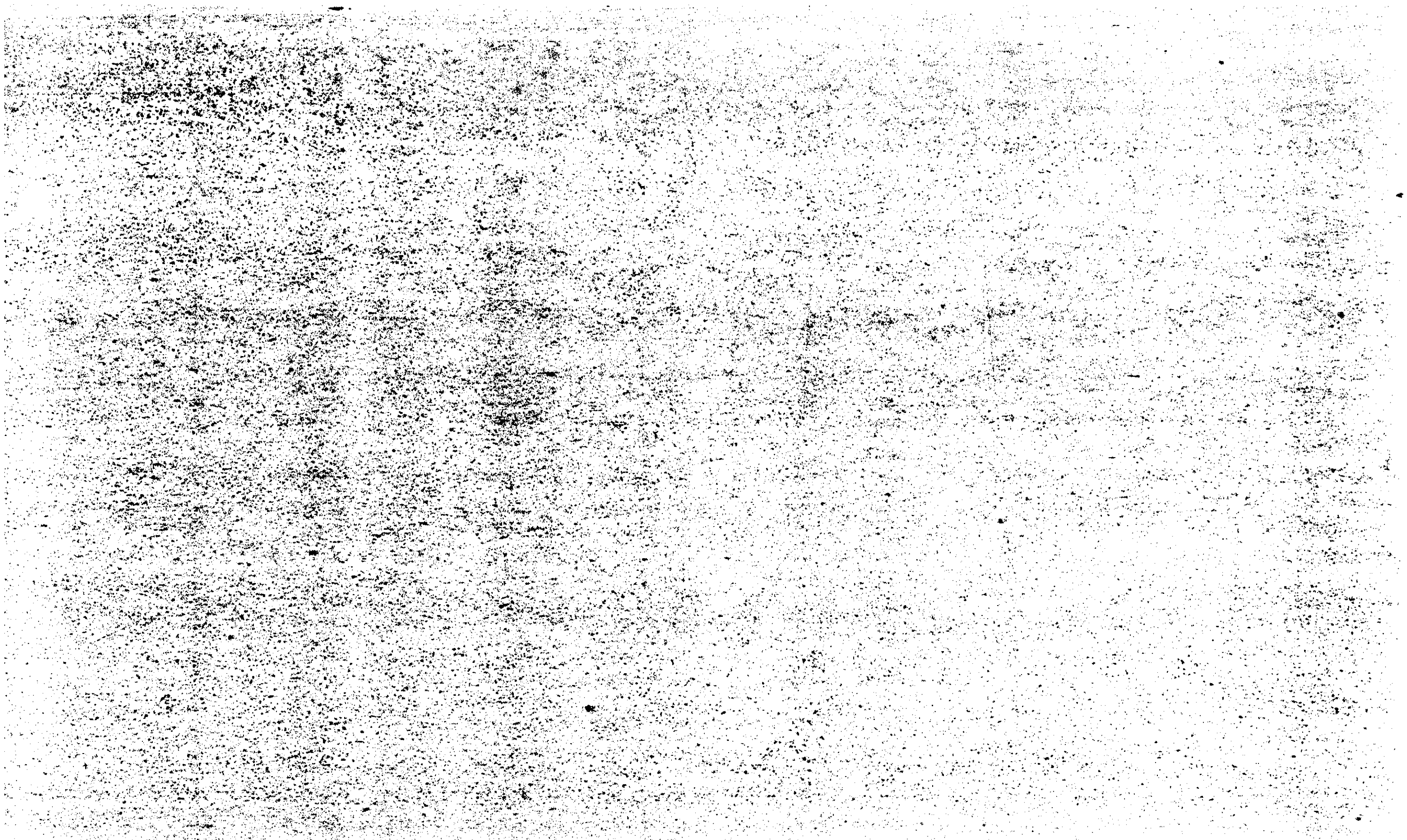
3.2.2.239.13 Diagnostic Messages Generated. -

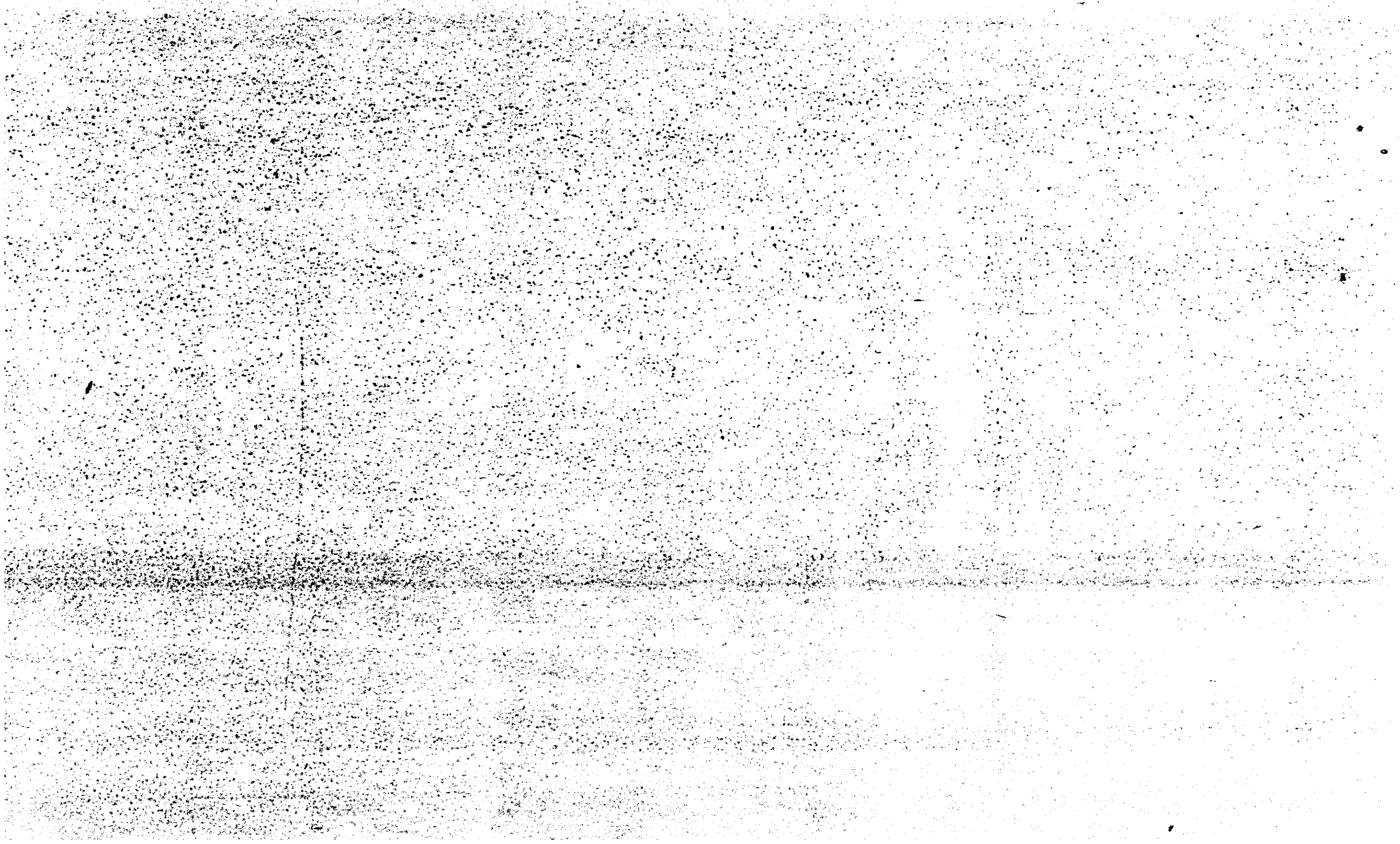
CODE	SEVERITY N.W.E.S.F	TEXT
14125	E	representation specifications must only be given for types in the same declarative par
14126	E	address cannot be given for the named construct
14127	E	representation specifications must not be given for types with derived user defined subprograms

3.2.2.239.14 Examples of Data Structures. - None

3.2.2.240 Package. - redefs

3.2.2.240.1 Purpose. - Define a set of routines and auxiliary data structures to enforce Ada restrictions upon redeclarations.





3.2.2.241.5 Nested Within. - redefs

3.2.2.241.6 Host Dependencies. - None

3.2.2.241.7 Target Dependencies. - None

3.2.2.241.8 Subprogram Visibility. - Within Package Only.

3.2.2.241.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.241.10 Formal Parameters. -

```
redef :  in  cdm_types.c_node_ref; -- the type being redeclared

decl :  in  cdm_types.c_node_ref; -- the type declaration doing
                                 the redeclaring
```

3.2.2.241.11 Side-Effects. - Modifies unredeclared private/incomplete types list.

3.2.2.241.12 Algorithm. -

```
check that discriminant parts match for the two declarations, or
  both are void;
if ERROR> then
  invoke "diagnose.msg" with decl for error 14091;
  return;
end if;
if AN incomplete type redefinition> then
  delete from unredeclared incomplete types list,
  after finding the decl. on the list using
  "search.search_redeclare_list";
      reset sm attribuytes of the id that is redeclared to that of the current
id;
else -- a private type redeclaration
  invoke "search.search_redeclare_list" with unredeclared private
    types list to find and delete the decl on the list;
  check that the type redeclaring the private type is not a task type, a
  () -private type, something with a component of one of these types, or
```

3.2.2.240.2 Number of Subprograms. - 7

3.2.2.240.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.240.4 Additional Dependencies for Body. - vistree, search, exps, stm_s, derived, diagnose, new_nodes, get_as, put_as, get_sm, put_sm, get_ix, put_ix, get_vs, put_vs, get_ft, put_ft, list_seq, array_seq, cdm_b_str;

3.2.2.240.5 Package Specification. -

```
-- visible outside package
procedure ident ...;
```

3.2.2.240.6 Elaboration Code. - None

3.2.2.240.7 Examples of Data Structures. - None

3.2.2.241 Subprogram. - itype_redef

3.2.2.241.1 Purpose. - Checks type declarations that redeclare private or incomplete types.

3.2.2.241.2 Assumptions. - None

3.2.2.241.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.241.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.242.5 Nested Within. - redefs

3.2.2.242.6 Host Dependencies. - None

3.2.2.242.7 Target Dependencies. - None

3.2.2.242.8 Subprogram Visibility. - Within Package Only.

3.2.2.242.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.242.10 Formal Parameters. -

```
redefs :  in  cdm_types.c_node_ref; -- the redefinitions (in same
                                 decl part) to be considered

decl :  in  cdm_types.c_node_ref; -- the decl currently being processed
```

3.2.2.242.11 Side-Effects. - None

3.2.2.242.12 Algorithm. -

```
while REDEFS not at end of list> and
      REDEF is in same decl part as decl> loop
  if (REDEFINITION is a subprogram equivalent to the current one>)
    and not (THEY are equivalent and redef is a spec and decl is
             a body>) or
     (THEY are equivalent and redef is an implicitly defined
       subprogram and this is a package spec>) then
    invoke "diagnose.msg" with decl for error 14133;
    exit;
  end if;
  advance position in list;
end loop;
invoke "vistree.add_dnt_entry";
if NO error> then
  truncate redefs list at current position;
else set redefs to null;
end if;
return NEW dnt>;
```

3.2.2.242.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14133	E	illegal redeclaration in same declaration part

3.2.2.242.14 Examples of Data Structures. - None

3.2.2.243 Subprogram. - type_redef

3.2.2.243.1 Purpose. - Determines the legality of previous declarations of an identifier in the same decl part, and returns a list if they are legal. The list is a special format.

3.2.2.243.2 Assumptions. - None

3.2.2.243.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.243.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.243.5 Nested Within. - redefs

3.2.2.243.6 Host Dependencies. - None

3.2.2.243.7 Target Dependencies. - None

3.2.2.243.8 Subprogram Visibility. - Within Package Only.

3.2.2.243.9 Function/Procedure. - This subprogram is a Function with a cdm_types.c_node_ref result type.

3.2.2.243.10 Formal Parameters. -

```
redefs :  in  cdm_types.c_node_ref; -- a list of previous (same id)
                                       decls in some decl part

decl :  in  cdm_types.c_node_ref; -- the decl currently being processed
```

3.2.2.243.11 Side-Effects. - None

3.2.2.243.12 Algorithm. -

```
if FIRST declaration on redefs list is an incomplete type
    or a private type> then
  set redef parameter to this type;
else invoke "diagnose.msg" with decl for error 14134;
  set redef parameter to null;
end if;
invoke "vistree.add_dnt_entry";
return NEW dnt>;
```

3.2.2.243.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14134	E	illegal redeclaration in same declaration part

3.2.2.243.14 Examples of Data Structures. - None

3.2.2.244 Subprogram. - module_body_redef

3.2.2.244.1 Purpose. - Determine the legality of previous declarations of an identifier in the same decl part, and returns a list if they are legal. The list is a special format.

3.2.2.244.2 Assumptions. - None

3.2.2.244.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.244.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.244.5 Nested Within. - redefs

3.2.2.244.6 Host Dependencies. - None

3.2.2.244.7 Target Dependencies. - None

3.2.2.244.8 Subprogram Visibility. - Within Package Only.

3.2.2.244.9 Function/Procedure. - This subprogram is a Function with a cdm_types.c_node_ref result type.

3.2.2.244.10 Formal Parameters. -

```
redefs :  in  cdm_types.c_node_ref; -- a list of previous decls of the
                                   same id in the same decl part

decl :  in  cdm_types.c_node_ref; -- the decl currently being processed
```

3.2.2.244.11 Side-Effects. - None

3.2.2.244.12 Algorithm. -

```
if ((FIRST decl on redefs list is a package spec) and
      DECL is a package body)>) or
   ((FIRST decl or redefs list is a task spec) and
     (decl is a task body)>) then
  set redefs to this spec;
else invoke "diagnose.msg" with decl for error 14135
  set redefs to void;
end if;
invoke "vistree.add_dnt_entry";
return NEW dnt>;
```

3.2.2.244.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14135	E	illegal redeclaration in same declaration part

3.2.2.244.14 Examples of Data Structures. - None

3.2.2.245 Subprogram. - const_redef

3.2.2.245.1 Purpose. - Determine the legality of previous declarations of an identifier in the same decl part, and returns a list if they are legal. The list is a special format.

3.2.2.245.2 Assumptions. - None

3.2.2.245.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.245.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.245.5 Nested Within. - redefs

3.2.2.245.6 Host Dependencies. - None

3.2.2.245.7 Target Dependencies. - None

3.2.2.245.8 Subprogram Visibility. - Within Package Only.

3.2.2.245.9 Function/Procedure. - This subprogram is a Function with a cdm_types.c_node_ref result type.

3.2.2.245.10 Formal Parameters. -

```
redefs :  in  cdm_types.c_node_ref; -- a list of previous declarations
                                  in the same decl

decl :  in  cdm_types.c_node_ref; -- the decl current being processed
```

3.2.2.245.11 Side-Effects. - None

3.2.2.245.12 Algorithm. -

```
if FIRST declaration is a deferred constant> then
  set redef to the constant
else invoke " diagnose.msg" with decl for error 14136;
  set redef to void
end if;
invoke "vistree.add_dnt_entry";
return NEW dnt>;
```

3.2.2.245.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14136	E	illegal redeclaration in same declaration part

3.2.2.245.14 Examples of Data Structures. - None

3.2.2.246 Subprogram. - enum_literal_redef

3.2.2.246.1 Purpose. - Determine the legality of previous declarations of an identifier in the same declarative part, and returns a list if they are legal. The list is a special format.

3.2.2.246.2 Assumptions. - None

3.2.2.246.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.246.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.246.5 Nested Within. - redefs

3.2.2.246.6 Host Dependencies. - None

3.2.2.246.7 Target Dependencies. - None

3.2.2.246.8 Subprogram Visibility. - Within Package Only.

3.2.2.246.9 Function/Procedure. - This subprogram is a Function with a cdm_types.c_node_ref result type.

3.2.2.246.10 Formal Parameters. -

```
redefs :  in  cdm_types.c_node_ref; -- a list of decls of the same id
                                    in the same decl part

decl :  in  cdm_types.c_node_ref; -- the decl currently being processe
```

3.2.2.246.11 Side-Effects. - None

3.2.2.246.12 Algorithm. -

```
while SEQUENCE of redefinitions is not at end> and
      REDEF is in same decl part as current> loop
  if REDEFINITION list is not an overloadable type> then
    invoke "diagnose.msg" with decl for error 14137;
    exit;
  end if;
  advance position in sequence;
end loop;
invoke "vistree.add_dnt_entry";
if NO error> then
  truncate redefs list at the current position;
end if;
return VOID>;
```

3.2.2.246.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
14137	E	illegal redeclaration in same declaration part

3.2.2.246.14 Examples of Data Structures. - None

3.2.2.247 Subprogram. - ident

3.2.2.247.1 Purpose. - Checks legality of a certain spelling of a declaration id. Returns legal previous declarations. Inserts an entry for this decl in the current dnt. Creates a new dnt (visibility structure (if this is a scope) and returns it. All previous declarations returned are in the same decl part.

3.2.2.247.2 Assumptions. - None

3.2.2.247.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.247.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.247.5 Nested Within. - redefs

3.2.2.247.6 Host Dependencies. - None

3.2.2.247.7 Target Dependencies. - None

3.2.2.247.8 Subprogram Visibility. - Within Package Only.

3.2.2.247.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.247.10 Formal Parameters. -

```
ident :  in  cdm_types.c_node_ref; -- the id of the decl being declared
decl :  in  cdm_types.c_node_ref; -- the decl being declared
```

```
redef : in cdm_types.c_node_ref; -- a list of legal previous
                                    redeclarations
```

3.2.2.247.11 Side-Effects. - None

3.2.2.247.12 Algorithm. -

```
invoke "search.ident" to find previous declarations of id with
  redecl param set to true;
if NOT found> or
   FIRST declaration on list is not part of the same decl part
     as the current decl> then
  invoke "vistree.add_dnt_entry" returning a dnt for this
     scope if necessary returning the new dnt;
else
case CURRENT decl type> of
SUBPROGAM spec or body>: invoke "decls,subprog_definitions"
     returning redef and new dnt;
TYPE>: invoke "decls.type_redef" returning redef and new dnt;
CONSTANT>: invoke "decls.const_redef" returning redef and
     new dnt of void;
PACKAGE_BODY,TASK_BODY>: invoke "decls.module_body_redef"
     returning redef and new dnt;
ENUM.LITERAL>: invoke "decls.enum_lit_redef" set new dnt to void;
OTHERS>: mark decl as in error;
           set new dnt to void;
end case;
end if;
```

3.2.2.247.13 Diagnostic Messages Generated. - None

3.2.2.247.14 Examples of Data Structures. - None

3.2.2.248 Package. - prag_attr

3.2.2.248.1 Purpose. - A target dependant package to check pragmas and attributes and to perform name resolution on them. This is for the PDP11/70 target only, to be the only prag_attr package for that target.

3.2.2.248.2 Number of Subprograms. - 15

3.2.2.248.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.248.4 Additional Dependencies for Body. - exps

3.2.2.248.5 Package Specification. -

```
--visible outside package
procedure pragmat...
procedure attribute...
```

3.2.2.248.6 Elaboration Code. - None

3.2.2.248.7 Examples of Data Structures. - None

3.2.2.249 Subprogram. - pragmat

3.2.2.249.1 Purpose. - Perform name resolution upon pragmas all targets.

3.2.2.249.2 Assumptions. - None

3.2.2.249.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.249.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.249.5 Nested within. - prag_attr

3.2.2.249.6 Host Dependencies. - None

3.2.2.249.7 Target Dependencies. - None

3.2.2.249.8 Subprogram Visibility. - Outside Package.

3.2.2.249.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.249.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- a reference to the pragma to be checked
```

3.2.2.249.11 Side-Effects. - None

3.2.2.249.12 Algorithm. -

```
case THE_PRAGMA is> if
PAGE>:  invoke "prag_attr.page";
TITLE, include, memory_size, priority,
 storage_unit, pack>:
     invoke "prag_attr.general_pragma";
INTERFACE>:  invoke "prag_attr.interface";
OPTIMIZE>:  invoke "prag_attr.optimize";
SUPPRESS>:  invoke "prag_attr.suppress";
SYSTEM>:  invoke "prag_attr.system";
INLINE>:  invoke "prag_attr.inline";
CONTROLLED>:  invoke "prag_attr.controlled";
LIST>:  invoke "prag_attr.list";
OTHERS>:  invoke "Diagnose.msg" with the_pragma for error 14033;
end case;
```

3.2.2.249.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14033	W	Pragma not recognized

3.2.2.249.14 Examples of Data Structures. - None

3.2.2.250 Subprogram. - system

3.2.2.250.1 Purpose. - Perform name resolution upon the system pragma for PDP11/UNIX target.

3.2.2.250.2 Assumptions. - None

3.2.2.250.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.250.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.250.5 Nested Within. - prag_attr

3.2.2.250.6 Host Dependencies. - None

3.2.2.250.7 Target Dependencies. - for the PDP 11 Unix target as in Aspec.

3.2.2.250.8 Subprogram Visibility. - Within Package Only.

3.2.2.250.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.250.10 <u>Formal Parameters</u>. -

the_pragma : in c_node_ref; -- the system pragma to be processed

3.2.2.250.11 <u>Side-Effects</u>. - None

3.2.2.250.12 <u>Algorithm</u>. -

```
check that there is only one argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14038;
     return;
end if;
invoke "EXPS.exo" with argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14039;
     return;
end if;
if ARGUMENT not part of enumeration
     type system. system_name>
then
     invoke "Diagnose.msg" with the_pragma for error 14040;
     return;
end if;
if ARGUMENT not PDP70_UNIX or
     PDP70> then
     invoke "Diagnose.msg" with the_pragma for error 14041;
end if;
```

3.2.2.250.13 <u>Diagnostic Messages Generated</u>. -

	SEVERITY	
<u>CODE</u>	<u>N.W.E.S.F</u>	<u>TEXT</u>
14038	E	system pragma should have only one argument
14039	E	pragma argument invalid
14040	E	Argument should be the system name PDP70_UNIX or PDP70
14041	W	argument should be the system name PDP70_UNIX or PDP70

3.2.2.250.14 Examples of Data Structures. - None

3.2.2.251 Subprogram. - inline

3.2.2.251.1 Purpose. - Performs name resolution upon the inline pragma for all targets.

3.2.2.251.2 Assumptions. - None

3.2.2.251.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.251.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.251.5 Nested within. - prag_attr

3.2.2.251.6 Host Dependencies. - None

3.2.2.251.7 Target Dependencies. - None

3.2.2.251.8 Subprogram Visibility. - Within Package Only.

3.2.2.251.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.251.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the inline pragma to be processed
```

3.2.2.251.11 Side-Effects. - None

3.2.2.251.12 Algorithm. -

```
while ARGUMENT list not at end>
loop
     invoke "EXPS.exp" to identify argument
     if ERROR> then
          invoke "Diagnose.msg" with the_pragma for error 14050;
     return;
     end if;
     check  that the argument is the name of a nongeneric
     subprogram declared in the same declaration part;
     if ERROR>
     then
          invoke "Diagnose.msg" with the_pragma for error 14051;
          return;
     end if;
advance position in list;
end loop;
```

3.2.2.251.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
14050	E	pragma argument invalid
14051	E	inline argument must be a nongeneric subprogram name

3.2.2.251.14 Examples of Data Structures. - None

3.2.2.252 Subprogram. - controlled

3.2.2.252.1 Purpose. - Perform name resolution upon a controlled pragma for all targets.

3.2.2.252.2 Assumptions. - None

3.2.2.252.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.252.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.252.5 Nested Within. - prag_attr

3.2.2.252.6 Host Dependencies. - None

3.2.2.252.7 Target Dependencies. - None

3.2.2.252.8 Subprogram Visibility. - Within Package Only.

3.2.2.252.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.252.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the pragma to be processed
```

3.2.2.252.11 Side-Effects. - None

3.2.2.252.12 Algorithm. -

```
check that the controlled pragma has only one argument;
if ERROR> then
      invoke "Diagnose.msg" with the_pragma for error 14052;
      return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
      invoke "Diagnose.msg" with the_pragma for error 14053;
      return;
```

```
end if;
check that argument is only an access type (not derived from one) declared in
the
same declarative part;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14054;
end if;
```

3.2.2.252.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14052	E	pragma should have only one argument
14053	E	pragma argument invalid
14054	E	controlled argument must be an access type

3.2.2.252.14 Examples of Data Structures. - None

3.2.2.253 Subprogram. - list

3.2.2.253.1 Purpose. - Perform name resolution upon the list pragma for all targets.

3.2.2.253.2 Assumptions. - None

3.2.2.253.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.253.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.253.5 Nested within. - prag_attr

3.2.2.253.6 Host Dependencies. - None

3.2.2.253.7 Target Dependencies. - None

3.2.2.253.8 Subprogram Visibility. - Within Package Only.

3.2.2.253.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.253.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the list pragma to be processed
```

3.2.2.253.11 Side-Effects. - None

3.2.2.253.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose_msg" with the_pragma for error 14055;
     return;
end if;
check that this argument is spelled "on" or "off";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14056;
end if;
```

3.2.2.253.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14055	E	pragma should have only one argument
14056	E	arguments to the list pragma should be on or off

3.2.2.253.14 Examples of Data Structures. - None

3.2.2.254 Subprogram. - page

3.2.2.254.1 Purpose. - Perform name resolution upon the page pragma for all targets.

3.2.2.254.2 Assumptions. - None

3.2.2.254.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.254.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.254.5 Nested within. - prag_attr

3.2.2.254.6 Host Dependencies. - None

3.2.2.254.7 Target Dependencies. - None

3.2.2.254.8 Subprogram Visibility. - Within Package Only.

3.2.2.254.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.254.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the page pragma to be processed
```

3.2.2.254.11 Side-Effects. - None

3.2.2.254.12 Algorithm. -

```
check that the_pragma has no arguments
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14058;
end if;
```

3.2.2.254.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.E	TEXT
14058	E	page pragma should have no arguments

3.2.2.254.14 Examples of Data Structures. - None

3.2.2.255 Subprogram. - general_pragma

3.2.2.255.1 Purpose. - Performs name resolution upon the pragmas for all targets.

3.2.2.255.2 Assumptions. - None

3.2.2.255.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.255.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.255.5 Nested Within. - prag_attr

3.2.2.255.6 Host Dependencies. - None

3.2.2.255.7 Target Dependencies. - None

3.2.2.255.8 Subprogram Visibility. - Within Package Only.

3.2.2.255.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.255.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- reference to pragma to be processed
```

3.2.2.255.11 Side-Effects. - None

3.2.2.255.12 Algorithm. -

```
check that pragma has only one argument;
if ERROR> then
      invoke "Diagnose.msg" with the_pragma for error 14059;
      return;
end if;
invoke "EXPS.exp"  with argument;
if ERROR> then
      invoke "Diagnose.msg" with the_pragma for error 14060;
end if;
case PRAGMA name> of
PACK>:
     check that argument names a record or arry type but not a
     derived type;
     if ERROR> then
        invoke "diagnose.msg" for error 14891;
     end if;
    OTHERS> : null;
    end if;
```

3.2.2.255.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14059;	E	pragma should only have one argument
14060	E	pragma argument invalid
14891	E	pack pragma must be given for a non-derived record or array only

3.2.2.255.14 Examples of Data Structures. - None

3.2.2.256 Subprogram. - interface

3.2.2.256.1 Purpose. - Perform name resolution upon the interface pragma for the PDP11/70 UNIX target.

3.2.2.256.2 Assumptions. - None

3.2.2.256.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.256.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.256.5 Nested Within. - prag_attr

3.2.2.256.6 Host Dependencies. - None

3.2.2.256.7 Target Dependencies. - For the PDP 11/70 Unix target as in the Aspec.

3.2.2.256.8 Subprogram Visibility. - Within Package Only.

3.2.2.256.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.256.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the interface pragma to be processed
```

3.2.2.256.11 Side-Effects. - None

3.2.2.256.12 Algorithm. -

```
check that the spelled argument is spelled "c" only ;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14061;
     return;
end if;
invoke "EXPS.exp" with second argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14062;
     return;
end if;
check that second argument is only a subprogram name;
if ERROR> then
  invoke "diagnose.msg" for error 14057;
if ANY more arguments exist>
then
     invoke "Diagnose.msg" with the_pragma for error 14063;
end if;
```

3.2.2.256.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14057	E	second argument must be a subprogram name
14061	W	invalid language to be interfaced
14062	E	second interface argument invalid
14063	E	interface pragma should have only two arguments

3.2.2.256.14 Examples of Data Structures. - None

3.2.2.257 Subprogram. - optimize

3.2.2.257.1 Purpose. - Performs name resolution upon the optimize pragma for all targets.

3.2.2.257.2 Assumptions. - None

3.2.2.257.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.257.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.257.5 Nested Within. - prag_attr

3.2.2.257.6 Host Dependencies. - None

3.2.2.257.7 Target Dependencies. - None

3.2.2.257.8 Subprogram Visibility. - Within Package Only.

3.2.2.257.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.257.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the optimize pragma to be processed
```

3.2.2.257.11 Side-Effects. - None

3.2.2.257.12 Algorithm. -

```
check that here is only 1 argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14064;
     return;
end if;
check that argument is spelled either "time" or "space";
if ERROR> then
    invoke "Diagnose.msg" with the_pragma for error 14065;
end if;
```

3.2.2.257.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.E	TEXT
14064	E	optimize pragma should have only one argument
14065	E	pragma argument invalid

3.2.2.257.14 Examples of Data Structures. - None

3.2.2.258 Subprogram. - suppress

3.2.2.258.1 Purpose. - Perform name resolution upon the suppress pragma for all targets.

3.2.2.258.2 Assumptions. - None

3.2.2.258.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.258.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.258.5 Nested Within. - prag_attr

3.2.2.258.6 Host Dependencies. - None

3.2.2.258.7 Target Dependencies. - None

3.2.2.258.8 Subprogram Visibility. - Within Package Only.

3.2.2.258.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.258.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the suppress pragma to be processed
```

3.2.2.258.11 Side-Effects. - None

3.2.2.258.12 Algorithm. -

```
check that first argument is only a "check_name";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14066;
     return;
end if;
if SECOND argument is present>
then
     invoke "EXPS.exp" with second argument;
     if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14067;
     return;
     end if;
  check that name is only an object or type name;
  if ERROR> then
    invoke "diagnose.mg" with the_pragma  for error 14395;
  end if;
end if;
```

3.2.2.258.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14066	E	first argument to suppress pragma must be a check name
14895	E	second argument must be a type or object name
14067	E	second argument invalid

3.2.2.258.14 Examples of Data Structures. - None

3.2.2.259 Subprogram. - attribute

3.2.2.259.1 Purpose. - Perform name resolution on and checks legality of an attribute that has no parameters, for the PDP70_UNIX target.

3.2.2.259.2 Assumptions. - None

3.2.2.259.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.259.4 Used Recursively. - This subprogram is used recursively.

3.2.2.259.5 Nested Within. - prag_attr

3.2.2.259.6 Host Dependencies. - None

3.2.2.259.7 Target Dependencies. - For the PDP 11/70 Unix target, as in the Aspec.

3.2.2.259.8 Subprogram Visibility. - Within Package Only.

3.2.2.259.9 Function/Procedure. - This subprogram is a Function with a c_node_ref result type.

3.2.2.259.10 Formal Parameters. -

```
attr : in  c_node_ref; -- the attribute to be processed
```

3.2.2.259.11 Side-Effects. - None

3.2.2.259.12 Algorithm. -

```
     invoke "exps.exp" with as_name of attr an
     names as context.
     if ERROR> then mark attr as in error.
              return VOID>; end if;
     case ATTR id> of
     ADDRESS> : name must indicate an objector subprog only.
     BASE> :  name must indicate a type or subtype that
            is not a task.
     SIZE> : name must indicate a non task type or subtype
            or an object.
     IMAGE, value> :
            name must indicate a scalar type or subtype.
     FIRST, last> :
            name must indicate a scalar type or subtype or
            an array type or subtype constrained
        only) or an object of array type or subtype.
     POS, val, pred, succ> :
            name must indicate a discrete type or subtype.
     DELTA, actual_delta, bits> :
            name must indicate a fixed point type.
     LARGE, macnine_rounds> :
            name must indicate a fixed or flp type.
     DIGIT, mantissa, emax, small,
            eosilon, machine_radix, machine_mantissa,
            machine_emax, machine_emin, machine_overflows> :
            name must indicate a floating point type or
            subtyoe only.
    LENGTH, range> : name must indicate an array
            type or subtype, or an object thereof,
            constrained only.
    CONSTRAINED> :  name must indicate a type with discriminants.
    POSITION, first_bit, last_bit> :
            name must indicate a record component.
    STORAGE_SIZE> : name must indicate an access type.
    TERMINATED, priority, failure, storage_size> :
            name must indicate a task object or type.
    COUNT> :
            name must indicate an entry.
    BASE> :
            name must be a type or subtype.
    OTHERS> : error;
    end case;
    if ERROR> and PRAGMA_FLAG not set> then
        invoke "diagnose.msg" with attr for error 14072
            return VOID>;
            else mark attr as error;
            return VOID>;
    end if;
    return attribute result type.
```

3.2.2.259.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14072	E	illegal attribute of this name.

3.2.2.259.14 Examples of Data Structures. - None

3.2.2.260 Package. - prag_attr

3.2.2.260.1 Purpose. - A target dependant package to check pragmas and attributes and to perform name resolution on them. This is for the ROLM1602_SA target only. Only this prag_attr package will be included for that target.

3.2.2.260.2 Number of Subprograms. - 15

3.2.2.260.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.260.4 Additional Dependencies for Body. - exps

3.2.2.260.5 Package Specification. -

```
  --visible outside package
  procedure pragmat...
  procedure attribute...
```

3.2.2.260.6 Elaboration Code. - None

3.2.2.260.7 Examples of Data Structures. - None

3.2.2.261 Subprogram. - pragmat

3.2.2.261.1 Purpose. - Perform name resolution upon pragmas all targets.

3.2.2.261.2 Assumptions. - None

3.2.2.261.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.261.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.261.5 Nested within. - prag_attr

3.2.2.261.6 Host Dependencies. - None

3.2.2.261.7 Target Dependencies. - None

3.2.2.261.8 Subprogram Visibility. - Outside Package.

3.2.2.261.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.261.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- a reference to the pragma to be checked
```

3.2.2.261.11 Side-Effects. - None

3.2.2.261.12 Algorithm. -

```
case THE_PRAGMA is> if
PAGE>:  invoke "prag_attr.page";
TITLE, include, memory_size, priority,
 storage_unit, pack>:
     invoke "prag_attr.general_pragma";
INTERFACE>:  invoke "prag_attr.interface";
OPTIMIZE>:  invoke "prag_attr.optimize";
SUPPRESS>:  invoke "prag_attr.suppress";
SYSTEM>:  invoke "prag_attr.system";
INLINE>:  invoke "prag_attr.inline";
CONTROLLED>:  invoke "prag_attr.controlled";
LIST>:  invoke "prag_attr.list";
OTHERS>:  invoke "Diagnose.msg" with the_pragma for error 14033;
end case;
```

3.2.2.261.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14033	W	Pragma not recognized

3.2.2.261.14 Examples of Data Structures. - None

3.2.2.262 Subprogram. - system

3.2.2.262.1 Purpose. - Perform name resolution upon the system pragma for ROLM 16028_SA target.

3.2.2.262.2 Assumptions. - None

3.2.2.262.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.262.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.262.5 Nested Within. - prag_attr

3.2.2.262.6 Host Dependencies. - None

3.2.2.262.7 Target Dependencies. - For the ROLM 1602 stand alone target as in Aspec.

3.2.2.262.8 Subprogram Visibility. - Within Package Only.

3.2.2.262.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.262.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the system pragma to be processed
```

3.2.2.262.11 Side-Effects. - None

3.2.2.262.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14034;
     return;
end if;
invoke "EXPS.EXP" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14035;
     return;
end if;
if ARGUMENT not part of enumeration type system
     system_name>
then
     invoke "Diagnose.msg" with the_pragma for error 14036;
     return;
end if
if ARGUMENT not ROLM1602B_SA or
     ROLM1602B
then
     invoke "Diagnose.msg" with the_pragma for error 14037;
end if;
```

3.2.2.262.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14034	E	system pragma should have only one argument
14035	E	Pragma argument invalid

14036	E	Argument should be the system name ROLM16023_SA or ROLM16028
14037	W	Argument should be the system name ROLM16023-SA or ROLM16028

3.2.2.262.14 Examples of Data Structures. - None

3.2.2.263 Subprogram. - inline

3.2.2.263.1 Purpose. - Performs name resolution upon the inline pragma for all targets.

3.2.2.263.2 Assumptions. - None

3.2.2.263.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.263.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.263.5 Nested Within. - prag_attr

3.2.2.263.6 Host Dependencies. - None

3.2.2.263.7 Target Dependencies. - None

3.2.2.263.8 Subprogram Visibility. - Within Package Only.

3.2.2.263.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.263.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the inline pragma to be processed
```

3.2.2.263.11 Side-Effects. - None

3.2.2.263.12 Algorithm. -

```
while ARGUMENT list not at end>
loop
     invoke "EXPS.exp" to identify argument
     if ERROR> then
          invoke "Diagnose.msg" with the_pragma for error 14050;
     return;
     end if;
     check  that the argument is the name of a nongeneric
     subprogram declared in the same declaration part;
     if ERROR>
     then
          invoke "Diagnose.msg" with the_pragma for error 14051;
          return;
     end if;
advance position in list;
end loop;
```

3.2.2.263.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14050	E	pragma argument invalid
14051	E	inline argument must be a nongeneric subprogram name

3.2.2.263.14 Examples of Data Structures. - None

3.2.2.264 Subprogram. - controlled

3.2.2.264.1 Purpose. - Perform name resolution upon a controlled pragma for all targets.

3.2.2.264.2 Assumptions. - None

3.2.2.264.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.264.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.264.5 Nested within. - prag_attr

3.2.2.264.6 Host Dependencies. - None

3.2.2.264.7 Target Dependencies. - None

3.2.2.264.8 Subprogram Visibility. - Within Package Only.

3.2.2.264.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.264.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the pragma to be processed
```

3.2.2.264.11 Side-Effects. - None

3.2.2.264.12 Algorithm. -

```
check that the controlled pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14052;
     return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14053;
     return;
end if;
check that argument is only an access type (not derived from one) declared in
the
same declarative part;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14054;
end if;
```

3.2.2.264.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14052	E	pragma should have only one argument
14053	E	pragma argument invalid
14054	E	controlled argument must be an access type

3.2.2.264.14 Examples of Data Structures. - None

3.2.2.265 Subprogram. - list

3.2.2.265.1 Purpose. - Perform name resolution upon the list pragma for all targets.

3.2.2.265.2 Assumptions. - None

3.2.2.265.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.265.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.265.5 Nested Within. - prag_attr

3.2.2.265.6 Host Dependencies. - None

3.2.2.265.7 Target Dependencies. - None

3.2.2.265.8 Subprogram Visibility. - Within Package Only.

3.2.2.265.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.265.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the list pragma to be processed
```

3.2.2.265.11 Side-Effects. - None

3.2.2.265.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose_msg" with the_pragma for error 14055;
     return;
end if;
check that this argument is spelled "on" or "off";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14056;
end if;
```

3.2.2.265.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14055	E	pragma should have only one argument
14056	E	arguments to the list pragma should be on or off

3.2.2.265.14 Examples of Data Structures. - None

3.2.2.266 Subprogram. - page

3.2.2.266.1 Purpose. - Perform name resolution upon the page pragma for all targets.

3.2.2.266.2 Assumptions. - None

3.2.2.266.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.266.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.266.5 Nested Within. - prag_attr

3.2.2.266.6 Host Dependencies. - None

3.2.2.266.7 Target Dependencies. - None

3.2.2.266.8 Subprogram Visibility. - Within Package Only.

3.2.2.266.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.266.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the page pragma to be processed
```

3.2.2.266.11 Side-Effects. - None

3.2.2.266.12 Algorithm. -

```
check that the_oragma has no arguments
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14058;
end if;
```

3.2.2.266.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
14058	E	page pragma should have no arguments

3.2.2.266.14 Examples of Data Structures. - None

3.2.2.267 Subprogram. - general_pragma

3.2.2.267.1 Purpose. - Performs name resolution upon the oragmas for all targets.

3.2.2.267.2 Assumptions. - None

3.2.2.267.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.267.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.267.5 Nested Within. - prag_attr

3.2.2.267.6 Host Dependencies. - None

3.2.2.267.7 Target Dependencies. - None

3.2.2.267.8 Subprogram Visibility. - Within Package Only.

3.2.2.267.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.267.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- reference to pragma to be processed
```

3.2.2.267.11 Side-Effects. - None

3.2.2.267.12 Algorithm. -

```
check that pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14059;
     return;
end if;
invoke "EXPS.exp"  with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14060;
end if;
```

```
case PRAGMA name> of
PACK>:
     check that argument names a record or arry type but not a
     derived type;
     if ERROR> then
        invoke "diagnose.msg" for error 14891;
     end if;
   OTHERS> : null;
   end if;
```

3.2.2.267.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14059;	E	pragma should only have one argument
14060	E	pragma argument invalid
14891	E	pack pragma must be given for a non-derived record or array only

3.2.2.267.14 Examples of Data Structures. - None

3.2.2.268 Subprogram. - optimize

3.2.2.268.1 Purpose. - Performs name resolution upon the optimize pragma for all targets.

3.2.2.268.2 Assumptions. - None

3.2.2.268.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.268.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.268.5 Nested Within. - prag_attr

3.2.2.268.6 Host Dependencies. - None

3.2.2.268.7 Target Dependencies. - None

3.2.2.268.8 Subprogram Visibility. - Within Package Only.

3.2.2.268.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.268.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the optimize pragma to be processed
```

3.2.2.268.11 Side-Effects. - None

3.2.2.268.12 Algorithm. -

```
check that here is only 1 argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14064;
     return;
end if;
check that argument is spelled either "time" or "space";
if ERROR> then
    invoke "Diagnose.msg" with the_pragma for error 14065;
end if;
```

3.2.2.268.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14064	E	optimize pragma should have only one argument
14065	E	pragma argument invalid

3.2.2.268.14 Examples of Data Structures. - None

3.2.2.269 Subprogram. - suppress

3.2.2.269.1 Purpose. - Perform name resolution upon the suppress pragma for all targets.

3.2.2.269.2 Assumptions. - None

3.2.2.269.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.269.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.269.5 Nested Within. - prag_attr

3.2.2.269.6 Host Dependencies. - None

3.2.2.269.7 Target Dependencies. - None

3.2.2.269.8 Subprogram Visibility. - Within Package Only.

3.2.2.269.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.269.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the suppress pragma to be processed
```

3.2.2.269.11 <u>Side-Effects</u>. - None

3.2.2.269.12 <u>Algorithm</u>. -

```
check that first argument is only a "check_name";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14066;
     return;
end if;
if SECOND argument is present>
then
     invoke "EXPS.exp" with second argument;
     if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14067;
     return;
     end if;
  check that name is only an object or type name;
  if ERROR> then
    invoke "diagnose.mg" with the_pragma  for error 14895;
  end if;
end if;
```

3.2.2.269.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
14066	E	first argument to suppress pragma must be a check name
14895	E	second argument must be a type or object name
14067	E	second argument invalid

3.2.2.269.14 <u>Examples of Data Structures</u>. - None

3.2.2.270 <u>Subprogram</u>. - attribute

3.2.2.270.1 Purpose. - Perform name resolution on and checks legality of an attribute that has no parameters, for the VAX/VMS target, the VAX_SA target, the ROLM 16028_SA target, and the ROLM1666_SA target.

3.2.2.270.2 Assumptions. - None

3.2.2.270.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.270.4 Used Recursively. - This subprogram is used recursively.

3.2.2.270.5 Nested Within. - prag_attr

3.2.2.270.6 Host Dependencies. - None

3.2.2.270.7 Target Dependencies. - For all targets except the PDP 11/70 Unix.

3.2.2.270.8 Subprogram Visibility. - Within Package Only.

3.2.2.270.9 Function/Procedure. - This subprogram is a Function with a c_node_ref result type.

3.2.2.270.10 Formal Parameters. -

```
attr : in  c_node_ref; -- the attribute to be processed
```

3.2.2.270.11 Side-Effects. - None

3.2.2.270.12 Algorithm. -

```
invoke "exps.exp" with as_name of attr an
names as context.
if ERROR> then mark attr as in error.
          return VOID>; end if;
case ATTR id> of
ADDRESS> : name must indicate an objector subprog only.
BASE> :  name must indicate a type or subtype that
      is not a task.
SIZE> : name must indicate a non task type or subtype
      or an object.
IMAGE, value> :
      name must indicate a scalar type or subtype.
FIRST, last> :
      name must indicate a scalar type or subtype or
      an array type or subtype constrained
   only) or an object of array type or subtype.
POS, val, pred, succ> :
      name must indicate a discrete type or subtype.
DELTA, actual_delta, bits> :
      name must indicate a fixed point type.
LARGE, machine_rounds> :
      name must indicate a fixed or flp type.
DIGIT, mantissa, emax, small,
      epsilon, machine_radix, machine_mantissa,
      machine_emax, machine_emin, machine_overflows> :
      name must indicate a floating point type or
      subtype only.
LENGTH, range> : name must indicate an array
      type or subtype, or an object thereof,
      constrained only.
CONSTRAINED> :  name must indicate a type with discriminants.
POSITION, first_bit, last_bit> :
      name must indicate a record component.
STORAGE_SIZE> : name must indicate an access type.
TERMINATED, priority, failure, storage_size> :
      name must indicate a task object or type.
COUNT> :
      name must indicate an entry.
BASE> :
      name must be a type or subtype.
DISP> : name must indicate a (potential) parameter to a
      code statement;
OTHERS> : error;
end case;
if ERROR> and PRAGMA_FLAG not set> then
   invoke "diagnose.msg" with attr for error 14006
      return VOID>;
      else mark attr as error;
      return VOID>;
end if;
return attribute result type.
```

3.2.2.270.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14006	E	illegal attribute of this name.

3.2.2.270.14 Examples of Data Structures. - None

3.2.2.271 Subprogram. - interface

3.2.2.271.1 Purpose. - Perform name resolution upon the interface pragma for the VAX and ROLM targets.

3.2.2.271.2 Assumptions. - None

3.2.2.271.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.271.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.271.5 Nested within. - prag_attr

3.2.2.271.6 Host Dependencies. - None

3.2.2.271.7 Target Dependencies. - For all targets except the PDP 11/70.

3.2.2.271.8 Subprogram Visibility. - Within Package Only.

3.2.2.271.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.271.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the interface pragma to be processed
```

3.2.2.271.11 Side-Effects. - None

3.2.2.271.12 Algorithm. -

```
invoke "EXPS.exp" with second argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14062;
     return;
end if;
check that second argument is only a subprogram name;
if ERROR> then
  invoke "diagnose.msg" for error 14057;
if ANY more arguments exist>
then
     invoke "Diagnose.msg" with the_pragma for error 14063;
end if;
```

3.2.2.271.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
14057	E	second argument must be a subprogram name
14061	W	invalid language to be interfaced
14062	E	second interface argument invalid
14063	E	interface pragma should have only two arguments

3.2.2.271.14 Examples of Data Structures. - None

3.2.2.272 Package. - prag_attr

3.2.2.272.1 Purpose. - A target dependant package to check pragmas and attributes and to perform name resolution on them. This is for the ROLM1666_SA target only. Only this prag_attr package will be included for that target.

3.2.2.272.2 Number of Subprograms. - 15

3.2.2.272.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.272.4 Additional Dependencies for Body. - exps

3.2.2.272.5 Package Specification. -

```
  --visible outside package
  procedure pragmat...
  procedure attribute...
```

3.2.2.272.6 Elaboration Code. - None

3.2.2.272.7 Examples of Data Structures. - None

3.2.2.273 Subprogram. - pragmat

3.2.2.273.1 Purpose. - Perform name resolution upon pragmas all targets.

3.2.2.273.2 Assumptions. - None

3.2.2.273.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.273.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.273.5 Nested Within. - prag_attr

3.2.2.273.6 Host Dependencies. - None

3.2.2.273.7 Target Dependencies. - None

3.2.2.273.8 Subprogram Visibility. - Outside Package.

3.2.2.273.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.273.10 Formal Parameters. -

the_pragma : in c_node_ref; -- a reference to the pragma to be checked

3.2.2.273.11 Side-Effects. - None

3.2.2.273.12 Algorithm. -

```
case THE_PRAGMA is> if
PAGE>:  invoke "prag_attr.page";
TITLE, include, memory_size, priority,
 storage_unit, pack>:
     invoke "prag_attr.general_pragma";
INTERFACE>:  invoke "prag_attr.interface";
OPTIMIZE>:  invoke "prag_attr.optimize";
SUPPRESS>:  invoke "prag_attr.suppress";
SYSTEM>:  invoke "prag_attr.system";
INLINE>:  invoke "prag_attr.inline";
CONTROLLED>:  invoke "prag_attr.controlled";
LIST>:  invoke "prag_attr.list";
OTHERS>:  invoke "Diagnose.msg" with the_pragma for error 14033;
```

end case;

3.2.2.273.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14033	W	Pragma not recognized

3.2.2.273.14 Examples of Data Structures. - None

3.2.2.274 Subprogram. - inline

3.2.2.274.1 Purpose. - Performs name resolution upon the inline pragma for all targets.

3.2.2.274.2 Assumptions. - None

3.2.2.274.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.274.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.274.5 Nested Within. - prag_attr

3.2.2.274.6 Host Dependencies. - None

3.2.2.274.7 Target Dependencies. - None

3.2.2.274.8 Subprogram Visibility. - Within Package Only.

3.2.2.274.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.274.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the inline pragma to be processed
```

3.2.2.274.11 Side-Effects. - None

3.2.2.274.12 Algorithm. -

```
while ARGUMENT list not at end>
loop
     invoke "EXPS.exp" to identify argument
     if ERROR> then
          invoke "Diagnose.msg" with the_pragma for error 14050;
     return;
     end if;
     check  that the argument is the name of a nongeneric
     subprogram declared in the same declaration part;
     if ERROR>
     then
          invoke "Diagnose.msg" with the_pragma for error 14051;
          return;
     end if;
advance position in list;
end loop;
```

3.2.2.274.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14050	E	pragma argument invalid
14051	E	inline argument must be a nongeneric subprogram name

3.2.2.274.14 Examples of Data Structures. - None

3.2.2.275 Subprogram. - controlled

3.2.2.275.1 Purpose. - Perform name resolution upon a controlled pragma for all targets.

3.2.2.275.2 Assumptions. - None

3.2.2.275.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.275.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.275.5 Nested Within. - prag_attr

3.2.2.275.6 Host Dependencies. - None

3.2.2.275.7 Target Dependencies. - None

3.2.2.275.8 Subprogram Visibility. - Within Package Only.

3.2.2.275.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.275.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the pragma to be processed
```

3.2.2.275.11 <u>Side-Effects</u>. - None

3.2.2.275.12 <u>Algorithm</u>. -

```
check that the controlled pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14052;
     return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14053;
     return;
end if;
check that argument is only an access type (not derived from one) declared in
the
same declarative part;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14054;
end if;
```

3.2.2.275.13 <u>Diagnostic Messages Generated</u>. -

	SEVERITY	
<u>CODE</u>	<u>N.W.E.S.F</u>	<u>TEXT</u>
14052	E	pragma should have only one argument
14053	E	pragma argument invalid
14054	E	controlled argument must be an access type

3.2.2.275.14 <u>Examples of Data Structures</u>. - None

3.2.2.276 <u>Subprogram</u>. - list

3.2.2.276.1 <u>Purpose</u>. - Perform name resolution upon the list pragma for all targets.

3.2.2.276.2 Assumptions. - None

3.2.2.276.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.276.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.276.5 Nested Within. - prag_attr

3.2.2.276.6 Host Dependencies. - None

3.2.2.276.7 Target Dependencies. - None

3.2.2.276.8 Subprogram Visibility. - Within Package Only.

3.2.2.276.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.276.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the list pragma to be processed
```

3.2.2.276.11 Side-Effects. - None

3.2.2.276.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose_msg" with the_pragma for error 14055;
     return;
end if;
check that this argument is spelled "on" or "off";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14056;
end if;
```

3.2.2.276.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14055	E	pragma should have only one argument
14056	E	arguments to the list pragma should be on or off

3.2.2.276.14 Examples of Data Structures. - None

3.2.2.277 Subprogram. - page

3.2.2.277.1 Purpose. - Perform name resolution upon the page pragma for all targets.

3.2.2.277.2 Assumptions. - None

3.2.2.277.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.277.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.277.5 Nested within. - prag_attr

3.2.2.277.6 Host Dependencies. - None

3.2.2.277.7 Target Dependencies. - None

3.2.2.277.8 Subprogram Visibility. - Within Package Only.

3.2.2.277.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.277.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the page pragma to be processed
```

3.2.2.277.11 Side-Effects. - None

3.2.2.277.12 Algorithm. -

```
check that the_pragma has no arguments
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14058;
end if;
```

3.2.2.277.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14058	E	page pragma should have no arguments

3.2.2.277.14 Examples of Data Structures. - None

3.2.2.278 Subprogram. - general_pragma

3.2.2.278.1 Purpose. - Performs name resolution upon the pragmas for all targets.

3.2.2.278.2 Assumptions. - None

3.2.2.278.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.278.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.278.5 Nested Within. - prag_attr

3.2.2.278.6 Host Dependencies. - None

3.2.2.278.7 Target Dependencies. - None

3.2.2.278.8 Subprogram Visibility. - Within Package Only.

3.2.2.278.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.278.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- reference to pragma to be processed
```

3.2.2.278.11 Side-Effects. - None

3.2.2.278.12 Algorithm. -

```
check that pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14059;
     return;
end if;
invoke "EXPS.exp"  with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14060;
end if;
```

```
case PRAGMA name> of
PACK>:
    check that argument names a record or arry type but not a
    derived type;
    if ERROR> then
       invoke "diagnose.msg" for error 14891;
    end if;
   OTHERS> : null;
   end if;
```

3.2.2.278.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14059;	E	pragma should only have one argument
14060	E	pragma argument invalid
14891	E	pack pragma must be given for a non-derived record or array only

3.2.2.278.14 Examples of Data Structures. - None

3.2.2.279 Subprogram. - optimize

3.2.2.279.1 Purpose. - Performs name resolution upon the optimize pragma for all targets.

3.2.2.279.2 Assumptions. - None

3.2.2.279.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.279.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.279.5 Nested within. - prag_attr

3.2.2.279.6 Host Dependencies. - None

3.2.2.279.7 Target Dependencies. - None

3.2.2.279.8 Subprogram Visibility. - Within Package Only.

3.2.2.279.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.279.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the optimize pragma to be processed
```

3.2.2.279.11 Side-Effects. - None

3.2.2.279.12 Algorithm. -

```
check that here is only 1 argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14064;
     return;
end if;
check that argument is spelled either "time" or "space";
if ERROR> then
    invoke "Diagnose.msg" with the_pragma for error 14065;
end if;
```

3.2.2.279.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14064	E	optimize pragma should have only one argument
14065	E	pragma argument invalid

3.2.2.279.14 Examples of Data Structures. - None

3.2.2.280 Subprogram. - suppress

3.2.2.280.1 Purpose. - Perform name resolution upon the suppress pragma for all targets.

3.2.2.280.2 Assumptions. - None

3.2.2.280.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.280.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.280.5 Nested within. - prag_attr

3.2.2.280.6 Host Dependencies. - None

3.2.2.280.7 Target Dependencies. - None

3.2.2.280.8 Subprogram Visibility. - Within Package Only.

3.2.2.280.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.280.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the suppress pragma to be processed
```

3.2.2.280.11 Side-Effects. - None

3.2.2.280.12 Algorithm. -

```
check that first argument is only a "check_name";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14066;
     return;
end if;
if SECOND argument is present>
then
     invoke "EXPS.exp" with second argument;
     if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14067;
     return;
     end if;
  check that name is only an object or type name;
  if ERROR> then
    invoke "diagnose.mg" with the_pragma  for error 14895;
  end if;
end if;
```

3.2.2.280.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14066	E	first argument to suppress pragma must be a check name
14895	E	second argument must be a type or object name
14067	E	second argument invalid

3.2.2.280.14 Examples of Data Structures. - None

3.2.2.281 Subprogram. - system

3.2.2.281.1 Purpose. - Perform name resolution upon the system pragma for ROLM1666 target.

3.2.2.281.2 Assumptions. - None

3.2.2.281.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.281.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.281.5 Nested within. - prag_attr

3.2.2.281.6 Host Dependencies. - None

3.2.2.281.7 Target Dependencies. - For the ROLM 1666 target as in the Aspec.

3.2.2.281.8 Subprogram Visibility. - Within Package Only.

3.2.2.281.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.281.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the system pragma to be processed
```

3.2.2.281.11 Side-Effects. - None

3.2.2.281.12 Algorithm. -

```
check that there is only one argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14068;
     return;
```

```
end if;
invoke "EXPS.exo" with argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14069;
     return;
end if;
if ARGUMENT not part of enumeration type system system_name>
then
     invoke "Diagnose.msg" with the_pragma for error 14070;
     return;
end if;
if ARGUMENT not ROLM1666_SA or
ROLM1666>
then
     invoke "Diagnose.msg" with the_pragma for error 14071;
end if;
```

3.2.2.281.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.E	TEXT
14068	E	system pragma should hava only one argument
14069	E	pragma argument invalid
14070	W	argument should be the system name ROLM1666_SA or ROLM1666
14071	W	argument should be the system name ROLM1666B_SA or ROLM1666

3.2.2.281.14 Examples of Data Structures. - None

3.2.2.282 Subprogram. - attribute

3.2.2.282.1 Purpose. - Perform name resolution on and checks legality of an attribute that has no parameters, for the VAX/VMS target, the VAX_SA target, the ROLM 1602B_SA target, and the ROLM1666_SA target.

3.2.2.282.2 Assumptions. - None

3.2.2.282.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.282.4 Used Recursively. - This subprogram is used recursively.

3.2.2.282.5 Nested Within. - prag_attr

3.2.2.282.6 Host Dependencies. - None

3.2.2.282.7 Target Dependencies. - For all targets except the PDP 11/70 Unix.

3.2.2.282.8 Subprogram Visibility. - Within Package Only.

3.2.2.282.9 Function/Procedure. - This subprogram is a Function with a c_node_ref result type.

3.2.2.282.10 Formal Parameters. -

```
attr : in  c_node_ref; -- the attribute to be processed
```

3.2.2.282.11 Side-Effects. - None

3.2.2.282.12 Algorithm. -

```
     invoke "exps.exp" with as_name of attr an
     names as context.
     if ERROR> then mark attr as in error.
             return VOID>; end if;
     case ATTR id> of
     ADDRESS> : name must indicate an objector subprog only.
     BASE> :  name must indicate a type or subtype that
            is not a task.
```

```
SIZE> : name must indicate a non task type or subtype
      or an object.
IMAGE, value> :
      name must indicate a scalar type or subtype.
FIRST, last> :
      name must indicate a scalar type or subtype or
      an array type or subtype constrained
   only) or an object of array type or subtype.
POS, val, pred, succ> :
      name must indicate a discrete type or subtype.
DELTA, actual_delta, bits> :
      name must indicate a fixed point type.
LARGE, machine_rounds> :
      name must indicate a fixed or flp type.
DIGIT, mantissa, emax, small,
      epsilon, machine_radix, machine_mantissa,
      machine_emax, machine_emin, machine_overflows> :
      name must indicate a floating point type or
      subtype only.
LENGTH, range> : name must indicate an array
      type or subtype, or an object thereof,
      constrained only.
CONSTRAINED> :  name must indicate a type with discriminants.
POSITION, first_bit, last_bit> :
      name must indicate a record component.
STORAGE_SIZE> : name must indicate an access type.
TERMINATED, priority, failure, storage_size> :
      name must indicate a task object or type.
COUNT> :
      name must indicate an entry.
BASE> :
      name must be a type or subtype.
DISP> : name must indicate a (potential) parameter to a
      code statement;
OTHERS> : error;
end case;
if ERROR> and PRAGMA_FLAG not set> then
   invoke "diagnose.msg" with attr for error 14006
      return VOID>;
      else mark attr as error;
      return VOID>;
end if;
return attribute result type.
```

3.2.2.282.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14006	E	illegal attribute of this name.

3.2.2.282.14 Examples of Data Structures. - None

3.2.2.283 Subprogram. - interface

3.2.2.283.1 Purpose. - Perform name resolution upon the interface pragma for the VAX and ROLM targets.

3.2.2.283.2 Assumptions. - None

3.2.2.283.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.283.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.283.5 Nested within. - prag_attr

3.2.2.283.6 Host Dependencies. - None

3.2.2.283.7 Target Dependencies. - For all targets except the PDP 11/70.

3.2.2.283.8 Subprogram Visibility. - Within Package Only.

3.2.2.283.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.283.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the interface pragma to be processed
```

3.2.2.283.11 Side-Effects. - None

3.2.2.283.12 Algorithm. -

```
invoke "EXPS.exo" with second argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14062;
     return;
end if;
check that second argument is only a subprogram name;
if ERROR> then
  invoke "diagnose.msg" for error 14057;
if ANY more arguments exist>
then
     invoke "Diagnose.msg" with the_pragma for error 14063;
end if;
```

3.2.2.283.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14057	E	second argument must be a subprogram name
14061	W	invalid language to be interfaced
14062	E	second interface argument invalid
14063	E	interface pragma should have only two arguments

3.2.2.283.14 Examples of Data Structures. - None

3.2.2.284 Package. - prag_attr

3.2.2.284.1 Purpose. - A target dependant package to check pragmas and attributes and to perform name resolution on them. This is for the VAX780_VMS target only. Only this prag_attr package will be included for that target.

3.2.2.284.2 Number of Subprograms. - 11

3.2.2.284.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.284.4 Additional Dependencies for Body. - exps

3.2.2.284.5 Package Specification. -

```
  --visible outside package
  procedure pragmat...
  procedure attribute...
```

3.2.2.284.6 Elaboration Code. - None

3.2.2.284.7 Examples of Data Structures. - None

3.2.2.285 Subprogram. - pragmat

3.2.2.285.1 Purpose. - Perform name resolution upon pragmas all targets.

3.2.2.285.2 Assumptions. - None

3.2.2.285.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.285.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.285.5 Nested within. - prag_attr

3.2.2.285.6 Host Dependencies. - None

3.2.2.285.7 Target Dependencies. - None

3.2.2.285.8 Subprogram Visibility. - Outside Package.

3.2.2.285.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.285.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- a reference to the pragma to be checked
```

3.2.2.285.11 Side-Effects. - None

3.2.2.285.12 Algorithm. -

```
case THE_PRAGMA is> if
PAGE>:  invoke "prag_attr.page";
TITLE, include, memory_size, priority,
 storage_unit, pack>:
     invoke "prag_attr.general_pragma";
INTERFACE>:  invoke "prag_attr.interface";
OPTIMIZE>:  invoke "prag_attr.optimize";
SUPPRESS>:  invoke "prag_attr.suppress";
SYSTEM>:  invoke "prag_attr.system";
INLINE>:  invoke "prag_attr.inline";
CONTROLLED>:  invoke "prag_attr.controlled";
LIST>:  invoke "prag_attr.list";
OTHERS>:  invoke "Diagnose.msg" with the_pragma for error 14033;
end case;
```

3.2.2.285.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
14033	W	Pragma not recognized

3.2.2.285.14 <u>Examples of Data Structures</u>. - None

3.2.2.286 <u>Subprogram</u>. - system

3.2.2.286.1 <u>Purpose</u>. - Perform name resolution upon the system pragma for VAX780_VMS target.

3.2.2.286.2 <u>Assumptions</u>. - None

3.2.2.286.3 <u>Implementation Language</u>. - This subprogram is written in Ada.

3.2.2.286.4 <u>Used Recursively</u>. - This subprogram is not used recursively.

3.2.2.286.5 <u>Nested within</u>. - prag_attr

3.2.2.286.6 <u>Host Dependencies</u>. - None

3.2.2.286.7 <u>Target Dependencies</u>. - For the Vax 780 VMS target as in the Aspec.

3.2.2.286.8 <u>Subprogram Visibility</u>. - Within Package Only.

3.2.2.286.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.286.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the system pragma to be processed
```

3.2.2.286.11 Side-Effects. - None

3.2.2.286.12 Algorithm. -

```
check that there is only one argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14046;
     return;
end if;
invoke "EXPS.exp" with argument
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14047;
     return;
end if;
if ARGUMENT not part of
     enumeration type system
     system_name>
then
     invoke "Diagnose.msg" with the_pragma for error 14048;
     return;
end if;
if ARGUMENT not VAX780_VMS or
     VAX780
then
     invoke "Diagnose.msg" with the_pragma for error 14049;
end if;
```

3.2.2.286.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14046	E	system pragma should have only one argument
14047	E	pragma argument invalid
14048	E	argument must be the system name VAX780_VMS or VAX780

14049 * argument should be the system name PDP70_UNIX or
@@@@@ PDP70

3.2.2.286.14 Examples of Data Structures. - None

3.2.2.287 Subprogram. - inline

3.2.2.287.1 Purpose. - Performs name resolution upon the inline pragma for all targets.

3.2.2.287.2 Assumptions. - None

3.2.2.287.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.287.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.287.5 Nested Within. - prag_attr

3.2.2.287.6 Host Dependencies. - None

3.2.2.287.7 Target Dependencies. - None

3.2.2.287.8 Subprogram Visibility. - Within Package Only.

3.2.2.287.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.287.10 <u>Formal Parameters</u>. -

the_pragma : in c_node_ref; -- the inline pragma to be processed

3.2.2.287.11 <u>Side-Effects</u>. - None

3.2.2.287.12 <u>Algorithm</u>. -

```
while ARGUMENT list not at end>
loop
     invoke "EXPS.exp" to identify argument
     if ERROR> then
          invoke "Diagnose.msg" with the_pragma for error 14050;
     return;
     end if;
     check  that the argument is the name of a nongeneric
     subprogram declared in the same declaration part;
     if ERROR>
     then
          invoke "Diagnose.msg" with the_pragma for error 14051;
          return;
     end if;
advance position in list;
end loop;
```

3.2.2.287.13 <u>Diagnostic Messages Generated</u>. -

	SEVERITY	
<u>CODE</u>	<u>N.W.E.S.F</u>	<u>TEXT</u>
14050	E	pragma argument invalid
14051	E	inline argument must be a nongeneric subprogram name

3.2.2.287.14 <u>Examples of Data Structures</u>. - None

3.2.2.288 Subprogram. - controlled

3.2.2.288.1 Purpose. - Perform name resolution upon a controlled pragma for all targets.

3.2.2.288.2 Assumptions. - None

3.2.2.288.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.288.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.288.5 Nested Within. - prag_attr

3.2.2.288.6 Host Dependencies. - None

3.2.2.288.7 Target Dependencies. - None

3.2.2.288.8 Subprogram Visibility. - Within Package Only.

3.2.2.288.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.288.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the pragma to be processed
```

3.2.2.288.11 Side-Effects. - None

3.2.2.288.12 Algorithm. -

```
check that the controlled pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14052;
     return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14053;
     return;
end if;
check that argument is only an access type (not derived from one) declared in
the
same declarative part;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14054;
end if;
```

3.2.2.288.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14052	E	pragma should have only one argument
14053	E	pragma argument invalid
14054	E	controlled argument must be an access type

3.2.2.288.14 Examples of Data Structures. - None

3.2.2.289 Subprogram. - list

3.2.2.289.1 Purpose. - Perform name resolution upon the list pragma for all targets.

3.2.2.289.2 Assumptions. - None

3.2.2.289.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.289.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.289.5 Nested Within. - prag_attr

3.2.2.289.6 Host Dependencies. - None

3.2.2.289.7 Target Dependencies. - None

3.2.2.289.8 Subprogram Visibility. - Within Package Only.

3.2.2.289.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.289.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the list pragma to be processed
```

3.2.2.289.11 Side-Effects. - None

3.2.2.289.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose_msg" with the_pragma for error 14055;
     return;
end if;
check that this argument is spelled "on" or "off";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14056;
end if;
```

3.2.2.289.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14055	E	pragma should have only one argument
14056	E	arguments to the list pragma should be on or off

3.2.2.289.14 Examples of Data Structures. - None

3.2.2.290 Subprogram. - page

3.2.2.290.1 Purpose. - Perform name resolution upon the page pragma for all targets.

3.2.2.290.2 Assumptions. - None

3.2.2.290.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.290.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.290.5 Nested Within. - prag_attr

3.2.2.290.6 Host Dependancies. - None

3.2.2.290.7 Target Dependencies. - None

3.2.2.290.8 Subprogram Visibility. - Within Package Only.

3.2.2.290.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.290.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the page pragma to be processed
```

3.2.2.290.11 Side-Effects. - None

3.2.2.290.12 Algorithm. -

```
check that the_pragma has no arguments
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14058;
end if;
```

3.2.2.290.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N,W,E,S,F	TEXT
14058	E	page pragma should have no arguments

3.2.2.290.14 Examples of Data Structures. - None

3.2.2.291 Subprogram. - general_pragma

3.2.2.291.1 Purpose. - Performs name resolution upon the pragmas for all targets.

3.2.2.291.2 Assumptions. - None

3.2.2.291.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.291.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.291.5 Nested Within. - prag_attr

3.2.2.291.6 Host Dependencies. - None

3.2.2.291.7 Target Dependencies. - None

3.2.2.291.8 Subprogram Visibility. - Within Package Only.

3.2.2.291.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.291.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- reference to pragma to be processed
```

3.2.2.291.11 Side-Effects. - None

3.2.2.291.12 Algorithm. -

```
check that pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14059;
     return;
end if;
invoke "EXPS.exp"  with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14060;
end if;
```

```
case PRAGMA name> of
PACK>:
     check that argument names a record or arry type but not a
     derived type;
     if ERROR> then
        invoke "diagnose.msg" for error 14891;
     end if;
   OTHERS> : null;
   end if;
```

3.2.2.291.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14059;	E	pragma should only have one argument
14060	E	pragma argument invalid
14891	E	pack pragma must be given for a non-derived record or array only

3.2.2.291.14 Examples of Data Structures. - None

3.2.2.292 Subprogram. - optimize

3.2.2.292.1 Purpose. - Performs name resolution upon the optimize pragma for all targets.

3.2.2.292.2 Assumptions. - None

3.2.2.292.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.292.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.292.5 Nested within. - prag_attr

3.2.2.292.6 Host Dependencies. - None

3.2.2.292.7 Target Dependencies. - None

3.2.2.292.8 Subprogram Visibility. - Within Package Only.

3.2.2.292.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.292.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the optimize pragma to be processed
```

3.2.2.292.11 Side-Effects. - None

3.2.2.292.12 Algorithm. -

```
check that here is only 1 argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14064;
     return;
end if;
check that argument is spelled either "time" or "space";
if ERROR> then
    invoke "Diagnose.msg" with the_pragma for error 14065;
end if;
```

3.2.2.292.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14064	E	optimize pragma should have only one argument
14065	E	pragma argument invalid

3.2.2.292.14 Examples of Data Structures. - None

3.2.2.293 Subprogram. - suppress

3.2.2.293.1 Purpose. - Perform name resolution upon the suppress pragma for all targets.

3.2.2.293.2 Assumptions. - None

3.2.2.293.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.293.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.293.5 Nested within. - prag_attr

3.2.2.293.6 Host Dependencies. - None

3.2.2.293.7 Target Dependencies. - None

3.2.2.293.8 Subprogram Visibility. - Within Package Only.

3.2.2.293.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.293.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the suppress pragma to be processed
```

3.2.2.293.11 Side-Effects. - None

3.2.2.293.12 Algorithm. -

```
check that first argument is only a "check_name";
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14066;
     return;
end if;
if SECOND argument is present>
then
     invoke "EXPS.exp" with second argument;
     if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14067;
     return;
     end if;
  check that name is only an object or type name;
  if ERROR> then
    invoke "diagnose.mg" with the_pragma  for error 14895;
  end if;
end if;
```

3.2.2.293.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14066	E	first argument to suppress pragma must be a check name
14895	E	second argument must be a type or object name
14067	E	second argument invalid

3.2.2.293.14 Examples of Data Structures. - None

3.2.2.294 Subprogram. - attribute

3.2.2.294.1 Purpose. - Perform name resolution on and checks legality of an attribute that has no parameters, for the VAX/VMS target, the VAX_SA target, the ROLM 16028_SA target, and the ROLM1666_SA target.

3.2.2.294.2 Assumptions. - None

3.2.2.294.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.294.4 Used Recursively. - This subprogram is used recursively.

3.2.2.294.5 Nested Within. - prag_attr

3.2.2.294.6 Host Dependencies. - None

3.2.2.294.7 Target Dependencies. - For all targets except the PDP 11/70 Unix.

3.2.2.294.8 Subprogram Visibility. - Within Package Only.

3.2.2.294.9 Function/Procedure. - This subprogram is a Function with a c_node_ref result type.

3.2.2.294.10 Formal Parameters. -

```
attr : in  c_node_ref; -- the attribute to be processed
```

3.2.2.294.11 Side-Effects. - None

3.2.2.294.12 Algorithm. -

```
invoke "exps.exp" with as_name of attr an
names as context.
if ERROR> then mark attr as in error.
          return VOID>; end if;
case ATTR id> of
ADDRESS> : name must indicate an objector subprog only.
BASE> :  name must indicate a type or subtype that
        is not a task.
SIZE> : name must indicate a non task type or subtype
        or an object.
IMAGE, value> :
        name must indicate a scalar type or subtype.
FIRST, last> :
        name must indicate a scalar type or subtype or
        an array type or subtype constrained
    only) or an object of array type or subtype.
POS, val, pred, succ> :
        name must indicate a discrete type or subtype.
DELTA, actual_delta, bits> :
        name must indicate a fixed point type.
LARGE, machine_rounds> :
        name must indicate a fixed or flp type.
DIGIT, mantissa, emax, small,
        epsilon, machine_radix, machine_mantissa,
        machine_emax, machine_emin, machine_overflows> :
        name must indicate a floating point type or
        subtype only.
LENGTH, range> : name must indicate an array
        type or subtype, or an object thereof,
        constrained only.
CONSTRAINED> :  name must indicate a type with discriminants.
POSITION, first_bit, last_bit> :
        name must indicate a record component.
STORAGE_SIZE> : name must indicate an access type.
TERMINATED, priority, failure, storage_size> :
        name must indicate a task object or type.
COUNT> :
        name must indicate an entry.
BASE> :
        name must be a type or subtype.
DISP> : name must indicate a (potential) parameter to a
        code statement;
OTHERS> : error;
end case;
if ERROR> and PRAGMA_FLAG not set> then
    invoke "diagnose.msg" with attr for error 14006
        return VOID>;
        else mark attr as error;
        return VOID>;
end if;
return attribute result type.
```

3.2.2.294.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14006	E	illegal attribute of this name.

3.2.2.294.14 Examples of Data Structures. - None

3.2.2.295 Subprogram. - interface

3.2.2.295.1 Purpose. - Perform name resolution upon the interface pragma for the VAX and ROLM targets.

3.2.2.295.2 Assumptions. - None

3.2.2.295.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.295.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.295.5 Nested Within. - prag_attr

3.2.2.295.6 Host Dependencies. - None

3.2.2.295.7 Target Dependencies. - For all targets except the PDP 11/70.

3.2.2.295.8 Subprogram Visibility. - Within Package Only.

3.2.2.295.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.295.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the interface pragma to be processed
```

3.2.2.295.11 Side-Effects. - None

3.2.2.295.12 Algorithm. -

```
invoke "EXPS.exp" with second argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14062;
     return;
end if;
check that second argument is only a subprogram name;
if ERROR> then
  invoke "diagnose.msg" for error 14057;
if ANY more arguments exist>
then
     invoke "Diagnose.msg" with the_pragma for error 14063;
end if;
```

3.2.2.295.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14057	E	second argument must be a subprogram name
14061	W	invalid language to be interfaced
14062	E	second interface argument invalid
14063	E	interface pragma should have only two arguments

3.2.2.295.14 Examples of Data Structures. - None

3.2.2.296 Package. - prag_attr

3.2.2.296.1 Purpose. - A target dependant package to check pragmas and attributes and to perform name resolution on them. This is for the VAX780_SA target only. Only this package will be included for this target.

3.2.2.296.2 Number of Subprograms. - 15

3.2.2.296.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.296.4 Additional Dependencies for Body. - exps

3.2.2.296.5 Package Specification. -

```
--visible outside package
procedure pragmat...
procedure attribute...
```

3.2.2.296.6 Elaboration Code. - None

3.2.2.296.7 Examples of Data Structures. - None

3.2.2.297 Subprogram. - pragmat

3.2.2.297.1 Purpose. - Perform name resolution upon pragmas all targets.

3.2.2.297.2 Assumptions. - None

3.2.2.297.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.297.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.297.5 Nested Within. - prag_attr

3.2.2.297.6 Host Dependencies. - None

3.2.2.297.7 Target Dependencies. - None

3.2.2.297.8 Subprogram Visibility. - Outside Package.

3.2.2.297.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.297.10 Formal Parameters. -

the_pragma : in c_node_ref; -- a reference to the pragma to be checked

3.2.2.297.11 Side-Effects. - None

3.2.2.297.12 Algorithm. -

```
case THE_PRAGMA is> if
PAGE>:  invoke "prag_attr.page";
TITLE, include, memory_size, priority,
 storage_unit, pack>:
'    invoke "prag_attr.general_pragma";
INTERFACE>:  invoke "prag_attr.interface";
OPTIMIZE>:  invoke "prag_attr.optimize";
SUPPRESS>:  invoke "prag_attr.suppress";
SYSTEM>:  invoke "prag_attr.system";
INLINE>:  invoke "prag_attr.inline";
CONTROLLED>:  invoke "prag_attr.controlled";
LIST>:  invoke "prag_attr.list";
OTHERS>:  invoke "Diagnose.msg" with the_pragma for error 14033;
```

end case;

3.2.2.297.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14033	W	Pragma not recognized

3.2.2.297.14 Examples of Data Structures. - None

3.2.2.298 Subprogram. - system

3.2.2.298.1 Purpose. - Perform name resolution upon the system pragma for VAX780_SA target.

3.2.2.298.2 Assumptions. - None

3.2.2.298.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.298.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.298.5 Nested Within. - prag_attr

3.2.2.298.6 Host Dependencies. - None

3.2.2.298.7 Target Dependencies. - For the VAX 780 stand alone target as in the Aspec.

3.2.2.298.8 Subprogram Visibility. - Within Package Only.

3.2.2.298.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.298.10 Formal Parameters. -

the_pragma : in c_node_ref; -- the system pragma to be processed

3.2.2.298.11 Side-Effects. - None

3.2.2.298.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14042;
     return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14043;
     return;
end if;
if ARGUMENT not part of enumeration type system
     system_name> then
     invoke "Diagnose.msg" with the_pragma for error 14044;
     return;
end if;
if ARGUMENT not VAX780_SA or VAX780> then
     invoke "Diagnose.msg" with the_pragma for error 14045;
end if;
```

3.2.2.298.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14042	E	system pragma should have only one argument
14043	E	pragma argument invalid
14044	E	argument must be the system name VAX7780_SA or VAX780

14045 W argument must be the system name VAX7780_SA or VAX780

3.2.2.298.14 Examples of Data Structures. - None

3.2.2.299 Subprogram. - inline

3.2.2.299.1 Purpose. - Performs name resolution upon the inline pragma for all targets.

3.2.2.299.2 Assumptions. - None

3.2.2.299.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.299.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.299.5 Nested within. - prag_attr

3.2.2.299.6 Host Dependencies. - None

3.2.2.299.7 Target Dependencies. - None

3.2.2.299.8 Subprogram Visibility. - Within Package Only.

3.2.2.299.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.299.10 Formal Parameters. -

the_pragma : in c_node_ref; -- the inline pragma to be processed

3.2.2.299.11 Side-Effects. - None

3.2.2.299.12 Algorithm. -

```
while ARGUMENT list not at end>
loop
     invoke "EXPS.exp" to identify argument
     if ERROR> then
          invoke "Diagnose.msg" with the_pragma for error 14050;
     return;
     end if;
     check  that the argument is the name of a nongeneric
     subprogram declared in the same declaration part;
     if ERROR>
     then
          invoke "Diagnose.msg" with the_pragma for error 14051;
          return;
     end if;
advance position in list;
end loop;
```

3.2.2.299.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
14050	E	pragma argument invalid
14051	E	inline argument must be a nongeneric subprogram name

3.2.2.299.14 Examples of Data Structures. - None

3.2.2.300 Subprogram. - controlled

3.2.2.300.1 Purpose. - Perform name resolution upon a controlled pragma for all targets.

3.2.2.300.2 Assumptions. - None

3.2.2.300.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.300.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.300.5 Nested within. - prag_attr

3.2.2.300.6 Host Dependencies. - None

3.2.2.300.7 Target Dependencies. - None

3.2.2.300.8 Subprogram Visibility. - Within Package Only.

3.2.2.300.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.300.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the pragma to be processed
```

3.2.2.300.11 Side-Effects. - None

3.2.2.300.12 Algorithm. -

```
check that the controlled pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14052;
     return;
end if;
invoke "EXPS.exp" with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14053;
     return;
end if;
check that argument is only an access type (not derived from one) declared in
the
same declarative part;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14054;
end if;
```

3.2.2.300.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14052	E	pragma should have only one argument
14053	E	pragma argument invalid
14054	E	controlled argument must be an access type

3.2.2.300.14 Examples of Data Structures. - None

3.2.2.301 Subprogram. - list

3.2.2.301.1 Purpose. - Perform name resolution upon the list pragma for all targets.

3.2.2.301.2 Assumptions. - None

3.2.2.301.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.301.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.301.5 Nested Within. - prag_attr

3.2.2.301.6 Host Dependencies. - None

3.2.2.301.7 Target Dependencies. - None

3.2.2.301.8 Subprogram Visibility. - Within Package Only.

3.2.2.301.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.301.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the list pragma to be processed
```

3.2.2.301.11 Side-Effects. - None

3.2.2.301.12 Algorithm. -

```
check that there is only one argument;
if ERROR> then
      invoke "Diagnose_msg" with the_pragma for error 14055;
      return;
end if;
check that this argument is spelled "on" or "off";
if ERROR> then
      invoke "Diagnose.msg" with the_pragma for error 14056;
end if;
```

3.2.2.301.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14055	E	pragma should have only one argument
14056	E	arguments to the list pragma should be on or off

3.2.2.301.14 Examples of Data Structures. - None

3.2.2.302 Subprogram. - page

3.2.2.302.1 Purpose. - Perform name resolution upon the page pragma for all targets.

3.2.2.302.2 Assumptions. - None

3.2.2.302.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.302.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.302.5 Nested Within. - prag_attr

3.2.2.302.6 Host Dependencies. - None

3.2.2.302.7 Target Dependencies. - None

3.2.2.302.8 Subprogram Visibility. - Within Package Only.

3.2.2.302.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.302.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the page pragma to be processed
```

3.2.2.302.11 Side-Effects. - None

3.2.2.302.12 Algorithm. -

```
check that the_pragma has no arguments
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14058;
end if;
```

3.2.2.302.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14058	E	page pragma should have no arguments

3.2.2.302.14 Examples of Data Structures. - None

3.2.2.303 Subprogram. - general_pragma

3.2.2.303.1 Purpose. - Performs name resolution upon the pragmas for all targets.

3.2.2.303.2 Assumptions. - None

3.2.2.303.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.303.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.303.5 Nested Within. - prag_attr

3.2.2.303.6 Host Dependencies. - None

3.2.2.303.7 Target Dependencies. - None

3.2.2.303.8 Subprogram Visibility. - Within Package Only.

3.2.2.303.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.303.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- reference to pragma to be processed
```

3.2.2.303.11 Side-Effects. - None

3.2.2.303.12 Algorithm. -

```
check that pragma has only one argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14059;
     return;
end if;
invoke "EXPS.exp"  with argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14060;
end if;
```

```
case PRAGMA name> of
PACK>:
     check that argument names a record or arry type but not a
     derived type;
     if ERROR> then
        invoke "diagnose.msg" for error 14891;
     end if;
   OTHERS> : null;
   end if;
```

3.2.2.303.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14059;	E	pragma should only have one argument
14060	E	pragma argument invalid
14891	E	pack pragma must be given for a non-derived record or array only

3.2.2.303.14 Examples of Data Structures. - None

3.2.2.304 Subprogram. - optimize

3.2.2.304.1 Purpose. - Performs name resolution upon the optimize pragma for all targets.

3.2.2.304.2 Assumptions. - None

3.2.2.304.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.304.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.304.5 Nested Within. - prag_attr

3.2.2.304.6 Host Dependencies. - None

3.2.2.304.7 Target Dependencies. - None

3.2.2.304.8 Subprogram Visibility. - Within Package Only.

3.2.2.304.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.304.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the optimize pragma to be processed
```

3.2.2.304.11 Side-Effects. - None

3.2.2.304.12 Algorithm. -

```
check that here is only 1 argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14064;
     return;
end if;
check that argument is spelled either "time" or "space";
if ERROR> then
    invoke "Diagnose.msg" with the_pragma for error 14065;
end if;
```

3.2.2.304.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14064	E	optimize pragma should have only one argument
14065	E	pragma argument invalid

3.2.2.304.14 Examples of Data Structures. - None

3.2.2.305 Subprogram. - suporess

3.2.2.305.1 Purpose. - Perform name resolution upon the suppress pragma for all targets.

3.2.2.305.2 Assumptions. - None

3.2.2.305.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.305.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.305.5 Nested Within. - prag_attr

3.2.2.305.6 Host Dependencies. - None

3.2.2.305.7 Target Dependencies. - None

3.2.2.305.8 Subprogram Visibility. - Within Package Only.

3.2.2.305.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.305.10 Formal Parameters. -

```
the_pragma :  in c_node_ref; -- the suppress pragma to be processed
```

2.2.305.11 Side-Effects. - None

2.2.305.12 Algorithm. -

```
eck that first argument is only a "check_name";
 ERROR> then
   invoke "Diagnose.msg" with the_pragma for error 14066;
   return;
d if;
 SECOND argument is present>
en
   invoke "EXPS.exp" with second argument;
   if ERROR> then
   invoke "Diagnose.msg" with the_pragma for error 14067;
   return;
   end if;
check that name is only an object or type name;
if ERROR> then
  invoke "diagnose.mg" with the_pragma  for error 14895;
end if;
d if;
```

2.2.305.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14066	E	first argument to suppress pragma must be a check name
14895	E	second argument must be a type or object name
14067	E	second argument invalid

2.2.305.14 Examples of Data Structures. - None

2.2.306 Subprogram. - attribute

3.2.2.306.1 Purpose. - Perform name resolution on and checks legality of attribute that has no parameters, for the VAX/VMS target, the VAX_SA target, ROLM 16028_SA target, and the ROLM1666_SA target.

3.2.2.306.2 Assumptions. - None

3.2.2.306.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.306.4 Used Recursively. - This subprogram is used recursively.

3.2.2.306.5 Nested Within. - prag_attr

3.2.2.306.6 Host Dependencies. - None

3.2.2.306.7 Target Dependencies. - For all targets except the PDP 11/70 Unix

3.2.2.306.8 Subprogram Visibility. - Within Package Only.

3.2.2.306.9 Function/Procedure. - This subprogram is a Function with c_node_ref result type.

3.2.2.306.10 Formal Parameters. -

attr : in c_node_ref; -- the attribute to be processed

3.2.2.306.11 Side-Effects. - None

3.2.2.306.12 Algorithm. -

```
invoke "exps.exp" with as_name of attr an
names as context.
if ERROR> then mark attr as in error.
         return VOID>; end if;
case ATTR id> of
ADDRESS> : name must indicate an objector subprog only.
BASE> :  name must indicate a type or subtype that
      is not a task.
SIZE> : name must indicate a non task type or subtype
      or an object.
IMAGE, value> :
      name must indicate a scalar type or subtype.
FIRST, last> :
      name must indicate a scalar type or subtype or
      an array type or subtype constrained
   only) or an object of array type or subtype.
POS, val, pred, succ> :
      name must indicate a discrete type or subtype.
DELTA, actual_delta, bits> :
      name must indicate a fixed point type.
LARGE, machine_rounds> :
      name must indicate a fixed or flp type.
DIGIT, mantissa, emax, small,
      epsilon, machine_radix, machine_mantissa,
      machine_emax, machine_emin, machine_overflows> :
      name must indicate a floating point type or
      subtype only.
LENGTH, range> : name must indicate an array
      type or subtype, or an object thereof,
      constrained only.
CONSTRAINED> :  name must indicate a type with discriminants.
POSITION, first_bit, last_bit> :
      name must indicate a record component.
STORAGE_SIZE> : name must indicate an access type.
TERMINATED, priority, failure, storage_size> :
      name must indicate a task object or type.
COUNT> :
      name must indicate an entry.
BASE> :
      name must be a type or subtype.
DISP> : name must indicate a (potential) parameter to a
      code statement;
OTHERS> : error;
end case;
if ERROR> and PRAGMA_FLAG not set> then
   invoke "diagnose.msg" with attr for error 14006
      return VOID>;
      else mark attr as error;
      return VOID>;
end if;
return attribute result type.
```

3.2.2.306.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.E	TEXT
14006	E	illegal attribute of this name.

3.2.2.306.14 Examples of Data Structures. - None

3.2.2.307 Subprogram. - interface

3.2.2.307.1 Purpose. - Perform name resolution upon the interface pragma for the VAX and ROLM targets.

3.2.2.307.2 Assumptions. - None

3.2.2.307.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.307.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.307.5 Nested Within. - prag_attr

3.2.2.307.6 Host Dependencies. - None

3.2.2.307.7 Target Dependencies. - For all targets except the PDP 11/70.

3.2.2.307.8 Subprogram Visibility. - Within Package Only.

3.2.2.307.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.307.10 Formal Parameters. -

the_pragma : in c_node_ref; -- the interface pragma to be processed

3.2.2.307.11 Side-Effects. - None

3.2.2.307.12 Algorithm. -

```
invoke "EXPS.exp" with second argument;
if ERROR> then
     invoke "Diagnose.msg" with the_pragma for error 14062;
     return;
end if;
check that second argument is only a subprogram name;
if ERROR> then
  invoke "diagnose.msg" for error 14057;
if ANY more arguments exist>
then
     invoke "Diagnose.msg" with the_pragma for error 14063;
end if;
```

3.2.2.307.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
14057	E	second argument must be a subprogram name
14061	W	invalid language to be interfaced
14062	E	second interface argument invalid
14063	E	interface pragma should have only two arguments

3.2.2.307.14 Examples of Data Structures. - None

3.2.2.308 Package. - maint_n

3.2.2.308.1 Purpose. - Implement maintenance options for name resolution.

3.2.2.308.2 Number of Subprograms. - 1

3.2.2.308.3 Dependencies on Other Packages for Spec. - cdm_types

3.2.2.308.4 Additional Dependencies for Body. - get_as,put_as,get_sm,put_sm,get_lx,put_lx,get_vs,put_vs, get_ft,put_ft,list_seq,array_seq,cdm_str_pack,decls,vistree

3.2.2.308.5 Package Specification. -

```
package maint_n is
    -- visible outside package
    procedure trace ...
end maint_n;
```

3.2.2.308.6 Elaboration Code. - None

3.2.2.308.7 Examples of Data Structures. - None

3.2.2.309 Subprogram. - trace

3.2.2.309.1 Purpose. - Implement the trace maintenance option for context processing.

3.2.2.309.2 Assumptions. - None

3.2.2.309.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.309.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.309.5 Nested Within. - maint_n

3.2.2.309.6 Host Dependencies. - None

3.2.2.309.7 Target Dependencies. - None

3.2.2.309.8 Subprogram Visibility. - Outside Package.

3.2.2.309.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.309.10 Formal Parameters. -

```
caller : in string (1..10); -- the name of the calling routine
current_stmt : in cdm_types.c_node_ref; -- the diana statement
          -- currently being processed
```

3.2.2.309.11 Side-Effects. - None

3.2.2.309.12 Algorithm. -

```
if IN correct current compilation unit source statement range> then
    while NOT at end of trace option list> loop
       case TRACE option> of
       A> : DUMP vistree node indicated the variable
          "context.current_scope">;
       B> : DUMP whole vistree beginning at
          "vistree.visibility_root">;
```

```
        C> : DUMP diana tree of vistree node indicated by
           "context.current_scope">;
        D> : DUMP entire diana tree of unit indicated by
           "context.current_scope">;
        E> : DUMP entire Diana tree corresponding to the
           vistree beginning at "vistree.visibility_root">;
        F> : DUMP list indicated by variable
           "vistree.incomplete_types_unredeclared">;
        G> : DUMP list indicated by variable
           "vistree.deferred_constants_unredeclared">;
        H> : DUMP name of calling routine>;
        I> : DUMP current source statement number>;
        J> : DUMP vistree from nearest enclosing unit to
           current scope>;
        K> : DUMP Diana from nearest enclosing unit to
           current scope>;
        L> : DUMP list indicated by variable
           "decls.private_types_unredeclared">;
        M> : DUMP list indicated by variable
           "vistree.aggr_poss_list">;
        N> : DUMP variable "vistree.access_cnt" and
           list "vistree.access_list">;
        O> : DUMP diana tree of current source statement>;
        P> : DUMP list indicated by variable
           "decls.label_s_unredeclared">;
        Q> : DUMP the variables listed below:
           "vistree.scope_innermost",
           "vistree.block_innermost,
           "vistree.unit_innermost">;
        R> : DUMP the variables listed below:
           "vistree.current_id_s",
           "vistree.discriminant_flag",
           "vistree.generic_flag",
           "vistree.private_flag",
           "vistree.param_flag">;
        end case;
        advance position in trace option list;
    end loop;
end if;
```

3.2.2.309.13 Diagnostic Messages Generated. - None

3.2.2.309.14 Examples of Data Structures. - None

3.2.2.310 Package. - ovp_expr

3.2.2.310.1 Purpose. - Perform overloading resolution on Diana nodes which appear in expressions.

3.2.2.310.2 Number of Subprograms. - 22

3.2.2.310.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.310.4 Additional Dependencies for Body. - ovp_util, cdm_as, ovp_clas, static_exp, stats

3.2.2.310.5 Package Specification. -

```
package OVP_EXPR is
-- see overview in package OVP_TRAV
    procedure OV_USED_OBID ...
    procedure OV_INDEXED ...
    procedure OV_SLICE ...
    procedure OV_NULL_ACC ...
    procedure OV_USED_CHAR ...
    procedure OV_NUM_LIT ...
    procedure OV_ALLOCATOR ...
    procedure OV_SELECTED ...
    procedure OV_ATTRIBUTE ...
    procedure OV_ATTR_CALL ...
    procedure OV_ALL ...
    procedure OV_STR_LIT ...
    procedure OV_AGG ...
    procedure OV_BINARY ...
    procedure OV_MEMBERSHP ...
    procedure OV_PARENTHZD ...
    procedure OV_CONVERS ...
    procedure OV_QUALIFIED ...
    procedure OV_FUNC_CALL ...
    procedure OV_RANGE ...
    procedure OV_DSCR_AGG ...
        procedure OV_APPLY...
end OVP_EXPR;
```

3.2.2.310.6 Elaboration Code. - None

3.2.2.310.7 Examples of Data Structures. - None

3.2.2.311 Subprogram. - ov_dscr_agg

3.2.2.311.1 Purpose. - Perform overloading resolution on a node of type 'dscrmt_aggregate'.

3.2.2.311.2 Assumptions. - None

3.2.2.311.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.311.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.311.5 Nested Within. - ovp_ctxt

3.2.2.311.6 Host Dependencies. - None

3.2.2.311.7 Target Dependencies. - None

3.2.2.311.8 Subprogram Visibility. - Outside Package.

3.2.2.311.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.311.10 Formal Parameters. -

node : in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.311.11 Side-Effects. - None

3.2.2.311.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    for EACH element of as_list> loop
       if DISCRIMINANT name has appeared before> then
       ERROR 15000>
       end if;
       OVERLOAD (expression in list element>, TYPE of corresponding discrim.>);
    end loop;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.311.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15000	E	Discriminant name appears twice

3.2.2.311.14 Examples of Data Structures. - None

3.2.2.312 <u>Subprogram</u>. - ov_used_objid

3.2.2.312.1 <u>Purpose</u>. - Perform overloading resolution on a node of type 'used_object_id'.

3.2.2.312.2 <u>Assumptions</u>. - None

3.2.2.312.3 <u>Implementation Language</u>. - This subprogram is written in Ada.

3.2.2.312.4 <u>Used Recursively</u>. - This subprogram is not used recursively.

3.2.2.312.5 <u>Nested Within</u>. - ovp_expr

3.2.2.312.6 <u>Host Dependencies</u>. - None

3.2.2.312.7 <u>Target Dependencies</u>. - None

3.2.2.312.8 <u>Subprogram Visibility</u>. - Outside Package.

3.2.2.312.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.312.10 <u>Formal Parameters</u>. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.
```

```
resolved :  out  boolean; -- set to false if the expression can't be
                            resolved on the first two passes.
```

3.2.2.312.11 Side-Effects. - None

3.2.2.312.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    GET sm_exp_type from sm_def>

    if SM_EXP_TYPE doesn't fit RESULT_REQ> then
       RESOLVED := FALSE;
       if PASS > 2 then
          ERROR # 15001>
       end if;
       return;
    else
       PARAM_AVAIL := SM_EXP_TYPE>
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " resolved = ", resolved,
          " param_avail = ", param_avail>
end if;
```

3.2.2.312.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15001	E	Type of variable illegal for context

3.2.2.312.14 Examples of Data Structures. - None

3.2.2.313 Subprogram. - ov_indexed

3.2.2.313.1 Purpose. - Perform overloading resolution on a node of type 'indexed'.

3.2.2.313.2 Assumptions. - None

3.2.2.313.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.313.4 Used Recursively. - This subprogram is used recursively.

3.2.2.313.5 Nested within. - ovp_expr

3.2.2.313.6 Host Dependencies. - None

3.2.2.313.7 Target Dependencies. - None

3.2.2.313.8 Subprogram Visibility. - Outside Package.

3.2.2.313.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.313.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref; -- diana node to be resolved.

pass : in integer range 1..4; -- pass of overloading algorithm.

result_req : in cdm_type.c_node_ref; -- type of result required in
                                         context in which this node
```

```
                                              appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                               result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                            resolved on the first two passes.
```

3.2.2.313.11 Side-Effects. - None

3.2.2.313.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    -- resolve array name
    if PASS = 1 or PASS = 3 then
       CL_NAME (AS_NAME>, PASS, ANY type>, IND_PARM, RESOLVED);
    else
       CL_NAME (AS_NAME>, PASS, SM_VALUE>, IND_PARM, RESOLVED);

    if |IND_PARM| = 1 then
       FIND component type of array>
       if COMPONENT type doesn't match RESULT_REQUIRED> then
          RESOLVED := FALSE;
          if PASS = 4 then
             ERROR # 15002>
          end if;
          PARAM_AVAIL := EMPTY>;
          return;
          end if;

    -- save IND_PARM for next pass (use sm_value field)
    sm_value := IND_PARM;
    sm_exp_type := COMPONENT type>;
       -- resolve index expressions
       for EACH index :> loop
          CL_EXP (ITH element of as_exp_s>, PASS,
                   TYPE of ith index>, IND_PARM, RESOLVED>;
       PARAM_AVAIL := COMPONENT type>;
    else
       -- array name not resolved
       RESOLVED := FALSE;
```

```
        if PASS = 4 then
           ERROR # 15003>
        end if;
        PARAM_AVAIL := EMPTY>;
     end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.313.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15002	E	Array component type does not match context requirements.
15003	E	Array type not resolvable.

3.2.2.313.14 Examples of Data Structures. - None

3.2.2.314 Subprogram. - ov_slice

3.2.2.314.1 Purpose. - Perform overloading resolution on a node of type 'slice'.

3.2.2.314.2 Assumptions. - None

3.2.2.314.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.314.4 Used Recursively. - This subprogram is used recursively.

3.2.2.314.5 Nested Within. - ovp_expr

3.2.2.314.6 Host Dependencies. - None

3.2.2.314.7 Target Dependencies. - None

3.2.2.314.8 Subprogram Visibility. - Outside Package.

3.2.2.314.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.314.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.314.11 Side-Effects. - None

3.2.2.314.12 Algorithm. -

```
if MAINTENANCE option 8> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
```

```
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
     --Resolve array name
     if PASS = 1 or PASS = 3
        CL_NAME (AS_NAME>, PASS, ANY type>, SLC_PARM, RESOLVED);
     else
        CL_NAME (AS_NAME>, PASS, SM_VALUE>, SLC_PARM, RESOLVED);
     if !SLC_PARM! = 1 then
        FIND component type of array>
        if COMPONENT type doesn't match RESULT_REQUIRED> then
           RESOLVED := FALSE;
           if PASS = 4 then
              ERROR # 15004>
           end if;
           PARAM_AVAIL := EMPTY>:
           return;
        end if;
        -- Save SLC_PARM for next pass (Use sm_value field)
        sm_value := SLC_PARM;
        sm_exp_type := COMPONENT type>;
        -- Resolve discrete range
        CL_DSCRT_RNG (AS_DSCRT_RANGE>, PASS, ARRAY component type>,
                              IND_PARM, RESOLVED);
        PARAM_AVAIL := COMPONENT type>;
     else
        -- array name not resolved
        RESOLVED := FALSE;
        if PASS = 4 then
           ERROR # 15005>
        end if;
        PARAM_AVAIL := EMPTY>;
     end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.314.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
15004	E	Array component type does not match context requirements
15005	E	Array type not resolvable

3.2.2.314.14 Examples of Data Structures. - None

3.2.2.315 Subprogram. - ov_selected

3.2.2.315.1 Purpose. - Perform overloading resolution on a node of type 'selected'.

3.2.2.315.2 Assumptions. - None

3.2.2.315.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.315.4 Used Recursively. - This subprogram is used recursively.

3.2.2.315.5 Nested within. - ovp_expr

3.2.2.315.6 Host Dependencies. - None

3.2.2.315.7 Target Dependencies. - None

3.2.2.315.8 Subprogram Visibility. - Outside Package.

3.2.2.315.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.315.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
```

```
                                                  appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                              result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.315.11 Side-Effects. - None

3.2.2.315.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
-- Resolve record or access variable name
if PASS = 1 or PASS = 3 then
    CL_NAME(AS_NAME>, PASS, ANY type>, SEL_PARM, RESOLVED);
else
    CL_NAME(AS_NAME>, PASS, SM_VALUE>, SEL_PARM, RESOLVED);

if |SEL_PARM| = 1 then
    FIND type of record being selected>
    FIND type of component named in as_designator>
    if COMPONENT type doesn't match RESULT_REQ> then
       RESOLVED := FALSE
       if PASS = 4
          then ERROR 15006>
       end if;
       PARAM_AVAIL := EMPTY>;
       return;
    end if;
    --Save SEL_PARM for next pass (Use sm_value field)
    sm_value := SEL_PARM;
    sm_exp_type := COMPONENT type>;
    PARAM_AVAIL := COMPONENT type>;
else
    -- Record name not resolved
    RESOLVED := FALSE
    if PASS = 4 then
       ERROR 15007>
    end if;
    PARAM_AVAIL := EMPTY>
```

```
end if;
if MAINTENANCE option 5> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.315.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15006	E	Selected component type does not match context requirements.
15007	E	Selected component name not resolvable.

3.2.2.315.14 Examples of Data Structures. - None

3.2.2.316 Subprogram. - ov_all

3.2.2.316.1 Purpose. - Perform overloading resolution on a node of type 'all'.

3.2.2.316.2 Assumptions. - None

3.2.2.316.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.316.4 Used Recursively. - This subprogram is used recursively.

3.2.2.316.5 Nested Within. - ovp_expr

3.2.2.316.6 Host Dependencies. - None

3.2.2.316.7 Target Dependencies. - None

3.2.2.316.8 Subprogram Visibility. - Outside Package.

3.2.2.316.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.316.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.316.11 Side-Effects. - None

3.2.2.316.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    -- Resolve access variable name
    if PASS = 1 or PASS = 3 then
       CL_NAME (AS_NAME>, PASS, ANY_TYPE>, ALL_PARM, RESOLVED);
    else
```

```
      CL_NAME (Ass_name>, PASS, SM_VALUE>, ALL_PARM, RESOLVED);
   if |ALL_PARM| = 1 then
      -- Save ALL_PARM for next pass
      sm_value := ALL_PARM;
      if ALL_PARM isn't an access type> then
         RESOLVED := FALSE;
         if PASS = 4 then
            ERROR 15008>
         end if;
         PARAM_AVAIL := EMPTY>;
         return;
      end if
      sm_exp_type := TYPE accessed by ALL_PARM>;
      PARAM_AVAIL := sm_exp_type;
   else
      -- Access variable not resolved
      RESOLVED := FALSE;
      if PASS = 4 then
         ERROR 15009>
      end if;
      PARAM_AVAIL := EMPTY>;
   end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.316.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15008	E	Access type required
15009	E	Selected component name not resolved

3.2.2.316.14 Examples of Data Structures. - None

3.2.2.317 Subprogram. - ov_attribute

3.2.2.317.1 Purpose. - Perform overloading resolution on a node of type 'attribute'.

3.2.2.317.2 Assumptions. - None

3.2.2.317.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.317.4 Used Recursively. - This subprogram is used recursively.

3.2.2.317.5 Nested Within. - ovp_expr

3.2.2.317.6 Host Dependencies. - None

3.2.2.317.7 Target Dependencies. - None

3.2.2.317.8 Subprogram Visibility. - Outside Package.

3.2.2.317.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.317.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_basics_.c_node_ref; -- seq of types which may
                                              result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.317.11 Side-Effects. - None

3.2.2.317.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
     if SM_EXP_TYPE not already determined> then
        if AS_ID is FIRST or LAST> then
           -- Find type T of as_name
           if AS_NAME is an attribute> then -- must be BASE
              T := as_name of attribute;
           elsif AS_NAME is an object> then
              T := TYPE of object>;
           end if;
           sm_exp_type := T;
        else
           sm_exo_type := TYPE corres. to as_id> -- Table lookup
        end if;
     end if;

     if SM_EXP_TYPE matches RESULT_REQUIRED> then
        PARAM_AVAIL := sm_exp_type;
     else
        RESOLVED := FALSE;
        if PASS = 4 then
           ERROR 15010>
        end if;
     end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.317.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15010	E	Attribute type does not match context requirements

3.2.2.317.14 Examples of Data Structures. - None

3.2.2.318 Subprogram. - ov_attr_call

3.2.2.318.1 Purpose. - Perform overloading resolution on a node of type 'attribute call'.

3.2.2.318.2 Assumptions. - None

3.2.2.318.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.318.4 Used Recursively. - This subprogram is used recursively.

3.2.2.318.5 Nested within. - ovp_expr

3.2.2.318.6 Host Dependencies. - None

3.2.2.318.7 Target Dependencies. - None

3.2.2.318.8 Subprogram Visibility. - Outside Package.

3.2.2.318.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.318.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
```

```
                                                       appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                                         result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                                 resolved on the first two passes.
```

3.2.2.318.11 Side-Effects. - None

3.2.2.318.12 Algorithm. -

```
if MAINTENANCE option 6> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
-- Find type T of as_name
    if AS_NAME is an attribute> then -- must be BASE
       T := AS_NAME of attribute>;
    elsif AS_NAME is an object> then
       T := TYPE of object>;
    end if;

    COMPUTE type needed for as_exp> -- Depends on attribute

    CL_EXP (AS_EXP>, PASS, NEEDED, ATTR_PARM, RESOLVED);

    --Compute PARAM_AVAIL
    if IMAGE> then
       PARAM_AVAIL := STRING>;
    elsif POS> then
       PARAM_AVAIL := INTEGER>;
    elsif VALUE, VAL, PRED, SUCC> then
       PARAM_AVAIL := T>;
    else -- must be FIRST, LAST, LENGTH, RANGE
       COMPUTE value of index position J, -- call static
                  expr. eval. if index is a literal exp.>
       PARAM_AVAIL := TYPE of Jth index of T>
    end if;

    if PARAM_AVAIL does not match RESULT_REQ> then
       RESOLVED := FALSE;
       if PASS = 4 then
```

```
          ERROR 15011>
        end if;
     else
        sm_exp_type := PARAM_AVAIL;
     end if;
if MAINTENANCE option 8> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.318.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15011	E	Type of attribute call does not match context requirements.

3.2.2.318.14 Examples of Data Structures. - None

3.2.2.319 Subprogram. - ov_num_lit

3.2.2.319.1 Purpose. - Perform overloading resolution on a node of type 'numeric_literal'.

3.2.2.319.2 Assumptions. - None

3.2.2.319.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.319.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.319.5 Nested Within. - ovp_expr

3.2.2.319.6 Host Dependencies. - None

3.2.2.319.7 Target Dependencies. - None

3.2.2.319.8 Subprogram Visibility. - Outside Package.

3.2.2.319.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.319.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.319.11 Side-Effects. - None

3.2.2.319.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
```

```
     SET sm_exp_type to universal_integer or universal_real, depending
         on symbol representation.>

     if SM_EXP_TYPE matches RESULT_REQ> then
        if RESULT_REQ = ANY_TYPE> then
           PARAM_AVAIL := sm_exp_type;
        else
           PARAM_AVAIL := RESULT_REQ; -- implicit type conversion
        end if;
     else
        RESOLVED := FALSE;
        if PASS = 4 then
           ERROR 15012>
        end if;
     end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.319.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15012	E	Type of numeric literal does not match context requirements.

3.2.2.319.14 Examples of Data Structures. - None

3.2.2.320 Subprogram. - ov_used_char

3.2.2.320.1 Purpose. - Perform overloading resolution on a node of type 'used_char'.

3.2.2.320.2 Assumptions. - None

3.2.2.320.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.320.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.320.5 Nested Within. - ovp_expr

3.2.2.320.6 Host Dependencies. - None

3.2.2.320.7 Target Dependencies. - None

3.2.2.320.8 Subprogram Visibility. - Outside Package.

3.2.2.320.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.320.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.320.11 Side-Effects. - None

3.2.2.320.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    REMOVE from the CDL all definitions which don't match RESULT_REQ>
    PARAM_AVAIL := CDL>;
    if |CDL| = 1 then
       SET sm_exp_type>
    else
       RESOLVED := FALSE;
       if PASS = 4 then
          ERROR 15013>
       end if;
    end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.320.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15013	E	Enumeration literal unresolvable or of incorrect type

3.2.2.320.14 Examples of Data Structures. - None

3.2.2.321 Subprogram. - ov_null_acc

3.2.2.321.1 Purpose. - Perform overloading resolution on a node of type 'null_access'.

3.2.2.321.2 Assumptions. - None

3.2.2.321.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.321.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.321.5 Nested Within. - ovp_expr

3.2.2.321.6 Host Dependencies. - None

3.2.2.321.7 Target Dependencies. - None

3.2.2.321.8 Subprogram Visibility. - Outside Package.

3.2.2.321.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.321.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                           context in which this node
                                           appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                              result after first pass.
```

```
resolved :  out  boolean; -- set to false if the expression can't abe
                             resolved on the first two passes.
```

3.2.2.321.11 Side-Effects. - None

3.2.2.321.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
    if RESULT_REQ = "any_type" then
       if |CDL| = 1 then
          sm_exp_type := CDL>;
       else
          RESOLVED := FALSE;
          if PASS = 4 then
             ERROR 15014>
          end if;
       end if;
    else
       if RESULT_REQ is an access type> then
          PARAM_AVAIL := RESULT_REQ;
          sm_exp_type := PARAM_AVAIL;
       else
          RESOLVED := FALSE;
          PARAM_AVAIL := EMPTY>;
       end if;
    end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.321.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15014	E	'Null' not resolvable to a unique access type

3.2.2.321.14 Examples of Data Structures. - None

3.2.2.322 Subprogram. - ov_str_lit

3.2.2.322.1 Purpose. - Perform overloading resolution on a node of type "string_literal'.

3.2.2.322.2 Assumptions. - None

3.2.2.322.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.322.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.322.5 Nested within. - ovp_expr

3.2.2.322.6 Host Dependencies. - None

3.2.2.322.7 Target Dependencies. - None

3.2.2.322.8 Subprogram Visibility. - Outside Package.

3.2.2.322.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.322.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                         context in which this node
                                         appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                           result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.322.11 Side-Effects. - None

3.2.2.322.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
if RESULT_REQ is not "any_type"> then
     if RESULT_REQ is not an array of enumerations> then
        RESOLVED := FALSE;
        if PASS = 4 then
           ERROR 15015>
        end if;
        return;
     end if;

else -- "any_type"
     if LX_SYMREP has characters from array component type> then
        sm_exp_type := RESULT_REQ;
```

```
         PARAM_AVAIL := RESULT_REQ;
      else
         RESOLVED := FALSE;
         if PASS = 4 then
            ERROR 15015>
         end if;
      end if;
   else -- "any_type"
      PARAM_AVAIL := EMPTY>
      for EACH element of CDL> loop
         if LIST element is not an array of enumerations> then
            REMOVE from list>
         end if;
         if LX symrep has characters only from array component
                     type> then
            APPEND this type to PARAM_AVAIL.>
         end if;
      end loop;
      if |PARAM_AVAIL| /= 1 then
         RESOLVED := FALSE;
         if PASS = 4 then
            ERROR 15016>
         end if;
      else
         sm_exp_type := PARAM_AVAIL;
      end if;
   end if;
   if MAINTENANCE option B> then
      PRINT "leave", cdm_type.get_node_id (node), "node",
            "  node = ", node,
            "  resolved = ", resolved,
            "  param_avail = ", param_avail>
   end if;
```

3.2.2.322.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
15015	E	String literal does not match context type requirements
15016	E	String literal does not match context type requirements
15016	E	String literal not resolvable

3.2.2.322.14 Examples of Data Structures. - None

3.2.2.323 Subprogram. - ov_aggregate

3.2.2.323.1 Purpose. - Perform overloading resolution on a node of type 'aggregate'.

3.2.2.323.2 Assumptions. - None

3.2.2.323.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.323.4 Used Recursively. - This subprogram is used recursively.

3.2.2.323.5 Nested within. - ovp_expr

3.2.2.323.6 Host Dependencies. - None

3.2.2.323.7 Target Dependencies. - None

3.2.2.323.8 Subprogram Visibility. - Outside Package.

3.2.2.323.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.323.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                           context in which this node
```

```
                                                  appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                            resolved on the first two passes.
```

3.2.2.323.11 Side-Effects. - None

3.2.2.323.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
REMOVE from the CDL all definitions which don't match RESULT_REQ>
if !CDL! = 0 tnen
    RESOLVED := FALSE;
    if PASS = 4 then
       ERROR 15017>
    end if;
    return;
end if;
if !CDL! = 1 then
    -- only one possible type for aggregate
    SM_EXP_TYPE := CDL>;
    if ARRAY aggregate> then
    for EACH component> loop
       if THERE is a choice expression> then
       CL_EXP (CHOICE_EXPRESSION>, PASS, TYPE of index
                             CHOICE_PARM, RESOLVED);
       end if;
       CL_EXP (COMPONENET_EXPRESSION>, PASS, TYPE of index>,
            COMP_PARM, RESOLVED);
    end loop;
    PARAM_AVAIL := sm_exp_type;
else
    -- record aggregate
    for EACH component> loop
       if DISCRIMINANT or fixed component> then
          CL_EXP (COMPONENENT>, PASS, TYPE of record component>,
             AGG_PARM, RESOLVED);
       else
```

```
          CL_EXP (COMPONENT>, PASS, "any type",
             AGG_PARM, RESOLVED);
       end if;
    end loop;
    if TYPE of non-fixed components do not match any variant> then
       RESOLVED := FALSE;
       PARAM_AVAIL := NULL>;
    else
       PARAM_AVAIL := sm_exp_type;
    end if;
end if;
else
    -- more than one possible resolution
    RESOLVED := FALSE;
    if PASS = 4 then
       ERROR 15018>
       return
    end if;
    -- try to reslove components
    COMPONENT_LIST := EMPTY>;
    for EACH component> loop
       -- resolve choice (might be an array aggregate)
       if CHOICE present> then
          CL_EXP (CHOICE_EXPR>, PASS, ANY_TYPE>,
                         CHOICE_PARM, RESOLVED);
       end if;
       -- resolve component expression
       CL_EXP (COMPONENET_EXPR>, PASS, ANY type>,
            COMPONENT_PARM, RESOLVED);
       APPEND COMPONENT_PARM to COMPONENT_LIST>,
    end loop;
    -- See which aggregate definitions are still possible.
    PARAM_AVAIL := EMPTY>;
    for EACH composite type on the CDL> loop
       if COMPONENT_LIST matches components of composite type> then
          APPEND composite type to PARAM_AVAIL>
       end if;
    end loop;
end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.323.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15017	E	Aggregate type does not match context type requirements.
15018	E	Aggregate not resolvable.

3.2.2.323.14 Examples of Data Structures. - None

3.2.2.324 Subprogram. - ov_binary

3.2.2.324.1 Purpose. - Perform overloading resolution on a node of type 'binary'.

3.2.2.324.2 Assumptions. - None

3.2.2.324.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.324.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.324.5 Nested Within. - ovp_expr

3.2.2.324.6 Host Dependencies. - None

3.2.2.324.7 Target Dependencies. - None

3.2.2.324.8 Subprogram Visibility. - Outside Package.

3.2.2.324.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.324.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_lnode_ref; -- type of result required in
                                            context in which this node
                                            appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                             result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.324.11 Side-Effects. - None

3.2.2.324.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
CL_EXP (AS_EXP1>, PASS, "BOOLEAN", DUMMY_PARM, RESOLVED);

CL_EXP (AS_EXP2>, PASS, "BOOLEAN", DUMMY_PARM, RESOLVED);

PARAM_AVAIL := "BOOLEAN";
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.324.13 Diagnostic Messages Generated. - None

3.2.2.324.14 Examples of Data Structures. - None

3.2.2.325 Subprogram. - ov_membershp

3.2.2.325.1 Purpose. - Perform overloading resolution on a node of type 'membership'.

3.2.2.325.2 Assumptions. - None

3.2.2.325.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.325.4 Used Recursively. - This subprogram is used recursively.

3.2.2.325.5 Nested Within. - ovp_expr

3.2.2.325.6 Host Dependencies. - None

3.2.2.325.7 Target Dependencies. - None

3.2.2.325.8 Subprogram Visibility. - Outside Package.

3.2.2.325.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.325.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.325.11 Side-Effects. - None

3.2.2.325.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
if PASS = 1 or PASS = 3 then
    CL_TYPE_RNG (AS_TYPE_RANGE>, PASS, ANY type>, RNG_PARM, RESOLVED);
    CL_EXP (as_exp, PASS, ANY type>, EXP_PARM, RESOLVED);
    REMOVE from EXP_PARM, everything which isn't in RNG_PARM
else
    CL_TYPE_RNG (AS_TYPE_RANGE>, PASS, SM_EXP_TYPE>, RNG_PARM, RESOLVED);
    CL_EXP (AS_EXP>, PASS, EXP_PARM, RESOLVED);
end if;

SET sm_exp_type to "boolean">
if not (|EXP_PARM| = 1 and EXP_PARM = RNG_PARM) then
    RESOLVED := FALSE;
    if PASS = 1 or PASS = 3 then
       SET sm_exp_type to EXP_PARM> -- will be reset later
    elsif PASS = 4 then
       ERROR 15019>
    end if;
end if;

PARAM_AVAIL := "BOOLEAN"
```

```
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.325.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15019	E	Unresolvable membership operator

3.2.2.325.14 Examples of Data Structures. - None

3.2.2.326 Subprogram. - ov_parenthzd

3.2.2.326.1 Purpose. - Perforn overloading resolution on a node of type 'parenthesized'.

3.2.2.326.2 Assumptions. - None

3.2.2.326.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.326.4 Used Recursively. - This subprogram is used recursively.

3.2.2.326.5 Nested Within. - ovp_expr

3.2.2.326.6 Host Dependencies. - None

3.2.2.326.7 Target Dependencies. - None

3.2.2.326.8 Subprogram Visibility. - Outside Package.

3.2.2.326.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.326.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                             result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.326.11 Side-Effects. - None

3.2.2.326.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
CL_EXP (AS_EXP>, PASS, RESULT_REQ, PARAM_AVAIL, RESOLVED);
if |PARAM_AVAIL| = 1 then
    SET sm_exp_type to PARAM_AVAIL>
end if;
```

```
if MAINTENANCE option 8> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
           "  node = ", node,
           "  resolved = ", resolved,
           "  param_avail = ", param_avail>
end if;
```

3.2.2.326.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.326.14 <u>Examples of Data Structures</u>. - None

3.2.2.327 <u>Subprogram</u>. - ov_convers

3.2.2.327.1 <u>Purpose</u>. - Perform overloading resolution on a node of type 'conversion'.

3.2.2.327.2 <u>Assumptions</u>. - None

3.2.2.327.3 <u>Implementation Language</u>. - This subprogram is written in Ada.

3.2.2.327.4 <u>Used Recursively</u>. - This subprogram is used recursively.

3.2.2.327.5 <u>Nested Within</u>. - ovp_expr

3.2.2.327.6 <u>Host Dependencies</u>. - None

3.2.2.327.7 <u>Target Dependencies</u>. - None

3.2.2.327.8 Subprogram Visibility. - Outside Package.

3.2.2.327.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.327.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                             context in which this node
                                             appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                              result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.327.11 Side-Effects. - None

3.2.2.327.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
if PASS = 1 or PASS = 3 then
    OV_EXPR (AS_EXP>, PASS, ANY type>, CON_PARM, RESOLVED);
    SET sm_exp_type to CON_PARM>
else
    OV_EXPR (AS_EXP>, PASS, SM_EXP_TYPE>, CON_PARM, RESOLVED);
    SET sm_exp_type to CON_PARM>
if !CON_PARM! /= 1 then
    RESOLVED := FALSE;
    if PASS = 4 then
       ERROR 15020>
    end if;
end if;
```

```
if MAINTENANCE option 8> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
          "  node = ", node,
          "  resolved = ", resolved,
          "  param_avail = ", param_avail>
end if;
```

3.2.2.327.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15020	E	Unresolvable expression.

3.2.2.327.14 Examples of Data Structures. - None

3.2.2.328 Subprogram. - ov_qualified

3.2.2.328.1 Purpose. - Perform overloading resolution on a node of type 'qualified'.

3.2.2.328.2 Assumptions. - None

3.2.2.328.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.328.4 Used Recursively. - This subprogram is used recursively.

3.2.2.328.5 Nested Within. - ovp_expr

3.2.2.328.6 Host Dependencies. - None

3.2.2.328.7 Target Dependencies. - None

3.2.2.328.8 Subprogram Visibility. - Outside Package.

3.2.2.328.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.328.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                             result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.328.11 Side-Effects. - None

3.2.2.328.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
if PASS = 1 then
    SET sm_dxp_type to type of as_name>
end if;
```

```
CL_EXP (AS_EXP>, PASS, SM_EXP_TYPE>, QUAL_PARM, RESOLVED);
if PASS = 4 then
     if QUAL_PARM /= sm_exp_type then
        ERROR 15021>
     end if;
end if;
if MAINTENANCE option 8> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
           "  node = ", node,
           "  resolved = ", resolved,
           "  param_avail = ", param_avail>
end if;
```

3.2.2.328.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
15021	E	Unresolvable expression.

3.2.2.328.14 Examples of Data Structures. - None

3.2.2.329 Subprogram. - ov_func_call

3.2.2.329.1 Purpose. - Perform overloading resolution on a node of type 'function_call'. This procedure will also be used for nodes of type 'procedure_call'.

3.2.2.329.2 Assumptions. - This procedure will be invoked for as many passes as are necessary to resolve the function reference or to detect an error. RESOLVED has been set to TRUE.

3.2.2.329.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.329.4 Used Recursively. - This subprogram is used recursively.

3.2.2.329.5 Nested Within. - ovp_expr

3.2.2.329.6 Host Dependencies. - None

3.2.2.329.7 Target Dependencies. - None

3.2.2.329.8 Subprogram Visibility. - Outside Package.

3.2.2.329.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.329.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.329.11 Side-Effects. - None

3.2.2.329.12 Algorithm. -

```
if MAINTENANCE option 8> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
```

```
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
-- The portion of the CDL (Candidate Definition List) which
-- contains names made visible by use clauses is used only
-- when PASS = 3 or PASS = 4

REMOVE from the CDL all definitions which don't match RESULT_REQ.>
if length (CDL) = 0 then
     --  no type-compatible definitions for this node
     RESOLVED := FALSE;
     if PASS > 2 then
        ERROR 15022>
     end if;
     return;
elsif length (CDL) = 1 then
     -- only one definition.  Resolve this node.
     SET sm_def_occurrence, sm_exp_type for NODE>
     for EACH actual parameter> loop
        CL_EXP (ACTUAL parameter node>, PASS, TYPE of corresp.
                              formal parm.>, DUMMY PARAM_AVAIL>, RESOLVED);
     end loop;
else
     -- more than one possible definition
     RESOLVED := FALSE;
     if PASS = 4 then
        ERROR 15023>
        return;
     end if;

     -- visit actual parameters
     PARM_LIST := EMPTY>    -- list of sets of types
     for EACH actual parameter> loop
        CL_EXP (ACTUAL parameter node>, PASS, ANY type>,
                              SON_PARM, RESOLVED);
        APPEND SON_PARM to PARM_LIST.>
     end loop;

     -- See which definitions are still possible
     PARM_AVAIL := EMPTY>
     for EACH element of CDL> loop
        if FORMALS don't match PARM_LIST> then
           REMOVE this element from the CDL.>
        else
           if THIS is a function_call> then
              APPEND result_type for this element (of CDL)
                                   to PARAM_AVAIL>
           end if;
        end if;
     end loop;
end if;
if MAINTENANCE option B> then
```

```
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.329.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15022	E	Subprogram call cannot match context requirements
15023	E	Ambiguous subprogram reference

3.2.2.329.14 Examples of Data Structures. - None

3.2.2.330 Subprogram. - ov_range

3.2.2.330.1 Purpose. - Perform overloading resolution on a node of type 'range'.

3.2.2.330.2 Assumptions. - None

3.2.2.330.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.330.4 Used Recursively. - This subprogram is used recursively.

3.2.2.330.5 Nested Within. - ovp_expr

3.2.2.330.6 Host Dependencies. - None

3.2.2.330.7 Target Dependencies. - None

3.2.2.330.8 Subprogram Visibility. - Outside Package.

3.2.2.330.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.330.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                              result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.330.11 Side-Effects. - None

3.2.2.330.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
          " node = ", node,
          " pass = ", pass,
          " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
if PASS = 1 or PASS = 3 then
    CL_EXP (AS_EXP1>, PASS, RESULT_REQ, EX1_PARM, RESOLVED);
    CL_EXP (AS_EXP2>, PASS, RESULT_REQ, EX2_PARM, RESOLVED);
else
```

```
    CL_EXP (AS_EXP1>, PASS, SM_BASE_TYPE>, EX1_PARM, RESOLVED);
    CL_EXP (AS_EXP2>, PASS, SM_BASE_TYPE>, EX2_PARM, RESOLVED);
T := EX1_PARM intersects EX2_PARM>:
if |T| = 1 then
    sm_base_type := T;
    PARAM_AVAIL := T;
else
    RESOLVED := FALSE;
    if PASS = 4 then
       ERROR 15024>
    end if;
end if;
if MAINTENANCE option 8> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.330.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15024	E	Range cannot be resolved

3.2.2.330.14 Examples of Data Structures. - None

3.2.2.331 Subprogram. - ov_allocator

3.2.2.331.1 Purpose. - Perform overloading resolution on a node of type 'allocator'.

3.2.2.331.2 Assumptions. - None

3.2.2.331.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.331.4 Used Recursively. - This subprogram is used recursively.

3.2.2.331.5 Nested Within. - ovp_expr

3.2.2.331.6 Host Dependencies. - None

3.2.2.331.7 Target Dependencies. - None

3.2.2.331.8 Subprogram Visibility. - Outside Package.

3.2.2.331.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.331.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- diana node to be resolved.

pass :  in  integer range 1..4; -- pass of overloading algorithm.

result_req :  in  cdm_type.c_node_ref; -- type of result required in
                                          context in which this node
                                          appears.

param_avail :  out  cdm_type.c_node_ref; -- seq of types which may
                                            result after first pass.

resolved :  out  boolean; -- set to false if the expression can't be
                             resolved on the first two passes.
```

3.2.2.331.11 Side-Effects. - None

3.2.2.331.12 Algorithm. -

```
if MAINTENANCE option B> then
   PRINT "enter", cdm_type.get_node_id (node), "node",
         " node = ", node,
         " pass = ", pass,
         " result_req = ", result_req>
end if;
if MAINTENANCE option D> then
   PRINT candidate definition list>
end if;
FIND type T for as_name.>
if AS_ACCESS_CONSTRAINT not void> then
   if T is a scalar> then
      CL_EXP (AS_ACCESS_CONSTRAINT, PASS, T, ACC_PARM, RESOLVED);
   elsif T is a record> then
      if not ALREADY a discriminant constraint> then
         if LENGTH of as_access_constraint = # of
                              discriminants> then
            CONVERT to discriminant constraint>
         end if;
      end if;

      if DISCRIMINANT constraint> then
         for EACH discriminant> loop
            CL_EXP (DISCRIMINANT>, PASS, CORRESP. type>,
                       ACC_PARM, RESOLVED);
         end loop;
      else
         OV_AGGREGATE (AS_ACCESS_CONSTRAINT>, PASS, T,
                          ACC_PARM, RESOLVED);
      end if;
   elsif T is an array> then
      if AS_ACCESS_CONSTRAINT is an index constraint> then
         for EACH index> loop
            CL_DSCRT_RNG (INDEX>, PASS, CORRESP. type>,
                             ACC_PARM, RESOLVED);
         end loop;
      else
         CL_EXP (AS_ACCESS_CONSTRAINT>, PASS, T,
                    ACC_PARM, RESOLVED);
      end if;
   end if;
end if;


-- Compute allocator type
REMOVE from CDL all definitions which don't match RESULT_REQ>
if !CDL! = 1 then
   PARAM_AVAIL := RESULT_REQ;
   sm_exp_type := RESULT_REQ;
else
   RESOLVED := FALSE;
```

```
    if PASS = 4 then
       ERROR 15025>
    end if;
end if;
if MAINTENANCE option B> then
   PRINT "leave", cdm_type.get_node_id (node), "node",
         "  node = ", node,
         "  resolved = ", resolved,
         "  param_avail = ", param_avail>
end if;
```

3.2.2.331.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15025	E	Type of allocator not resolvable

3.2.2.331.14 Examples of Data Structures. - None

3.2.2.332 Subprogram. - ov_apply

3.2.2.332.1 Purpose. - Perform overloading resolution on a node of type 'apply' which appears in an expression.

3.2.2.332.2 Assumptions. - None

3.2.2.332.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.332.4 Used Recursively. - This subprogram is used recursively.

3.2.2.332.5 Nested within. - ovp_expr

3.2.2.332.6 Host Dependencies. - None

3.2.2.332.7 Target Dependencies. - None

3.2.2.332.8 Subprogram Visibility. - Outside Package.

3.2.2.332.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.332.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- diana node to be resolved
pass : in  integer range 1..4; -- pass of overloading algorithm
result_req : in  cdm_type.c_node_ref; -- type of result required in context
                                         in which this node appears
param_avail : out  cdm_type.c_node_ref; -- seq of types which may result
                                           after first pass
resolved : out  boolean; -- set to false if the expression can't be
                            resolved on the first two passes
```

3.2.2.332.11 Side-Effects. - None

3.2.2.332.12 Algorithm. -

```
-- This routine will be called only once for each apply
-- Determine what type of 'apply' it is

if NAME attribute is a used_name_id node> then
   CONVERT 'apply' to a 'function_call'>
   INVOKE statistics recorder>
   INVOKE ov_func_call >
elsif PARAM_ASSOC_S is a range or the range attribute> then
      CONVERT 'apply' to 'slice'>
      INVOKE statistics recorder>
      INVOKE ov_slice>
else
      CONVERT 'apply' to 'indexed'>
      INVOKE statistics recorder>
      INVOKE ov_indexed>
end if;
```

3.2.2.332.13 Diagnostic Messages Generated. - None

3.2.2.332.14 Examples of Data Structures. - None

3.2.2.333 Package. - ovp_ctxt

3.2.2.333.1 Purpose. - Perform overloading resolution on Diana nodes which impose context requirements on expressons they contain.

3.2.2.333.2 Number of Subprograms. - 25

3.2.2.333.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.333.4 Additional Dependencies for Body. - ovp_clas, ovp_expr, ovp_util, cdm_as, ovp_trav, stats

3.2.2.333.5 Package Specification. -

```
package ovp_ctxt is
-- see overview in package ovp_trav
     procedure ov_pragma ...
     procedure ov_constant ...
     procedure ov_var ...
     procedure ov_number ...
     procedure ov_constrnd ...
     procedure ov_integer ...
     procedure ov_floattype ...
     procedure ov_fixedtype ...
     procedure ov_floatcon ...
     procedure ov_fixedcon ...
     procedure ov_var_part ...
     procedure ov_assign ...
     procedure ov_cond_cls ...
     procedure ov_case ...
```

```
      procedure ov_for_rev ...
      procedure ov_exit ...
      procedure ov_return ...
      procedure ov_in ...
      procedure ov_proc_call ...
      procedure ov_entry ...
      procedure ov_entry_cal ...
      procedure ov_delay ...
      procedure ov_select_cl ...
      procedure ov_simp_rep ...
      procedure ov_apply ...
end ovp_ctxt;
```

3.2.2.333.6 Elaboration Code. - None

3.2.2.333.7 Examples of Data Structures. - None

3.2.2.334 Subprogram. - ov_pragma

3.2.2.334.1 Purpose. - Perform overloading resolution on a node of type 'pragma'.

3.2.2.334.2 Assumptions. - None

3.2.2.334.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.334.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.334.5 Nested Within. - ovp_ctxt

3.2.2.334.6 Host Dependencies. - None

3.2.2.334.7 Target Dependencies. - None

3.2.2.334.8 Subprogram Visibility. - Outside Package.

3.2.2.334.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.334.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.334.11 Side-Effects. - None

3.2.2.334.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
RESOLVE as_param_assoc_s>
if PRAGMA include> then
      if ARGUMENT type not string> then
            ERROR  15026>
      end if;
elsif
      PRAGMA memory_size> then
      if ARGUMENT type not integer> then
          ERROR 15027>
      end if;
elsif PRAGMA priority> then
      if ARGUMENT not of type priority> then
          ERROR 15027>
      end if;
elsif PRAGMA storage_unit> then
      if ARGUMENT not of an integer type> then
           ERROR 15027>
      end if;
end if;
```

3.2.2.334.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15026	E	string required
15027	E	integer type required

3.2.2.334.14 Examples of Data Structures. - None

3.2.2.335 Subprogram. - ov_constant

3.2.2.335.1 Purpose. - Perform overloading resolution on a node of type 'constant'.

3.2.2.335.2 Assumptions. - None

3.2.2.335.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.335.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.335.5 Nested Within. - ovp_ctxt

3.2.2.335.6 Host Dependencies. - None

3.2.2.335.7 Target Dependencies. - None

3.2.2.335.8 <u>Subprogram Visibility</u>. - Outside Package.

3.2.2.335.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.335.10 <u>Formal Parameters</u>. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.335.11 <u>Side-Effects</u>. - None

3.2.2.335.12 <u>Algorithm</u>. -

```
overload (AS_OBJECT_DEF>, AS_TYPE_SPEC>);
```

3.2.2.335.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.335.14 <u>Examples of Data Structures</u>. - None

3.2.2.336 <u>Subprogram</u>. - ov_var

3.2.2.336.1 <u>Purpose</u>. - Perform overloading resolution on a node of type 'var'

3.2.2.336.2 <u>Assumptions</u>. - None

3.2.2.336.3 <u>Implementation Language</u>. - This subprogram is written in Ada.

3.2.2.336.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.336.5 Nested Within. - ovp_ctxt

3.2.2.336.6 Host Dependencies. - None

3.2.2.336.7 Target Dependencies. - None

3.2.2.336.8 Subprogram Visibility. - Outside Package.

3.2.2.336.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.336.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.336.11 Side-Effects. - None

3.2.2.336.12 Algorithm. -

```
overload (AS_OBJECT_DEF>, AS_TYPE_SPEC>);
```

3.2.2.336.13 Diagnostic Messages Generated. - None

3.2.2.336.14 Examples of Data Structures. - None

3.2.2.337 Subprogram. - ov_number

3.2.2.337.1 Purpose. - Perform overloading resolution on a node of type 'number'

3.2.2.337.2 Assumptions. - None

3.2.2.337.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.337.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.337.5 Nested Within. - ovp_ctxt

3.2.2.337.6 Host Dependencies. - None

3.2.2.337.7 Target Dependencies. - None

3.2.2.337.8 Subprogram Visibility. - Outside Package.

3.2.2.337.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.337.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.337.11 Side-Effects. - None

3.2.2.337.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP>, "any type");
if SM_EXP_TYPE of as_exp is not universal_integer or universal_real> then
     ERROR 15028>
end if;
```

3.2.2.337.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15028	E	literal expresson required

3.2.2.337.14 Examples of Data Structures. - None

3.2.2.338 Subprogram. - ov_constrnd

3.2.2.338.1 Purpose. - Perform overloading resolution on a node of type 'constrained'

3.2.2.338.2 Assumptions. - None

3.2.2.338.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.338.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.338.5 Nested within. - ovp_ctxt

3.2.2.338.6 Host Dependencies. - None

3.2.2.338.7 Target Dependencies. - None

3.2.2.338.8 Subprogram Visibility. - Outside Package.

3.2.2.338.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.338.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.338.11 Side-Effects. - None

3.2.2.338.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
GET type T of as_name>
cl_constrnt (AS_CONSTRAINT>, T);
```

3.2.2.338.13 Diagnostic Messages Generated. - None

3.2.2.338.14 Examples of Data Structures. - None

3.2.2.339 Subprogram. - ov_integer

3.2.2.339.1 Purpose. - Perform overloading resolution on a node of type 'integer'

3.2.2.339.2 Assumptions. - None

3.2.2.339.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.339.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.339.5 Nested Within. - ovp_ctxt

3.2.2.339.6 Host Dependencies. - None

3.2.2.339.7 Target Dependencies. - None

3.2.2.339.8 Subprogram Visibility. - Outside Package.

3.2.2.339.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.339.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.339.11 Side-Effects. - None

3.2.2.339.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
GET as_range>
```

```
overload (AS_EXP1>, "any type");
if SM_EXP_TYPE of as_exp1 not an integer type> then
     ERROR 15029>
end if;
overload (AS_EXP2>, "any type");
if SM_EXP_TYPE of as_exp2 not an integer type> then
     ERROR 15029>
end if;
```

3.2.2.339.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15029	E	integer type required

3.2.2.339.14 Examples of Data Structures. - None

3.2.2.340 Subprogram. - ov_floattype

3.2.2.340.1 Purpose. - Perform overloading resolution on a node of type 'float' which is used as a type specification.

3.2.2.340.2 Assumptions. - None

3.2.2.340.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.340.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.340.5 Nested Within. - ovp_ctxt

3.2.2.340.6 Host Dependencies. - None

3.2.2.340.7 Target Dependencies. - None

3.2.2.340.8 Subprogram Visibility. - Outside Package.

3.2.2.340.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.340.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.340.11 Side-Effects. - None

3.2.2.340.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP>, "any type";
if TYPE of as_exp not an integer type> then
      ERROR 15030>
end if;
GET as_range_void>
if NOT void> then
      overload (AS_EXP1>, "any type");
      if TYPE of expl not a real type> then
            ERROR 15031>
      end if;
      overload (AS_EXP2>, any type");
      if TYPE of exp2 not a real type> then
            ERROR 15031>
      end if;
end if;
```

3.2.2.340.13 Diagnostic Messages Generated -

CODE	SEVERITY N.W.E.S.F	TEXT
15030	E	integer required
15031	E	real type required

3.2.2.340.14 Examples of Data Structures. - None

3.2.2.341 Subprogram. - ov_fixedtype

3.2.2.341.1 Purpose. - Perform overloading resolution on a node of type 'fixed' which is used as a type specification.

3.2.2.341.2 Assumptions. - None

3.2.2.341.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.341.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.341.5 Nested Within. - ovp_ctxt

3.2.2.341.6 Host Dependencies. - None

3.2.2.341.7 Target Dependencies. - None

3.2.2.341.8 Subprogram Visibility. - Outside Package.

3.2.2.341.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.341.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.341.11 Side-Effects. - None

3.2.2.341.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP>, "any type");
if TYPE of as_exp not a real type> then
     ERROR 15030>
end if;
GET as_range_void>
if NOT void> then
     overload (AS_EXP1>, "any type");
     if TYPE of exp1 not a real type> then
          ERROR 15031>
     end if;
     overload (AS_EXP2>, "any type");
     if TYPE of exp2 not a real type> then
          ERROR 15031>
     end if;
end if;
```

3.2.2.341.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15030	E	integer type required
15031	E	real type required

3.2.2.341.14 Examples of Data Structures. - None

3.2.2.342 Subprogram. - ov_floatcon

3.2.2.342.1 Purpose. - Perform overloading resolution on a node of type 'float' which appears as a constraint.

3.2.2.342.2 Assumptions. - None

3.2.2.342.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.342.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.342.5 Nested Within. - ovp_ctxt

3.2.2.342.6 Host Dependencies. - None

3.2.2.342.7 Target Dependencies. - None

3.2.2.342.8 Subprogram Visibility. - Outside Package.

3.2.2.342.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.342.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
result_req :  in  cdm_type.c_node_ref;  -- type of result
                                          required in context
                                          in which this node
                                          appears
```

3.2.2.342.11 Side-Effects. - None

3.2.2.342.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP>, "any type");
if TYPE of as_exp not an integer type> then
     ERROR 15030>
end if;
if AS_RANGE_VOID not void> then
     overloadrng (AS_RANGE_VOID>, TYPE of as_exp>);
```

3.2.2.342.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
15030	E	integer type expected

3.2.2.342.14 Examples of Data Structures. - None

3.2.2.343 Subprogram. - ov_fixedcon

3.2.2.343.1 Purpose. - Perform overloading resolution on a node of type 'fixed' which appears as a constraint.

3.2.2.343.2 Assumptions. - None

3.2.2.343.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.343.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.343.5 Nested within. - ovp_ctxt

3.2.2.343.6 Host Dependencies. - None

3.2.2.343.7 Target Dependencies. - None

3.2.2.343.8 Subprogram Visibility. - Outside Package.

3.2.2.343.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.343.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
result_req :  in  cdm_type.c_node_ref;  -- type of result
                                           required in context
                                           in which this node
                                           appears
```

3.2.2.343.11 Side-Effects. - None

3.2.2.343.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP>, "any type");
if TYPE of as_exp not a real type> then
     ERROR 15031>
end if;
if AS_RANGE_VOID not void>, then
     overloadrng (AS_RANGE_VOID>, result_req);
```

3.2.2.343.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15031	E	real type required

3.2.2.343.14 Examples of Data Structures. - None

3.2.2.344 Subprogram. - ov_var_part

3.2.2.344.1 Purpose. - Perform overloading resolution on a node of type 'variant part'.

3.2.2.344.2 Assumptions. - None

3.2.2.344.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.344.4 Used Recursively. - This subprogram is used recursively.

3.2.2.344.5 Nested within. - ovp_ctxt

3.2.2.344.6 Host Dependencies. - None

3.2.2.344.7 Target Dependencies. - None

3.2.2.344.8 Subprogram Visibility. - Outside Package.

3.2.2.344.9 Function/Procedure. - This suoprogram is a Procedure.

3.2.2.344.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.344.11 Side-Effects. - None

3.2.2.344.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_tyoe.get_node_id (node), "node">
end if;
GET type T of discriminant named in as_name>
for EACH element in as_variant_s> loop
     ov_choices (AS_CHOICE_S of list element>, T);
     ov_record (AS_RECORD of list element>);
end loop;
```

3.2.2.344.13 Diagnostic Messages Generated. - None

3.2.2.344.14 Examples of Data Structures. - None

3.2.2.345 Subprogram. - ov_assign

3.2.2.345.1 Purpose. - Perform overloading resolution on a node of type 'assign'.

3.2.2.345.2 Assumptions. - None

3.2.2.345.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.345.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.345.5 Nested within. - ovp_ctxt

3.2.2.345.6 Host Dependencies. - None

3.2.2.345.7 Target Dependencies. - None

3.2.2.345.8 Subprogram Visibility. - Outside Package.

3.2.2.345.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.345.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.345.11 Side-Effects. - None

3.2.2.345.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
pass := 1
resolved := true;
while (pass 4 and resolved = true) loop
      resolved := false
      cl_exp (as_name, pass, "any type", left_parm, left_
            resolved);
      if !left_parm! = 1 then
            T := left_parm;
            cl_exp (as_exp, pass, T, right_parm, right_resolved);
      else
```

```
          -- must get resolution of LHS from RHS.
          cl_exp (as_exp, pass, "any type", right_parm, right_resolved);
          T := LEFT_PARM intersects right_parm>;
     end if;
     if |T| = 1 then
          -- run second (or fourth) pass, if necessary
          if not left_resolved then
               cl_exp (as_name pass+1, T, left_parm, resolved);
          end if;
          if not right_resolved then
               cl_exp (as_name, pass+1, T, right_parm, resolved);
          end if;
     else
          resolved := false;
     end if;
     pass := pass+2;
end loop;
if resolved = false then
     ERROR 15032>
end if;
```

3.2.2.345.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
15032	E	type of assignment expression does not match type of target

3.2.2.345.14 <u>Examples of Data Structures</u>. - None

3.2.2.346 <u>Subprogram</u>. - ov_cond_cls

3.2.2.346.1 <u>Purpose</u>. - Perform overloading resolution on a node of type 'cond_clause'.

3.2.2.346.2 <u>Assumptions</u>. - None

3.2.2.346.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.346.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.346.5 Nested Within. - ovp_ctxt

3.2.2.346.6 Host Dependencies. - None

3.2.2.346.7 Target Dependencies. - None

3.2.2.346.8 Subprogram Visibility. - Outside Package.

3.2.2.346.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.346.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  - node to be resolved
```

3.2.2.346.11 Side-Effects. - None

3.2.2.346.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP_VOID>, "boolean");
RESOLVE as_stm_s>
```

3.2.2.346.13 Diagnostic Messages Generated. - None

3.2.2.346.14 Examples of Data Structures. - None

3.2.2.347 Subprogram. - ov_case

3.2.2.347.1 Purpose. - Perform overloading resolution on a node of type 'case'

3.2.2.347.2 Assumptions. - None

3.2.2.347.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.347.4 Used Recursively. - This subprogram is used recursively.

3.2.2.347.5 Nested Within. - ovp_ctxt

3.2.2.347.6 Host Dependencies. - None

3.2.2.347.7 Target Dependencies. - None

3.2.2.347.8 Subprogram Visibility. - Outside Package.

3.2.2.347.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.347.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.347.11 Side-Effects. - None

3.2.2.347.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP, "any type");
T := TYPE of as_exp>;
GET as_alternative_s>
for EACH element of as_list> loop
      GET as_choice_s>
      for EACH element of as_list> loop
            cl_choice (LIST element>, T);
      end loop;
      RESOLVE as_stm_s>
end loop;
```

3.2.2.347.13 Diagnostic Messages Generated. - None

3.2.2.347.14 Examples of Data Structures. -

none

3.2.2.348 Subprogram. - ov_for_rev

3.2.2.348.1 Purpose. - Perform overload resolution on a node of type 'for', or of type 'reverse'.

3.2.2.348.2 Assumptions. - None

3.2.2.348.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.348.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.348.5 Nested Within. - ovp_ctxt

3.2.2.348.6 Host Dependencies. - None

3.2.2.348.7 Target Dependencies. - None

3.2.2.348.8 Subprogram Visibility. - Outside Package.

3.2.2.348.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.348.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.348.11 Side-Effects. - None

3.2.2.348.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
GET as_dscrt_range>
if as_dscrt_range is 'constrained' then
'     RESOLVE as_dscrt_range>
else
      -- must be 'range' node
      overloadrng (AS_DSCRT_RANGE>, "any type");
      if TYPE of as_dscrt_range is universal integer
           SET type of loop parameter (as_id) to integer>
      else
           SET type of loop parameter (as_id) to type of dscrt_
```

```
                range>
    end if;
end if;
```

3.2.2.348.13 Diagnostic Messages Generated. - None

3.2.2.348.14 Examples of Data Structures. - None

3.2.2.349 Subprogram. - ov_exit

3.2.2.349.1 Purpose. - Perform overloading resolution on a node of type 'exit'

3.2.2.349.2 Assumptions. - None

3.2.2.349.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.349.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.349.5 Nested Within. - ovp_ctxt

3.2.2.349.6 Host Dependencies. - None

3.2.2.349.7 Target Dependencies. - None

3.2.2.349.8 Subprogram Visibility. - Outside Package.

3.2.2.349.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.349.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref; -- node tobe resolved
```

3.2.2.349.11 Side-Effects. - None

3.2.2.349.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP_VOID>, boolean);
```

3.2.2.349.13 Diagnostic Messages Generated. - None

3.2.2.349.14 Examples of Data Structures. - None

3.2.2.350 Subprogram. - ov_return

3.2.2.350.1 Purpose. - Perform overloading resolution on a node of type 'return'

3.2.2.350.2 Assumptions. - None

3.2.2.350.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.350.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.350.5 Nested Within. - ovp_ctxt

3.2.2.350.6 Host Dependencies. - None

3.2.2.350.7 Target Dependencies. - None

3.2.2.350.8 Subprogram Visibility. - Outside Package.

3.2.2.350.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.350.10 Formal Parameters. -

```
node :  in cdm_type.c_node_ref;  -- node to be resolved.
```

3.2.2.350.11 Side-Effects. - None

3.2.2.350.12 Algorithm. -

```
overload (AS_EXP_TYPE>, RESULT type of function>);
```

3.2.2.350.13 Diagnostic Messages Generated. - None

3.2.2.350.14 Examples of Data Structures. - None

3.2.2.351 Subprogram. - ov_in

3.2.2.351.1 Purpose. - Perform overloading resolution on a node of type 'in'

3.2.2.351.2 Assumptions. - None

3.2.2.351.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.351.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.351.5 Nested Within. - ovp_ctxt

3.2.2.351.6 Host Dependencies. - None

3.2.2.351.7 Target Dependencies. - None

3.2.2.351.8 Subprogram Visibility. - Outside Package.

3.2.2.351.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.351.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  - node to be resolvedq
```

3.2.2.351.11 Side-Effects. - None

3.2.2.351.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
RESOLVE as_type_spec>
overload (AS_EXP_VOID>, AS_TYPE_SPEC>);
```

3.2.2.351.13 Diagnostic Messages Generated. - None

3.2.2.351.14 Examples of Data Structures. - None

3.2.2.352 Subprogram. - ov_proc_call

3.2.2.352.1 Purpose. - Perform overloading resolution on a node of type 'procedure call'

3.2.2.352.2 Assumptions. - None

3.2.2.352.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.352.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.352.5 Nested Within. - ovp_ctxt

3.2.2.352.6 Host Dependencies. - None

3.2.2.352.7 Target Dependencies. - None

3.2.2.352.8 Subprogram Visibility. - Outside Package.

3.2.2.352.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.352.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.352.11 Side-Effects. - None

3.2.2.352.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (node, "any type");
if NODE was resolved to an entry> then
     CHANGE Diana tree to an entry_call node>
end if;
```

3.2.2.352.13 Diagnostic Messages Generated. - None

3.2.2.352.14 Examples of Data Structures. - None

3.2.2.353 Subprogram. - ov_entry

3.2.2.353.1 Purpose. - Perform overloading resolution on a node of type 'entry'

3.2.2.353.2 Assumptions. - None

3.2.2.353.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.353.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.353.5 Nested within. - ovp_ctxt

3.2.2.353.6 Host Dependencies. - None

3.2.2.353.7 Target Dependencies. - None

3.2.2.353.8 Subprogram Visibility. - Outside Package.

3.2.2.353.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.353.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.353.11 Side-Effects. - None

3.2.2.353.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
nt:= NODE type of as_dscrt_range_void>
case nt of
     when constrained =>
          RESOLVE as_dscrt_range_void>
     when range =>
          overloadrng (AS_DSCRT_RANGE_VOID>, "any type");
          if TYPE of range is universal literal> then
               SET type to integer>
          end if;
     when void =>
```

```
        return;
end case;
```

3.2.2.353.13 Diagnostic Messages Generated. - None

3.2.2.353.14 Examples of Data Structures. - None

3.2.2.354 Subprogram. - ov_entry_cal

3.2.2.354.1 Purpose. - Perform overloading resolution on a node of type 'entry_call"

3.2.2.354.2 Assumptions. - None

3.2.2.354.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.354.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.354.5 Nested within. - ovp_ctxt

3.2.2.354.6 Host Dependencies. - None

3.2.2.354.7 Target Dependencies. - None

3.2.2.354.8 Subprogram Visibility. - Outside Package.

3.2.2.354.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.354.10 Formal Parameters. -

```
node :  in cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.354.11 Side-Effects. - None

3.2.2.354.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
-- This node will be here only if name resolution has uniquely
-- determined that as_name is an entry.
for EACH actual parameter in as_param_assoc_s> loop
     overload (ACTUAL parameter>, TYPE of corresponding
          parameter>);
end loop;
```

3.2.2.354.13 Diagnostic Messages Generated. - None

3.2.2.354.14 Examples of Data Structures. - None

3.2.2.355 Subprogram. - ov_delay

3.2.2.355.1 Purpose. - Perform overloading resolution on a node of type 'delay'

3.2.2.355.2 Assumptions. - None

3.2.2.355.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.355.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.355.5 Nested Within. - ovp_ctxt

3.2.2.355.6 Host Dependencies. - None

3.2.2.355.7 Target Dependencies. - None

3.2.2.355.8 Subprogram Visibility. - Outside Package.

3.2.2.355.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.355.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.355.11 Side-Effects. - None

3.2.2.355.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (node, "duration");
```

3.2.2.355.13 Diagnostic Messages Generated. - None

3.2.2.355.14 Examples of Data Structures. - None

3.2.2.356 Subprogram. - ov_select_cl

3.2.2.356.1 Purpose. - Perform overloading resolution on a node of type 'select_clause'

3.2.2.356.2 Assumptions. - None

3.2.2.356.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.356.4 Used Recursively. - This subprogram is used recursively.

3.2.2.356.5 Nested Within. - ovp_ctxt

3.2.2.356.6 Host Dependencies. - None

3.2.2.356.7 Target Dependencies. - None

3.2.2.356.8 Subprogram Visibility. - Outside Package.

3.2.2.356.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.356.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.356.11 Side-Effects. - None

3.2.2.356.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
overload (AS_EXP_VOID>, boolean);
RESOLVE as_stms_s>
```

3.2.2.356.13 Diagnostic Messages Generated. - None

3.2.2.356.14 Examples of Data Structures. - None

3.2.2.357 Subprogram. - ov_simp_rep

3.2.2.357.1 Purpose. - Perform overloading resolution on a node of type 'simple_rep'

3.2.2.357.2 Assumptions. - None

3.2.2.357.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.357.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.357.5 Nested Within. - ovp_ctxt

3.2.2.357.6 Host Dependencies. - None

3.2.2.357.7 Target Dependencies. - None

3.2.2.357.8 Subprogram Visibility. - Outside Package.

3.2.2.357.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.357.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref;  -- node to be resolved
```

3.2.2.357.11 Side-Effects. - None

3.2.2.357.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at", cdm_type.get_node_id (node), "node">
end if;
if AS_NAME is an attribute> then
     -- length specification
     overload (AS_EXP>, "any type");
     if ATTRIBUTE is actual_delta> then
          if AS_EXP is not real> then
              ERROR 15031>
          end if;
     else
          if AS_EXP is not an integer type> then
               ERROR 15030>
          end if;
     end if;
else
     -- enumeration type representation
     overload (AS_EXP>, ARRAY of universal_integer>);
end if;
```

3.2.2.357.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
15031	E	real expression required

15030 E integer expression required

3.2.2.357.14 Examples of Data Structures. - None

3.2.2.358 Subprogram. - ov_apply

3.2.2.358.1 Purpose. - Perform overloading resolution on a node of type 'apply'.

3.2.2.358.2 Assumptions. - None

3.2.2.358.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.358.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.358.5 Nested Within. - ovp_ctxt

3.2.2.358.6 Host Dependencies. - None

3.2.2.358.7 Target Dependencies. - None

3.2.2.358.8 Subprogram Visibility. - Outside Package.

3.2.2.358.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.358.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- diana node to be resolved
pass : in  integer range 1..4; -- pass of overloading algorithm
result_req : in  cdm_type.c_node_ref; -- type of result required in
                                         context in which this node appears
param_avail : out  cdm_type.c_node_ref; -- seq of types which may result
                                           after first pass
resolved : out  boolean; -- set to false if the expression can't be
                            resolved on the first two passes
```

3.2.2.358.11 Side-Effects. - None

3.2.2.358.12 Algorithm. -

```
overload (node, "any type");
if NODE was resolved to an entry> then
   CHANGE Diana node to an 'entry_call' node>
else
   CHANGE Diana node to a 'procedure_call' node>
end if;
INVOKE statistics recorder>
```

3.2.2.358.13 Diagnostic Messages Generated. - None

3.2.2.358.14 Examples of Data Structures. - None

3.2.2.359 Package. - ovp_trav

3.2.2.359.1 Purpose. - Traverse Diana nodes to perform overloading resolution.

3.2.2.359.2 Number of Subprograms. - 44

3.2.2.359.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.359.4 Additional Dependencies for Body. - ovp_clas, ovp_ctxt, cdm_as

3.2.2.359.5 Package Specification. -

```
package ovp_trav is
--    OVERVIEW OF THE OVERLOADING RESOLUTION FUNCTION
--
-- The overloading resolution function is organized as a recursive descent
-- traversal of the Diana tree.  It consists of five packages, OVP_TRAV,
-- OVP_CTXT, OVP_EXPR, OVP_CLAS, and OVP_UTIL.
--
-- OVP_TRAV:
--
-- The procedures of OVP_TRAV are used to traverse the portions of the
-- Diana tree which do not involve expressions.  Each procedure corresponds
-- to a Diana node type.  The algorithm for the procedure contains one
-- pseudo-code statement for each structural attribute of the node type.
-- For example, the procedure ov-function corresponds to the node type
-- 'function' which has attributes:
--
--      as_param_s : param_s
--      as_constrained_void : CONSTRAINED_VOID
--
-- The algorithm for ov_function is
--
--      RESOLVE as_param_s>
--      RESOLVE as_constrained_void>
--
-- The first statement means, "invoke the procedure which handles nodes of
-- type param_s."  The second means, "invoke the procedure which handles nodes
-- of the class CONSTRAINED_VOID."
--
--
-- OVP_CLAS:
--
-- The procedures of package OVP_CLAS handle Diana classes.  There is one
-- procedure for each class (except for one-element classes) which consists
-- of one choice for each node type of the class.  The pseudo-code
-- corresponding to this choice consists of a call to the procedure which
-- handles that node type.
--
--
-- OVP_CTXT:
--
-- Each procedure of the package OVP_CTXT handles a node type which has
-- an expression as one of its structural attributes.  In general, the
-- node type will impose some context requirement on the type required
-- for the expression.  This context requirement is passed to the
-- procedure OVP_UTIL.OVERLOAD or OVP_UTIL.OVERLOADRNG which handle
-- overloading of complete expressions.  Structural attributes which
-- are not expressions are handled as in OVP_TRAV.  For example,
-- OVP_COND_CLS, which handles nodes of type 'cond_clause', has the
-- algorithm:
--
--      OVERLOAD (AS_EXP_VOID>, "BOOLEAN");
--      RESOLVE as_stm_s>
--
-- The first line means, "perform overloading resolution on the as_exp_void
-- attribute, requiring a BOOLEAN result type."  The second line means,
-- "invoke the procedure which handles nodes of type 'stm_s'".
--
--
-- OVP_EXPR and OVP_UTIL:
--
-- The procedures of the package OVP_EXPR correspond th Diana nodes which
-- are part of expressions.  The procedure OV_FUNC_CALL, which is typical of
-- the procedures in this package, has the same purpose as most of the
-- published overloading resolution algorithms.  That is, it describes the
-- action to be taken during one "pass" for one function invocation.
-- To perform a pass over an entire expression which consists of only function
-- calls, OV_FUNC_CALL is invoked recursively, with one invocation for each
-- "pass".  (Here, one pass is defined as a top-down traversal followed by a
-- bottom-up traversal of a complete expression.)  To complete overloading
-- resolution, four passes may be required (two which ignore use clauses,
-- and two which take use clauses into account).  The procedure
-- OVP_UTIL.OVERLOAD (or OVP_UTIL.OVERLOADRNG) acts as a driver for the
-- four passes over a complete expression.
--
-- In the published literature, other components of expressions
-- (aggregates, literals, allocators, etc.) are not introduced into the
-- overloading algorithms since then are said to be logically equivalent to
-- function calls.  This can lead to a simple implementation of
-- overloading resolution if the internal representation makes all
-- expression components look like function calls.  Since this is not the
-- case with Diana, the package OVP_EXPR contains a different procedure for
-- each node type which can be part of an expression.  All of these procedures
-- are meant to call each other recursively, and to be driven by
-- OVP_UTIL.OVERLOAD or OVP_UTIL.OVERLOADRNG.
   procedure ov_derived...
   procedure ov_array...
   procedure ov_dscrtrngs...
   procedure ov_record...
   procedure ov_vars...
   procedure ov_access...
   procedure ov_stm_s...
   procedure ov_if...
   procedure ov_loop...
   procedure ov_while...
   procedure ov_block...
   procedure ov_subp_dcl...
   procedure ov_procedure...
   procedure ov_function...
   procedure ov_param_s...
   procedure ov_in_out...
   procedure ov_out...
   procedure ov_subp_body...
   procedure ov_pack_spec...
   procedure ov_pack_dcl...
   procedure ov_decl_s...
   procedure ov_dcl_rep_s...
   procedure ov_pack_body...
   procedure ov_accept...
   procedure ov_select...
   procedure ov_cond_entr...
   procedure ov_timed_ent...
   procedure ov_abort...
   procedure ov_pragma_s...
   procedure ov_comp_unit...
   procedure ov_subunit...
   procedure ov_gen_asscs...
   procedure ov_rec_rep...
   procedure ov_comp_rep...
   procedure ov_address...
   procedure ov_code...
   procedure ov_task_body...
   procedure ov_type...
   procedure ov_task_dcl...
   procedure ov_subtype...
   procedure ov_labeled...
   procedure ov_named_stm...
   procedure ov_raise...
   procedure ov_task_spec...
end ovp_trav;
```

3.2.2.359.6 Elaboration Code. - None

3.2.2.359.7 Examples of Data Structures. - None

3.2.2.360 Subprogram. - ov_derived

3.2.2.360.1 Purpose. - Perform overloading resolution on a node of type 'derived'.

3.2.2.360.2 Assumptions. - None

3.2.2.360.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.360.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.360.5 Nested Within. - ovp_trav

3.2.2.360.6 Host Dependencies. - None

3.2.2.360.7 Target Dependencies. - None

3.2.2.360.8 Subprogram Visibility. - Outside Package.

3.2.2.360.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.360.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref ; -- node to be resolved.
```

3.2.2.360.11 Side-Effects. - None

3.2.2.360.12 Algorithm. -

```
RESOLVE as_constrained>
```

3.2.2.360.13 Diagnostic Messages Generated. - None

3.2.2.360.14 Examples of Data Structures. - None

3.2.2.361 Subprogram. - ov_array

3.2.2.361.1 Purpose. - Perform overloading resolution on a node of type 'array'.

3.2.2.361.2 Assumptions. - None

3.2.2.361.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.361.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.361.5 Nested within. - ovp_trav

3.2.2.361.6 Host Dependencies. - None

3.2.2.361.7 Target Dependencies. - None

3.2.2.361.8 Subprogram Visibility. - Outside Package.

3.2.2.361.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.361.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.361.11 Side-Effects. - None

3.2.2.361.12 Algorithm. -

```
for EACH element of as_dscrt_range> loop
    overloadrng (LIST element>, "any type">;
    if TYPE of list element is universal_integer> then
       SET to integer.>
    end if;
end loop;
RESOLVE as_dscrt_range>
RESOLVE as_constrained>
```

3.2.2.361.13 Diagnostic Messages Generated. - None

3.2.2.361.14 Examples of Data Structures. - None

3.2.2.362 Subprogram. - ov_dscrtrngs

3.2.2.362.1 Purpose. - Perform overloading resolution on a node of type 'dscrt_range_s'.

3.2.2.362.2 Assumptions. - None

3.2.2.362.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.362.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.362.5 Nested within. - ovp_trav

3.2.2.362.6 Host Dependencies. - None

3.2.2.362.7 Target Dependencies. - None

3.2.2.362.8 Subprogram Visibility. - Outside Package.

3.2.2.362.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.362.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.362.11 Side-Effects. - None

3.2.2.362.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.362.13 Diagnostic Messages Generated. - None

3.2.2.362.14 Examples of Data Structures. - None

3.2.2.363 Subprogram. - ov_record

3.2.2.363.1 Purpose. - Perform overloading resolution on a node of type 'record'.

3.2.2.363.2 Assumptions. - None

3.2.2.363.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.363.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.363.5 Nested Within. - ovp_trav

3.2.2.363.6 Host Dependencies. - None

3.2.2.363.7 Target Dependencies. - None

3.2.2.363.8 Subprogram Visibility. - Outside Package.

3.2.2.363.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.363.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.363.11 Side-Effects. - None

3.2.2.363.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.363.13 Diagnostic Messages Generated. - None

3.2.2.363.14 Examples of Data Structures. - None

3.2.2.364 Subprogram. - ov_vars

3.2.2.364.1 Purpose. - Perform overloading resolution on a node of type 'var_s'.

3.2.2.364.2 Assumptions. - None

3.2.2.364.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.364.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.364.5 Nested within. - ovp_trav

3.2.2.364.6 Host Dependencies. - None

3.2.2.364.7 Target Dependencies. - None

3.2.2.364.8 Subprogram Visibility. - Outside Package.

3.2.2.364.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.364.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; --  node to be resolved.
```

3.2.2.364.11 Side-Effects. - None

3.2.2.364.12 Algorithm. -

```
for EACH element of as_list>  loop
    RESOLVE list element>
end loop;
```

3.2.2.364.13 Diagnostic Messages Generated. - None

3.2.2.364.14 Examples of Data Structures. - None

3.2.2.365 Subprogram. - ov_access

3.2.2.365.1 Purpose. - Perform overloading resolution on a node of type 'access'.

3.2.2.365.2 Assumptions. - None

3.2.2.365.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.365.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.365.5 Nested Within. - ovp_trav

3.2.2.365.6 Host Dependencies. - None

3.2.2.365.7 Target Dependencies. - None

3.2.2.365.8 Subprogram Visibility. - Outside Package.

3.2.2.365.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.365.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.365.11 Side-Effects. - None

3.2.2.365.12 Algorithm. -

```
RESOLVE as_constrained>
```

3.2.2.365.13 Diagnostic Messages Generated. - None

3.2.2.365.14 Examples of Data Structures. - None

3.2.2.366 Subprogram. - ov_stm_s

3.2.2.366.1 Purpose. - Perform overloading resolution on a node of type 'stm_s'.

3.2.2.366.2 Assumptions. - None

3.2.2.366.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.366.4 Used Recursively. - This subprogram is used recursively.

3.2.2.366.5 Nested Within. - ovp_trav

3.2.2.366.6 Host Dependencies. - None

3.2.2.366.7 Target Dependencies. - None

3.2.2.366.8 Subprogram Visibility. - Outside Package.

3.2.2.366.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.366.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.366.11 Side-Effects. - None

3.2.2.366.12 Algorithm. -

```
for EACH element of as_list > loop
    RESOLVE list element>
end loop;
```

3.2.2.366.13 Diagnostic Messages Generated. - None

3.2.2.366.14 Examples of Data Structures. - None

3.2.2.367 Subprogram. - ov_if

3.2.2.367.1 Purpose. - Perform overloading resolution on a node of type 'of'.

3.2.2.367.2 Assumptions. - None

3.2.2.367.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.367.4 Used Recursively. - This subprogram is used recursively.

3.2.2.367.5 Nested Within. - ovp_trav

3.2.2.367.6 Host Dependencies. - None

3.2.2.367.7 Target Dependencies. - None

3.2.2.367.8 Subprogram Visibility. - Outside Package.

3.2.2.367.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.367.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.367.11 Side-Effects. - None

3.2.2.367.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.367.13 Diagnostic Messages Generated. - None

3.2.2.367.14 Examples of Data Structures. - None

3.2.2.368 Subprogram. - ov_loop

3.2.2.368.1 Purpose. - Perform overloading resolution on a node of type 'loop'.

3.2.2.368.2 Assumptions. - None

3.2.2.368.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.368.4 Used Recursively. - This subprogram is used recursively.

3.2.2.368.5 Nested Within. - ovp_trav

3.2.2.368.6 Host Dependencies. - None

3.2.2.368.7 Target Dependencies. - None

3.2.2.368.8 Subprogram Visibility. - Outside Package.

3.2.2.368.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.368.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.368.11 Side-Effects. - None

3.2.2.368.12 Algorithm. -

```
RESOLVE as_iteration.>
RESOLVE as_stm_s.>
```

3.2.2.368.13 Diagnostic Messages Generated. - None

3.2.2.368.14 Examples of Data Structures. - None

3.2.2.369 Subprogram. - ov_while

3.2.2.369.1 Purpose. - Perform overloading resolution on a node of type 'while'.

3.2.2.369.2 Assumptions. - None

3.2.2.369.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.369.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.369.5 Nested Within. - ovp_trav

3.2.2.369.6 Host Dependencies. - None

3.2.2.369.7 Target Dependencies. - None

3.2.2.369.8 Subprogram Visibility. - Outside Package.

3.2.2.369.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.369.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.369.11 Side-Effects. - None

3.2.2.369.12 Algorithm. -

```
OVERLOAD (AS_EXP), boolean);
```

3.2.2.369.13 Diagnostic Messages Generated. - None

3.2.2.369.14 Examples of Data Structures. - None

3.2.2.370 Subprogram. - ov_block

3.2.2.370.1 Purpose. - Perform overloading resolution on a node of type 'block'.

3.2.2.370.2 Assumptions. - None

3.2.2.370.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.370.4 Used Recursively. - This subprogram is used recursively.

3.2.2.370.5 Nested within. - ovp_trav

3.2.2.370.6 Host Dependencies. - None

3.2.2.370.7 Target Dependencies. - None

3.2.2.370.8 Subprogram Visibility. - Outside Package.

3.2.2.370.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.370.10 Formal Parameters. -

node: in cdm_type.c_node_ref; node to be resolved.

3.2.2.370.11 Side-Effects. - None

3.2.2.370.12 Algorithm. -

```
RESOLVE as_item_s>
RESOLVE as_stm_s>
RESOLVE as_alternative_s>
```

3.2.2.370.13 Diagnostic Messages Generated. - None

3.2.2.370.14 Examples of Data Structures. -

None

3.2.2.371 Subprogram. - ov_subo_dcl

3.2.2.371.1 Purpose. - Perform overloading resolution on a node of type 'subprogram_dccl'.

3.2.2.371.2 Assumptions. - None

3.2.2.371.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.371.4 Used Recursively. - This subprogram is used recursively.

3.2.2.371.5 Nested within. - ovp_trav

3.2.2.371.6 Host Dependencies. - None

3.2.2.371.7 Target Dependencies. - None

3.2.2.371.8 Subprogram Visibility. - Outside Package.

3.2.2.371.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.371.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.371.11 Side-Effects. - None

3.2.2.371.12 Algorithm. -

```
RESOLVE as_header>
```

3.2.2.371.13 Diagnostic Messages Generated. - None

3.2.2.371.14 Examples of Data Structures. - None

3.2.2.372 Subprogram. - ov_procedure

3.2.2.372.1 Purpose. - Perform overloading resolution on a node of type 'procedure'.

3.2.2.372.2 Assumptions. - None

3.2.2.372.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.372.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.372.5 Nested within. - ovp_trav

3.2.2.372.6 Host Dependencies. - None

3.2.2.372.7 Target Dependencies. - None

3.2.2.372.8 Subprogram Visibility. - Outside Package.

3.2.2.372.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.372.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.372.11 Side-Effects. - None

3.2.2.372.12 Algorithm. -

RESOLVE as_param_s>

3.2.2.372.13 Diagnostic Messages Generated. - None

3.2.2.372.14 Examples of Data Structures. - None

3.2.2.373 Subprogram. - ov_function

3.2.2.373.1 Purpose. - Perform overloading resolution on a node of type 'function'.

3.2.2.373.2 Assumptions. - None

3.2.2.373.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.373.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.373.5 Nested Within. - ovp_trav

3.2.2.373.6 Host Dependencies. - None

3.2.2.373.7 Target Dependencies. - None

3.2.2.373.8 Subprogram Visibility. - Outside Package.

3.2.2.373.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.373.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.373.11 Side-Effects. - None

3.2.2.373.12 Algorithm. -

```
RESOLVE as_param_s>
RESOLVE as_constrained_void>
```

3.2.2.373.13 Diagnostic Messages Generated. - None

3.2.2.373.14 Examples of Data Structures. - None

3.2.2.374 Subprogram. - ov_param_s

3.2.2.374.1 Purpose. - Perform overloading resolution on a node of type 'param_s'.

3.2.2.374.2 Assumptions. - None.

3.2.2.374.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.374.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.374.5 Nested within. - ovp_trav

3.2.2.374.6 Host Dependencies. - None

3.2.2.374.7 Target Dependencies. - None

3.2.2.374.8 Subprogram Visibility. - Outside Package.

3.2.2.374.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.374.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.374.11 Side-Effects. - None

3.2.2.374.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.374.13 Diagnostic Messages Generated. - None

3.2.2.374.14 Examples of Data Structures. - None

3.2.2.375 Subprogram. - ov_in_out

3.2.2.375.1 Purpose. - Perform overloading resolution on a node of type 'in_out'.

3.2.2.375.2 Assumptions. - None

3.2.2.375.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.375.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.375.5 Nested Within. - ovp_trav

3.2.2.375.6 Host Dependencies. - None

3.2.2.375.7 Target Dependencies. - None

3.2.2.375.8 Subprogram Visibility. - Outside Package.

3.2.2.375.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.375.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.375.11 Side-Effects. - None

3.2.2.375.12 Algorithm. -

```
RESOLVE as_type_spcc>
RESOLVE as_exp_void>
```

3.2.2.375.13 Diagnostic Messages Generated. - None

3.2.2.375.14 Examples of Data Structures. - None

3.2.2.376 Subprogram. - ov_out

3.2.2.376.1 Purpose. - Perform overloading resolution on a node of type 'out'.

3.2.2.376.2 Assumptions. - None

3.2.2.376.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.376.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.376.5 Nested within. - ovp_trav

3.2.2.376.6 Host Dependencies. - None

3.2.2.376.7 Target Dependencies. - None

3.2.2.376.8 Subprogram Visibility. - Outside Package.

3.2.2.376.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.376.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.376.11 Side-Effects. - None

3.2.2.376.12 Algorithm. -

RESOLVE as_type_spcc>

3.2.2.376.13 Diagnostic Messages Generated. - None

3.2.2.376.14 Examples of Data Structures. - None

3.2.2.377 Subprogram. - ov_subp_body

3.2.2.377.1 Purpose. - Perform overloading resolution on a node of type 'subprogram_body'.

3.2.2.377.2 Assumptions. - None

3.2.2.377.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.377.4 Used Recursively. - This subprogram is used recursively.

3.2.2.377.5 Nested Within. - ovp_trav

3.2.2.377.6 Host Dependencies. - None

3.2.2.377.7 Target Dependencies. - None

3.2.2.377.8 Subprogram Visibility. - Outside Package.

3.2.2.377.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.377.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.377.11 Side-Effects. - None

3.2.2.377.12 Algorithm. -

```
RESOLVE as_header>
RESOLVE as_block_stub>
```

3.2.2.377.13 Diagnostic Messages Generated. - None

3.2.2.377.14 Examples of Data Structures. - None

3.2.2.378 Subprogram. - ov_pack_dcl

3.2.2.378.1 Purpose. - Perform overloading resolution on a node of type 'package_decl'.

3.2.2.378.2 Assumptions. - None

3.2.2.378.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.378.4 Used Recursively. - This subprogram is used recursively.

3.2.2.378.5 Nested within. - ovp_trav

3.2.2.378.6 Host Dependencies. - None

3.2.2.378.7 Target Dependencies. - None

3.2.2.378.8 Subprogram Visibility. - Outside Package.

3.2.2.378.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.378.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.378.11 Side-Effects. - None

3.2.2.378.12 Algorithm. -

```
RESOLVE as_package_def>
```

3.2.2.378.13 Diagnostic Messages Generated. - None

3.2.2.378.14 Examples of Data Structures. - None

3.2.2.379 Subprogram. - ov_pack_spec

3.2.2.379.1 Purpose. - Perform overloading resolution on a node of type 'package_spec'.

3.2.2.379.2 Assumptions. - None

3.2.2.379.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.379.4 Used Recursively. - This subprogram is used recursively.

3.2.2.379.5 Nested within. - ovp_trav

3.2.2.379.6 Host Dependencies. - None

3.2.2.379.7 Target Dependencies. - None

3.2.2.379.8 Subprogram Visibility. - Outside Package.

3.2.2.379.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.379.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.379.11 Side-Effects. - None

3.2.2.379.12 Algorithm. -

```
RESOLVE as_decl_s>
RESOLVE as_decl_rep_s>
```

3.2.2.379.13 Diagnostic Messages Generated. - None

3.2.2.379.14 Examples of Data Structures. - None

3.2.2.380 Subprogram. - ov_decl_s

3.2.2.380.1 Purpose. - Perform overloading resolution on a node of type 'decl_s'.

3.2.2.380.2 Assumptions. - None

3.2.2.380.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.380.4 Used Recursively. - This subprogram is used recursively.

3.2.2.380.5 Nested Within. - ovp_trav

3.2.2.380.6 Host Dependencies. - None

3.2.2.380.7 Target Dependencies. - None

3.2.2.380.8 Subprogram Visibility. - Outside Package.

3.2.2.380.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.380.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.380.11 Side-Effects. - None

3.2.2.380.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.380.13 Diagnostic Messages Generated. - None

3.2.2.380.14 Examples of Data Structures. - None

3.2.2.381 Subprogram. - ov_dcl_rep_s

3.2.2.381.1 Purpose. - Perform overloading resolution on a node of type 'decl_rep_s'.

3.2.2.381.2 Assumptions. - None

3.2.2.381.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.381.4 Used Recursively. - This subprogram is used recursively.

3.2.2.381.5 Nested Within. - ovp_trav

3.2.2.381.6 Host Dependencies. - None

3.2.2.381.7 Target Dependencies. - None

3.2.2.381.8 Subprogram Visibility. - Outside Package.

3.2.2.381.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.381.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.381.11 Side-Effects. - None

3.2.2.381.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.381.13 Diagnostic Messages Generated. - None

3.2.2.381.14 Examples of Data Structures. - None

3.2.2.382 Subprogram. - ov_pack_body

3.2.2.382.1 Purpose. - Perform overloading resolution on a node of type 'package_body'.

3.2.2.382.2 Assumptions. - None

3.2.2.382.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.382.4 Used Recursively. - This subprogram is used recursively.

3.2.2.382.5 Nested Within. - ovp_trav

3.2.2.382.6 Host Dependencies. - None

3.2.2.382.7 Target Dependencies. - None

3.2.2.382.8 Subprogram Visibility. - Outside Package.

3.2.2.382.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.382.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.382.11 Side-Effects. - None

3.2.2.382.12 Algorithm. -

RESOLVE as_block_stub>

3.2.2.382.13 Diagnostic Messages Generated. - None

3.2.2.382.14 Examples of Data Structures. - None

3.2.2.383 Subprogram. - ov_accept

3.2.2.383.1 Purpose. - Perform overloading resolution on a node of type 'accept'.

3.2.2.383.2 Assumptions. - None

3.2.2.383.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.383.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.383.5 Nested Within. - ovp_trav

3.2.2.383.6 Host Dependencies. - None

3.2.2.383.7 Target Dependencies. - None

3.2.2.383.8 Subprogram Visibility. - Outside Package.

3.2.2.383.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.383.10 Formal Parameters. -

```
node:  in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.383.11 Side-Effects. - None

3.2.2.383.12 Algorithm. -

```
RESOLVE as_name>
RESOLVE as_param_s>
RESOLVE as_stm_s>
```

3.2.2.383.13 Diagnostic Messages Generated. - None

3.2.2.383.14 Examples of Data Structures. - None

3.2.2.384 Subprogram. - ov_select

3.2.2.384.1 Purpose. - Perform overloading resolution on a node of type 'select'.

3.2.2.384.2 Assumptions. - None

3.2.2.384.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.384.4 Used Recursively. - This subprogram is used recursively.

3.2.2.384.5 Nested Within. - ovp_trav

3.2.2.384.6 Host Dependencies. - None

3.2.2.384.7 Target Dependencies. - None

3.2.2.384.8 Subprogram Visibility. - Outside Package.

3.2.2.384.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.384.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.384.11 Side-Effects. - None

3.2.2.384.12 Algorithm. -

```
GET as_cond_clause_s>
for EACH element of as_list> loop
    RESOLVE list element>
end loop;

RESOLVE as_stm_s>
```

3.2.2.384.13 Diagnostic Messages Generated. - None

3.2.2.384.14 Examples of Data Structures. - None

3.2.2.385 Subprogram. - ov_cond_entr

3.2.2.385.1 Purpose. - Perform overloading resolution on a node of type 'cond_entry'.

3.2.2.385.2 Assumptions. - None

3.2.2.385.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.385.4 Used Recursively. - This subprogram is used recursively.

3.2.2.385.5 Nested within. - ovp_trav

3.2.2.385.6 Host Dependencies. - None

3.2.2.385.7 Target Dependencies. - None

3.2.2.385.8 Subprogram Visibility. - Outside Package.

3.2.2.385.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.385.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.385.11 Side-Effects. - None

3.2.2.385.12 Algorithm. -

```
RESOLVE as_stm_s1>
RESOLVE as_stm_s2>
```

3.2.2.385.13 Diagnostic Messages Generated. - None

3.2.2.385.14 Examples of Data Structures. - None

3.2.2.386 Subprogram. - ov_timed_ent

3.2.2.386.1 Purpose. - Perform overloading resolution on a node of type 'timed_entry'.

3.2.2.386.2 Assumptions. - None

3.2.2.386.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.386.4 Used Recursively. - This subprogram is used recursively.

3.2.2.386.5 Nested Within. - ovp_trav

3.2.2.386.6 Host Dependencies. - None

3.2.2.386.7 Target Dependencies. - None

3.2.2.386.8 Subprogram Visibility. - Outside Package.

3.2.2.386.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.386.10 Formal Parameters. -

```
node: in cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.386.11 Side-Effects. - None

3.2.2.386.12 Algorithm. -

```
RESOLVE as_stm_s1>
RESOLVE as_stm_s2>
```

3.2.2.386.13 Diagnostic Messages Generated. - None

3.2.2.386.14 Examples of Data Structures. - None

3.2.2.387 Subprogram. - ov_abort

3.2.2.387.1 Purpose. - Perform overloading resolution on a node of type 'abort'.

3.2.2.387.2 Assumptions. - None

3.2.2.387.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.387.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.387.5 Nested Within. - ovp_trav

3.2.2.387.6 Host Dependencies. - None

3.2.2.387.7 Target Dependencies. - None

3.2.2.387.8 Subprogram Visibility. - Outside Package.

3.2.2.387.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.387.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.387.11 Side-Effects. - None

3.2.2.387.12 Algorithm. -

```
RESOLVE as_name_as>
```

3.2.2.387.13 Diagnostic Messages Generated. - None

3.2.2.387.14 Examples of Data Structures. - None

3.2.2.388 Subprogram. - ov_praema_s

3.2.2.388.1 Purpose. - Perform overloading resolution on a node of type 'pragma_s'.

3.2.2.388.2 Assumptions. - None

3.2.2.388.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.388.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.388.5 Nested Within. - ovp_trav

3.2.2.388.6 Host Dependencies. - None

3.2.2.388.7 Target Dependencies. - None

3.2.2.388.8 Subprogram Visibility. - Outside Package.

3.2.2.388.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.388.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.388.11 Side-Effects. - None

3.2.2.388.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.388.13 Diagnostic Messages Generated. - None

3.2.2.388.14 Examples of Data Structures. - None

3.2.2.389 Subprogram. - ov_comp_unit

3.2.2.389.1 Purpose. - Perform overloading resolution on a node of type 'comp_unit'.

3.2.2.389.2 Assumptions. - None

3.2.2.389.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.389.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.389.5 Nested Within. - ovp_trav

3.2.2.389.6 <u>Host Dependencies</u>. - None

3.2.2.389.7 <u>Target Dependencies</u>. - None

3.2.2.389.8 <u>Subprogram Visibility</u>. - Outside Package.

3.2.2.389.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.389.10 <u>Formal Parameters</u>. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.389.11 <u>Side-Effects</u>. - None

3.2.2.389.12 <u>Algorithm</u>. -

```
RESOLVE as_pragma_s>
RESOLVE as_unit_body>
```

3.2.2.389.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.389.14 <u>Examples of Data Structures</u>. - None

3.2.2.390 <u>Subprogram</u>. - ov_subunit

3.2.2.390.1 <u>Purpose</u>. - Perform overloading resolution on a node of type 'subunit'.

3.2.2.390.2 Assumptions. - None

3.2.2.390.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.390.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.390.5 Nested Within. - ovp_trav

3.2.2.390.6 Host Dependencies. - None

3.2.2.390.7 Target Dependencies. - None

3.2.2.390.8 Subprogram Visibility. - Outside Package.

3.2.2.390.9 Function/Procedure. - This suborogram is a Procedure.

3.2.2.390.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.390.11 Side-Effects. - None

3.2.2.390.12 Algorithm. -

```
RESOLVE as_subunit_body>
```

3.2.2.390.13 Diagnostic Messages Generated. - None

3.2.2.390.14 Examples of Data Structures. - None

3.2.2.391 Subprogram. - ov_gen_asscs

3.2.2.391.1 Purpose. - Perform overloading resolution on a node of type 'generic_assoc_s'.

3.2.2.391.2 Assumptions. - None

3.2.2.391.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.391.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.391.5 Nested Within. - ovp_trav

3.2.2.391.6 Host Dependencies. - None

3.2.2.391.7 Target Dependencies. - None

3.2.2.391.8 Subprogram Visibility. - Outside Package.

3.2.2.391.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.391.10 Formal Parameters. -

node: in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.391.11 Side-Effects. - None

3.2.2.391.12 Algorithm. -

```
for EACH element of as_list> loop
    RESOLVE list element>
end loop;
```

3.2.2.391.13 Diagnostic Messages Generated. - None

3.2.2.391.14 Examples of Data Structures. - None

3.2.2.392 Subprogram. - ov_rec_rep

3.2.2.392.1 Purpose. - Perform overloading resolution on a node of type 'record_rep'.

3.2.2.392.2 Assumptions. - None

3.2.2.392.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.392.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.392.5 Nested within. - ovp_trav

3.2.2.392.6 Host Dependencies. - None

3.2.2.392.7 Target Dependencies. - None

3.2.2.392.8 Subprogram Visibility. - Outside Package.

3.2.2.392.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.392.10 Formal Parameters. -

```
node:  in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.392.11 Side-Effects. - None

3.2.2.392.12 Algorithm. -

```
RESOLVE as_exp_void>
if AS_EXP_VOID not an integer type> then
   ERROR 15030>
end if;
GET as_comp_rep_s>
for EACH element of as_list>  loop
    RESOLVE list element>
end loop;
```

3.2.2.392.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
15030	E	integer expression required

3.2.2.392.14 Examples of Data Structures. - None

3.2.2.393 Subprogram. - ov_comp_rep

3.2.2.393.1 Purpose. - Perform overloading resolution on a node of type 'comp_rep'.

3.2.2.393.2 Assumptions. - None

3.2.2.393.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.393.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.393.5 Nested Within. - ovp_trav

3.2.2.393.6 Host Dependencies. - None

3.2.2.393.7 Target Dependencies. - None

3.2.2.393.8 Subprogram Visibility. - Outside Package.

3.2.2.393.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.393.10 Formal Parameters. -

```
node: in  cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.393.11 Side-Effects. - None

3.2.2.393.12 Algorithm. -

```
RESOLVE as_exp>
if TYPE of as_exp not an integer> then
   ERROR 15030>
end if;

RESOLVE as_range>
if TYPE of as_range not an integer> then
   ERROR 15030>
end if;
```

3.2.2.393.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15030	E	integer expression required

3.2.2.393.14 Examples of Data Structures. - None

3.2.2.394 Subprogram. - ov_address

3.2.2.394.1 Purpose. - Perform overloading resolution on a node of type 'address'.

3.2.2.394.2 Assumptions. - None

3.2.2.394.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.394.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.394.5 Nested Within. - ovp_trav

3.2.2.394.6 Host Dependencies. - None

3.2.2.394.7 Target Dependencies. - None

3.2.2.394.8 Subprogram Visibility. - Outside Package.

3.2.2.394.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.394.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- node to be resolved
```

3.2.2.394.11 Side-Effects. - None

3.2.2.394.12 Algorithm. -

```
RESOLVE as_exp>
if AS_EXP not of an integer type> then
  ERROR 15030>
end if;
```

3.2.2.394.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15030	E	integer expression required

3.2.2.394.14 Examples of Data Structures. - None

3.2.2.395 Subprogram. - ov_code

3.2.2.395.1 Purpose. - Perform overloading resolution on a node of type 'code'.

3.2.2.395.2 Assumptions. - None

3.2.2.395.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.395.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.395.5 Nested Within. - ovp_trav

3.2.2.395.6 Host Dependencies. - None

3.2.2.395.7 Target Dependencies. - None

3.2.2.395.8 Subprogram Visibility. - Outside Package.

3.2.2.395.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.395.10 Formal Parameters. -

```
node :  in  cdm_type.c_node_ref; -- node to be resolved
```

3.2.2.395.11 Side-Effects. - None

3.2.2.395.12 Algorithm. -

```
GET as_name> -- should be a type name
overload (AS_EXP,as_name>);
```

3.2.2.395.13 Diagnostic Messages Generated. - None

3.2.2.395.14 Examples of Data Structures. - None

3.2.2.396 Subprogram. - ov_task_body

3.2.2.396.1 Purpose. - Perform overloading resolution on a node of type 'task_body'.

3.2.2.396.2 Assumptions. - None

3.2.2.396.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.396.4 Used Recursively. - This subprogram is used recursively.

3.2.2.396.5 Nested Within. - ovp_trav

3.2.2.396.6 Host Dependencies. - None

3.2.2.396.7 Target Dependencies. - None

3.2.2.396.8 Subprogram Visibility. - Outside Package.

3.2.2.396.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.396.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref; -- node to be resolved.
```

3.2.2.396.11 Side-Effects. - None

3.2.2.396.12 Algorithm. -

```
RESOLVE as_block_stub>
```

3.2.2.396.13 Diagnostic Messages Generated. - None

3.2.2.396.14 Examples of Data Structures. - None

3.2.2.397 Subprogram. - ov_type

3.2.2.397.1 Purpose. - Perform overloading resolution on a node of type 'type'.

3.2.2.397.2 Assumptions. - None

3.2.2.397.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.397.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.397.5 Nested Within. - ovp_trav

3.2.2.397.6 Host Dependencies. - None

3.2.2.397.7 Target Dependencies. - None

3.2.2.397.8 Subprogram Visibility. - Outside Package.

3.2.2.397.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.397.10 Formal Parameters. -

node : in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.397.11 Side-Effects. - None

3.2.2.397.12 Algorithm. -

```
RESOLVE as_type_spec>
```

3.2.2.397.13 Diagnostic Messages Generated. - None

3.2.2.397.14 Examples of Data Structures. - None

3.2.2.398 Subprogram. - ov_task_decl

3.2.2.398.1 Purpose. - Perform overloading resolution on a node of type 'task_decl'.

3.2.2.398.2 Assumptions. - None

3.2.2.398.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.398.4 Used Recursively. - This subprogram is used recursively.

3.2.2.398.5 Nested Within. - ovp_trav

3.2.2.398.6 Host Dependencies. - None

3.2.2.398.7 Target Dependencies. - None

3.2.2.398.8 Subprogram Visibility. - Outside Package.

3.2.2.398.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.398.10 Formal Parameters. -

node : in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.398.11 Side-Effects. - None

3.2.2.398.12 Algorithm. -

```
RESOLVE as_task_def>
```

3.2.2.398.13 Diagnostic Messages Generated. - None

3.2.2.398.14 Examples of Data Structures. - None

3.2.2.399 Subprogram. - ov_subtype

3.2.2.399.1 Purpose. - Perform overloading resolution on a node of type 'subtype'

3.2.2.399.2 Assumptions. - None

3.2.2.399.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.399.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.399.5 Nested Within. - ovp_trav

3.2.2.399.6 Host Dependencies. - None

3.2.2.399.7 Target Dependencies. - None

3.2.2.399.8 Subprogram Visibility. - Outside Package.

3.2.2.399.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.399.10 Formal Parameters. -

node : in cdm_type.c_node_ref; -- node to be resolved.

3.2.2.399.11 Side-Effects. - None

3.2.2.399.12 Algorithm. -

RESOLVE as_constrained>

3.2.2.399.13 Diagnostic Messages Generated. - None

3.2.2.399.14 Examples of Data Structures. - None

3.2.2.400 Subprogram. - ov_labeled

3.2.2.400.1 Purpose. - Perform overloading resolution on a node of type 'labeled'

3.2.2.400.2 Assumptions. - None

3.2.2.400.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.400.4 Used Recursively. - This subprogram is used recursively.

3.2.2.400.5 Nested Within. - ovp_trav

3.2.2.400.6 Host Dependencies. - None

3.2.2.400.7 Target Dependencies. - None

3.2.2.400.8 Subprogram Visibility. - Outside Package.

3.2.2.400.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.400.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref;  -- node to be resolved.
```

3.2.2.400.11 Side-Effects. - None

3.2.2.400.12 Algorithm. -

```
RESOLVE as_stm>
```

3.2.2.400.13 Diagnostic Messages Generated. - None

3.2.2.400.14 Examples of Data Structures. - None

3.2.2.401 Subprogram. - ov_named_stm

3.2.2.401.1 Purpose. - Perform overloading resolution on a node of type 'named_stm'.

3.2.2.401.2 Assumptions. - None

3.2.2.401.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.401.4 Used Recursively. - This subprogram is used recursively.

3.2.2.401.5 Nested Within. - ovp_trav

3.2.2.401.6 Host Dependencies. - None

3.2.2.401.7 Target Dependencies. - None

3.2.2.401.8 Subprogram Visibility. - Outside Package.

3.2.2.401.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.401.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref;  -- node to be resolved.
```

3.2.2.401.11 Side-Effects. - None

3.2.2.401.12 Algorithm. -

RESOLVE as_stm.>

3.2.2.401.13 Diagnostic Messages Generated. - None

3.2.2.401.14 Examples of Data Structures. - None

3.2.2.402 Subprogram. - ov_raise

3.2.2.402.1 Purpose. - Perform overloading resolution on a node of type 'raise'

3.2.2.402.2 Assumptions. - None

3.2.2.402.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.402.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.402.5 Nested Within. - ovp_trav

3.2.2.402.6 Host Dependencies. - None

3.2.2.402.7 Target Dependencies. - None

3.2.2.402.8 Subprogram Visibility. - Outside Package.

3.2.2.402.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.402.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref;  -- node to be resolved.
```

3.2.2.402.11 Side-Effects. - None

3.2.2.402.12 Algorithm. -

```
RESOLVE as_name_void>
```

3.2.2.402.13 Diagnostic Messages Generated. - None

3.2.2.402.14 Examples of Data Structures. - None

3.2.2.403 Subprogram. - ov_task_spec

3.2.2.403.1 Purpose. - Perform overloading resolution on a node of type 'task_spec'.

3.2.2.403.2 Assumptions. - None

3.2.2.403.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.403.4 Used Recursively. - This subprogram is used recursively.

3.2.2.403.5 Nested within. - ovp_trav

3.2.2.403.6 Host Dependencies. - None

3.2.2.403.7 Target Dependencies. - None

3.2.2.403.8 Subprogram Visibility. - Outside Package.

3.2.2.403.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.403.10 Formal Parameters. -

```
node : in cdm_type.c_node_ref;  -- node to be resolved.
```

3.2.2.403.11 Side-Effects. - None

3.2.2.403.12 Algorithm. -

```
RESOLVE decl_rep_s.>
```

3.2.2.403.13 Diagnostic Messages Generated. - None

3.2.2.403.14 Examples of Data Structures. - None

3.2.2.404 Package. - ovp_util

3.2.2.404.1 Purpose. - Provide utility support for overloading resolution routines.

3.2.2.404.2 Number of Subprograms. - 2

3.2.2.404.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.404.4 Additional Dependencies for Body. - None

3.2.2.404.5 Package Specification. -

```
package ovp_util is
-- see overview in package ovp_util
procedure OVERLOAD...
procedure OVERLOADRNG...
end  ovp_util;
```

3.2.2.404.6 Elaboration Code. - None

3.2.2.404.7 Examples of Data Structures. - None

3.2.2.405 Subprogram. - overload

3.2.2.405.1 Purpose. - Peform overloading resolution on an expression is not nested in another expression.

3.2.2.405.2 Assumptions. - None

3.2.2.405.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.405.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.405.5 Nested Within. - ovp_util

3.2.2.405.6 Host Dependencies. - None

3.2.2.405.7 Target Dependencies. - None

3.2.2.405.8 Subprogram Visibility. - Outside Package.

3.2.2.405.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.405.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref ; -- expression node to be resolved.
result_req : in  cdm_type.c_node_ref ; -- type of result required
                                            in which this node appears.
```

3.2.2.405.11 Side-Effects. - None

3.2.2.405.12 Algorithm. -

```
if MAINTENANCE option c> then
   PRINT "overload of", node, "with result_req = ", result_req>
end if;
if NODE = 'void'> then
   return;
end if;
resolved := true;
cl_exp (node, 1, result_req, param_avail, resolved);
if MAINTENANCE option c> then
   PRINT "after pass 1, resolved = ", resolved,
                            "param_avail = ", param_avail>
```

```
end if;
if resolved = true then
   return;
end if;
if !param_avail! = 1 then
   resolved := true;
   cl_exp (node, 2, result_req, param_avail, resolved);
   if MAINTENANCE option c> then
      PRINT "after pass 2, resolved = ", resolved,
                              "param_avail = ", param_avail>
   end if;
   if resolved = true then
      return;
   end if;
end if;
resolved := true;
cl_exp (node, 3, result_req, param_avail, resolved);
if MAINTENANCE option c> then
   PRINT "after pass 3, resolved = ", resolved,
                           "param_avail = ", param_avail>
end if;
if resolved = true then
   return;
end if;
if !param_avail! = 1 then
   resolved := true;
   cl_exp (node, 4, result_req, param_avail, resolved);
   if MAINTENANCE option c> then
      PRINT "after pass 4, resolved = ", resolved,
                              "param_avail = ", param_avail>
   end if;
else
   ERROR 15033>
end if;
```

3.2.2.405.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15033	E	overloaded expression cannot be resolved

3.2.2.405.14 Examples of Data Structures. - None

3.2.2.406 Subprogram. - overloadrng

3.2.2.406.1 Purpose. - Perform overloading resolution on a range which is not nested in an expression.

3.2.2.406.2 Assumptions. - None

3.2.2.406.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.406.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.406.5 Nested Within. - ovp_util

3.2.2.406.6 Host Dependencies. - None

3.2.2.406.7 Target Dependencies. - None

3.2.2.406.8 Subprogram Visibility. - Outside Package.

3.2.2.406.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.406.10 Formal Parameters. -

```
node : in  cdm_type.c_node_ref ; -- range node to be resolved
result_req : in  cdm_type.c_node_ref ; -- type of result required in
                                          context in which this node appears.
```

3.2.2.406.11 Side-Effects. - None

3.2.2.406.12 Algorithm. -

```
if MAINTENANCE option c> then
   PRINT "overloadrng of", node, "with result_req", result_req>
end if;
resolved := true;
ov_range ( node, 1, result_req, param_avail, resolved);
if MAINTENANCE option c> then
   PRINT "after pass 1. resolved = ", resolved
                          "param_avail = ", param_avail>
end if;
if resolved = true then
   return;
end if;
if !param_avail! = 1 then
   resolved := true;
   ov_range ( node, 2, result_req, parma_avail, resolved);
   if MAINTENANCE option c> then
      PRINT "after pass 2. resolved = ", resolved,
                              "param_avail = ", param_avail>
   end if;
   if resolved = true then
      return;
   end if;
end if;
resolved := true;
ov_range ( node, 3, result_req, parma_avail, resolved);
if MAINTENANCE option c> then
   PRINT "after pass 3. resolved = ", resolved,
                          "param_avail = ", param_avail>
end if;
if resolved = true then
   return;
end if;
if !param_avail! = 1 then
   resolved := true;
   ov_range ( node, 4, result_req, param_avail, resolved);
   if MAINTENANCE option c>  then
      PRINT "after pass 4. resolved = ", resolved,
                               "param_avail = ", param_avail>
   end if;
else
   ERROR 15034>
end if;
```

3.2.2.406.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
15034	E	range expressions cannot be resolved

3.2.2.406.14 Examples of Data Structures. - None

3.2.2.407 Package. - ovp_clas

3.2.2.407.1 Purpose. - Implement the overloading resolution function for Diana classes.

3.2.2.407.2 Number of Subprograms. - 27

3.2.2.407.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.407.4 Additional Dependencies for Body. - ovp_ctxt, ovp_trav, ovp_expr

3.2.2.407.5 Package Specification. -

```
package ovp_clas is
   -- See overview in package ovp_trav
     procedure cl_acc_const;
     procedure cl_blit_stub;
     procedure cl_choice
     procedure cl_cnst_void;
     procedure cl_constrnt;
     procedure cl_decl;
     procedure cl_decl_rep;
     procedure cl_dscrt_rng;
     procedure cl_dscrt_r_v;
     procedure cl_exp;
     procedure cl_exp_void;
     procedure cl_gen_parm;
     procedure cl_header;
     procedure cl_item;
```

```
      procedure cl_iteration;
      procedure cl_name;
      procedure cl_name_void;
      procedure cl_pack_def;
      procedure cl_param;
      procedure cl_rng_void;
      procedure cl_rep;
      procedure cl_stm;
      procedure cl_sbunt_body;
      procedure cl_type_rnf;
      procedure cl_type_spec;
      procedure cl_unit_body;
      procedure cl_error;
end ovp_clas;
```

3.2.2.407.6 Elaboration Code. - None

3.2.2.407.7 Examples of Data Structures. - None

3.2.2.408 Subprogram. - cl_acc_const

3.2.2.408.1 Purpose. - Invoke an overloading resolution procedure for a node in the class access_constraint.

3.2.2.408.2 Assumptions. - subtree (the first parameter) is in the class access_constraint.

3.2.2.408.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.408.4 Used Recursively. - This subprogram is used recursively.

3.2.2.408.5 Nested Within. - ovp_clas

3.2.2.408.6 Host Dependencies. - None

3.2.2.408.7 Target Dependencies. - None

3.2.2.408.8 Subprogram Visibility. - Outside Package.

3.2.2.408.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.408.10 Formal Parameters. -

```
subtree :  in  com_type.c_node_ref;  -- reference to diana node
```

3.2.2.408.11 Side-Effects. - None

3.2.2.408.12 Algorithm. -

```
case NODE type of subtree> is
      when VOID> =>
            return
      when DSCRT_RANGE_S> =>
            INVOKE ov_dscrtrngs>
      when others =>
            INVOKE cl_exp>
end case;
```

3.2.2.408.13 Diagnostic Messages Generated. - None

3.2.2.408.14 Examples of Data Structures. - None

3.2.2.409 Subprogram. - cl_blk_stub

3.2.2.409.1 Purpose. - Invoke an overloading resolution procedure for a node in the class block_stub.

3.2.2.409.2 Assumptions. - subtree (the first parameter) is in the class block_stub.

3.2.2.409.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.409.4 Used Recursively. - This subprogram is used recursively.

3.2.2.409.5 Nested Within. - ovp_clas

3.2.2.409.6 Host Dependencies. - None

3.2.2.409.7 Target Dependencies. - None

3.2.2.409.8 Subprogram Visibility. - Outside Package.

3.2.2.409.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.409.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- references to diana node
```

3.2.2.409.11 Side-Effects. - None

3.2.2.409.12 Algorithm. -

```
case NODE type of suotree> is
     when BLOCK>
          INVOKE ov_block>
     when otners =>
          return
end case;
```

3.2.2.409.13 Diagnostic Messages Generated. - None

3.2.2.409.14 Examples of Data Structures. - None

3.2.2.410 Subprogram. - cl_choice

3.2.2.410.1 Purpose. - Invoke an overloading resolution procedure for a node in the class choice.

3.2.2.410.2 Assumptions. - None

3.2.2.410.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.410.4 Used Recursively. - This suborogram is used recursively.

3.2.2.410.5 Nested Within. - ovp_clas

3.2.2.410.6 Host Dependencies. - None

3.2.2.410.7 Target Dependencies. - None

3.2.2.410.8 Subprogram Visibility. - Outside Package.

3.2.2.410.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.410.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- diana node to be resolved
result_req :  in  cdm_type.c_node_ref;  -- type required for choice
```

3.2.2.410.11 Side-Effects. - None

3.2.2.410.12 Algorithm. -

```
case NODE type of subtree> is
      when OTHERS> =>
           return;
      when CONSTRAINED> =>
           INVOKE ov_constrnd>
      when RANGE> =>
           overloadrng (subtree, result_req);
      when others =>
           overload (subtree, result_req);
end case;
```

3.2.2.410.13 Diagnostic Messages Generated. - None

3.2.2.410.14 Examples of Data Structures. - None

3.2.2.411 Subprogram. - cl_cnst_void

3.2.2.411.1 Purpose. - Invoke an overloading resolution procedure for a node in the class constrained_void.

3.2.2.411.2 Assumptions. - subtree (the first parameter) is in the class constrained_void.

3.2.2.411.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.411.4 Used Recursively. - This subprogram is used recursively.

3.2.2.411.5 Nested Within. - ovp_clas

3.2.2.411.6 Host Dependencies. - None

3.2.2.411.7 Target Dependencies. - None

3.2.2.411.8 Subprogram Visibility. - Outside Package.

3.2.2.411.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.411.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- reference to diana node
```

3.2.2.411.11 Side-Effects. - None

3.2.2.411.12 <u>Algorithm</u>. -

```
case NODE type of subtree> is
     when CONSTRAINED> =>
          INVOKE ov_constrnd>
     when VOID> =>
          return;
     when others =>
          INVOKE cl_error>
end case;
```

3.2.2.411.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.411.14 <u>Examples of Data Structures</u>. - None

3.2.2.412 <u>Subprogram</u>. - cl_constrnt

3.2.2.412.1 <u>Purpose</u>. - Invoke an overloading resolution procedure for a node in the class constraint.

3.2.2.412.2 <u>Assumptions</u>. - subtree (the first parameter) is in the class constraint.

3.2.2.412.3 <u>Implementation Language</u>. - This subprogram is written in Ada.

3.2.2.412.4 <u>Used Recursively</u>. - This subprogram is used recursively.

3.2.2.412.5 <u>Nested Within</u>. - ovp_clas

3.2.2.412.6 <u>Host Dependencies</u>. - None

3.2.2.412.7 Target Dependencies. - None

3.2.2.412.8 Subprogram Visibility. - Outside Package.

3.2.2.412.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.412.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- reference to diana node
```

3.2.2.412.11 Side-Effects. - None

3.2.2.412.12 Algorithm. -

```
case NODE type of subtree> is
      when RANGE> =>
            INVOKE ov_range>
      when DSCRT_AGG> =>
            INVOKE ov_dscr_agg>
      when DSCRT_RANGE_S> =>
            INVOKE ov_dscrt_rngs>
      when FIXED> =>
            INVOKE ov_fixedcon>
      when FLOAT>
            INVOKE ov_floatcon>
      when VOID>
            return;
      when others =>
            INVOKE cl_error>
end case;
```

3.2.2.412.13 Diagnostic Messages Generated. - None

3.2.2.412.14 Examples of Data Structures. - None

3.2.2.413 Subprogram. - cl_decl

3.2.2.413.1 Purpose. - Invoke an overloading resolution procedure for a node in the class decl.

3.2.2.413.2 Assumptions. - subtree (first parameter) is in the class decl.

3.2.2.413.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.413.4 Used Recursively. - This subprogram is used recursively.

3.2.2.413.5 Nested Within. - ovp_clas

3.2.2.413.6 Host Dependencies. - None

3.2.2.413.7 Target Dependencies. - None

3.2.2.413.8 Subprogram Visibility. - Within Package Only.

3.2.2.413.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.413.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- reference to diana node
```

3.2.2.413.11 Side-Effects. - None

3.2.2.413.12 Algorithm. -

```
case NODE type of subtree> is
      when CONSTANT> =>
            INVOKE ov_constant>
      when VAR> =>
            INVOKE ov_var>
      when NUMBER> =>
            INVOKE ov_number>
      when TYPE> =>
            INVOKE ov_type>
      when SUBTYPE> =>
            INVOKE ov_subtype>
      when SUBPROGAM_DECL> =>
            INVOKE ov_subp_dcl>
      when PACKAGE_DECL> =>
            INVOKE ov_pack_dcl>
      when TASK_DECL> =>
            INVOKE ov_task_dcl>
      when USE> =>
            return;
      when EXCEPTION> =>
            return;
      when PRAGMA> =>
            INVOKE ov_pragma>
      when others =>
            INVOKE cl_error>
end case;
```

3.2.2.413.13 Diagnostic Messages Generated. - None

3.2.2.413.14 Examples of Data Structures. - None

3.2.2.414 Subprogram. - cl_decl_rep

3.2.2.414.1 Purpose. - Invoke an overloading resolution for a node in the class decl_rep.

3.2.2.414.2 Assumptions. - subtree (the first parameter) is in the class decl_rep .

3.2.2.414.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.414.4 Used Recursively. - This subprogram is used recursively.

3.2.2.414.5 Nested within. - ovp_clas

3.2.2.414.6 Host Dependencies. - None

3.2.2.414.7 Target Dependencies. - None

3.2.2.414.8 Subprogram Visibility. - Outside Package.

3.2.2.414.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.414.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- reference to diana node
```

3.2.2.414.11 Side-Effects. - None

3.2.2.414.12 Algorithm. -

```
case NODE type of subtree> is
     when ADDRESS> =>
          INVOKE ov_address>
     when RECORD_REP> =>
          INVOKE ov_rec_rep>
     when SIMPLE_REP> =>
          INVOKE ov_simp_rep>
     when others =>
```

```
          INVOKE cl_decl>
end case;
```

3.2.2.414.13 Diagnostic Messages Generated. - None

3.2.2.414.14 Examples of Data Structures. - None

3.2.2.415 Subprogram. - cl_dscrt_rng

3.2.2.415.1 Purpose. - Invoke correct overloading resolution routine for a node of class dscrt_range.

3.2.2.415.2 Assumptions. - None

3.2.2.415.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.415.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.415.5 Nested Within. - ovp_clas

3.2.2.415.6 Host Dependencies. - None

3.2.2.415.7 Target Dependencies. - None

3.2.2.415.8 Subprogram Visibility. - Outside Package.

3.2.2.415.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.415.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- node to be resolved
pass : in  integer range 1..4; -- pass of overloading algorithm
result_req : in  cdm_type.c_node_ref; -- type of result required in context
                                         in which this node appears
param_avail : out  cdm_type.c_node_ref; -- seq of types which may result
                                            after first pass
resolved : out  boolean; -- set to false if the expression can't be resolved
                            on the first two passes
```

3.2.2.415.11 Side-Effects. - None

3.2.2.415.12 Algorithm. -

```
case NODE type of subtree> is
     when INDEX> =>
          return;
     when CONSTRAINED> =>
          ov_constrnd (subtree)
     when RANGE> =>
          overloadrng (subtree, "any type");
          if RANGE expressions are universal integer> then
               SET to predefined integer>
          end if;
     when others =>
          INVOKE cl_error>
end case;
```

3.2.2.415.13 Diagnostic Messages Generated. - None

3.2.2.415.14 Examples of Data Structures. - None

3.2.2.416 Subprogram. - cl_dscrt_r_v

3.2.2.416.1 Purpose. - Invoke an overloading procedure for a node in the class DSCRT_RANGE_VOID.

3.2.2.416.2 Assumptions. - SUBTREE (the first parameter) is in the class DSCRT_RANGE_VOID.

3.2.2.416.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.416.4 Used Recursively. - This subprogram is used recursively.

3.2.2.416.5 Nested Within. - ovp_clas

3.2.2.416.6 Host Dependencies. - None

3.2.2.416.7 Target Dependencies. - None

3.2.2.416.8 Subprogram Visibility. - Outside Package.

3.2.2.416.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.416.10 Formal Parameters. -

```
subtree: in cdm_type.c_node_ref; -- reference to diana node.
pass : in  integer 1..4; -- pass of overloading algorithm
result_req : in  cdm_type.c_node_ref; -- type of result required in context
                                         in which this node appears
param_avail : out  cdm_type.c_node_ref; -- seq of types which may result
                                           after first pass
resolved : out  booleand; -- set to false if the expression can't be resolved
                             on the first two passes
```

3.2.2.416.11 Side-Effects. - None

3.2.2.416.12 Algorithm. -

```
case NODE type of subtree> is
  when VOID> =>
    return;
  when others =>
    INVOKE cl_dscrt_rng>
end case;
```

3.2.2.416.13 Diagnostic Messages Generated. - None

3.2.2.416.14 Examples of Data Structures. - None

3.2.2.417 Subprogram. - cl_exp

3.2.2.417.1 Purpose. - Invoke an overloading resolution procedure for a node in the class EXP.

3.2.2.417.2 Assumptions. - SUBTREE (the first parameter) is in the class EXP.

3.2.2.417.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.417.4 Used Recursively. - This subprogram is used recursively.

3.2.2.417.5 Nested Within. - ovp_clas

3.2.2.417.6 Host Dependencies. - None

3.2.2.417.7 Target Dependencies. - None

3.2.2.417.8 Subprogram Visibility. - Outside Package.

3.2.2.417.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.417.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.417.11 Side-Effects. - None

3.2.2.417.12 Algorithm. -

```
case NODE type of subtree> is
  when AGGREGATE> =>
     INVOKE ov_aggregate>
  when ALLOCATOR> =>
     INVOKE ov_allocator>
  when BINARY> =>
     INVOKE ov_binary>
  when MEMBERSHIP> =>
     INVOKE ov_membership>
  when NULL_ACCESS> =>
     INVOKE ov_null_acc>
  when PARANTHESIZED> =>
     INVOKE ov_parenthzd>
  when QUALIFIED> =>
     INVOKE ov_qualified>
  when NUMERIC_LITERAL> =>
     INVOKE ov_num_lit>
  when STRING_LITERAL> =>
     INVOKE ov_str_lit>
  when USED_CHAR> =>
     INVOKE ov_used_char>
  when OTHERS> =>
     INVOKE cl_name>
end case;
```

3.2.2.417.13 Diagnostic Messages Generated. - None

3.2.2.417.14 Examples of Data Structures. - None

3.2.2.418 Subprogram. - cl_exp_void

3.2.2.418.1 Purpose. - Invoke an overloading resolution procedure for a node in the class exp_void.

3.2.2.418.2 Assumptions. - SUBTREE (the first parameter) is in the class EXP_VOID.

3.2.2.418.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.418.4 Used Recursively. - This subprogram is used recursively.

3.2.2.418.5 Nested Within. - ovp_clas

3.2.2.418.6 Host Dependencies. - None

3.2.2.418.7 Target Dependencies. - None

3.2.2.418.8 Subprogram Visibility. - Outside Package.

3.2.2.418.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.418.10 Formal Parameters. -

suotree: in cdm_type.c_node_ref; reference to diana node.

3.2.2.418.11 Side-Effects. - None

3.2.2.418.12 Algorithm. -

```
if NODE type of suotree> = VOID> then
   return
else
   INVOKE cl_exn>
end if;
```

3.2.2.418.13 Diagnostic Messages Generated. - None

3.2.2.418.14 Examples of Data Structures. - None

3.2.2.419 Subprogram. - cl_gen_parm

3.2.2.419.1 Purpose. - Invoke an overloading resolution procedure for a node in the class GENERIC_PARAM.

3.2.2.419.2 Assumptions. - SUBTREE (the first parameter) is in the class GENERIC_PARAM.

3.2.2.419.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.419.4 Used Recursively. - This suoprogram is used recursively.

3.2.2.419.5 Nested within. - ovo_clas

3.2.2.419.6 Host Dependencies. - None

3.2.2.419.7 Target Dependencies. - None

3.2.2.419.8 Subprogram Visibility. - Outside Package.

3.2.2.419.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.419.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.419.11 Side-Effects. - None

3.2.2.419.12 Algorithm. - .

```
case NODE type of Subtree> is
  when IN> =>
     INVOKE ov_in>
  when IN_OUT> =>
     INVOKE ov_out>
  when SUBPROGRAM_DECL> =>
     INVOKE ov_subp_dcl>
  when TYPE> =>
     INVOKE ov_type>
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.419.13 Diagnostic Messages Generated. - None

3.2.2.419.14 Examples of Data Structures. - None

3.2.2.420 Subprogram. - cl_header

3.2.2.420.1 Purpose. - Invoke an overloading resolution procedure for a node in the class HEADER.

3.2.2.420.2 Assumptions. - SUBTREE (the first parameter) is in the class HEADER.

3.2.2.420.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.420.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.420.5 Nested within. - ovp_clas

3.2.2.420.6 Host Dependencies. - None

3.2.2.420.7 Target Dependencies. - None

3.2.2.420.8 Subprogram Visibility. - Outside Package.

3.2.2.420.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.420.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.420.11 Side-Effects. - None

3.2.2.420.12 Algorithm. -

```
case NODE type of subtree> is
  when FUNCTION> =>
     INVOKE ov_function>
  when PROCEDURE> =>
     INVOKE ov_procedure>
  when ENTRY> =>
     INVOKE ov_entry>
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.420.13 Diagnostic Messages Generated. - None

3.2.2.420.14 Examples of Data Structures. - None

3.2.2.421 Subprogram. - cl_item

3.2.2.421.1 Purpose. - Invoke an overloading resolution procedure for a node in the class ITEM.

3.2.2.421.2 Assumptions. - SUBTREE (the first parameter) is in the class ITEM.

3.2.2.421.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.421.4 Used Recursively. - This subprogram is used recursively.

3.2.2.421.5 Nested Within. - ovp_clas

3.2.2.421.6 Host Dependencies. - None

3.2.2.421.7 Target Dependencies. - None

3.2.2.421.8 Subprogram Visibility. - Outside Package.

3.2.2.421.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.421.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.421.11 Side-Effects. - None

3.2.2.421.12 Algorithm. -

```
case NODE type of subtree> is
  when PACKAGE_BODY> =>
     INVOKE ov_pack_body>
  when SUBPROGRAM_BODY> =>
     INVOKE ov_subp_body>
  when TASK_BODY> =>
     INVOKE ov_task_body>
  when USE> =>
     return
  when ADDRESS> =>
     INVOKE ov_address>
  when RECORD_REP> =>
     INVOKE ov_rec_rep>
  when SIMPLE_REP> =>
     INVOKE ov_simp_rep>
  when others =>
     INVOKE cl_decl>
end case;
```

3.2.2.421.13 Diagnostic Messages Generated. - None

3.2.2.421.14 Examples of Data Structures. - None

3.2.2.422 Subprogram. - cl_iteration

3.2.2.422.1 Purpose. - Invoke an overloading resolution procedure for a node in the class ITERATION.

3.2.2.422.2 Assumptions. - SUBTREE (the first parameter) is in the class ITERATION.

3.2.2.422.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.422.4 Used Recursively. - This subprogram is used recursively.

3.2.2.422.5 Nested Within. - ovp_clas

3.2.2.422.6 Host Dependencies. - None

3.2.2.422.7 Target Dependencies. - None

3.2.2.422.8 Subprogram Visibility. - Outside Package.

3.2.2.422.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.422.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.422.11 Side-Effects. - None

3.2.2.422.12 Algorithm. -

```
case NODE type of subtree> is
  when FOR | reverse> =>
     INVOKE ov_for_rev>
  when WHILE> =>
     INVOKE ov_while>
  when VOID> =>
     return;
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.422.13 Diagnostic Messages Generated. - None

3.2.2.422.14 Examples of Data Structures. - None

3.2.2.423 Subprogram. - cl_name

3.2.2.423.1 Purpose. - Invoke an overloading resolution procedure for a node in the class NAME.

3.2.2.423.2 Assumptions. - SUBTREE (in first parameter) is in the class NAME.

3.2.2.423.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.423.4 Used Recursively. - This subprogram is used recursively.

3.2.2.423.5 Nested Within. - ovp_clas

3.2.2.423.6 Host Dependencies. - None

3.2.2.423.7 Target Dependencies. - None

3.2.2.423.8 Subprogram Visibility. - Outside Package.

3.2.2.423.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.423.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.423.11 Side-Effects. - None

3.2.2.423.12 Algorithm. -

```
case NODE type of subtree> is
  when ALL> =>
     INVOKE ov_all>
  when FUNCTION_CALL> =>
     INVOKE ov_func_call>
  when INDEXED> =>
    INVOKE ov_indexed>
  when SELECTED> =>
    INVOKE ov_selected>
  when SLICE> =>
    INVOKE ov_slice>
  when ATTRIBUTE> =>
    INVOKE ov_attribute>
  when ATTRIBUTE_CALL> =>
     INVOKE ov_attr_call>
  when others =>
     INVOKE cl_error>
```

end case;

3.2.2.423.13 Diagnostic Messages Generated. - None

3.2.2.423.14 Examples of Data Structures. - None

3.2.2.424 Subprogram. - cl_name_void

3.2.2.424.1 Purpose. - Invoke an overloading resolution procedure for a node in the class NAME_VOID.

3.2.2.424.2 Assumptions. - SUBTREE (the first parameter) is in the class NAME_VOID.

3.2.2.424.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.424.4 Used Recursively. - This subprogram is used recursively.

3.2.2.424.5 Nested within. - ovp_clas

3.2.2.424.6 Host Dependencies. - None

3.2.2.424.7 Target Dependencies. - None

3.2.2.424.8 Subprogram Visibility. - Outside Package.

3.2.2.424.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.424.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.424.11 Side-Effects. - None

3.2.2.424.12 Algorithm. -

```
case NODE type of subtree> is
  when VOID> =>
   return;
  when others =>
   INVOKE cl_name>
end case;
```

3.2.2.424.13 Diagnostic Messages Generated. - None

3.2.2.424.14 Examples of Data Structures. - None

3.2.2.425 Subprogram. - cl_pack_def

3.2.2.425.1 Purpose. - Invoke an overloading resolution procedure for a node in the class PACKAGE_DEF.

3.2.2.425.2 Assumptions. - SUBTREE (the first parameter) is in the class PACKAGE_DEF.

3.2.2.425.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.425.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.425.5 Nested within. - ovp_clas

3.2.2.425.6 Host Dependencies. - None

3.2.2.425.7 Target Dependencies. - None

3.2.2.425.8 Subprogram Visibility. - Outside Package.

3.2.2.425.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.425.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.425.11 Side-Effects. - None

3.2.2.425.12 Algorithm. -

```
case NODE type of subtree> is
  when PACKAGE_SPEC> =>
    INVOKE ov_pack_spec>
  when others =>
    return;
end case;
```

3.2.2.425.13 Diagnostic Messages Generated. - None

3.2.2.425.14 Examples of Data Structures. - None

3.2.2.426 Subprogram. - cl_param

3.2.2.426.1 Purpose. - Invoke an overloading resolution procedure for a node in the class PARAM.

3.2.2.426.2 Assumptions. - SUBTREE (the first parameter) is in the class PARAM.

3.2.2.426.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.426.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.426.5 Nested within. - ovp_clas

3.2.2.426.6 Host Dependencies. - None

3.2.2.426.7 Target Dependencies. - None

3.2.2.426.8 Subprogram Visibility. - Outside Package.

3.2.2.426.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.426.10 Formal Parameters. -

```
subtree : in  cdm_type.c_node_ref; -- reference to
                                      diana node.
```

3.2.2.426.11 Side-Effects. - None

3.2.2.426.12 Algorithm. -

```
case NODE type of subtree> is
  when IN> =>
     INVOKE ov_in>
  when OUT> =>
     INVOKE ov_out>
  when IN_OUT> =>
     INVOKE ov_in_out>
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.426.13 Diagnostic Messages Generated. - None

3.2.2.426.14 Examples of Data Structures. - None

3.2.2.427 Subprogram. - cl_rng_void

3.2.2.427.1 Purpose. - Invoke an overloading resolution procedure for a node in the class RANGE_VOID.

3.2.2.427.2 Assumptions. - SUBTREE (the first parameter) is in the class RANGE_VOID.

3.2.2.427.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.427.4 Used Recursively. - This subprogram is used recursively.

3.2.2.427.5 Nested within. - ovp_clas

3.2.2.427.6 Host Dependencies. - None

3.2.2.427.7 Target Dependencies. - None

3.2.2.427.8 Subprogram Visibility. - Outside Package.

3.2.2.427.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.427.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.427.11 Side-Effects. - None

3.2.2.427.12 Algorithm. -

```
case NODE type of subtree> is
  when VOID> =>
     return;
  when RANGE> =>
     INVOKE ov_range>
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.427.13 Diagnostic Messages Generated. - None

3.2.2.427.14 Examples of Data Structures. - None

3.2.2.428 Subprogram. - cl_rep

3.2.2.428.1 Purpose. - Invoke an overloading resolution procedure for a node in the class REP.

3.2.2.428.2 Assumptions. - SUBTREE (the first parameter) is in the class REP.

3.2.2.428.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.428.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.428.5 Nested Within. - ovp_clas

3.2.2.428.6 Host Dependencies. - None

3.2.2.428.7 Target Dependencies. - None

3.2.2.428.8 Subprogram Visibility. - Outside Package.

3.2.2.428.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.428.10 Formal Parameters. -

```
subtree : in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.428.11 Side-Effects. - None

3.2.2.428.12 Algorithm. -

```
case NODE type of subtree> is
  when ADDRESS> =>
     INVOKE ov_address>
  when RECORD_REP> =>
     INVOKE ov_rec_rep>
  when SIMPLE_REP> =>
     INVOKE ov_simp_rep>
  when others =>
     INVOKE cl_error>
end case;
```

3.2.2.428.13 Diagnostic Messages Generated. - None

3.2.2.428.14 Examples of Data Structures. - None

3.2.2.429 Subprogram. - cl_stm

3.2.2.429.1 Purpose. - Invoke an overloading resolution procedure for a node in the class STM.

3.2.2.429.2 Assumptions. - SUBTREE (the first parameter) is in the class STM.

3.2.2.429.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.429.4 Used Recursively. - This subprogram is used recursively.

3.2.2.429.5 Nested Within. - ovp_clas

3.2.2.429.6 Host Dependencies. - None

3.2.2.429.7 Target Dependencies. - None

3.2.2.429.8 Subprogram Visibility. - Outside Package.

3.2.2.429.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.429.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.429.11 Side-Effects. - None

3.2.2.429.12 Algorithm. -

```
case NODE type of subtree> is
  when LOOP> =>
    INVOKE ov_loop>
  when ABORT> =>
    returns;
  when ACCEPT> =>
    INVOKE ov_accept>
  when ASSIGN> =>
    INVOKE ov_assign>
  when BLOCK> =>
    INVOKE ov_block>
  when CASE> =>
    INVOKE ov_case>
  when CODE> =>
    INVOKE ov_code>
  when COND_ENTRY> =>
    INVOKE ov_cond_entr>
  when DELAY> =>
    INVOKE ov_delay>
  when ENTRY_CALL> =>
    INVOKE ov_entry_cal>
  when EXIT> =>
    INVOKE ov_exit>
  when GOTO> =>
    INVOKE ov_goto>
```

```
   when IF> =>
     INVOKE ov_if>
   when LABELED> =>
     INVOKE ov_labeled>
   when NAMED_STM> =>
     INVOKE ov_named_stm>
   when NULL_STM> =>
     return;
   when PRAGMA> =>
     INVOKE ov_pragma>
   when PROCEDURE_CALL> =>
     INVOKE ov_proc_call>
   when RAISE> =>
     INVOKE ov_raise>
   when RETURN> =>
     INVOKE ov_return>
   when SELECT> =>
     INVOKE ov_select>
   when TERMINATE> =>
     return;
   when TIMED_ENTRY> =>
     INVOKE ov_timed_ent>
   when others =>
     INVOKE cl_error>
end case;
```

3.2.2.429.13 Diagnostic Messages Generated. - None

3.2.2.429.14 Examples of Data Structures. - None

3.2.2.430 Subprogram. - cl_sbunt_body

3.2.2.430.1 Purpose. - Invoke an overloading resolution procedure for a node in the class SUBUNIT_BODY.

3.2.2.430.2 Assumptions. - SUBTREE (the first parameter) is in the class SUBUNIT_BODY.

3.2.2.430.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.430.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.430.5 Nested Within. - ovp_clas

3.2.2.430.6 Host Dependencies. - None

3.2.2.430.7 Target Dependencies. - None

3.2.2.430.8 Subprogram Visibility. - Outside Package.

3.2.2.430.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.430.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; reference to diana node.

3.2.2.430.11 Side-Effects. - None

3.2.2.430.12 Algorithm. -

```
case NODE type of subtree> is
  when SUBPROGRAM_BODY> =>
    INVOKE ov_subp_body>
  when PACKAGE_BODY> =>
    INVOKE ov_pack_body>
  when TASK_BODY> =>
    INVOKE ov_task_body>
  when others =>
    INVOKE cl_error>
end case;
```

3.2.2.430.13 Diagnostic Messages Generated. - None

3.2.2.430.14 Examples of Data Structures. - None

3.2.2.431 Subprogram. - cl_type_rng

3.2.2.431.1 Purpose. - Invoke an overloading resolution procedure for a node in the class TYPE_RANGE.

3.2.2.431.2 Assumptions. - SUBTREE (the first parameter) is in the class TYPE_RANGE.

3.2.2.431.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.431.4 Used Recursively. - This subprogram is used recursively.

3.2.2.431.5 Nested Within. - ovp_clas

3.2.2.431.6 Host Dependencies. - None

3.2.2.431.7 Target Dependencies. - None

3.2.2.431.8 Subprogram Visibility. - Outside Package.

3.2.2.431.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.431.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.431.11 Side-Effects. - None

3.2.2.431.12 Algorithm. -

```
case NODE type of subtree> is
  when RANGE> =>
    INVOKE ov_range>
  when others =>
    INVOKE ov_constrnd>
end case;
```

3.2.2.431.13 Diagnostic Messages Generated. - None

3.2.2.431.14 Examples of Data Structures. - None

3.2.2.432 Subprogram. - cl_type_spec

3.2.2.432.1 Purpose. - Invoke an overloading resolution procedure for a node in the class TYPE_SPEC.

3.2.2.432.2 Assumptions. - SUBTREE (the first parameter) is in the class TYPE_SPEC.

3.2.2.432.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.432.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.432.5 Nested Within. - ovp_clas

3.2.2.432.6 Host Dependencies. - None

3.2.2.432.7 Target Dependencies. - None

3.2.2.432.8 Subprogram Visibility. - Outside Package.

3.2.2.432.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.432.10 Formal Parameters. -

```
subtree: in  cdm_type.c_node_ref; -- reference to diana node.
```

3.2.2.432.11 Side-Effects. - None

3.2.2.432.12 Algorithm. -

```
case NODE type of subtree> is
  when CONSTRAINED> =>
    INVOKE ov_constrnd>
  when ACCESS> =>
    INVOKE ov_access>
  when ARRAY> =>
    INVOKE ov_array>
  when DERIVED> =>
    INVOKE ov_derived>
  when FIXED> =>
    INVOKE ov_fixed>
  when FLOAT> =>
    INVOKE ov_float>
  when INTEGER> =>
    INVOKE ov_integer>
  when L_PRIVATE> |PRIVATE> =>
    return;
  when RECORD> =>
    INVOKE ov_record>
  when TASK_SPED> =>
    INVOKE ov_task_spec>
  when others =>
    INVOKE cl_error>
end case;
```

3.2.2.432.13 Diagnostic Messages Generated. - None

3.2.2.432.14 Examples of Data Structures. - None

3.2.2.433 Subprogram. - cl_unit_body

3.2.2.433.1 Purpose. - Invoke an overloading resolution procedure for a node in the class UNIT_BODY.

3.2.2.433.2 Assumptions. - SUBTREE (the first parameter) is in the class UNIT_BODY.

3.2.2.433.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.433.4 Used Recursively. - This subprogram is used recursively.

3.2.2.433.5 Nested Within. - ovp_clas

3.2.2.433.6 Host Dependencies. - None

3.2.2.433.7 Target Dependencies. - None

3.2.2.433.8 Subprogram Visibility. - Outside Package.

3.2.2.433.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.433.10 Formal Parameters. -

subtree: in cdm_type.c_node_ref; -- reference to diana node.

3.2.2.433.11 Side-Effects. - None

3.2.2.433.12 Algorithm. -

```
case NODE type of subtree> is
  when PACKAGE_BODY> =>
    INVOKE ov_pack_body>
  when PACKAGE_DECL> =>
    INVOKE ov_pack_dcl>
  when SUBUNIT> =>
    INVOKE ov_subunit>
  when SUBPROGRAM_BODY> =>
    INVOKE ov_subp_body>
  when SUBPROGRAM_DECL> =>
    INVOKE ov_subp_decl>
  when others =>
    INVOKE cl_error>
end case;
```

3.2.2.433.13 Diagnostic Messages Generated. - None

3.2.2.433.14 Examples of Data Structures. - None

3.2.2.434 Subprogram. - cl_error

3.2.2.434.1 Purpose. - Report an error from an overloading class procedure.

3.2.2.434.2 Assumptions. - None

3.2.2.434.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.434.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.434.5 Nested Within. - ovp_clas

3.2.2.434.6 Host Dependencies. - None

3.2.2.434.7 Target Dependencies. - None

3.2.2.434.8 Subprogram Visibility. - Within Package Only.

3.2.2.434.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.434.10 Formal Parameters. - None

3.2.2.434.11 Side-Effects. - None

3.2.2.434.12 Algorithm. -

ERROR #15500>

3.2.2.434.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
15500	S	inconsistent node type

3.2.2.434.14 Examples of Data Structures. - None

3.2.2.435 Package. - static_exp

3.2.2.435.1 Purpose. - This package provides the capability to perform static expression evaluation on all static within a given subtree.

3.2.2.435.2 Number of Subprograms. - 23

3.2.2.435.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.435.4 Additional Dependencies for Body. - con_man

3.2.2.435.5 Package Specification. -

```
-- when used by overloading resolution, eval is called with the
-- exp in the attribute call. When used in the static expression
-- evaluation function, eval is called with the root of the
-- current container.
procedure eval (subtree: in cdm_type.c_node_ref);
```

3.2.2.435.6 Elaboration Code. - None

3.2.2.435.7 Examples of Data Structures. - None

3.2.2.436 Subprogram. - eval

3.2.2.436.1 Purpose. - This routine evaluates all the static expressions (section 4.9 of rlm) contained within the given Diana subtree.

3.2.2.436.2 Assumptions. - None

3.2.2.436.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.436.4 Used Recursively. - This subprogram is used recursively.

3.2.2.436.5 Nested Within. - static_exp

3.2.2.436.6 Host Dependencies. - None

3.2.2.436.7 Target Dependencies. - None

3.2.2.436.8 Subprogram Visibility. - Outside Package.

3.2.2.436.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.436.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;  -- points to subtree, all of whose
                                        contained static expressions are
                                        to be evaluated.
```

3.2.2.436.11 Side-Effects. - None

3.2.2.436.12 Algorithm. -

```
if SUBTREE /= void then
     case SUBTREE'NODE_TYPE> is
          when"aggregate" => aggregate (subtree)
          when "allocator" => allocator (subtree)
          when "binary" => binary (subtree)
          when "conversion" => conversion (subtree)
          when "null_access" => null_access (subtree)
          when "numeric_literal" => numeric_literal (subtree)
          when "parenthisized" => parenthisized (subtree)
          when "qualified" => qualified (subtree)
          when "string_literal" => string_literal (subtree)
          when "used_char" => used_char (subtree)
          when "used_object_id" => used_object_id (subtree)
          when "all"            => all (subtree)
          when "attribute"      => attribute (subtree)
          when "attribute_call" => attribute_call (subtree)
          when "function_call"  => function_call (subtree)
          when "indexed"        => indexed (subtree)
          when "selected"       => selected (subtree)
          when "slice"          => slice (subtree)
     end case
end if
```

3.2.2.436.13 Diagnostic Messages Generated. - None

3.2.2.436.14 Examples of Data Structures. - None

3.2.2.437 Subprogram. - aggregate

3.2.2.437.1 Purpose. - This routine evaluates the "aggregate" nodes in expressions.

3.2.2.437.2 Assumptions. - None

3.2.2.437.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.437.4 Used Recursively. - This subprogram is used recursively.

3.2.2.437.5 Nested Within. - static_exp

3.2.2.437.6 Host Dependencies. - None

3.2.2.437.7 Target Dependencies. - None

3.2.2.437.8 Subprogram Visibility. - Within Package Only.

3.2.2.437.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.437.10 Formal Parameters. -

subtree : in cdm_type.c_node_ref;

3.2.2.437.11 Side-Effects. - None

3.2.2.437.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.sm_constraint)
     number of elements := 0
     all_subscripts_static := true
     exception_raised := false
     for  ALL elements in subtree.as_list > loop
          number of elements := number of elements + 1
          if ELEMENT' node_type = named"> then
               eval (element.as_exp)
               if element.as_exp.sm_value = void then
                    all_subscripts_static := false
               elsif element.as_exp.sm_value = exception_raised then
                    exception_raised := true
               end if
          else
               eval (element)
               SET all_subscripts_static and exception raised as above.>
          end if
     end loop
     if all_subscripts_static then
          if exception_raised then
               subtree.sm_value := exception_raised.
          else
               CREATE an array sequence of "number of elements" length.>
               for ALL elements in subtree.as_list> loop
```

```
                  insert element or element.as_exp into array sequence
              end loop
              subtree.sm_value := ARRAY sequence + constraint.>
         end if
    end if
end if
```

3.2.2.437.13 Diagnostic Messages Generated. - None

3.2.2.437.14 Examples of Data Structures. - None

3.2.2.438 Subprogram. - all1

3.2.2.438.1 Purpose. - This routine evaluates the "all" nodes in expressions.

3.2.2.438.2 Assumptions. - None

3.2.2.438.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.438.4 Used Recursively. - This subprogram is used recursively.

3.2.2.438.5 Nested Within. - static_exp

3.2.2.438.6 Host Dependencies. - None

3.2.2.438.7 Target Dependencies. - None

3.2.2.438.8 Subprogram Visibility. - Within Package Only.

3.2.2.438.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.438.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref;
```

3.2.2.438.11 Side-Effects. - None

3.2.2.438.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_name)
end if
-- note we assume null.all is void.
```

3.2.2.438.13 Diagnostic Messages Generated. - None

3.2.2.438.14 Examples of Data Structures. - None

3.2.2.439 Subprogram. - allocator

3.2.2.439.1 Purpose. - This routine evaluates the "allocator" nodes in expressions.

3.2.2.439.2 Assumptions. - None

3.2.2.439.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.439.4 Used Recursively. - This subprogram is used recursively.

3.2.2.439.5 Nested within. - static_exp

3.2.2.439.6 Host Dependencies. - none

3.2.2.439.7 Target Dependencies. - None

3.2.2.439.8 Subprogram Visibility. - Within Package Only.

3.2.2.439.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.439.10 Formal Parameters. -

```
subtree : in  cdm_type.c_node_ref;
```

3.2.2.439.11 Side-Effects. - None

3.2.2.439.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_access_constraint)
end if
```

3.2.2.439.13 Diagnostic Messages Generated. - None

3.2.2.439.14 <u>Examples of Data Structures</u>. - None

3.2.2.440 <u>Subprogram</u>. - binary

3.2.2.440.1 <u>Purpose</u>. - This routine evaluates the "binary" nodes in expressions.

3.2.2.440.2 <u>Assumptions</u>. - None

3.2.2.440.3 <u>Implementation Language</u>. - This Subprogram is written in Ada.

3.2.2.440.4 <u>Used Recursively</u>. - This subprogram is used recursively.

3.2.2.440.5 <u>Nested Within</u>. - static_exp

3.2.2.440.6 <u>Host Dependencies</u>. - None

3.2.2.440.7 <u>Target Dependencies</u>. - None

3.2.2.440.8 <u>Subprogram Visibility</u>. - Within Package Only.

3.2.2.440.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.440.10 <u>Formal Parameters</u>. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.440.11 Side-Effects. - None

3.2.2.440.12 Algorithm. -

```
if subtree.sm_value = void.then
      eval (subtree.as_expl)
      eval (subtree.as_exp2)
      if SUBTREE.AS_BINARY_OP'NODE type = "and_then"> then
            if subtree.as_expl.sm_value = 0 FALSE> then
                 subtree.sm_value := subtree.as_expl.sm_value
            else
                 subtree.sm_value := subtree.as_exp2.sm_value
            end if
      else -- binary_op is "or_else".
            if subtree.as_expl.sm_value = 0 FALSE> then
                 subtree.sm_value := subtree.exp2.sm_value
            else
                 subtree.sm_value := subtree.expl.sm_value
            end if
      end if
end if
```

3.2.2.440.13 Diagnostic Messages Generated. - None

3.2.2.440.14 Examples of Data Structures. - None

3.2.2.441 Subprogram. - conversion

3.2.2.441.1 Purpose. - This routine evaluates "conversion" nodes in expressions.

3.2.2.441.2 Assumptions. - None

3.2.2.441.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.441.4 Used Recursively. - This subprogram is used recursively.

3.2.2.441.5 Nested Within. - static_exp

3.2.2.441.6 Host Dependencies. - None

3.2.2.441.7 Target Dependencies. - None

3.2.2.441.8 Subprogram Visibility. - Within Package Only.

3.2.2.441.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.441.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.441.11 Side-Effects. - None

3.2.2.441.12 Algorithm. -

```
if subtree.sm_value = void then
   eval (subtree.as_exp)
   if SUBTREE.AS_EXP /= void and then subtree.as_name.
   sm_def.sm_type_spec
   if SUBTREE.AS_EXP is a raised exception> then
   is a static type mark> then
          subtree.as_value := RAISED exception>
      elsif CHECK_CONSTRAINT called on subtree.as_exp and
          subtree.as_name.sm_defn.sm_type_spec> then
          -- constraint is satisfied
          if THE type mark is an array type with different bounds> then
             subtree.sm_value := MODIFY bounds on subtree.as_exp.sm_value>
          else
             subtree.sm_value := subtree.as_exp.sm_value
          end if
      else
          subtree.sm_value := exception raised
      end if
   end if
end if
```

3.2.2.441.13 Diagnostic Messages Generated. - None

3.2.2.441.14 Examples of Data Structures. - None

3.2.2.442 Subprogram. - function_call

3.2.2.442.1 Purpose. - This routine evaluates the "function_call" nodes in expression

3.2.2.442.2 Assumptions. - None

3.2.2.442.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.442.4 Used Recursively. - This subprogram is used recursively.

3.2.2.442.5 Nested Within. - static_exp

3.2.2.442.6 Host Dependencies. - None

3.2.2.442.7 Target Dependencies. - None

3.2.2.442.8 Subprogram Visibility. - Within Package Only.

3.2.2.442.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.442.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.442.11 Side-Effects. - None

3.2.2.442.12 Algorithm. -

```
if subtree.sm_value = void then
     INVOKE eval on all the parameters of this function call
          and count the number of parameters while doing so>
     if FUNCTION call is built in and first_parameter.sm_value /= void
            and (number of params =2 => second param /= void)> then
          if first_param.sm_value = exception raised then
               subtree.sm_value := first_param.sm_value
          elsif number of params = 1 then
               case FUNCTION call> is
               when "_" => con_man.minus(function call.sm_value,
                    first-param.sm_value)
                    -- and similarily for abs and .+
                end case
          elsif SECOND_PARAM.SM_VALUE = exception raised> then
                subtree.sm_value := second_param.sm_value
          else
                case FUNCTION call> is
                when "+" => con_man.add(first param.sm_value,
                     second_param.sm_value)
                     -- and similarily for other predefined binary operators
                end case
          end if
     end if
end if
```

3.2.2.442.13 Diagnostic Messages Generated. - None

3.2.2.442.14 Examples of Data Structures. - None

3.2.2.443 Subprogram. - indexed

3.2.2.443.1 Purpose. - This routine evaluates the "indexed" nodes in expressions.

3.2.2.443.2 Assumptions. - None

3.2.2.443.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.443.4 Used Recursively. - This subprogram is used recursively.

3.2.2.443.5 Nested Within. - static_exp

3.2.2.443.6 Host Dependencies. - None

3.2.2.443.7 Target Dependencies. - None

3.2.2.443.8 Subprogram Visibility. - Within Package Only.

3.2.2.443.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.443.10 Formal Parameters. -

```
subtree : in  cdm_type.c_node_ref;
```

3.2.2.443.11 Side-Effects. - None

3.2.2.443.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_name)
     all_subscript_defined := all_subscripts_static := true
     for ALL "exp"s in subtree.as_exp_s.as_list> loop
          eval (exp)
          if exp.sm_value = void then
               all_subscripts_static := false
          elsif EXP.SM_VALUE = exception raised> then
               all_subscript_defined := false
          end if
     end loop
     if all_subscripts_static then
          if not all_subscript_defined then
               subtree.sm_value := EXCEPTION raised>
          else
               current_value := subtree.as_name.sm_value
               for ALL "exp"s in subtree.as_exp_s.as_list>
                    current_value := CURRENT_VALUE indexed by exp.sm_value
                         or exception raised if subscript out of bounds.
               end loop
               subtree.sm_value := current_value
          end if
     end if
end if
```

3.2.2.443.13 Diagnostic Messages Generated. - None

3.2.2.443.14 Examples of Data Structures. - None

3.2.2.444 Subprogram. - membership

3.2.2.444.1 Purpose. - This routine evaluates "membership" nodes in expressions.

3.2.2.444.2 Assumptions. - None

3.2.2.444.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.444.4 Used Recursively. - This subprogram is used recursively.

3.2.2.444.5 Nested Within. - static_exp

3.2.2.444.6 Host Dependencies. - None

3.2.2.444.7 Target Dependencies. - None

3.2.2.444.8 Subprogram Visibility. - Within Package Only.

3.2.2.444.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.444.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.444.11 Side-Effects. - None

3.2.2.444.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_exp)
     if subtree.as_exp /= void and then
              SUBTREE.AS_TYPE range is static> then
          if SUBTREE.AS_EXP is a raised exception> then
              subtree.sm_value := RAISED exception>
          elsif
              SUBTREE.AS_EXP satisfies subtree.as_type_
                   range's constraint (use check_constraint)> then
              subtree.sm_value := true
          else
              subtree.sm_value := false
          end if
```

```
    end if
end if
```

3.2.2.444.13 Diagnostic Messages Generated. - None

3.2.2.444.14 Examples of Data Structures. - None

3.2.2.445 Subprogram. - null_access

3.2.2.445.1 Purpose. - This routine evaluates the "null_access" nodes in expressions.

3.2.2.445.2 Assumptions. - None

3.2.2.445.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.445.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.445.5 Nested Within. - static_exp

3.2.2.445.6 Host Dependencies. - None

3.2.2.445.7 Target Dependencies. - None

3.2.2.445.8 Subprogram Visibility. - Within Package Only.

3.2.2.445.9 Function/Procedure. - This suborogram is a Procedure.

3.2.2.445.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref
```

3.2.2.445.11 Side-Effects. - None

3.2.2.445.12 Algorithm. -

```
if subtree.sm_value = void then
     subtree.sm_value := THE value 0>
end if
```

3.2.2.445.13 Diagnostic Messages Generated. - None

3.2.2.445.14 Examples of Data Structures. - None

3.2.2.446 Subprogram. - numeric_literal

3.2.2.446.1 Purpose. - This routine evaluates the "numerical_literal" nodes in expressions.

3.2.2.446.2 Assumptions. - None

3.2.2.446.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.446.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.446.5 Nested within. - static_exp

3.2.2.446.6 Host Dependencies. - None

3.2.2.446.7 Target Dependencies. - None

3.2.2.446.8 Subprogram Visibility. - Within Package Only.

3.2.2.446.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.446.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.446.11 Side-Effects. - None

3.2.2.446.12 Algorithm. -

```
if subtree.sm_value = void then
     subtree.sm_value := con_man.conv_numeric_literal(subtree.lx_num_rep)
end if
```

3.2.2.446.13 Diagnostic Messages Generated. - None

3.2.2.446.14 Examples of Data Structures. - None

3.2.2.447 Subprogram. - other_nodes

3.2.2.447.1 Purpose. - Evaluate non_expression nodes (ie nodes that don't have "sm_value" attribute)

3.2.2.447.2 Assumptions. - None

3.2.2.447.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.447.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.447.5 Nested Within. - static_exp

3.2.2.447.6 Host Dependencies. - None

3.2.2.447.7 Target Dependencies. - None

3.2.2.447.8 Subprogram Visibility. - Within Package Only.

3.2.2.447.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.447.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.447.11 Side-Effects. - None

3.2.2.447.12 Algorithm. -

```
case GET arity of subtree> is
     nullary ==> null
     unary   ==> eval (subtree.sona)
     binary  ==> eval (subtree.son a)
```

```
                eval (subtree.son b)
      ternary ==> EVAL son a, son b, son c as above>
      abbitrary ==>
                -- then must be a sequence node
                temp := FIRST element of sequence subtree.as_list>
                loop
                     exit when temp = void
                     eval (temp)
                     temp := NEXT element of sequence (or void
                         if none)>
                end loop
end case
```

3.2.2.447.13 Diagnostic Messages Generated. - None

3.2.2.447.14 Examples of Data Structures. - None

3.2.2.448 Subprogram. - parenthesized

3.2.2.448.1 Purpose. - This routine evaluates the "parenthesized" nodes in expressions.

3.2.2.448.2 Assumptions. - None

3.2.2.448.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.448.4 Used Recursively. - This subprogram is used recursively.

3.2.2.448.5 Nested Within. - static_exp

3.2.2.448.6 Host Dependencies. - None

3.2.2.448.7 Target Dependencies. - None

3.2.2.448.8 Subprogram Visibility. - Within Package Only.

3.2.2.448.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.448.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.448.11 Side-Effects. - None

3.2.2.448.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_exp)
     subtree.sm_value := subtree.as_exp.sm_value
end if
```

3.2.2.448.13 Diagnostic Messages Generated. - None

3.2.2.448.14 Examples of Data Structures. - None

3.2.2.449 Subprogram. - qualified

3.2.2.449.1 Purpose. - Thie routine evaluates "qualified" nodes.

3.2.2.449.2 Assumptions. - None

3.2.2.449.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.449.4 Used Recursively. - This subprogram is used recursively.

3.2.2.449.5 Nested Within. - static_exp

3.2.2.449.6 Host Dependencies. - None

3.2.2.449.7 Target Dependencies. - None

3.2.2.449.8 Subprogram Visibility. - Within Package Only.

3.2.2.449.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.449.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.449.11 Side-Effects. - None

3.2.2.449.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_exp)
     constraint_check (subtree.as_exp.sm_value,
                       subtree.as_name.sm_defn.sm_type_spec,
                       valid)
     if valid then
          subtree.sm_value := subtree.as_exp.sm_value
     else
          subtree.sm_value := RAISED exception (constraint error)>
```

```
    end if
end if
```

3.2.2.449.13 Diagnostic Messages Generated. - None

3.2.2.449.14 Examples of Data Structures. - None

3.2.2.450 Subprogram. - selected

3.2.2.450.1 Purpose. - This routine evaluates the "selected" nodes in expressions.

3.2.2.450.2 Assumptions. - None

3.2.2.450.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.450.4 Used Recursively. - This subprogram is used recursively.

3.2.2.450.5 Nested Within. - static_exp

3.2.2.450.6 Host Dependencies. - none

3.2.2.450.7 Target Dependencies. - None

3.2.2.450.8 Subprogram Visibility. - Within Package Only.

3.2.2.450.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.450.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref;
```

3.2.2.450.11 Side-Effects. - None

3.2.2.450.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_name)
     if subtree.as_name.sm_value /= void then
          if SUBTREE.AS_NAME.SM_VALUE is exception raised> then
               subtree.sm_value := subtree.as_name.sm_value
          else
               -- translate field selector into on subscript
               record_mode := subtree.as_name.sm_exp_type
               subscript :=0
               loop
                    get next id in record_node def (be it a comp_id, dsrcmt_id,
                         variant_port or null_comp)
                    exit when next_id = null_comp or no more ids
                    if next_id = variant then
                         loop until subscript of coresponding disciminant is
                              found
                         use that subscript to index aggregate and hence choice
                              appropriate variant.
                         get next_id in this variant
                    else
                         subscipt := subscript := subscript +1
                         exit when next_id = subtree.as_designator.sm_defn
                    end if
               end loop
               if COMP_ID or dscrmt_id found in loop above> then
                    subtree.sm_value := index aggregate with subscript.
               else
                    subtree.sm_value := "constaint error raised"
               end if
          end if
     end if
end if
```

3.2.2.450.13 Diagnostic Messages Generated. - None

3.2.2.450.14 Examples of Data Structures. - None

3.2.2.451 Subprogram. - slice

3.2.2.451.1 Purpose. - This routine evaluates the "slice" nodes in expressions.

3.2.2.451.2 Assumptions. - None

3.2.2.451.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.451.4 Used Recursively. - This subprogram is used recursively.

3.2.2.451.5 Nested Within. - static_exp

3.2.2.451.6 Host Dependencies. - None

3.2.2.451.7 Target Dependencies. - None

3.2.2.451.8 Subprogram Visibility. - Within Package Only.

3.2.2.451.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.451.10 Formal Parameters. -

```
subtree :  in  com_type.c_node_ref;
```

3.2.2.451.11 Side-Effects. - None

3.2.2.451.12 Algorithm. -

```
if subtree.sm_value = void then
     eval (subtree.as_name)
     eval (subtree.as_dscrt_range)
end if
```

3.2.2.451.13 Diagnostic Messages Generated. - None

3.2.2.451.14 Examples of Data Structures. - None

3.2.2.452 Subprogram. - string_literal

3.2.2.452.1 Purpose. - This routine evaluates the "string_literal" nodes in expressions.

3.2.2.452.2 Assumptions. - None

3.2.2.452.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.452.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.452.5 Nested within. - static_exp

3.2.2.452.6 Host Dependencies. - None

3.2.2.452.7 Target Dependencies. - None

3.2.2.452.8 Subprogram Visibility. - Within Package Only.

3.2.2.452.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.452.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.452.11 Side-Effects. - None

3.2.2.452.12 Algorithm. -

```
if subtree.sm_value = void then
     for EACH character in subtree.lx_sym_rep> loop
          temp_value := APPEND translated character onto
               temp_value>
     end loop
     subtree.sm_value := temp_value
end if
```

3.2.2.452.13 Diagnostic Messages Generated. - None

3.2.2.452.14 Examples of Data Structures. -

none

3.2.2.453 Subprogram. - used_char

3.2.2.453.1 Purpose. - This routine evaluates the "used_char" nodes in expressions.

3.2.2.453.2 Assumptions. - None

3.2.2.453.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.453.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.453.5 Nested Within. - static_exp

3.2.2.453.6 Host Dependencies. - None

3.2.2.453.7 Target Dependencies. - None

3.2.2.453.8 Subprogram Visibility. - Within Package Only.

3.2.2.453.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.453.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref;
```

3.2.2.453.11 Side-Effects. - None

3.2.2.453.12 Algorithm. -

```
if subtree.sm_value = void then
   subtree.sm_value := CREATE a "small_int" node with positive sign
        and sm_digit = subtree.sm_defn.sm_pos>
end if
```

3.2.2.453.13 Diagnostic Messages Generated. - None

3.2.2.453.14 Examples of Data Structures. - None

3.2.2.454 Subprogram. - used_object_id

3.2.2.454.1 Purpose. - This routine evaluates the "used_object_id" nodes in expressions.

3.2.2.454.2 Assumptions. - None

3.2.2.454.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.454.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.454.5 Nested Within. - static_exp

3.2.2.454.6 Host Dependencies. - None

3.2.2.454.7 Target Dependencies. - None

3.2.2.454.8 Subprogram Visibility. - Within Package Only.

3.2.2.454.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.454.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref;
```

3.2.2.454.11 Side-Effects. - None

3.2.2.454.12 Algorithm. -

```
if subtree.sm_value = void then
     def_id := subtree.sm_defn
     if DEF_ID'NODE_TYPE = "number_id"> then
          subtree.sm_value := def_id.sm_init_exp.sm_value
     elsif DEF_ID'NODE_TYPE = "const_id"> then
          subtree.sm_value := def_id.sm_obj_def.sm_value
     elsif DEF_ID'NODE_TYPE = "enum_id"> then
          subtree.sm_value := CREATE a "small_int" node with positive sign
                and sm_digit = def_id.sm_pos>
     end if
end if
```

3.2.2.454.13 Diagnostic Messages Generated. - None

3.2.2.454.14 Examples of Data Structures. - None

3.2.2.455 Subprogram. - constraint_check

3.2.2.455.1 Purpose. - This routine checks to see if the given value satisfies any constraint given in the type_specification.

3.2.2.455.2 Assumptions. - None

3.2.2.455.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.455.4 Used Recursively. - This subprogram is used recursively.

3.2.2.455.5 Nested Within. - static_exp

3.2.2.455.6 Host Dependencies. - None

3.2.2.455.7 Target Dependencies. - None

3.2.2.455.8 Subprogram Visibility. - Within Package Only.

3.2.2.455.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.455.10 Formal Parameters. -

```
val :  in  value_type;
type_spec :  in cdm_type.c_node_ref;
valid :  out  boolean;  -- is true if the constraint is satisfied
```

3.2.2.455.11 Side-Effects. - None

3.2.2.455.12 Algorithm. -

```
case TYPE_SPEC'NODE_TYPE> is
   "integer" => valid := range_check(val, type_spec.as_range)
   "float" => if type_spec.as_range_void /= void then
                 valid := range_check (val, type_spec.as_range_void)
         else
                 valid := range_check(val,type_spec.sm_
                    type_struct.as_range_void)
```

```
        end  if
    "fixed"   => if type_spec.as_range_void /= void then
                    valid := range_check(val,type_spec.as_range_void)
        end if
    "constrained" => INVOKE range_check, array_check or
                     discriminant_check depending on whether "sm_constrained"
                     attribute is range, index or discriminant constrained.
    "derived  => SAME as above except use "as_constrained"> attribute.
    "array"   => invoke array_check
    "record"  => INVOKE discriminant_check>
    others    => valid := true
end case
```

3.2.2.455.13 Diagnostic Messages Generated. - None

3.2.2.455.14 Examples of Data Structures. - None

3.2.2.456 Subprogram. - range_check

3.2.2.456.1 Purpose. - This routine checks to see if the given value lies within the given range.

3.2.2.456.2 Assumptions. - None

3.2.2.456.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.456.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.456.5 Nested Within. - static_exp

3.2.2.456.6 Host Dependencies. - None

3.2.2.456.7 Target Dependencies. - None

3.2.2.456.8 Subprogram Visibility. - Within Package Only.

3.2.2.456.9 Function/Procedure. - This subprogram is a Function with a boolean result type.

3.2.2.456.10 Formal Parameters. -

```
val : in value_type;
range_node : in cdm_type.c_node_ref;
```

3.2.2.456.11 Side-Effects. - None

3.2.2.456.12 Algorithm. -

```
if val >- range_node.as_exp1.sm_value and then
   val - range_node.as_exp2.sm_valaue then
        return (true)
else
        return (false)
endif.
```

3.2.2.456.13 Diagnostic Messages Generated. - None

3.2.2.456.14 Examples of Data Structures. - None

3.2.2.457 Subprogram. - array_check

3.2.2.457.1 Purpose. - This routine checks to see if the given value belongs to the given array type definition.

3.2.2.457.2 Assumptions. - None

3.2.2.457.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.457.4 Used Recursively. - This subprogram is used recursively.

3.2.2.457.5 Nested Within. - static_exp

3.2.2.457.6 Host Dependencies. - None

3.2.2.457.7 Target Dependencies. - None

3.2.2.457.8 Subprogram Visibility. - Within Package Only.

3.2.2.457.9 Function/Procedure. - This subprogram is a Function with a boolean result type.

3.2.2.457.10 Formal Parameters. -

```
val : in value_type;
discrete_range_left : in cdm_type.c_node_ref;
component_spec : in cdm_type.c_node_ref;
```

3.2.2.457.11 Side-Effects. - None

3.2.2.457.12 Algorithm. -

```
if DISCRETE_RANGE_LEFT is empty> then
     return (true)
else
     discrete_range := GET next "dscrt_range" node in discrete_
          ranges_left>
     if VAL is not as multiple value> then return (false) end if
     if DISCRETE_RANGE'NODE_TYPE = range> then
          RETURN false if bounds in val don't equal discrete_
               range.as.exp1.sm_value and discrete_range.as_exp2.
               sm_value.
     elsif DISCRETE_RANGE'NODE_TYPE = constrained> then
          range := discrete_range.sm_constrained
          RETURN false if bounds in val don't equal range.as_exp1.sm_value
               and range.as_exp2.sm_value>
     end if
          GET the list of leftover discrete ranges>
          for EACH "element_value" in val> loop
               if not array_check (element_value, leftover_discrete_ranges,
                         component_spec) then
                    return (false)
               end if
          end loop
          return (true)
end if
```

3.2.2.457.13 Diagnostic Messages Generated. - None

3.2.2.457.14 Examples of Data Structures. - None

3.2.2.458 Subprogram. - discriminant_check

3.2.2.458.1 Purpose. - This routine checks to see if the given value has the given discriminant values.

3.2.2.458.2 Assumptions. - None

3.2.2.458.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.458.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.458.5 Nested Within. - static_exp

3.2.2.458.6 Host Dependencies. - None

3.2.2.458.7 Target Dependencies. - None

3.2.2.458.8 Subprogram Visibility. - Within Package Only.

3.2.2.458.9 Function/Procedure. - This subprogram is a Function with a boolean result type.

3.2.2.458.10 Formal Parameters. -

```
val : in value_type;
dscrmt_aggregate : in cdm_type.c-node_ref;
dscrmt_def : in cdm_type.c_node_ref; -- points to a ??_s.as_list list
                              sequence which contains " dscrmt_id"s
```

3.2.2.458.11 Side-Effects. - None

3.2.2.458.12 Algorithm. -

```
subscript := 0
dscrmt_defs_remaining := dscrmt_def
for EACH "element" in the sequence dscrmt_aggregate> loop
     if ELEMENT'NODE_TYPE /= "named"> then
          subscript := subscript +1
          ADVANCE dscrmt_defs_remaining list by one>
          if VAL subscripted by subscript /= element.sm_value> then
               return (false)
          end if
```

```
    else -- "element" is a "named" node.
         for EACH "name" in element.as_choice_s.as_list> loop
              subscript := ?; current_dscrmt_id := dscrmt_defs_ remaining
                   while name.sm_defn /= current_dsrcmt_id loop
                   subscript := subscript +1
                        get current_dscrmt_id := next dscrmt_id in list,
                        return (false) if none left.
              end loop
              if VAL subscripted by subscript /= element.as_exp.
                   sm_value then
                   return (false)
              end if
         end loop
    end if
end loop
```

3.2.2.458.13 Diagnostic Messages Generated. - None

3.2.2.458.14 Examples of Data Structures. - None

3.2.2.459 Package. - target_type

3.2.2.459.1 Purpose. - This package determines the parent types for derived types which are specified in one of the following forms:

(a) "TYPE X IS RANGE A..B;" (b) "TYPE X IS DIGITS N;"

3.2.2.459.2 Number of Subprograms. - 2

3.2.2.459.3 Dependencies on Other Packages for Spec. - cdm_type, con_man

3.2.2.459.4 Additional Dependencies for Body. - none

3.2.2.459.5 Package Specification. -

```
procedure integer_type (integer: in cdm_type.c_node_ref);
procedure float_type (float_node: in cdm_type.c_node_ref);

-- non-visible constants

short_float_digits : value
float_digits : value
long_float_digits : value
short_int_first : value
short_int_last : value
int_first : value
int_last : value
long_int_first : value
long_int_last : value
```

3.2.2.459.6 Elaboration Code. - INITIALIZE the non-visible constants from info in standard>

3.2.2.459.7 Examples of Data Structures. - None

3.2.2.460 Subprogram. - integer_type

3.2.2.460.1 Purpose. - This routine takes a node of type "integer" (representing an integer type definition) and fills in its TYPE_SPEC fields (sm_type_struct in the "integer" node and the sm_base_type attribute in the "range" node) to point to the appropriate integer definition.

3.2.2.460.2 Assumptions. - None

3.2.2.460.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.460.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.460.5 Nested Within. - target_type

3.2.2.460.6 Host Dependencies. - None

3.2.2.460.7 Target Dependencies. - None

3.2.2.460.8 Subprogram Visibility. - Outside Package.

3.2.2.460.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.460.10 Formal Parameters. -

```
integer : in cdm_type.c_node_ref; -- points to an integer type definition whose
                                     parent is to be determined.
```

3.2.2.460.11 Side-Effects. - None

3.2.2.460.12 Algorithm. -

```
-- This routine needs 6 constants.
-- The constants are:
-- short_int_last, int_last, long_int_last, short_int_first,
-- int_first and long_int_first.

range := integer.as_range
VERIFY that range.as_expl.sm_value /= void and range.as_exp2.sm_value
                                                        /= void>

if RANGE.AS_EXP2.SM_VALUE > short_int_last> then
     if RANGE.AS_EXP2.SM_VALUE > int_last> then
          temp := large
          EMIT error 16001 if range.as_exp2.sm_value  LONG_INT_LAST>
     else
          temp := medium
     end if
```

```
else
   temp := short
end if

if temp = short and RANGE.AS_EXP1.SM_VALUE  SHORT_INT_FIRST> then
   if RANGE.AS_EXP1.SM_VALUE  INT_FIRST> then
      temp := large
   else
      temp := medium
   end if
elsif
      temp := medium and RANGE.AS_EXP1SM_VALUE  INT_FIRST> then
      temp := large
end if
SET range.sm_base_type and integer.sm_type_struct according to the
           value of temp>
```

3.2.2.460.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
16000	E	integer type definition does not have static range elements
16001	E	integer type definition exceeds accuracy of this target

3.2.2.460.14 Examples of Data Structures. - None

3.2.2.461 Subprogram. - float_type

3.2.2.461.1 Purpose. - This routine takes a node of type "float" (representing a floating point definition) and fills in its TYPE_SPEC fields to point to the appropriate floating point definition.

3.2.2.461.2 __Assumptions__. - None

3.2.2.461.3 __Implementation Language__. - This subprogram is written in Ada.

3.2.2.461.4 __Used Recursively__. - This subprogram is not used recursively.

3.2.2.461.5 __Nested Within__. - target_type

3.2.2.461.6 __Host Dependencies__. - None

3.2.2.461.7 __Target Dependencies__. - None

3.2.2.461.8 __Subprogram Visibility__. - Outside Package.

3.2.2.461.9 __Function/Procedure__. - This subprogram is a Procedure.

3.2.2.461.10 __Formal Parameters__. -

```
float_node : in cdm_type.c_node_ref; -- points to the floating point
                                        type definition whose parent
                                        is to be determined
```

3.2.2.461.11 __Side-Effects__. - None

3.2.2.461.12 __Algorithm__. -

```
-- This routine needs the constants:
-- short_float_digits, float_digits and long_float_digits.

if float_node.as_exp.sm_value /= void then
     V := float.as_exp.sm_value
     if V _= short_float_digits> then
        float_node.as_exp.sm_base_type := float_node.sm_type_struct :=
                                        pointer to short float in standard
    elsif V _= int_float_digits> then
      -- similarity for float

    elsif V _= long_float_digits> then
      -- similiarty for long_float
    else
      EMIT error message that def's exceeds targets accuracy>
    endif
else
   EMIT error message that digits field s not static>
end if
```

3.2.2.461.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
16002	E	floating point constraint in floating point definition is not static
16003	E	floating point definition exceeds accuracy of target machine

3.2.2.461.14 Examples of Data Structures. - None

3.2.2.462 Package. - stmtchk

3.2.2.462.1 Purpose. - Define data structures to be used by all statement checking routines.

3.2.2.462.2 Number of Subprograms. - None

3.2.2.462.3 Dependencies on Other Packages for Spec. - cdm_type

3.2.2.462.4 Additional Dependencies for Body. - None

3.2.2.462.5 Package Specification. -

```
--      OVERVIEW OF STATEMENT CHECKING FUNCTION
--
-- The statement checking function is organized a a recursive descent
-- traversal of the Diana tree.   The following subprograms are present:
--
-- 1. There is one procedure for each Diana node type.
--         - This procedure is named SC_diana_name, where "diana_name"
--           is the name of the node, as defined in the Diana Reference
--           manual.
--         - There are two parameters:
--              a. A c_node_ref to the node to be checked
--              b. A variable of type CONTEXT which tessl where in the
--                 compilation this node appears (e.g. in a task
--                 specification, in a declarative part, etc.)
--         - Each procedure calls the procedures corresponding to the
--           structural attributes of the node being checked and then
--           checks the node itself.  The calls to other procedures are
--           repsented by the pseudo-code lines:
--
--             CHECK as_structural_attribute_name>
--
-- 2. There is one procedure for each Diana class
--           - This procedure is named CL_diana_name, where "diana_name" is
--             the name of the class, as defined in the Diana Reference
--             Manual.
--           - This procedure consists of a case statement which has one
--             branch for each element of the class.  In each branch, there
--             is a call to one traversal routine of type  1 (above).
--
package stmtchk is
    inner_seq:  cdm_type.c_node_ref;
    --pointer to list of statement sequences
      type contexts is
       (in_decl_part,    --in declarative part of a package
       in_task_spec,  --in a task specification
       in_select   --in a select statement
       in_gen_parm,    --in a generic parameter
       stat_range, --static range required
       stat_exp,   --static expression required
       discrim_exp,    --requires expression which
               depends only on discriminants
       arr_rec_parm,  --actual parameter of array or
               record type
       arr_rec_ret);  --retrun for array or record type
    type context_rec is
         NAME:  stmtchk.contexts;
             set_used: boolean;
```

```
          vis_priv:  boolean;
          accept_ct:  integer;
          delay_ct:  integer:
          term_ct:  integer:
          pack_body_name:  cdm_basics.c_node_ref;
   type SEQ_USAGE is
     (EXCP_SEQ, LOOP_SEQ ... etc.);
   type SEQ_INFO is record
     USAGE : SEQ_USAGE;
     HAS_CODE : BOOLEAN;
     HAS_NON_CODE : BOOLEAN;
end record;
end context_rec;
end stmtchk;
```

3.2.2.462.6 Elaboration Code. -

LIST of statement sequences := empty>
CALL_RELATION list := empty>
ELABORATION_CALL list := empty>

3.2.2.462.7 Examples of Data Structures. - list of statement sequences: (maintained in container).

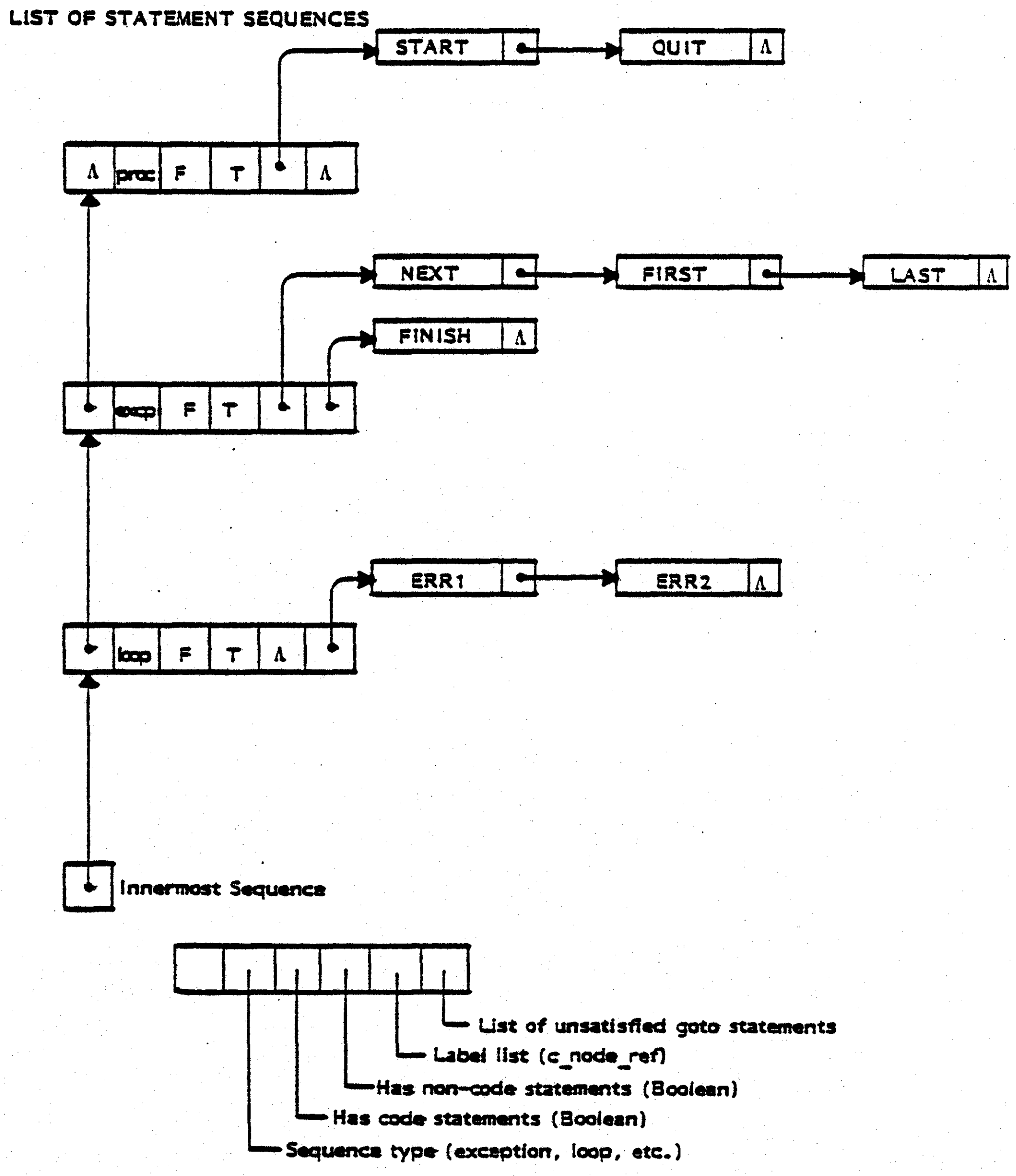

LIST OF STATEMENT SEQUENCES
START
QUIT
Λ
proc
F
T
NEXT
FIRST
LAST
FINISH
excp
ERR1
ERR2
loop
Innermost Sequence
List of unsatisfied goto statements
Label list (c_node_ref)
Has non-code statements (Boolean)
Has code statements (Boolean)
Sequence type (exception, loop, etc.)

Figure 3-7. Statement Checking Function

3.2.2.463 Package. - scp_decl

3.2.2.463.1 Purpose. - Implement the statement checking function for nodes involved with declarations.

3.2.2.463.2 Number of Subprograms. - 46

3.2.2.463.3 Dependencies on Other Packages for Spec. - cdm_type, stmtchk

3.2.2.463.4 Additional Dependencies for Body. - cdm_as, sc_class, diagnose

3.2.2.463.5 Package Specification. -

```
package scp_decl is
    procedure  sc_constant;
    procedure  sc_var;
    procedure  sc_number;
    procedure  sc_type;
    procedure  sc_subtype;
    procedure  sc_constrnd;
    procedure  sc_derived
    procedure  sc_range;
    procedure  sc_integer;
    procedure  sc_floatt;
    procedure  sc_floatc;
    procedure  sc_sc_fixed;
    procedure  sc_array;
    procedure  sc_dscrt_r_s;
    procedure  sc_record;
    procedure  sc_variant_p;
    procedure  sc_variant_s;
    procedure  sc_variant;
    procedure  sc_access;
    procedure  sc_item_s;
    procedure  sc_procedure;
    procedure  sc_function;
    procedure  sc_param_s;
    procedure  sc_in
    procedure  sc_out
    procedure  sc_subp_body;
    procedure  sc_pack_decl;
    procedure  sc_pack_spec;
    procedure  sc_decl_s;
    procedure  sc_decl_reps;
    procedure  sc_pack_body;
    procedure  sc_private;
    procedure  sc_task_decl;
    procedure  sc_task_spec
    procedure  sc_task_body
    procedure  sc_entry
    procedure  sc_generic;
    procedure  sc_gen_parms;
    procedure  sc_sim_rep;
    procedure  sc_rec_rep;
    procedure  sc_comp_reps;
    procedure  sc_comp_rep;
    procedure  sc_address;
    procedure  sc_dscrm_agg;
    procedure  sc_subp_decl;
    procedure  sc_rename;
    end scp_decl;
```

3.2.2.463.6 Elaboration Code. - None

3.2.2.463.7 Examples of Data Structures. - None

3.2.2.464 Subprogram. - sc_constant

3.2.2.464.1 Purpose. - Perform statement checking for a node of type 'constant'.

3.2.2.464.2 Assumptions. - None

3.2.2.464.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.464.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.464.5 Nested within. - scp_decl

3.2.2.464.6 Host Dependencies. - None

3.2.2.464.7 Target Dependencies. - None

3.2.2.464.8 Subprogram Visibility. - Outside Package.

3.2.2.464.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.464.10 Formal Parameters. -

```
subtree:  in  cdm_type.c_node_ref; -- reference to diana node
                                       to be checked.

context:  in  stmtchk.context_rec; -- defines context in which
                                      this node appears.
```

3.2.2.464.11 Side-Effects. - None

3.2.2.464.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node already flagged as in error> then
  return;
end if;
CHECK  as_type_spec>
CHECK as_object_def>
if SM_OBJECT_DEF is void> then
  error  17000
end if;
if OBJECT is of a task type> then
   ERROR  17002>
end if;
if OBJECT is limited private or has
```

```
    limited private components> then
   if NOT in private part of package> then
      ERROR  17003>
   end if;
end if;
if UNCONSTRAINED array type> then
  if INITIALIZING array is "ragged"> then
    error  17003
  end if;
end if;
```

3.2.2.464.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17000	E	constant has no value
17003	E	invalid array structure
17001	E	task type can not be initialized
17002	E	limited private type can not be initialized

3.2.2.464.14 Examples of Data Structures. - None

3.2.2.465 Subprogram. - sc_var

3.2.2.465.1 Purpose. - Perform statement checking for a node of type 'var'.

3.2.2.465.2 Assumptions. - None

3.2.2.465.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.465.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.465.5 Nested Within. - scp_decl

3.2.2.465.6 Host Dependencies. - None

3.2.2.465.7 Target Dependencies. - None

3.2.2.465.8 Subprogram Visibility. - Outside Package.

3.2.2.465.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.465.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       checked.

context:  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.465.11 Side-Effects. - None

3.2.2.465.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node already flagged as in error> then
  return;
end if;
CHECK as_type_spec>
CHECK as_object_def>
if UNCONSTRAINED array type> then
  ERROR  17004>
end if;
```

```
if OBJECT has limited private type> and NOT in private part
                                              of package> then
  ERROR  17005>
end if;
if OBJECT has a task type> then
   ERROR  17006>
```

3.2.2.465.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
17004	E	variable has unconstrained array type
17005	E	invalid initialization of object of limited private type
17006	E	task type can not be initialized

3.2.2.465.14 Examples of Data Structures. - None

3.2.2.466 Subprogram. - sc_number

3.2.2.466.1 Purpose. - Perform statement checking on a node of type 'number'.

3.2.2.466.2 Assumptions. - None

3.2.2.466.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.466.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.466.5 Nested Within. - scp-decl

3.2.2.466.6 Host Dependencies. - None

3.2.2.466.7 Target Dependencies. - None

3.2.2.466.8 Subprogram Visibility. - Outside Package.

3.2.2.466.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.466.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked.

context:  in stmtchk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.466.11 Side-Effects. - None

3.2.2.466.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node already flagged as in error> then
  return;
end if;
CHECK as_exp with static_required context>
```

3.2.2.466.13 Diagnostic Messages Generated. - None

3.2.2.466.14 Examples of Data Structures. - None

3.2.2.467 Subprogram. - sc_type

3.2.2.467.1 Purpose. - Perform statement checking on a node of type 'type'.

3.2.2.467.2 Assumptions. - None

3.2.2.467.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.467.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.467.5 Nested Within. - scp_decl

3.2.2.467.6 Host Dependencies. - None

3.2.2.467.7 Target Dependencies. - None

3.2.2.467.8 Subprogram Visibility. - Outside Package.

3.2.2.467.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.467.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.467.11 Side-Effects. - None

3.2.2.467.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_var_s>
CONSTRUCT discriminant list from as_var_s>
CHECK as_type_spec>
```

3.2.2.467.13 Diagnostic Messages Generated. - None

3.2.2.467.14 Examples of Data Structures. - None

3.2.2.468 Subprogram. - sc_subtype

3.2.2.468.1 Purpose. - Perform statement checking on a node of type 'subtype'.

3.2.2.468.2 Assumptions. - None

3.2.2.468.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.468.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.468.5 Nested within. - scp_decl

3.2.2.468.6 Host Dependencies. - None

3.2.2.468.7 Target Dependencies. - None

3.2.2.468.8 Subprogram Visibility. - Outside Package.

3.2.2.468.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.468.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.468.11 Side-Effects. - None

3.2.2.468.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_constraint>
```

3.2.2.468.13 Diagnostic Messages Generated. - None

3.2.2.468.14 Examples of Data Structures. - None

3.2.2.469 <u>Subprogram</u>. - sc_constrnd

3.2.2.469.1 <u>Purpose</u>. - Perform statement checking on a node of type 'constrained'.

3.2.2.469.2 <u>Assumptions</u>. - None

3.2.2.469.3 <u>Implementation Language</u>. - This Subprogram is written in Ada.

3.2.2.469.4 <u>Used Recursively</u>. - This subprogram is used recursively.

3.2.2.469.5 <u>Nested Within</u>. - scp_decl

3.2.2.469.6 <u>Host Dependencies</u>. - None

3.2.2.469.7 <u>Target Dependencies</u>. - None

3.2.2.469.8 <u>Subprogram Visibility</u>. - Outside Package.

3.2.2.469.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.469.10 <u>Formal Parameters</u>. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  inout stmtchk.context_rec; -- defines context in which this
                                        this node appears
```

3.2.2.469.11 Side-Effects. - None

3.2.2.469.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_constraint>
if NAME is already constrained> then
  ERROR  17007>
end if;
```

3.2.2.469.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
17007	E	additional constraints not allowed

3.2.2.469.14 Examples of Data Structures. - None

3.2.2.470 Subprogram. - sc_derived

3.2.2.470.1 Purpose. - Perform statement checking on a node of type 'derived'.

3.2.2.470.2 Assumptions. - None

3.2.2.470.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.470.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.470.5 Nested Within. - scp_decl

3.2.2.470.6 Host Dependencies. - None

3.2.2.470.7 Target Dependencies. - None

3.2.2.470.8 Subprogram Visibility. - Outside Package.

3.2.2.470.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.470.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked.

context: inout stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.470.11 Side-Effects. - None

3.2.2.470.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_constrained>
```

3.2.2.470.13 Diagnostic Messages Generated. - None

3.2.2.470.14 Examples of Data Structures. - None

3.2.2.471 Subprogram. - sc_range

3.2.2.471.1 Purpose. - Perform statement checking on a node of type 'range'.

3.2.2.471.2 Assumptions. - None

3.2.2.471.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.471.4 Used Recursively. - This subprogram is used recursively.

3.2.2.471.5 Nested Within. - scp_decl

3.2.2.471.6 Host Dependencies. - None

3.2.2.471.7 Target Dependencies. - None

3.2.2.471.8 Subprogram Visibility. - Outside Package.

3.2.2.471.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.471.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.471.11 Side-Effects. - None

3.2.2.471.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp1 with static requirement from context>
CHECK as_exp2 with static requirement from context>
```

3.2.2.471.13 Diagnostic Messages Generated. - None

3.2.2.471.14 Examples of Data Structures. - None

3.2.2.472 Subprogram. - sc_integer

3.2.2.472.1 Purpose. - Perform statement checking on a node of type 'integer'.

3.2.2.472.2 Assumptions. - None

3.2.2.472.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.472.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.472.5 Nested Within. - scp_decl

3.2.2.472.6 Host Dependencies. - None

3.2.2.472.7 Target Dependencies. - None

3.2.2.472.8 Subprogram Visibility. - Outside Package.

3.2.2.472.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.472.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                        to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                         this node appears
```

3.2.2.472.11 Side-Effects. - None

3.2.2.472.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_range, static context>
```

3.2.2.472.13 Diagnostic Messages Generated. - None

3.2.2.472.14 Examples of Data Structures. - None

3.2.2.473 Subprogram. - sc_floatt

3.2.2.473.1 Purpose. - Perform statement checking on a node of type 'float,' which is used as a type_spec.

3.2.2.473.2 Assumptions. - None

3.2.2.473.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.473.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.473.5 Nested Within. - scp_decl

3.2.2.473.6 Host Dependencies. - None

3.2.2.473.7 Target Dependencies. - None

3.2.2.473.8 Subprogram Visibility. - Outside Package.

3.2.2.473.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.473.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  inout context_rec; -- defines context in which
                                this node appears
```

3.2.2.473.11 Side-Effects. - None

3.2.2.473.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error then
  return;
end if;
CHECK as_exp with static context>
if VALUE of as_exp is non_positive> then
  error  17008
end if;
if RANGE is null> then
   ERROR  17009>
end if;
CHECK as_range_void with static context>
```

3.2.2.473.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17008	E	non_negative value required for digits
17009	E	range is empty

3.2.2.473.14 Examples of Data Structures. - None

3.2.2.474 Subprogram. - sc_floatc

3.2.2.474.1 Purpose. - Perform statement checking on a node of type 'float', which is used as a constraint.

3.2.2.474.2 Assumptions. - None

3.2.2.474.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.474.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.474.5 Nested Within. - scp_decl

3.2.2.474.6 Host Dependencies. - None

3.2.2.474.7 Target Dependencies. - None

3.2.2.474.8 Subprogram Visibility. - Outside Package.

3.2.2.474.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.474.10 Formal Parameters. -

```
subtree:  in com_type.c_node_ref;  -- reference to diana node
                                       to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.474.11 Side-Effects. - None

3.2.2.474.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp with static context>
if VALUE of as_exp  0 or value of as_exp> max.digits> then
  ERROR  17010>
end if;
```

3.2.2.474.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17010	E	invalid constraint value

3.2.2.474.14 Examples of Data Structures. - None

3.2.2.475 Subprogram. - sc_fixed

3.2.2.475.1 Purpose. - Perform statement checking on a node of type 'fixed'.

3.2.2.475.2 Assumptions. - None

3.2.2.475.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.475.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.475.5 Nested within. - scp_decl

3.2.2.475.6 Host Dependencies. - None

3.2.2.475.7 Target Dependencies. - None

3.2.2.475.8 Subprogram Visibility. - Outside Package.

3.2.2.475.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.475.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.475.11 Side-Effects. - None

3.2.2.475.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp with static context>
if VALUE of as_exp  system_min_delta> then
  ERROR  17011>
end if;
CHECK as_range_void with static context>
if VALUE of as_exp greater than max (abs (left range exp,
                                        right range exp) > then
  ERROR  17012>
end if;
```

3.2.2.475.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17011	E	delta expression too small
17012	E	delta expression too large

3.2.2.475.14 Examples of Data Structures. - None

3.2.2.476 Subprogram. - sc_array

3.2.2.476.1 Purpose. - Perform statement checking on a node of type 'array'.

3.2.2.476.2 Assumptions. - None

3.2.2.476.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.476.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.476.5 Nested within. - scp_decl

3.2.2.476.6 Host Dependencies. - None

3.2.2.476.7 Target Dependencies. - None

3.2.2.476.8 Subprogram Visibility. - Outside Package.

3.2.2.476.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.476.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.476.11 Side-Effects. - None

3.2.2.476.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_dscrt_range_s>
CHECK as_constrained>
if SUBTYPE indication (as constrained) not constrained> then
  ERROR  17013>
end if;
```

3.2.2.476.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17013	E	unconstrained array component

3.2.2.476.14 Examples of Data Structures. - None

3.2.2.477 Subprogram. - sc_dscrt_r_s

3.2.2.477.1 Purpose. - Perform statement checking on a node of type 'dscrt_rang_s'.

3.2.2.477.2 Assumptions. - None

3.2.2.477.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.477.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.477.5 Nested Within. - scp_decl

3.2.2.477.6 Host Dependencies. - None

3.2.2.477.7 Target Dependencies. - None

3.2.2.477.8 Subprogram Visibility. - Outside Package.

3.2.2.477.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.477.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.477.11 Side-Effects. - None

3.2.2.477.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK dscrt_range>
end loop;
```

3.2.2.477.13 Diagnostic Messages Generated. - None

3.2.2.477.14 Examples of Data Structures. - None

3.2.2.478 Subprogram. - sc_record

3.2.2.478.1 Purpose. - Perform statement checking on a node of type 'record'.

3.2.2.478.2 Assumptions. - None

3.2.2.478.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.478.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.478.5 Nested Within. - scp_decl

3.2.2.478.6 Host Dependencies. - None

3.2.2.478.7 Target Dependencies. - None

3.2.2.478.8 Subprogram Visibility. - Outside Package.

3.2.2.478.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.478.10 Formal Parameters. -

```
subtree:  in com_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                       this node appears.
```

3.2.2.478.11 Side-Effects. - None

3.2.2.478.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK comp>
end loop;
```

3.2.2.478.13 Diagnostic Messages Generated. - None

3.2.2.478.14 Examples of Data Structures. - None

3.2.2.479 Subprogram. - sc_variant_p

3.2.2.479.1 Purpose. - Perform statement checking on a node of type 'variant_part'.

3.2.2.479.2 Assumptions. - None

3.2.2.479.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.479.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.479.5 Nested Within. - scp_decl

3.2.2.479.6 Host Dependencies. - None

3.2.2.479.7 Target Dependencies. - None

3.2.2.479.8 Subprogram Visibility. - Outside Package.

3.2.2.479.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.479.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked

context:  inout stmtchk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.479.11 Side-Effects. - None

3.2.2.479.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
INITIALIZE discrete range coverage data structure>
CHECK as_variant_s>
```

3.2.2.479.13 Diagnostic Messages Generated. - None

3.2.2.479.14 Examples of Data Structures. - None

3.2.2.480 Subprogram. - sc_decl_s

3.2.2.480.1 Purpose. - Perform statement checking on a node of type 'decl_s'.

3.2.2.480.2 Assumptions. - None

3.2.2.480.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.480.4 Used Recursively. - This subprogram is used recursively.

3.2.2.480.5 Nested Within. - sco_decl

3.2.2.480.6 Host Dependencies. - None

3.2.2.480.7 Target Dependencies. - None

3.2.2.480.8 Subprogram Visibility. - Outside Package.

3.2.2.480.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.480.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana
                                       node to be checked

context:  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.480.11 Side-Effects. - None

3.2.2.480.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK DECL>
end loop;
```

3.2.2.480.13 Diagnostic Messages Generated. - None

3.2.2.480.14 Examples of Data Structures. - None

3.2.2.481 Subprogram. - sc_pack_body

3.2.2.481.1 Purpose. - Perform statement checking on a node of type 'package_body'.

3.2.2.481.2 Assumptions. - None

3.2.2.481.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.481.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.481.5 Nested Within. - scp_decl

3.2.2.481.6 Host Dependencies. - None

3.2.2.481.7 Target Dependencies. - None

3.2.2.481.8 Subprogram Visibility. - Outside Package.

3.2.2.481.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.481.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana
                                      node to be checked

context:  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.481.11 Side-Effects. - None

3.2.2.481.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_block_stub>
```

3.2.2.481.13 Diagnostic Messages Generated. - None

3.2.2.481.14 Examples of Data Structures. - None

3.2.2.482 Subprogram. - sc_private

3.2.2.482.1 Purpose. - Perform statement checking on a node of type 'private' or 'l-private'.

3.2.2.482.2 Assumptions. - None

3.2.2.482.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.482.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.482.5 Nested Within. - scp_decl

3.2.2.482.6 Host Dependencies. - None

3.2.2.482.7 Target Dependencies. - None

3.2.2.482.8 Subprogram Visibility. - Outside Package.

3.2.2.482.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.482.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked

context:  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.482.11 Side-Effects. - None

3.2.2.482.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
if CONTEXT not the visible part of a package spec> then
  ERROR  17014>
end if;
```

3.2.2.482.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17014	E	priv declaration does not appear in pkg spec.

3.2.2.482.14 Examples of Data Structures. - None

3.2.2.483 Subprogram. - sc_task_decl

3.2.2.483.1 Purpose. - Perform statement checking on a node of type 'task_decl'.

3.2.2.483.2 Assumptions. - None

3.2.2.483.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.483.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.483.5 Nested Within. - scp_decl

3.2.2.483.6 Host Dependencies. - None

3.2.2.483.7 Target Dependencies. - None

3.2.2.483.8 Subprogram Visibility. - Outside Package.

3.2.2.483.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.483.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.483.11 Side-Effects. - None

3.2.2.483.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_task_def>
```

3.2.2.483.13 Diagnostic Messages Generated. - None

3.2.2.483.14 Examples of Data Structures. - None

3.2.2.484 Subprogram. - sc_task_spec

3.2.2.484.1 Purpose. - Perform statement checking on a node of type 'task_spec'.

3.2.2.484.2 Assumptions. - None

3.2.2.484.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.484.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.484.5 Nested Within. - scp_decl

3.2.2.484.6 Host Dependencies. - None

3.2.2.484.7 Target Dependencies. - None

3.2.2.484.8 Subprogram Visibility. - Outside Package.

3.2.2.484.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.484.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  in stmtchk.context_rec;  -- defines context in which
                                     this node appears
```

3.2.2.484.11 Side-Effects. - None

3.2.2.484.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_decl_rep_s>
```

3.2.2.484.13 Diagnostic Messages Generated. - None

3.2.2.484.14 Examples of Data Structures. - None

3.2.2.485 Subprogram. - sc_task_body

3.2.2.485.1 Purpose. - Perform statement checking on a node of type 'task_body'.

3.2.2.485.2 Assumptions. - None

3.2.2.485.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.485.4 Used Recursively. - This subprogram is used recursively.

3.2.2.485.5 Nested Within. - scp_decl

3.2.2.485.6 Host Dependencies. - None

3.2.2.485.7 Target Dependencies. - None

3.2.2.485.8 Subprogram Visibility. - Outside Package.

3.2.2.485.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.485.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  in stmtchk.context_rec;  -- defines context in which
                                     this node appears
```

3.2.2.485.11 Side-Effects. - None

3.2.2.485.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_block_stub>
```

3.2.2.485.13 Diagnostic Messages Generated. - None

3.2.2.485.14 Examples of Data Structures. - None

3.2.2.486 Subprogram. - sc_entry

3.2.2.486.1 Purpose. - Perform statement checking on a node of type 'entry'.

3.2.2.486.2 Assumptions. - None

3.2.2.486.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.486.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.486.5 Nested within. - scp_decl

3.2.2.486.6 Host Dependencies. - None

3.2.2.486.7 Target Dependencies. - None

3.2.2.486.8 Subprogram Visibility. - Outside Package.

3.2.2.486.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.486.10 Formal Parameters. -

```
suotree:  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context:  in stmtchk.context_rec; -- defines contet in which
                                     this node appears
```

3.2.2.486.11 Side-Effects. - None

3.2.2.486.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_dscrt_range_void>
CHECK as_param_s>
if CONTEXT not a task spec> then
  ERROR  17015>
end if;
```

3.2.2.486.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17015	E	entry declaration allowed only in task specification

3.2.2.486.14 Examples of Data Structures. - None

3.2.2.487 Subprogram. - sc_decl_reps

3.2.2.487.1 Purpose. - Perform statement checking on a node of type 'decl_rep_s'.

3.2.2.487.2 Assumptions. - None

3.2.2.487.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.487.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.487.5 Nested Within. - scp_decl

3.2.2.487.6 Host Dependencies. - None

3.2.2.487.7 Target Dependencies. - None

3.2.2.487.8 Subprogram Visibility. - Outside Package.

3.2.2.487.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.487.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to be
                                        checked

context :  in  stmtchk.context_rec; -- defines context in which

                                       this node appears
```

3.2.2.487.11 Side-Effects. - None

3.2.2.487.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK DECL_REP>
end loop;
```

3.2.2.487.13 Diagnostic Messages Generated. - None

3.2.2.487.14 Examples of Data Structures. - None

3.2.2.488 Subprogram. - sc_generic

3.2.2.488.1 Purpose. - Perform statement checking on a node of type 'generic'.

3.2.2.488.2 Assumptions. - None

3.2.2.488.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.488.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.488.5 Nested within. - scp_decl

3.2.2.488.6 Host Dependencies. - None

3.2.2.488.7 Target Dependencies. - None

3.2.2.488.8 Subprogram Visibility. - Outside Package.

3.2.2.488.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.488.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.488.11 Side-Effects. - None

3.2.2.488.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
```

```
  return;
end if;
CHECK as_generic_param_s>
CHECK as_generic_header>
```

3.2.2.488.13 Diagnostic Messages Generated. - None

3.2.2.488.14 Examples of Data Structures. - None

3.2.2.489 Subprogram. - sc_gen_parms

3.2.2.489.1 Purpose. - Perform statement checking on a node of type 'generic_param_s'.

3.2.2.489.2 Assumptions. - None

3.2.2.489.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.489.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.489.5 Nested Within. - scp_decl

3.2.2.489.6 Host Dependencies. - None

3.2.2.489.7 Target Dependencies. - None

3.2.2.489.8 Subprogram Visibility. - Outside Package.

3.2.2.489.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.489.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context-rec; -- defines context in which
                                       this node appears
```

3.2.2.489.11 Side-Effects. - None

3.2.2.489.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK generic_param>
end loop;
```

3.2.2.489.13 Diagnostic Messages Generated. - None

3.2.2.489.14 Examples of Data Structures. - None

3.2.2.490 Subprogram. - sc_simp_rep

3.2.2.490.1 Purpose. - Perform statement checking on a node of type 'simple_rep'.

3.2.2.490.2 Assumptions. - None

3.2.2.490.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.490.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.490.5 Nested Within. - scp_decl

3.2.2.490.6 Host Dependencies. - None

3.2.2.490.7 Target Dependencies. - None

3.2.2.490.8 Subprogram Visibility. - Outside Package.

3.2.2.490.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.490.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.490.11 Side-Effects. - None

3.2.2.490.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_exp with static context>
if AS_NAME denotes an enumeration type> then
  for EACH element of as_exp> loop
    SET sm_rep for corresponding enumeration literal>
  end loop;
  for EACH literal> loop
    if SM_REP = (sm_rep for previous literal)> then
      ERROR  17016>
    end if;
  end loop;
else -- must be an attribute
  if SIZE attribute> then
    CHECK that   of bits is sufficient>
  end if;
end if;
```

3.2.2.490.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17016	E	inconsistent representation for enumeration literals

3.2.2.490.14 Examples of Data Structures. - None

3.2.2.491 Subprogram. - sc_rec_rep

3.2.2.491.1 Purpose. - Perform statement checking on a node of type 'record_rep'.

3.2.2.491.2 Assumptions. - None

3.2.2.491.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.491.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.491.5 Nested Within. - scp_decl

3.2.2.491.6 Host Dependencies. - None

3.2.2.491.7 Target Dependencies. - None

3.2.2.491.8 Subprogram Visibility. - Outside Package.

3.2.2.491.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.491.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                         node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.491.11 Side-Effects. - None

3.2.2.491.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp_void with static context>
CHECK as_comp_rep_s>
```

3.2.2.491.13 Diagnostic Messages Generated. - None

3.2.2.491.14 Examples of Data Structures. - None

3.2.2.492 Subprogram. - sc_comp_reps

3.2.2.492.1 Purpose. - Perform statement checking on a node of type 'comp_rep_s'.

3.2.2.492.2 Assumptions. - None

3.2.2.492.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.492.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.492.5 Nested Within. - scp_decl

3.2.2.492.6 Host Dependencies. - None

3.2.2.492.7 Target Dependencies. - None

3.2.2.492.8 Subprogram Visibility. - Outside Package.

3.2.2.492.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.492.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.492.11 Side-Effects. - None

3.2.2.492.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK comp_rep>
end loop;
```

3.2.2.492.13 Diagnostic Messages Generated. - None

3.2.2.492.14 Examples of Data Structures. - None

3.2.2.493 Subprogram. - sc_comp_rep

3.2.2.493.1 Purpose. - Perform statement checking on a node of type 'comp_rep'.

3.2.2.493.2 Assumptions. - None

3.2.2.493.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.493.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.493.5 Nested within. - scp_decl

3.2.2.493.6 Host Dependancies. - None

3.2.2.493.7 Target Dependencies. - None

3.2.2.493.8 Subprogram Visibility. - Outside Package.

3.2.2.493.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.493.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                         node to be checked

context :  in stmtchk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.493.11 Side-Effects. - None

3.2.2.493.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exo with static context>
CHECK as_range with static context>
```

3.2.2.493.13 Diagnostic Messages Generated. - None

3.2.2.493.14 Examples of Data Structures. - None

3.2.2.494 Subprogram. - sc_address

3.2.2.494.1 Purpose. - Perform statement checking on a node of type 'address'.

3.2.2.494.2 Assumptions. - None

3.2.2.494.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.494.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.494.5 Nested Within. - scp_decl

3.2.2.494.6 Host Dependencies. - None

3.2.2.494.7 Target Dependencies. - None

3.2.2.494.8 Subprogram Visibility. - Outside Package.

3.2.2.494.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.494.10 Formal Parameters. -

```
subtree :  in  cdm_type.c-node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.494.11 Side-Effects. - None

3.2.2.494.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp with static CONTEXT>
```

3.2.2.494.13 Diagnostic Messages Generated. - None

3.2.2.494.14 Examples of Data Structures. - None

3.2.2.495 Subprogram. - sc_variant_s

3.2.2.495.1 Purpose. - Perform statement checking on a node of type 'variant_s'.

3.2.2.495.2 Assumptions. - None

3.2.2.495.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.495.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.495.5 Nested Within. - scp_decl

3.2.2.495.6 Host Dependencies. - None

3.2.2.495.7 Target Dependencies. - None

3.2.2.495.8 Subprogram Visibility. - Outside Package.

3.2.2.495.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.495.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.495.11 Side-Effects. - None

3.2.2.495.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK VARIANT>
end loop;
```

3.2.2.495.13 Diagnostic Messages Generated. - None

3.2.2.495.14 Examples of Data Structures. - None

3.2.2.496 Subprogram. - sc_variant

3.2.2.496.1 Purpose. - Perform statement checking on a node of type 'variant'.

3.2.2.496.2 Assumptions. - None

3.2.2.496.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.496.4 Used Recursively. - This subprogram is used recursively.

3.2.2.496.5 Nested Within. - scp_decl

3.2.2.496.6 Host Dependencies. - None

3.2.2.496.7 Target Dependencies. - None

3.2.2.496.8 Subprogram Visibility. - Outside Package.

3.2.2.496.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.496.10 Formal Parameters. -

```
subtree :  cdm_type.c_node_ref;  -- reference to diana
                                    node to be checked

context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.496.11 Side-Effects. - None

3.2.2.496.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_choice_s> loop
  CHECK CHOICE, requiring static expression or name of static subtype>
  INCLUDE this choice is coverage data structure>
  if THERE is duplicate coverage> then
    ERROR  17017>
  end if;
end loop;
CHECK as_record with static or discriminant context>
if DISCRIMINANT subtype is static> then
  if RANGE not covered> then
    ERROR  17018>
  end if;
else
  if RANGE of base type not covered> then
```

```
    ERROR  17019>
  end if;
end if;
```

3.2.2.496.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17017	E	duplicate choice values in record variants
17108	E	discrete range not covered by choice values
17019	E	discrete range not covered by choice values

3.2.2.496.14 Examples of Data Structures. -

The discrete range coverage data structure is used to check that the choices in a case statement, aggregate, or variant record, do not overlap and cover the required range completely.
At any time during the scan of a sequence of choices, the data structure has a list of pairs of integers which represent the portions of the discrete range which have been covered so far.
As each choice is scanned, the list is updated so that adjacent choices are "collapsed" into one pair.

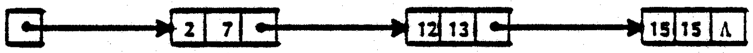
DISCRETE RANGE COVERAGE

A linked list of pairs of (integer) values (a_i, b_i), $i = 1, \ldots, n$
such that $a_i \leq b_i$ for all i, and $b_{i+1} + 1 < a_{i+1}$ for $i = 1, \ldots, n - 1$

After processing

```
    when 2 .. 7 =>
    when 12, 13 =>
    when 15 =>
```

the list would be



After processing

```
    when 14 =>
```

in addition to the previous choices, the data structure becomes:

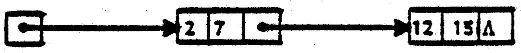


Figure 3-8. Statement Checking Function

3.2.2.497 Subprogram. - sc_access

3.2.2.497.1 Purpose. - Perform statement checking on a node of type 'access'.

3.2.2.497.2 Assumptions. - None

3.2.2.497.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.497.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.497.5 Nested within. - scp_decl

3.2.2.497.6 Host Dependencies. - None

3.2.2.497.7 Target Dependencies. - None

3.2.2.497.8 Subprogram Visibility. - Outside Package.

3.2.2.497.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.497.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana
                                          node to be checked

context :  inout  stmtchk.context_rec; -- defines context in which
                                           this node appears
```

3.2.2.497.11 Side-Effects. - None

3.2.2.497.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
if CONSTRAINT is present for type which is already constrained> then
   ERROR  17020>
end if;
CHECK as_constrained>
```

3.2.2.497.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17020	E	duplicate constraints

3.2.2.497.14 Examples of Data Structures. - None

3.2.2.498 Subprogram. - sc_item_s

3.2.2.498.1 Purpose. - Perform statement checking on a node of type 'item_s'.

3.2.2.498.2 Assumptions. - None

3.2.2.498.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.498.4 Used Recursively. - This subprogram is used recursively.

3.2.2.498.5 Nested Within. - scp_decl

3.2.2.498.6 Host Dependencies. - None

3.2.2.498.7 Target Dependencies. - None

3.2.2.498.8 Subprogram Visibility. - Outside Package.

3.2.2.498.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.498.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                         node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.498.11 Side-Effects. - None

3.2.2.498.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK ITEM>
end loop;
```

3.2.2.498.13 Diagnostic Messages Generated. - None

3.2.2.498.14 Examples of Data Structures. - sc_procedure

3.2.2.499 Subprogram. - sc_procedure

3.2.2.499.1 Purpose. - Perform statement checking on a node of type 'procedure'.

3.2.2.499.2 Assumptions. - None

3.2.2.499.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.499.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.499.5 Nested within. - scp_decl

3.2.2.499.6 Host Dependencies. - None

3.2.2.499.7 Target Dependencies. - None

3.2.2.499.8 Subprogram Visibility. - Outside Package.

3.2.2.499.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.499.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.499.11 Side-Effects. - None

3.2.2.499.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_param_s>
```

3.2.2.499.13 Diagnostic Messages Generated. - None

3.2.2.499.14 Examples of Data Structures. - None

3.2.2.500 Subprogram. - sc_function

3.2.2.500.1 Purpose. - Perform statement checking on a node of type 'function'.

3.2.2.500.2 Assumptions. - None

3.2.2.500.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.500.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.500.5 Nested Within. - scp_decl

3.2.2.500.6 Host Dependencies. - None

3.2.2.500.7 Target Dependencies. - None

3.2.2.500.8 Subprogram Visibility. - Outside Package.

3.2.2.500.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.500.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in stmtchk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.500.11 Side-Effects. - None

3.2.2.500.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_param_s>
CHECK as_constrained_void with static CONTEXT>
for EACH element of as_param_s> loop
  if NOT an IN parameter> then
    ERROR  17021>
  end if;
end loop;
```

3.2.2.500.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
17021	E	illegal mode for function parameter

3.2.2.500.14 Examples of Data Structures. - None

3.2.2.501 Subprogram. - sc_param_s

3.2.2.501.1 Purpose. - Perform statement checking on a node of type 'param_s'.

3.2.2.501.2 Assumptions. - None

3.2.2.501.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.501.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.501.5 Nested Within. - scp_decl

3.2.2.501.6 Host Dependencies. - None

3.2.2.501.7 Target Dependencies. - None

3.2.2.501.8 Subprogram Visibility. - Outside Package.

3.2.2.501.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.501.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.501.11 Side-Effects. - None

3.2.2.501.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK PARAM>
end loop;
```

3.2.2.501.13 Diagnostic Messages Generated. - None

3.2.2.501.14 Examples of Data Structures. - None

3.2.2.502 Subprogram. - sc_in

3.2.2.502.1 Purpose. - Perform statement checking on a node of type 'in'.

3.2.2.502.2 Assumptions. - None

3.2.2.502.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.502.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.502.5 Nested Within. - scp_decl

3.2.2.502.6 Host Dependencies. - None

3.2.2.502.7 Target Dependencies. - None

3.2.2.502.8 Subprogram Visibility. - Outside Package.

3.2.2.502.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.502.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.502.11 Side-Effects. - None

3.2.2.502.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_type_spec with static context>
CHECK as_exp_void with static CONTEXT>
if AS_EXP_VOID not void> then
  if AS_TYPE_SPEC is limited private> then
    ERROR  17022>
  end if;
end if;
```

3.2.2.502.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17022	E	illegal parameter initialization

3.2.2.502.14 Examples of Data Structures. - None

3.2.2.503 Subprogram. - sc_out

3.2.2.503.1 Purpose. - Perform statement checking on a node of type 'out' or 'in out'.

3.2.2.503.2 Assumptions. - None

3.2.2.503.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.503.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.503.5 Nested Within. - scp_decl

3.2.2.503.6 Host Dependencies. - None

3.2.2.503.7 Target Dependencies. - None

3.2.2.503.8 Subprogram Visibility. - Outside Package.

3.2.2.503.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.503.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.503.11 Side-Effects. - None

3.2.2.503.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_type_spec>
CHECK as_exp_void>
if AS_EXP_VOID not void> then
  ERROR  17023>
end if;
```

3.2.2.503.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17023	E	illegal parameter initialization

3.2.2.503.14 Examples of Data Structures. - None

3.2.2.504 Subprogram. - sc_subp_body

3.2.2.504.1 Purpose. - Perform statement checking on a node of type 'subprogram_body'.

3.2.2.504.2 Assumptions. - None

3.2.2.504.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.504.4 Used Recursively. - This subprogram is used recursively.

3.2.2.504.5 Nested Within. - scp_decl

3.2.2.504.6 Host Dependencies. - None

3.2.2.504.7 Target Dependencies. - None

3.2.2.504.8 Subprogram Visibility. - Outside Package.

3.2.2.504.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.504.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                         node to be checked

context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.504.11 Side-Effects. - None

3.2.2.504.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_header>
CHECK as_block_stub>
```

3.2.2.504.13 Diagnostic Messages Generated. - None

3.2.2.504.14 Examples of Data Structures. - None

3.2.2.505 Subprogram. - sc_pack_decl

3.2.2.505.1 Purpose. - Perform statement checking on a node of type 'package_decl'.

3.2.2.505.2 Assumptions. - None

3.2.2.505.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.505.4 Used Recursively. - This subprogram is used recursively.

3.2.2.505.5 Nested Within. - scp_decl

3.2.2.505.6 Host Dependencies. - None

3.2.2.505.7 Target Dependencies. - None

3.2.2.505.8 Subprogram Visibility. - Outside Package.

3.2.2.505.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.505.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to be
                                         checked
context :  in  stmt_chk.context_rec; -- defines context in which this
                                        node appears
```

3.2.2.505.11 Side-Effects. - None

3.2.2.505.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
```

CHECK as_package_def>

3.2.2.505.13 Diagnostic Messages Generated. - None

3.2.2.505.14 Examples of Data Structures. - None

3.2.2.506 Subprogram. - sc_pack_spec

3.2.2.506.1 Purpose. - Perform statement checking on a node of type 'package_spec'.

3.2.2.506.2 Assumptions. - None

3.2.2.506.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.506.4 Used Recursively. - This subprogram is used recursively.

3.2.2.506.5 Nested Within. - scp_decl

3.2.2.506.6 Host Dependencies. - None

3.2.2.506.7 Target Dependencies. - None

3.2.2.506.8 Subprogram Visibility. - Outside Package.

3.2.2.506.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.506.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked

context :  in  stmt_chk.context_rec; -- defines context in which this
                                        node appears
```

3.2.2.506.11 Side-Effects. - None

3.2.2.506.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_decl_s>
CHECK as_decl_rep_s>
```

3.2.2.506.13 Diagnostic Messages Generated. - None

3.2.2.506.14 Examples of Data Structures. - None

3.2.2.507 Subprogram. - sc_subp_decl

3.2.2.507.1 Purpose. - Perform statement checking on a node of type 'subprogram_decl'.

3.2.2.507.2 Assumptions. - None

3.2.2.507.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.507.4 Used Recursively. - This subprogram is used recursively.

3.2.2.507.5 Nested Within. - scp_decl

3.2.2.507.6 Host Dependencies. - None

3.2.2.507.7 Target Dependencies. - None

3.2.2.507.8 Subprogram Visibility. - Outside Package.

3.2.2.507.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.507.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to be
                                        checked
context :  inout  context_rec; -- defines context in which this
                                  node appears
```

3.2.2.507.11 Side-Effects. - None

3.2.2.507.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
```

CHECK as_header>
CHECK as_subprogram_def>

3.2.2.507.13 Diagnostic Messages Generated. - None

3.2.2.507.14 Examples of Data Structures. - None

3.2.2.508 Subprogram. - sc_dscrm_agg

3.2.2.508.1 Purpose. - Perform statement checking on a node of type 'discriminant aggregate'.

3.2.2.508.2 Assumptions. - None

3.2.2.508.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.508.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.508.5 Nested Within. - scp_decl

3.2.2.508.6 Host Dependencies. - None

3.2.2.508.7 Target Dependencies. - None

3.2.2.508.8 Subprogram Visibility. - Outside Package.

3.2.2.508.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.508.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked
context :  inout  context-rec; -- defines context in which this
                                  node appears
```

3.2.2.508.11 Side-Effects. - None

3.2.2.508.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
          "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH element of as_list> loop
  CHECK list element>
end loop;
```

3.2.2.508.13 Diagnostic Messages Generated. - None

3.2.2.508.14 Examples of Data Structures. - None

3.2.2.509 Subprogram. - sc_rename

3.2.2.509.1 Purpose. - Perform statement checking on a node of type 'rename'.

3.2.2.509.2 Assumptions. - None

3.2.2.509.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.509.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.509.5 Nested Within. - scp_decl

3.2.2.509.6 Host Dependencies. - None

3.2.2.509.7 Target Dependencies. - None

3.2.2.509.8 Subprogram Visibility. - Outside Package.

3.2.2.509.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.509.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- refernece to diana node to
                                         be checked
context :  inout  context_rec; -- defines context in which this
                                  node appears
```

3.2.2.509.11 Side-Effects. - None

3.2.2.509.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cde_type.get_node_id(node),
         "context = ", context>
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
```

3.2.2.509.13 Diagnostic Messages Generated. - None

3.2.2.509.14 Examples of Data Structures. - None

3.2.2.510 Package. - scp_expr

3.2.2.510.1 Purpose. - Implement the statement checking function for nodes involved with expressions.

3.2.2.510.2 Number of Subprograms. - 15

3.2.2.510.3 Dependencies on Other Packages for Spec. - cdm_type, stmt_chk

3.2.2.510.4 Additional Dependencies for Body. - cdm_as, sc_class, diagnose

3.2.2.510.5 Package Specification. -

```
package scp_expr is
        procedure  sc_attr_call;
        procedure  sc_aggregate;
        procedure  sc_binary
        procedure  sc_membership;
        procedure  sc_parenthzd;
        procedure  sc_type_conv;
        procedure  sc_qualified;
        procedure  sc_allocator;
        procedure  sc_indexed;
        procedure  sc_slice;
        procedure  sc_selected;
        procedure  sc_all;
        procedure  sc_attribute;
        procedure  sc_str_lit;
        procedure  sc_fcn_call;
        procedure  sc_usec_obj;
    end scp_expr;
```

3.2.2.510.6 Elaboration Code. - None

3.2.2.510.7 Examples of Data Structures. - None

3.2.2.511 Subprogram. - sc_attr_call

3.2.2.511.1 Purpose. - Perform statement checking on a node of type 'attribute_call'.

3.2.2.511.2 Assumptions. - None

3.2.2.511.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.511.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.511.5 Nested Within. - scp_expr

3.2.2.511.6 Host Dependencies. - None

3.2.2.511.7 Target Dependencies. - None

3.2.2.511.8 Subprogram Visibility. - Outside Package.

3.2.2.511.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.511.10 <u>Formal Parameters</u>. -

```
subtree:  in com_type.c_node_ref; -- reference to diana node
                                       to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.511.11 <u>Side-Effects</u>. - None

3.2.2.511.12 <u>Algorithm</u>. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
return;
end if;
CHECK as_exp_s with static context>
```

3.2.2.511.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.511.14 <u>Examples of Data Structures</u>. - None

3.2.2.512 <u>Subprogram</u>. - sc_aggregate

3.2.2.512.1 <u>Purpose</u>. - Perform statement checking on a node of type 'aggregate'.

3.2.2.512.2 <u>Assumptions</u>. - None

3.2.2.512.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.512.4 Used Recursively. - This subprogram is used recursively.

3.2.2.512.5 Nested Within. - scp_expr

3.2.2.512.6 Host Dependencies. - None

3.2.2.512.7 Target Dependencies. - None

3.2.2.512.8 Subprogram Visibility. - Outside Package.

3.2.2.512.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.512.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  inout stmt_chk.context_rec; -- defines context in
                                        which this node appears
```

3.2.2.512.11 Side-Effects. - None

3.2.2.512.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
  return;
end if;
```

```
has_names := false;
if RECORD aggregate> then
  FIND record type and discriminate>
  for EACH element of as_list> loop
  CHECK comp_assoc>
  if NOT a named association> then
     if has_names then
        ERROR  17107>
     end if;
  end if;
  if THIS is a discriminant component> then
    ADD to dscrmt_aggregate>
    if THIS component is for a variant part> then
      if NOT static> then
        error
      end if;
    end if;
    INDICATE that this component is 'covered' (in record type)>
  end if;
end loops
SET sm_constraint to dscrmt-aggregate)
if ALL components not covered> then
   ERROR  17100>
end if;
elsif ARRAY aggregate>
  SET up coverage data structure for index type>
  for EACH element of as_list> loop
      CHECK COMP_ASSOC>
      if MORE than one component and choice expr not static> then
         ERROR  17101>
      end if;
      MERGE choice value in coverage data structure>
  end loop;
  -- find array bounds
  if THERE is an 'others' choice> then
     GET bounds from context, if any>
  elsif NAMED notation used> then
     BOUNDS := smallest index .. largest index>
  else
     BOUNDS := S'FIRST .. S'FIRST + (  of components)-1>
  end if;
  if INDEX range not covered> then
     ERROR  17102>
  end if;
```

3.2.2.512.13 <u>Diagnostic Messages Generated</u>. -

CODE	SEVERITY N.W.E.S.F	TEXT
17101	E	static expression required

17100	E	values not provided for all record components
17102	E	values not provided for all array components
17107	E	named associations must follow positional associations

3.2.2.512.14 Examples of Data Structures. - Discrete range coverage data structure:
See example for SC_VARIANT

3.2.2.513 Subprogram. - sc_binary

3.2.2.513.1 Purpose. - Perform statement checking on a node of type 'binary'.

3.2.2.513.2 Assumptions. - None

3.2.2.513.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.513.4 Used Recursively. - This subprogram is used recursively.

3.2.2.513.5 Nested Within. - scp_expr

3.2.2.513.6 Host Dependencies. - None

3.2.2.513.7 Target Dependencies. - None

3.2.2.513.8 Subprogram Visibility. - Outside Package.

3.2.2.513.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.513.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana
                                     node to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.513.11 Side-Effects. - None

3.2.2.513.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
  return;
end if;
CHECK as_exp1>
CHECK as_exp2>
```

3.2.2.513.13 Diagnostic Messages Generated. - None

3.2.2.513.14 Examples of Data Structures. - None

3.2.2.514 Subprogram. - sc_membershp

3.2.2.514.1 Purpose. - Perform statement checking on a node of type 'membership'.

3.2.2.514.2 Assumptions. - None

3.2.2.514.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.514.4 Used Recursively. - This subprogram is used recursively.

3.2.2.514.5 Nested Within. - scp_expr

3.2.2.514.6 Host Dependencies. - None

3.2.2.514.7 Target Dependencies. - None

3.2.2.514.8 Subprogram Visibility. - Outside Package.

3.2.2.514.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.514.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node-ref; -- reference to diana node
                                        to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.514.11 Side-Effects. - None

3.2.2.514.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
```

```
end if;
if THIS node has been flagged as in error> then
  return;
end if;
CHECK as_exp>
CHECK as_type_range>
```

3.2.2.514.13 Diagnostic Messages Generated. - None

3.2.2.514.14 Examples of Data Structures. - None

3.2.2.515 Subprogram. - sc_type_conv

3.2.2.515.1 Purpose. - Perform statement checking on a node of type 'type_conversion'.

3.2.2.515.2 Assumptions. - None

3.2.2.515.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.515.4 Used Recursively. - This subprogram is used recursively.

3.2.2.515.5 Nested Within. - scp_expr

3.2.2.515.6 Host Dependencies. - None

3.2.2.515.7 Target Dependencies. - None

3.2.2.515.8 Subprogram Visibility. - Outside Package.

3.2.2.515.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.515.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                       to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.515.11 Side-Effects. - None

3.2.2.515.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has  been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_exp>
if SM_EXP_TYPE is an array type> then
  if CONSTRAINED array. then
    CHECK that each element of the sm_exp_type has a matching
    component in the as_exp>
  else
    CHANGE sm_exp_type to 'constrained ' and copy bounds
     of as_exp array>
  end if;
end if;
```

3.2.2.515.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17103	E	unmatched components in array type coversion

3.2.2.515.14 Examples of Data Structures. - None

3.2.2.516 Subprogram. - sc_qualified

3.2.2.516.1 Purpose. - Perform statement checking on a node of type 'qualified'.

3.2.2.516.2 Assumptions. - None

3.2.2.516.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.516.4 Used Recursively. - This subprogram is used recursively.

3.2.2.516.5 Nested Within. - scp_expr

3.2.2.516.6 Host Dependencies. - None

3.2.2.516.7 Target Dependencies. - None

3.2.2.516.8 Subprogram Visibility. - Outside Package.

3.2.2.516.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.516.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana node
                                     to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                     this node appears
```

3.2.2.516.11 Side-Effects. - None

3.2.2.516.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
        "context =", context>
end if;
if THIS node has been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_exp>
```

3.2.2.516.13 Diagnostic Messages Generated. - None

3.2.2.516.14 Examples of Data Structures. - None

3.2.2.517 Subprogram. - sc_allocator

3.2.2.517.1 Purpose. - Perform statement checking on a node of type 'allocator'.

3.2.2.517.2 Assumptions. - None

3.2.2.517.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.517.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.517.5 Nested Within. - scp_expr

3.2.2.517.6 Host Dependencies. - None

3.2.2.517.7 Target Dependencies. - None

3.2.2.517.8 Subprogram Visibility. - Outside Package.

3.2.2.517.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.517.10 Formal Parameters. -

```
subtree  :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked

context  :  in stmt_chk.context_rec; -- defines context in which
                                         this node appears
```

3.2.2.517.11 Side-Effects. - None

3.2.2.517.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_access_constraint>
if SM_TYPE_SPEC denotes an unconstrained private, record or
    array type> then
  if CONSTRAINT not present> then
    ERROR  17104>
  end if;
end if;
```

```
if SM_TYPE_SPEC denotes an unconstrained array> then
  if ARRAY is "ragged"> then
     ERROR  17105>
  end if;
end if;
```

3.2.2.517.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17104	E	constraint required for allocator
17105	E	illegal array structure

3.2.2.517.14 Examples of Data Structures. - None

3.2.2.518 Subprogram. - sc_slice

3.2.2.518.1 Purpose. - Perform statement checking on a node of type 'slice'.

3.2.2.518.2 Assumptions. - None

3.2.2.518.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.518.4 Used Recursively. - This subprogram is used recursively.

3.2.2.518.5 Nested Within. - scp_expr

3.2.2.518.6 Host Dependencies. - None

3.2.2.518.7 Target Dependencies. - None

3.2.2.518.8 Subprogram Visibility. - Outside Package.

3.2.2.518.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.518.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node to be
                                        checked

context :  inout  context_rec; -- defines context in which this node
                                  appears
```

3.2.2.518.11 Side-Effects. - None

3.2.2.518.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
   return;
end if;
CHECK as_name>
CHECK as_dscrt_range>
SET sm_constraint attribute to constraint created by slice>
```

3.2.2.518.13 Diagnostic Messages Generated. - None

3.2.2.518.14 Examples of Data Structures. - None

3.2.2.519 Subprogram. - sc_selected

3.2.2.519.1 Purpose. - Perform statement checking on a node of type 'selected'.

3.2.2.519.2 Assumptions. - None

3.2.2.519.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.519.4 Used Recursively. - This subprogram is used recursively.

3.2.2.519.5 Nested Within. - scp_expr

3.2.2.519.6 Host Dependencies. - None

3.2.2.519.7 Target Dependencies. - None

3.2.2.519.8 Subprogram Visibility. - Outside Package.

3.2.2.519.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.519.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                          be checked

context :  inout  context_rec; -- defines context in which this node
                                   appears
```

3.2.2.519.11 Side-Effects. - None

3.2.2.519.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
   return;
end if;
CHECK as_name>
```

3.2.2.519.13 Diagnostic Messages Generated. - None

3.2.2.519.14 Examples of Data Structures. - None

3.2.2.520 Subprogram. - sc_all

3.2.2.520.1 Purpose. - Perform statement checking on a node of type 'all'.

3.2.2.520.2 Assumptions. - None

3.2.2.520.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.520.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.520.5 Nested Within. - scp_expr

3.2.2.520.6 <u>Host Dependencies</u>. - None

3.2.2.520.7 <u>Target Dependencies</u>. - None

3.2.2.520.8 <u>Subprogram Visibility</u>. - Outside Package.

3.2.2.520.9 <u>Function/Procedure</u>. - This subprogram is a Procedure.

3.2.2.520.10 <u>Formal Parameters</u>. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked

context :  inout  context_rec; -- defines context in which this node
                                  appears
```

3.2.2.520.11 <u>Side-Effects</u>. - None

3.2.2.520.12 <u>Algorithm</u>. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
   return;
end if;
CHECK as_name>
CHECK as_designator>
```

3.2.2.520.13 <u>Diagnostic Messages Generated</u>. - None

3.2.2.520.14 Examples of Data Structures. - None

3.2.2.521 Subprogram. - sc_attribute

3.2.2.521.1 Purpose. - Perform statement checking on a node of type 'attribute'.

3.2.2.521.2 Assumptions. - None

3.2.2.521.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.521.4 Used Recursively. - This subprogram is used recursively.

3.2.2.521.5 Nested Within. - scp_expr

3.2.2.521.6 Host Dependencies. - None

3.2.2.521.7 Target Dependencies. - None

3.2.2.521.8 Subprogram Visibility. - Outside Package.

3.2.2.521.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.521.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node-ref; -- referance to diana node to
                                          be checked

context :  inout  context_rec; -- defines context in which this
                                   node appears
```

3.2.2.521.11 Side-Effects. - None

3.2.2.521.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
   return;
end if;
CHECK as_name>
CHECK as_id>
```

3.2.2.521.13 Diagnostic Messages Generated. - None

3.2.2.521.14 Examples of Data Structures. - None

3.2.2.522 Subprogram. - sc_str_lit

3.2.2.522.1 Purpose. - Perform statement checking on a node of type 'string literal'.

3.2.2.522.2 Assumptions. - None

3.2.2.522.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.522.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.522.5 Nested Within. - scp_expr

3.2.2.522.6 Host Dependencies. - None

3.2.2.522.7 Target Dependencies. - None

3.2.2.522.8 Subprogram Visibility. - Outside Package.

3.2.2.522.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.522.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to be
                                        checked

context :  inout  context_rec; -- defines context in which this node
                                  appears
```

3.2.2.522.11 Side-Effects. - None

3.2.2.522.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has  been flagged as in error> then
   return;
end if;
CREATE Diana nodes of form:
SET sm_constraint to these nodes>
```

3.2.2.522.13 Diagnostic Messages Generated. - None

3.2.2.522.14 Examples of Data Structures. - None

3.2.2.523 Subprogram. - sc_fcn_call

3.2.2.523.1 Purpose. - Perform statement checking on a node of type 'function_call'.

3.2.2.523.2 Assumptions. - None

3.2.2.523.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.523.4 Used Recursively. - This subprogram is used recursively.

3.2.2.523.5 Nested Within. - scp_expr

3.2.2.523.6 Host Dependencies. - None

3.2.2.523.7 Target Dependencies. - None

3.2.2.523.8 Subprogram Visibility. - Outside Package.

3.2.2.523.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.523.10 <u>Formal Parameters</u>. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked

context :  in  stmt_chk.context_rec; -- defines context in which this

                                       node appears
```

3.2.2.523.11 <u>Side-Effects</u>. - None

3.2.2.523.12 <u>Algorithm</u>. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has been flagged as in error> then
   return;
end if;
if THIS function is generic> then
   ERROR  17106>
end if;
CHECK as_param_assoc_s>
-- compute elaboration order information
ADD (containing_procedure_name calls as_name) to call_relation list>
if THIS is an elaboration context> then
   ADD as_name to elaboration_call list>
end if;
```

3.2.2.523.13 <u>Diagnostic Messages Generated</u> -

	SEVERITY	
CODE	N.d.E.S.F	TEXT
17106	E	generic function called

3.2.2.523.14 Examples of Data Structures. - None

3.2.2.524 Subprogram. - sc_indexed

3.2.2.524.1 Purpose. - Perform statement checking on a node of type 'indexed'.

3.2.2.524.2 Assumptions. - None

3.2.2.524.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.524.4 Used Recursively. - This subprogram is used recursively.

3.2.2.524.5 Nested Within. - scp_expr

3.2.2.524.6 Host Dependencies. - None

3.2.2.524.7 Target Dependencies. - None

3.2.2.524.8 Subprogram Visibility. - Within Package Only.

3.2.2.524.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.524.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                        to be checked

context :  inout  context-rec; -- defines context in which this
                                  node appears
```

3.2.2.524.11 Side-Effects. - None

3.2.2.524.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_exp_s>
```

3.2.2.524.13 Diagnostic Messages Generated. - None

3.2.2.524.14 Examples of Data Structures. - None

3.2.2.525 Subprogram. - sc_used_obj

3.2.2.525.1 Purpose. - Perform statement checking on a node of type 'used_object_id'.

3.2.2.525.2 Assumptions. - None

3.2.2.525.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.525.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.525.5 Nested within. - scp_expr

3.2.2.525.6 Host Dependencies. - None

3.2.2.525.7 Target Dependencies. - None

3.2.2.525.8 Subprogram Visibility. - Within Package Only.

3.2.2.525.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.525.10 Formal Parameters. -

```
subtree : in  cdm_type.c_node_ref; -- reference to diana node to be
                                      checked
context : in out  context_rec; -- defines context in which this node appears
```

3.2.2.525.11 Side-Effects. - None

3.2.2.525.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
   return;
end if;
INVOKE cross-reference recorder>
```

3.2.2.525.13 Diagnostic Messages Generated. - None

3.2.2.525.14 Examples of Data Structures. - None

3.2.2.526 Package. - scp_stmt

3.2.2.526.1 Purpose. - Implement the statement checking function for nodes involved with statements.

3.2.2.526.2 Number of Subprograms. - 24

3.2.2.526.3 Dependencies on Other Packages for Spec. - cdm_type, stmtchk

3.2.2.526.4 Additional Dependencies for Body. - cdm_as, sc_class, diagnose

3.2.2.526.5 Package Specification. -

```
package scp_stmt is
    procedure  sc_stmt_s;
    procedure  sc_labeled:
    procedure  sc_assign;
    procedure  sc_if;
    procedure  sc_cond_cl
    procedure  sc_case;
    procedure  sc_named_stm;
    procedure  sc_loop;
    procedure  sc_for_rev;
    procedure  sc_block;
    procedure  sc_exit;
    procedure  sc_return;
    procedure  sc_entry_cal;
    procedure  sc_accept;
    procedure  sc_delay;
    procedure  sc_select;
    procedure  sc_cond_entr;
    procedure  sc_timed_ent;
    procedure  sc_abort;
    procedure  sc_raise;
    procedure  sc_code;
    procedure  sc_goto;
    procedure sc_proc_call;
    procedure sc_terminate;
    end scp-stmt;
```

3.2.2.526.6 Elaboration Code. - None

3.2.2.526.7 Examples of Data Structures. - None

3.2.2.527 Subprogram. - sc_stm_s

3.2.2.527.1 Purpose. - Perform statement checking on a node of type 'stm_s'.

3.2.2.527.2 Assumptions. - None

3.2.2.527.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.527.4 Used Recursively. - This subprogram is used recursively.

3.2.2.527.5 Nested Within. - scp_stmt

3.2.2.527.6 Host Dependencies. - None

3.2.2.527.7 Target Dependencies. - None

3.2.2.527.8 Subprogram Visibility. - Outside Package.

3.2.2.527.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.527.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference diana node
                                          to be checked
context :  in out stmtchk.context_rec; -- defines context
                                           in which this
                                           node appears
```

3.2.2.527.11 Side-Effects. - The statement sequence list data structure is modified: new sequence list.

3.2.2.527.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CREATE new element of statement sequence list>
--complete info. for statement sequence data structure
if FIRST statement is a code stmt> then
  SET has code flag for this sequence>
else
  SET has_non_code flag for this sequence>
end if;
ifIN a select statement> then
  UPDATE   accepts,  delays,  terminates for select statements
   if first statement is accept,delay or terminate>
end if;
for EACH element of as_list>loop
  CHECK STM>
end loop;
if THIS sequence has "unclaimed" goto statements>then
  ERROR  17200 for each goto>
end if;
ifTHIS is a function body> then
  ifNO return stmt>then
   ERROR  17201>
  end if;
end if;
PROPAGATE "unchecked goto's and return_stmt flags>
DESTROY list for this seq>
```

3.2.2.527.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17200	E	invalid 'goto' destination
17201	E	no return statement in function body

3.2.2.527.14 Examples of Data Structures. - None

3.2.2.528 Subprogram. - sc_labeled

3.2.2.528.1 Purpose. - Perform statement checking on a node of type 'labeled'.

3.2.2.528.2 Assumptions. - None

3.2.2.528.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.528.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.528.5 Nested Within. - scp_stmt

3.2.2.528.6 Host Dependencies. - None

3.2.2.528.7 Target Dependencies. - None

3.2.2.528.8 Subprogram Visibility. - Outside Package.

3.2.2.528.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.528.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                        to be checked

context :  in  stmt_chk.context_rec; -- defined context in
                                       which this node
                                       appears
```

3.2.2.528.11 Side-Effects. - Label information is added to the list_of_statements_sequences data structure.

3.2.2.528.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
ADD as-id to list of labels for the current sequence>
CHECK as_stm>
```

3.2.2.528.13 Diagnostic Messages Generated. - None

3.2.2.528.14 Examples of Data Structures. - None

3.2.2.529 Subprogram. - sc_assign

3.2.2.529.1 Purpose. - Perform statement checking on a node of type 'assign'.

3.2.2.529.2 Assumptions. - None

3.2.2.529.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.529.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.529.5 Nested within. - scp_stmt

3.2.2.529.6 Host Dependencies. - None

3.2.2.529.7 Target Dependencies. - None

3.2.2.529.8 Subprogram Visibility. - Outside Package.

3.2.2.529.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.529.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana
                                         node to be checked

context :  in  stmt_chk.context_rec;  -- defines context
                                         in which this node
                                         appears
```

3.2.2.529.11 Side-Effects. - None

3.2.2.529.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_exp>
if TARGET of assignment is illegal*> then
  ERROR  17202>
end if;
if ARRAY assignments> then
  if   of components does not match> then
    ERROR  17203>
  end if;
end if;
```

* constant, task, in parameter, loop parm, discriminant, lim_private type, record with lim_private type as a component (except in corresp. body).

3.2.2.529.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
17202	E	invalid assignment target
17203	E	components do not match in array assignment.

3.2.2.529.14 Examples of Data Structures. - None

3.2.2.530 Subprogram. - sc_if

3.2.2.530.1 Purpose. - Perform statement checking on a node of type 'if'.

3.2.2.530.2 Assumptions. - None

3.2.2.530.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.530.4 Used Recursively. - This subprogram is used recursively.

3.2.2.530.5 Nested Within. - scp_stmt

3.2.2.530.6 Host Dependencies. - None

3.2.2.530.7 Target Dependencies. - None

3.2.2.530.8 Subprogram Visibility. - Outside Package.

3.2.2.530.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.530.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; --  reference to diana
                                          to be checked

context :  in stmt_chk.context_rec; -- defines context
                                      in which this node
                                      appears
```

3.2.2.530.11 Side-Effects. - None

3.2.2.530.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return:
end if;
for EACH element of as_list>loop
  CHECK cond_clause>
end loop;
```

3.2.2.530.13 Diagnostic Messages Generated. - None

3.2.2.530.14 Examples of Data Structures. - None

3.2.2.531 Subprogram. - sc_cond_cl

3.2.2.531.1 Purpose. - Perform statement checking on a node of type 'cond_clause'.

3.2.2.531.2 Assumptions. - None

3.2.2.531.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.531.4 Used Recursively. - This subprogram is used recursively.

3.2.2.531.5 Nested Within. - scp_stmt

3.2.2.531.6 Host Dependencies. - None

3.2.2.531.7 Target Dependencies. - None

3.2.2.531.8 Subprogram Visibility. - Outside Package.

3.2.2.531.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.531.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.531.11 Side-Effects. - None

3.2.2.531.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp_void>
CHECK as_stm_s>
```

3.2.2.531.13 Diagnostic Messages Generated. - None

3.2.2.531.14 Examples of Data Structures. - None

3.2.2.532 Subprogram. - sc_case

3.2.2.532.1 Purpose. - Perform statement checking for a node of type 'case'.

3.2.2.532.2 Assumptions. - None

3.2.2.532.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.532.4 Used Recursively. - This subprogram is used recursively.

3.2.2.532.5 Nested Within. - scp_stmt

3.2.2.532.6 Host Dependencies. - None

3.2.2.532.7 Target Dependencies. - None

3.2.2.532.8 Subprogram Visibility. - Outside Package.

3.2.2.532.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.532.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                       node to be checked
context:  in context_rec; -- defines context in which this node
                             appears
```

3.2.2.532.11 Side-Effects. - None

3.2.2.532.12 Algorithm. -

```
if NODE already marked as in error> then
  return;
end if;
CHECK as_exp>
if NOT a discrete type> then
  ERROR  17204>
end if;
INITIALIZE discrete range coverage data structure>
for EACH element of as_alternative_s>loop
  if OTHERS has appeared> then
   ERROR  17205>
  end if;
  for EACH element of as_choice_s>loop
      ifCHOICE expression is not static> then
      ERROR  17207>
    end if;
    INCLUDE this choice in coverage data structure>
    if THERE is duplicate coverage> then
      ERROR  17208>
    end if;
  end loop;
end loop;
if DISCRETE range is not entireley covered> then
  ERROR  17209>
end if;
```

3.2.2.532.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17204	E	non_discrete type in case expression
17207	E	non_static choice expression

17208	E	duplicate choice values in case statement
17205	E	'others' choice misplaces
17209	E	discrete range not covered

3.2.2.532.14 Examples of Data Structures. -
Discrete range coverage data structure:
See example for SC_VARIANT.

3.2.2.533 Subprogram. - sc_named_stm

3.2.2.533.1 Purpose. - Perform statement checking on a node of type 'named_stm'.

3.2.2.533.2 Assumptions. - None

3.2.2.533.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.533.4 Used Recursively. - This subprogram is used recursively.

3.2.2.533.5 Nested within. - scp_stmt

3.2.2.533.6 Host Dependencies. - None

3.2.2.533.7 Target Dependencies. - None

3.2.2.533.8 Subprogram Visibility. - Outside Package.

3.2.2.533.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.533.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context
                                       in which this node
                                       appears
```

3.2.2.533.11 Side-Effects. - None

3.2.2.533.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_stm>
```

3.2.2.533.13 Diagnostic Messages Generated. - None

3.2.2.533.14 Examples of Data Structures. - None

3.2.2.534 Subprogram. - sc_loop

3.2.2.534.1 Purpose. - Perform statement checking on a node of type 'loop'.

3.2.2.534.2 Assumptions. - None

3.2.2.534.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.534.4 Used Recursively. - This subprogram is used recursively.

3.2.2.534.5 Nested within. - scp_stmt

3.2.2.534.6 Host Dependencies. - None

3.2.2.534.7 Target Dependencies. - None

3.2.2.534.8 Subprogram Visibility. - Outside Package.

3.2.2.534.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.534.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                       node to be checked

context :  in stmt_chk.context_rec; -- defines context in
                                      which this node
                                      appears
```

3.2.2.534.11 Side-Effects. - None

3.2.2.534.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
           "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_iteration>
CHECK as stm_s>
```

3.2.2.534.13 Diagnostic Messages Generated. - None

3.2.2.534.14 Examples of Data Structures. - None

3.2.2.535 Subprogram. - sc_for_rev

3.2.2.535.1 Purpose. - Perform statement checking on a node of type 'for' or 'reverse'.

3.2.2.535.2 Assumptions. - None

3.2.2.535.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.535.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.535.5 Nested Within. - scp-stmt

3.2.2.535.6 Host Dependencies. - None

3.2.2.535.7 Target Dependencies. - None

3.2.2.535.8 Subprogram Visibility. - Outside Package.

3.2.2.535.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.535.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        to be checked

context :  in stmt_chk.context_rec; -- defines context in
                                       which this node appears
```

3.2.2.535.11 Side-Effects. - None

3.2.2.535.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_dscrt_range with static_required CONTEXT>
if LOOP parameter not of a discrete type> then
  ERROR  17210>
end if;
```

3.2.2.535.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
17210	E	non_discrete type for loop parameter

3.2.2.535.14 Examples of Data Structures. - None

3.2.2.536 Subprogram. - sc_block

3.2.2.536.1 Purpose. - Perform statement checking on a node of type 'block'.

3.2.2.536.2 Assumptions. - None

3.2.2.536.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.536.4 Used Recursively. - This subprogram is used recursively.

3.2.2.536.5 Nested Within. - scp_stmt

3.2.2.536.6 Host Dependencies. - None

3.2.2.536.7 Target Dependencies. - None

3.2.2.536.8 Subprogram Visibility. - Outside Package.

3.2.2.536.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.536.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                      node to be checked
context :  in stmt_chk.context_rec; -- defines context in
                                      which this node appears
```

3.2.2.536.11 Side-Effects. - None

3.2.2.536.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_item_s>
CHECK as_stm_s>
CHECK as_alternative_s>
```

3.2.2.536.13 Diagnostic Messages Generated. - None

3.2.2.536.14 Examples of Data Structures. - None

3.2.2.537 Subprogram. - sc_exit

3.2.2.537.1 Purpose. - Perform statement checking on a node of type 'exit'.

3.2.2.537.2 Assumptions. - None

3.2.2.537.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.537.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.537.5 Nested Within. - scp_stmt

3.2.2.537.6 Host Dependencies. - None

3.2.2.537.7 Target Dependencies. - None

3.2.2.537.8 Subprogram Visibility. - Outside Package.

3.2.2.537.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.537.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context
                                       in which this node
                                       appears
```

3.2.2.537.11 Side-Effects. - None

3.2.2.537.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_exp_void>
if AS_NAME_VOID is void> then
  if NOT in a loop sequence of statements> then
    ERROR  17211>
  end if;
end if;
```

3.2.2.537.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17211	E	'exit' must be from a loop

3.2.2.537.14 Examples of Data Structures. - None

3.2.2.538 Subprogram. - sc_return

3.2.2.538.1 Purpose. - Perform statement checking on a node of type 'return'.

3.2.2.538.2 Assumptions. - None

3.2.2.538.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.538.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.538.5 Nested within. - scp_stmt

3.2.2.538.6 Host Dependencies. - None

3.2.2.538.7 Target Dependencies. - None

3.2.2.538.8 Subprogram Visibility. - Outside Package.

3.2.2.538.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.538.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context in
                                       which this node appears
```

3.2.2.538.11 Side-Effects. - Statement sequence list is modified.

3.2.2.538.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT in a subprogram or accept statement> then
   ERROR 17227>
end if;
CHECK as_exp_void>
SET return_stmt_present flag in stmt_sequence node>
```

3.2.2.538.13 Diagnostic Messages Generated. - None

3.2.2.538.14 Examples of Data Structures. - None

3.2.2.539 Subprogram. - sc_goto

3.2.2.539.1 Purpose. - Perform statement checking on a node of type 'goto'.

3.2.2.539.2 Assumptions. - None

3.2.2.539.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.539.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.539.5 Nested Within. - scp_stmt

3.2.2.539.6 Host Dependencies. - None

3.2.2.539.7 Target Dependencies. - None

3.2.2.539.8 Subprogram Visibility. - Outside Package.

3.2.2.539.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.539.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana node
                                        to be checked
context :  in stmt_chk.context_rec; -- defines context
                                       in which this node
                                       appears
```

3.2.2.539.11 Side-Effects. - None

3.2.2.539.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
```

```
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH sequence_of_statements from innermost to
     first exception or outermost> loop
  if AS_NAME is in label list> then
    return;  --ok
  end if;
end loop;
ADD as_name to list of unchecked goto's for the current sequence
 of statements>
```

3.2.2.539.13 Diagnostic Messages Generated. - None

3.2.2.539.14 Examples of Data Structures. - None

3.2.2.540 Subprogram. - sc_entry_cal

3.2.2.540.1 Purpose. - Perform statement checking on a node of type 'entry_call'.

3.2.2.540.2 Assumptions. - None

3.2.2.540.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.540.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.540.5 Nested Within. - scp_stmt

3.2.2.540.6 Host Dependencies. - None

3.2.2.540.7 Target Dependencies. - None

3.2.2.540.8 Subprogram Visibility. - Outside Package.

3.2.2.540.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.540.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context
                                       in which this node
                                       appears
```

3.2.2.540.11 Side-Effects. - None

3.2.2.540.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_param_assoc_s>
```

3.2.2.540.13 Diagnostic Messages Generated. - None

3.2.2.540.14 Examples of Data Structures. - None

3.2.2.541 Subprogram. - sc_accept

3.2.2.541.1 Purpose. - Perform statement checking on a node of type 'accept'.

3.2.2.541.2 Assumptions. - None

3.2.2.541.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.541.4 Used Recursively. - This subprogram is used recursively.

3.2.2.541.5 Nested Within. - scp-stmt

3.2.2.541.6 Host Dependencies. - None

3.2.2.541.7 Target Dependencies. - None

3.2.2.541.8 Subprogram Visibility. - Outside Package.

3.2.2.541.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.541.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context in
                                       which this node appears
```

3.2.2.541.11 Side-Effects. - None

3.2.2.541.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_name>
CHECK as_param_s>
CHECK as_stm_s>
if NOT in a task body> then
  ERROR 17224>
end if;
if IDENTIFIER on end does not match as_name> then
   ERROR 17228>
end if;
```

3.2.2.541.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17224	E	accept statement does not appear in a task body
17228	E	end identifier does not match entry name

3.2.2.541.14 Examples of Data Structures. - None

3.2.2.542 Subprogram. - sc_delay

3.2.2.542.1 Purpose. - Perform statement checking on a node of type 'delay'.

3.2.2.542.2 Assumptions. - None

3.2.2.542.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.542.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.542.5 Nested within. - scp_stmt

3.2.2.542.6 Host Dependencies. - None

3.2.2.542.7 Target Dependencies. - None

3.2.2.542.8 Subprogram Visibility. - Outside Package.

3.2.2.542.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.542.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_cnk.context_rec; -- defines context in
                                       which this node appears
```

3.2.2.542.11 Side-Effects. - None

3.2.2.542.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
```

```
  return;
end if;
CHECK as_exp>
if NOT in a task body sequence> then
  ERROR  17212>
end if;
```

3.2.2.542.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17212	E	delay statement does not appear in a task body

3.2.2.542.14 Examples of Data Structures. - None

3.2.2.543 Subprogram. - sc_select

3.2.2.543.1 Purpose. - Perform statement checking on a node of type 'select'.

3.2.2.543.2 Assumptions. - None

3.2.2.543.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.543.4 Used Recursively. - This subprogram is used recursively.

3.2.2.543.5 Nested Within. - scp_stmt

3.2.2.543.6 Host Dependencies. - None

3.2.2.543.7 Target Dependencies. - None

3.2.2.543.8 Subprogram Visibility. - Outside Package.

3.2.2.543.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.543.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                        node to be checked
context :  in stmt_chk.context_rec; -- defines context in
                                       which this node appears
```

3.2.2.543.11 Side-Effects. - None

3.2.2.543.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_cond_clause>
CHECK as_stm_s>
if NUMBER of terminates >1> then
  ERROR  17213>
end if;
if NUMBER of terminates >0 and   of delays >0 > then
  ERROR  17214>
end if;
if AS_STM_S not empty> then --else present
  if NUMBER of terminates >0 or   of delays >0> then
    ERROR  17215>
  end if;
end if;
if NUMBER of terminates =1 and context is an inner block with task
    objects> then
  ERROR  17216>
end if;
if ENCLOSING unit not a task body> then
```

```
  ERROR  17217>
end if;
if NUMBER of accepts = 0> then
   ERROR  17218>
end if;
```

3.2.2.543.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.E	TEXT
17213	E	more than one terminate in a select statement
17214	E	terminate and delay in same select statement
17215	E	term. or del. present with 'else' in select stmt
17216	E	term. appears in inner block which decl. body task
17217	E	select statement does not appear in a task body
17218	E	at least one acccept statement required

3.2.2.543.14 Examples of Data Structures. - None

3.2.2.544 Subprogram. - sc_cond_entr

3.2.2.544.1 Purpose. - Perform statement checking on a node of type 'cond_entry'.

3.2.2.544.2 Assumptions. - None

3.2.2.544.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.544.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.544.5 Nested within. - scp_stmt

3.2.2.544.6 Host Dependencies. - None

3.2.2.544.7 Target Dependencies. - None

3.2.2.544.8 Subprogram Visibility. - Outside Package.

3.2.2.544.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.544.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                      node to be checked

context:  in stmt_chk.context_rec; -- defines context in which
                                      this node appears
```

3.2.2.544.11 Side-Effects. - None

3.2.2.544.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_stm_s1>
CHECK as_stm_s2>
if FIRST stmt of s1 not an entry call > then
  ERROR  17219>
end if;
```

```
if ENCLOSING unit is not a task body> then
   ERROR  17227>
end if;
```

3.2.2.544.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17219	E	first stmt in select sequence is not an entry call
17227	E	conditional entry does not appear in a task body

3.2.2.544.14 Examples of Data Structures. - None

3.2.2.545 Subprogram. - sc_timed_ent

3.2.2.545.1 Purpose. - Perform statement checking on a node of type 'timed entry'.

3.2.2.545.2 Assumptions. - None

3.2.2.545.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.545.4 Used Recursively. - This subprogram is used recursively.

3.2.2.545.5 Nested Within. - scp_stmt

3.2.2.545.6 Host Dependencies. - None

3.2.2.545.7 Target Dependencies. - None

3.2.2.545.8 Subprogram Visibility. - Outside Package.

3.2.2.545.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.545.10 Formal Parameters. -

```
subtree:  in cdm_type.c_node_ref; -- reference to diana
                                     node to be checked

context:  in stmt_chk.context_rec; -- defines context in which this
                                    node appears
```

3.2.2.545.11 Side-Effects. - None

3.2.2.545.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_stm_s1>
CHECK as-stm_s2>
if FIRST statement of s1 not an entry call> then
 ERROR  17220>
end if;
if FIRST statement of s2 not a delay> then
  ERROR  17221>
end if;
```

3.2.2.545.13 Diagnostic Messages Generated. -

CODE	SEVERITY N,W,E,S,F	TEXT
17220	E	entry call expected

17221 E delay statement expected

3.2.2.545.14 Examples of Data Structures. - None

3.2.2.546 Subprogram. - sc_abort

3.2.2.546.1 Purpose. - Perform statement checking on a node of type 'abort'.

3.2.2.546.2 Assumptions. - None

3.2.2.546.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.546.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.546.5 Nested Within. - scp_stmt

3.2.2.546.6 Host Dependencies. - None

3.2.2.546.7 Target Dependencies. - None

3.2.2.546.8 Subprogram Visibility. - Outside Package.

3.2.2.546.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.546.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana
                                       node to be checked

context :  in stmt_chk.context_rec; -- defines context in
                                      which this node
                                      appears
```

3.2.2.546.11 Side-Effects. - None

3.2.2.546.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
for EACH name in as_name_s> loop
  CHECK as_name>
  if NOT a task name> then
    ERROR  17222>
  end if;
end loop;
```

3.2.2.546.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
17222	E	task name required in abort statement

3.2.2.546.14 Examples of Data Structures. - None

3.2.2.547 Subprogram. - sc_code

3.2.2.547.1 Purpose. - Perform statement checking on a node of type 'code'.

3.2.2.547.2 Assumptions. - None

3.2.2.547.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.547.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.547.5 Nested Within. - scp_stmt

3.2.2.547.6 Host Dependencies. - None

3.2.2.547.7 Target Dependencies. - None

3.2.2.547.8 Subprogram Visibility. - Outside Package.

3.2.2.547.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.547.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                        to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                         this node appears
```

3.2.2.547.11 Side-Effects. - None

3.2.2.547.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
if THIS statement sequence has non_code> statements then
   ERROR  17223>
end if;
```

3.2.2.547.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17223	E	code statements mixed with non_code statements

3.2.2.547.14 Examples of Data Structures. - None

3.2.2.548 Subprogram. - sc_raise

3.2.2.548.1 Purpose. - Perform statement checking on a node of type 'raise'.

3.2.2.548.2 Assumptions. - None

3.2.2.548.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.548.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.548.5 Nested Within. - scp_stmt

3.2.2.548.6 Host Dependencies. - None

3.2.2.548.7 Target Dependencies. - None

3.2.2.548.8 Subprogram Visibility. - Outside Package.

3.2.2.548.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.548.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                        to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.548.11 Side-Effects. - None

3.2.2.548.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
if AS_NAME_VOID is void> then
  if ENCLOSING construct is not an exception handler> then
     ERROR  17224>
  end if;
end if;
```

3.2.2.548.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17224	E	unnamed raise stmt not appear in exception handler

3.2.2.548.14 Examples of Data Structures. - None

3.2.2.549 Subprogram. - sc_terminate

3.2.2.549.1 Purpose. - Perform statement checking on a node of type 'terminate'.

3.2.2.549.2 Assumptions. - None

3.2.2.549.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.549.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.549.5 Nested Within. - scp_stmt

3.2.2.549.6 Host Dependencies. - None

3.2.2.549.7 Target Dependencies. - None

3.2.2,549.8 Subprogram Visibility. - Within Package Only.

3.2.2.549.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.549.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                       be checked

context :  inout  context_rec; -- defines context in which this node
                                  appears
```

3.2.2.549.11 Side-Effects. - None

3.2.2.549.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
         "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT in a task body> then
  ERROR 17226>
end if;
```

3.2.2.549.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N.W.E.S.F	TEXT
17226	E	'terminate' not in a task body

3.2.2.549.14 Examples of Data Structures. - None

3.2.2.550 Subprogram. - sc_proc_call

3.2.2.550.1 Purpose. - Perform statement checking on a node of type 'procedure_call'.

3.2.2.550.2 Assumptions. - None

3.2.2.550.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.550.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.550.5 Nested Within. - scp_stmt

3.2.2.550.6 Host Dependencies. - None

3.2.2.550.7 Target Dependencies. - None

3.2.2.550.8 Subprogram Visibility. - Outside Package.

3.2.2.550.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.550.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked

context :  in  stmt_chk.context_rec; -- defines context in which this
                                        node appears
```

3.2.2.550.11 Side-Effects. - None

3.2.2.550.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node, "of type", cdm_type.get_node_id(node),
          "context =", context>
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_param_assoc_s>
if THIS procedure is generic> then
  ERROR 17225>
end if;
-- compute elaboration order information
ADD (containing_procedure_name calls as_name) to call_relation list>
if THIS is an elaboration context> then
   ADD as_name to elaboration_call list>
end if;
```

3.2.2.550.13 Diagnostic Messages Generated. -

	SEVERITY	
CODE	N,W,E,S,F	TEXT
17225	E	generic procedure called

3.2.2.550.14 Examples of Data Structures. - None

3.2.2.551 Package. - scp_prag

3.2.2.551.1 Purpose. - Implement the statement checking function for pragmas.

3.2.2.551.2 Number of Subprograms. - 7

3.2.2.551.3 Dependencies on Other Packages for Spec. - cdm_type,stmt_chk

3.2.2.551.4 Additional Dependencies for Body. - cdm_as, scp_tdeo, sc_class, diagnose

3.2.2.551.5 Package Specification. -

```
package scp_prag is
  procedure sc_pragma;
  procedure sc_title;
  procedure sc_mem_size;
  procedure sc_optimize;
  procedure sc_priority;
  procedure sc_strg_unit;
  procedure sc_system;
end scp_prag;
```

3.2.2.551.6 Elaboration Code. - None

3.2.2.551.7 Examples of Data Structures. - None

3.2.2.552 Subprogram. - sc_pragma

3.2.2.552.1 Purpose. - Perform statement checking on a node of type 'pragma'.

3.2.2.552.2 Assumptions. - None

3.2.2.552.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.552.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.552.5 Nested within. - scp_prag

3.2.2.552.6 Host Dependencies. - None

3.2.2.552.7 Target Dependencies. - None

3.2.2.552.8 Subprogram Visibility. - Outside Package.

3.2.2.552.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.552.10 Formal Parameters. -

```
subtree :  in  cdm_tyoe.c_node_ref; -- reference to diana node
                                         to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.552.11 Side-Effects. - None

3.2.2.552.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_param_assoc_s>
case PRAGMA name> of
  when PAGE> ! INCLUDE> ! LIST> =>
    return;
  when TITLE> =>
    INVOKE sc_title>
  when INLINE> =>
    return;
  when INTERFACE> =>
    return;
  when MEMORY_SIZE> =>
    INVOKE sc_mem_size>
  when OPTIMIZE> =>
```

```
    INVOKE sc_optimize>
  when PRIORITY> =>
    INVOKE sc_priority>
  when STORAGE_UNIT> =>
    invoke sc_stdr_unit>
  when SUPPRESS> =>
    INVOKE sc_suppress>
  when SYSTEM> =>
    INVOKE sc_system>
  when controlled> =>
    INVOKE sc_controlled>
  when PACK> =>
    INVOKE sc_pack>
end case;
```

3.2.2.552.13 Diagnostic Messages Generated. - None

3.2.2.552.14 Examples of Data Structures. - None

3.2.2.553 Subprogram. - sc_title

3.2.2.553.1 Purpose. - Perform statement checking on a TITLE pragma.

3.2.2.553.2 Assumptions. - None

3.2.2.553.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.553.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.553.5 Nested Within. - scp_prag

3.2.2.553.6 Host Dependencies. - None

3.2.2.553.7 Target Dependencies. - None

3.2.2.553.8 Subprogram Visibility. - Outside Package.

3.2.2.553.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.553.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                          to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                         this node appears
```

3.2.2.553.11 Side-Effects. - None

3.2.2.553.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT the first lexical unit> then
  ERROR  17300>
end if;
if ARG not a character string> then
  ERROR  17301>
end if;
```

3.2.2.553.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17300	W	title pragma misplaced
17301	W	invalid argument for title pragma

3.2.2.553.14 Examples of Data Structures. - None

3.2.2.554 Subprogram. - sc_mem_size

3.2.2.554.1 Purpose. - Perform statement checking on a MEMORY_SIZE pragma.

3.2.2.554.2 Assumptions. - None

3.2.2.554.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.554.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.554.5 Nested Within. - spc_prag

3.2.2.554.6 Host Dependencies. - None

3.2.2.554.7 Target Dependencies. - None

3.2.2.554.8 Subprogram Visibility. - Within Package Only.

3.2.2.554.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.554.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                        to be checked
context :  in  stmt_chk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.554.11 Side-Effects. - None

3.2.2.554.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT before a compilation unit> then
  ERROR  17302>
end if;
if ARGUMENT value not in range 0..system.memory_size> then
  ERROR  17303>
```

3.2.2.554.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17302	W	memory_size pragma misplaced
17303	W	invalid value for memory_size pragma

3.2.2.554.14 Examples of Data Structures. - None

3.2.2.555 Subprogram. - sc_optimize

3.2.2.555.1 Purpose. - Perform statement checking on an OPTIMIZE pragma.

3.2.2.555.2 Assumptions. - None

3.2.2.555.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.555.4 Used Recursively. - This suborogram is not used recursively.

3.2.2.555.5 Nested Within. - scp_prag

3.2.2.555.6 Host Dependencies. - None

3.2.2.555.7 Target Dependencies. - None

3.2.2.555.8 Subprogram Visibility. - Within Package Only.

3.2.2.555.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.555.10 Formal Parameters. -

```
suotree :  in  cdm_tyoe.c_node_ref; -- reference to diana node
                                        to be checked

context :  in  stmt_chk.context_rec;  -- defines context in which
                                        this node appears
```

3.2.2.555.11 Side-Effects. - None

3.2.2.555.12 Algorithm. -

```
if THIS node has already been  flagged as in error> then
  return;
end if;
if NOT in a declarative part> then
  ERROR  17304>
end if;
```

3.2.2.555.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17304	W	optimize pragma not in a declarative part

3.2.2.555.14 Examples of Data Structures. - None

3.2.2.556 Subprogram. - sc_priority

3.2.2.556.1 Purpose. - Perform statement checking on a PRIORITY pragma.

3.2.2.556.2 Assumptions. - None

3.2.2.556.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.556.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.556.5 Nested Within. - scp_prag

3.2.2.556.6 Host Dependencies. - None

3.2.2.556.7 Target Dependencies. - None

3.2.2.556.8 Subprogram Visibility. - Within Package Only.

3.2.2.556.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.556.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node to
                                        be checked

context :  in  stmt_chk.context_rec;  -- defines context in which
                                        this node appears
```

3.2.2.556.11 Side-Effects. - None

3.2.2.556.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT in a task spec or outermost decl part of library subprogram> then
  ERROR  17305>
end if;
if  VALUE not in range of subtype SYSTEM.PRIORITY> then
  ERROR  17306>
end if;
```

3.2.2.556.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17305	W	priority pragma misplaced
17306	W	invalid argument for priority pragma

3.2.2.556.14 Examples of Data Structures. - None

3.2.2.557 Subprogram. - sc_strg_unit

3.2.2.557.1 Purpose. - Perform statement checking on a STORAGE_UNIT pragma.

3.2.2.557.2 Assumptions. - None

3.2.2.557.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.557.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.557.5 Nested Within. - scp_pragma

3.2.2.557.6 Host Dependencies. - None

3.2.2.557.7 Target Dependencies. - None

3.2.2.557.8 Subprogram Visibility. - Within Package Only.

3.2.2.557.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.557.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                        to be checked

context :  in  stmt_chk.context_rec;  -- defines context in which
                                        this node appears
```

3.2.2.557.11 Side-Effects. - None

3.2.2.557.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT before a library unit> then
  ERROR  17307>
end if;
if VALUE of argument /= SYSTEM.STORAGE_UNIT> then
  ERROR  17308>
end if;
```

3.2.2.557.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17307	W	storage_unit pragma misplaced
17308	W	invalid storage_unit pragma argument

3.2.2.557.14 Examples of Data Structures. - None

3.2.2.558 Subprogram. - sc_system

3.2.2.558.1 Purpose. - Perform statement checking on a SYSTEM pragma.

3.2.2.558.2 Assumptions. - None

3.2.2.558.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.558.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.558.5 Nested Within. - scp_prag

3.2.2.558.6 Host Dependencies. - None

3.2.2.558.7 Target Dependencies. - None

3.2.2.558.8 Subprogram Visibility. - Within Package Only.

3.2.2.558.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.558.10 Formal Parameters. -

```
subtree :  in cdm_type.c_node_ref; -- reference to diana node
                                      to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                        this node appears
```

3.2.2.558.11 Side-Effects. - None

3.2.2.558.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT before a library unit> then
  ERROR  17309>
end if;
```

3.2.2.558.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17309	W	system pragma misplaced

3.2.2.558.14 Examples of Data Structures. - None

3.2.2.559 Package. - scp_comp

3.2.2.559.1 Purpose. - Implement the statement checking function for nodes involved with compilation units.

3.2.2.559.2 Number of Subprograms. - 3

3.2.2.559.3 Dependencies on Other Packages for Spec. - cdm_type, stmtchk

3.2.2.559.4 Additional Dependencies for Body. - cdm_as, sc_pragma

3.2.2.559.5 Package Specification. -

```
package scp_comp is
     procedure sc_comp_unit;
     procedure sc_pragma_s;
     procedure sc_subunit;
end scp_comp;
```

3.2.2.559.6 Elaboration Code. - None

3.2.2.559.7 Examples of Data Structures. - None

3.2.2.560 Subprogram. - sc_comp_unit

3.2.2.560.1 Purpose. - Perform statement checking on a node of type 'comp_unit'.

3.2.2.560.2 Assumptions. - None

3.2.2.560.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.560.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.560.5 Nested Within. - scp_comp

3.2.2.560.6 Host Dependencies. - None

3.2.2.560.7 Target Dependencies. - None

3.2.2.560.8 Subprogram Visibility. - Outside Package.

3.2.2.560.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.560.10 Formal Parameters. -

```
subtree : in cdm_type.c_node_ref; -- reference to diana node to
                                       be checked.
context : in stmt_chk.context_rec; -- defines context in which this
                                      node appears.
```

3.2.2.560.11 Side-Effects. - None

3.2.2.560.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node "of type", cdm_type.get_node_id,
          "context = " context >
end if;
if THIS node has already been flagged as in error> then
   return;
end if;
CHECK as_pragma_s>
CHECK as_unit_body>
-- compute elaboration information
PERFORM transitive closure on call_relation>
ADD to the CUT all library units P such that:
    P contains a subprigram Y such that:
      for some X in the elaboration_call list,
          (X calls Y) is in the transitive closure
          of the call_relation>
```

3.2.2.560.13 Diagnostic Messages Generated. - None

3.2.2.560.14 Examples of Data Structures. - None

3.2.2.561 Subprogram. - sc_pragma_s

3.2.2.561.1 Purpose. - Perform statement checking on a node of type 'pragma_s'.

3.2.2.561.2 Assumptions. - None

3.2.2.561.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.561.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.561.5 Nested Within. - sc_comp

3.2.2.561.6 Host Dependencies. - None

3.2.2.561.7 Target Dependencies. - None

3.2.2.561.8 Subprogram Visibility. - Outside Package.

3.2.2.561.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.561.10 Formal Parameters. -

```
subtree : in cdm_type.c_node_ref; -- reference to diana node to
                                     be checked
context : in stmt_chk.context_rec; -- defines context in which this
                                    node appears
```

3.2.2.561.11 Side-Effects. - None

3.2.2.561.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node "of type", cdm_type.get_node_id,
         "context = " context >
end if;
if THIS node has already been flagged as in error> then
   return;
end if;
for EACH element of as_list> loop
    CHECK PRAGMA>
end loop;
```

3.2.2.561.13 Diagnostic Messages Generated. - None

3.2.2.561.14 Examples of Data Structures. - None

3.2.2.562 Subprogram. - sc_subunit

3.2.2.562.1 Purpose. - Perform statement checking on a node of type 'subunit'.

3.2.2.562.2 Assumptions. - None

3.2.2.562.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.562.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.562.5 Nested Within. - sc_comp

3.2.2.562.6 Host Dependencies. - None

3.2.2.562.7 Target Dependencies. - None

3.2.2.562.8 Subprogram Visibility. - Outside Package.

3.2.2.562.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.562.10 Formal Parameters. -

```
subtree : in cdm_type.c_node_ref; -- reference to diana node to
                                      be checked
context : in stmt_chk.context_rec; -- defines context in which this
                                     node appears
```

3.2.2.562.11 Side-Effects. - None

3.2.2.562.12 Algorithm. -

```
if MAINTENANCE option A> then
   PRINT "at node", node "of type", cdm_type.get_node_id,
          "context = " context >
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
CHECK as_subunit_body>
```

3.2.2.562.13 Diagnostic Messages Generated. - None

3.2.2.562.14 Examples of Data Structures. - None

3.2.2.563 Package. - scp_tdep

3.2.2.563.1 Purpose. - Implement the statement checking function for constructs which require target dependent checks.

3.2.2.563.2 Number of Subprograms. - 2

3.2.2.563.3 Dependencies on Other Packages for Spec. - cdm_type, stmtchk

3.2.2.563.4 Additional Dependencies for Body. - cdm_as

3.2.2.563.5 Package Specification. -

```
package scp_tdep is
  procedure sc_suppress;
end scp_tdep;
```

3.2.2.563.6 Elaboration Code. - None

3.2.2.563.7 Examples of Data Structures. - None

3.2.2.564 Subprogram. - sc_suppress

3.2.2.564.1 Purpose. - Perform statement checking on a SUPPRESS program for PDP-11/70 UNIX, ROLM 1666, ROLM 16028 targets.

3.2.2.564.2 Assumptions. - None

3.2.2.564.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.564.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.564.5 Nested Within. - scp_tdep

3.2.2.564.6 Host Dependencies. - None

3.2.2.564.7 Target Dependencies. - PDP-11/70 UNIX, ROLM 1666, ROLM 16028

3.2.2.564.8 Subprogram Visibility. - Within Package Only.

3.2.2.564.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.564.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
                                         to be checked

context :  in  stmt_chk.context_rec; -- defines context in which
                                         this node appears
```

3.2.2.564.11 Side-Effects. - None

3.2.2.564.12 Algorithm. -

```
if MAINTEMANCE option A> then
   PRINT "SUPPRESS pragma">
end if;
if THIS node has already been flagged as in error> then
  return;
end if;
if NOT in a declarative part> then
   ERROR  53>
end if;
```

3.2.2.564.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
53	E	suppress pragma not in a declarative part

3.2.2.564.14 Examples of Data Structures. - None

3.2.2.565 Subprogram. - sc_suppress

3.2.2.565.1 Purpose. - Perform statement checking on a SUPPRESS pragma for VAX/VMS, BARE VAX targets.

3.2.2.565.2 Assumptions. - None

3.2.2.565.3 Implementation Language. - This subprogram is written in Ada.

3.2.2.565.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.565.5 Nested Within. - scp_tdep

3.2.2.565.6 Host Dependencies. - None

3.2.2.565.7 Target Dependencies. - VAX/VMS, BARE VAX

3.2.2.565.8 Subprogram Visibility. - Within Package Only.

3.2.2.565.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.565.10 Formal Parameters. -

```
subtree : in cdm_basics.c_node_ref; -- reference to diana node to be checked
context : in  stmt_chk.context_rec; -- defines context in which this
                                       node appears
```

3.2.2.565.11 Side-Effects. - None

3.2.2.565.12 Algorithm. -

```
if THIS node has already been flagged as in error> then
   return;
end if;
if NOT in a declarative part> then
   ERROR #53>
end if;
if ARGUMENT is division_check, overflow_check> then
   ERROR #54>
end if;
```

3.2.2.565.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
53	E	suppress pragma not in a declarative part
54	W	this check cannot be suppressed

3.2.2.565.14 Examples of Data Structures. - None

3.2.2.566 Package. - sc_class

3.2.2.566.1 Purpose. - Implement the statement checking function for Diana classes.

3.2.2.566.2 Number of Subprograms. - 31

3.2.2.566.3 Dependencies on Other Packages for Spec. - cdm_type, stmtchk

3.2.2.566.4 Additional Dependencies for Body. - scp_expr, scp_stmt, scp_decl, scp_prag, scp_comp

3.2.2.566.5 Package Specification. -

```
package sc_class is
    procedure  cl_acc_const;
    procedure  cl_blk_stub;
    procedure  cl_choice;
    procedure  cl_cnst_void;
    procedure  cl_constrnt;
    procedure  cl_decl;
    procedure  cl_decl_rep;
    procedure  cl_dscrt_rng;
    procedure  cl_dscrt_r_v;
    procedure  cl_exp;
    procedure  cl_exp-void;
    procedure  cl_gen_parm;
    procedure  cl_header;
    procedure  cl_item;
    procedure  cl_iteration;
    procedure  cl_name;
    procedure  cl_name_void;
    procedure  cl_pack_def;
    procedure  cl_param;
    procedure  cl_rng_void;
    procedure  cl_rep;
    procedure  cl_stm;
    procedure  cl_sbunt_body;
    procedure  cl_type_rng;
    procedure  cl_type spec;
    procedure  cl_unit_body;
    procedure  cl_comp;
    procedure  cl_comp_assoc;
    procedure  cl_gen_hdr;
    procedure  cl_subp_def;
    procedure  cl_task_spec;
end sc_class;
```

3.2.2.566.6 Elaboration Code. - None

3.2.2.566.7 Examples of Data Structures. - None

3.2.2.567 Subprogram. - ci_acc_const

3.2.2.567.1 Purpose. - Invoke a statement checking procedure for a node in the class ACCESS_CONSTRAINT.

3.2.2.567.2 Assumptions. - SUBTREE (the first parameter) is in the class ACCESS_CONSTRAINT.

3.2.2.567.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.567.4 Used Recursively. - This subprogram is used recursively.

3.2.2.567.5 Nested within. - sc_class

3.2.2.567.6 Host Dependencies. - None

3.2.2.567.7 Target Dependencies. - None

3.2.2.567.8 Subprogram Visibility. - Outside Package.

3.2.2.567.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.567.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                            this node appears
```

3.2.2.567.11 Side-Effects. - None

3.2.2.567.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when VOID> =>
    return
  when DSCRT_RANGE_S> =>
    INVOKE SC_DSCRT_R_S>
  when others =>
    INVOKE CL_EXP>
end case;
```

3.2.2.567.13 Diagnostic Messages Generated. - None

3.2.2.567.14 Examples of Data Structures. - None

3.2.2.568 Subprogram. - cl_blk_stub

3.2.2.568.1 Purpose. - Invoke a statement checking procedure for a node in the class BLOCK_STUB.

3.2.2.568.2 Assumptions. - SUBTREE (the first parameter) is in the class BLOCK_STUB.

3.2.2.568.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.568.4 Used Recursively. - This subprogram is used recursively.

3.2.2.568.5 Nested Within. - sc_class

3.2.2.568.6 Host Dependencies. - None

3.2.2.568.7 Target Dependencies. - None

3.2.2.568.8 Subprogram Visibility. - Outside Package.

3.2.2.568.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.568.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.568.11 Side-Effects. - None

3.2.2.568.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when BLOCK>
    INVOKE SC_BLOCK>
  when others =>
    return
end case;
```

3.2.2.568.13 Diagnostic Messages Generated. - None

3.2.2.568.14 Examples of Data Structures. - None

3.2.2.569 Subprogram. - cl_choice

3.2.2.569.1 Purpose. - Invoke a statement checking procedure for a node in the class CHOICE.

3.2.2.569.2 Assumptions. - SUBTREE (the first parameter) is in the class CHOICE.

3.2.2.569.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.569.4 Used Recursively. - This subprogram is used recursively.

3.2.2.569.5 Nested Within. - sc_class

3.2.2.569.6 Host Dependencies. - None

3.2.2.569.7 Target Dependencies. - None

3.2.2.569.8 Subprogram Visibility. - Outside Package.

3.2.2.569.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.569.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.569.11 Side-Effects. - None

3.2.2.569.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when OTHERS> =>
    return
  when CONSTRAINED> =>
    INVOKE SC_CONSTRND>
  when RANGE> =>
    INVOKE SC_RANGE>
  when INDEX> =>
    INVOKE SC_INDEX>
  when others =>
    INVOKE CL_EXP>
end case;
```

3.2.2.569.13 Diagnostic Messages Generated. - None

3.2.2.569.14 Examples of Data Structures. - None

3.2.2.570 Subprogram. - cl_cnst_void

3.2.2.570.1 Purpose. - Invoke a statement checking procedure for a node in the class CONSTRAINED_VOID.

3.2.2.570.2 Assumptions. - SUBTREE (the first parameter) is in the class CONSTRAINED_VOID.

3.2.2.570.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.570.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.570.5 Nested within. - sc_class

3.2.2.570.6 Host Dependencies. - None

3.2.2.570.7 Target Dependencies. - None

3.2.2.570.8 Subprogram Visibility. - Outside Package.

3.2.2.570.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.570.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                           this node appears
```

3.2.2.570.11 Side-Effects. - None

3.2.2.570.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when CONSTRAINED> =>
    INVOKE SC_CONSTRND>
  when VOID> =>
    return;
end case;
```

3.2.2.570.13 Diagnostic Messages Generated. - None

3.2.2.570.14 Examples of Data Structures. - None

3.2.2.571 Subprogram. - cl_constrnt

3.2.2.571.1 Purpose. - Invoke a statement checking procedure for a node in the class CONSTRAINT.

3.2.2.571.2 Assumptions. - SUBTREE (the first parameter) is in the class CONSTRAINT.

3.2.2.571.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.571.4 Used Recursively. - This subprogram is used recursively.

3.2.2.571.5 Nested Within. - sc_class

3.2.2.571.6 Host Dependencies. - None

3.2.2.571.7 Target Dependencies. - None

3.2.2.571.8 Subprogram Visibility. - Outside Package.

3.2.2.571.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.571.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.571.11 Side-Effects. - None

3.2.2.571.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when RANGE> =>
    INVOKE SC_RANGE>
  when DSCRMT_AGG> =>
    INVOKE SC_DSCRMT_AG>
  when DSCRT_RANGE_S> =>
    INVOKE SC_DSCRT_R_S>
  when FIXED> =>
    INVOKE SC_FIXED>
  when FLOAT> =>
    INVOKE SC_FLOATC>
  when VOID>
    return;
end case;
```

3.2.2.571.13 Diagnostic Messages Generated. - None

3.2.2.571.14 Examples of Data Structures. - None

3.2.2.572 Subprogram. - cl_decl

3.2.2.572.1 Purpose. - Invoke a statement checking procedure for a node in the class DECL.

3.2.2.572.2 Assumptions. - SUBTREE (first parameter) is in the class DECL.

3.2.2.572.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.572.4 Used Recursively. - This subprogram is used recursively.

3.2.2.572.5 Nested Within. - sc_class

3.2.2.572.6 Host Dependencies. - None

3.2.2.572.7 Target Dependencies. - None

3.2.2.572.8 Subprogram Visibility. - Outside Package.

3.2.2.572.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.572.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  in  context_rec; -- defines context in which this node

                                 appears
```

3.2.2.572.11 Side-Effects. - None

3.2.2.572.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when CONSTANT> =>
    INVOKE SC_CONSTANT>
  when VAR> =>
    INVOKE SC_VAR>
  when NUMBER> =>
    INVOKE SC_NUMBER>
  when TYPE> =>
    INVOKE SC_TYPE>
  when SUBTYPE> =>
    INVOKE SC_SUBTYPE>
  when SUBPROGRAM_DECL> =>
    INVOKE SC_SUBP_DECL>
  when PACKGE_DECL> =>
    INVOKE SC_PACK_DECL>
  when TASK_DECL> =>
   INVOKE SC_TASK_DECL>
  when USE> =>
      return;
  when EXCEPTION> =>
      return;
  when PRAGMA> =>
    INVOKE SC_PRAGMA>
end case;
```

3.2.2.572.13 Diagnostic Messages Generated. - None

3.2.2.572.14 Examples of Data Structures. - None

3.2.2.573 Subprogram. - cl_decl_rep

3.2.2.573.1 Purpose. - Invoke a statement checking procedure for a node in the class DECL_REP.

3.2.2.573.2 Assumptions. - SUBTREE (the first parameter) is in the class DECL_REP.

3.2.2.573.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.573.4 Used Recursively. - This subprogram is used recursively.

3.2.2.573.5 Nested Within. - sc_class

3.2.2.573.6 Host Dependencies. - None

3.2.2.573.7 Target Dependencies. - None

3.2.2.573.8 Subprogram Visibility. - Outside Package.

3.2.2.573.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.573.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.573.11 Side-Effects. - None

3.2.2.573.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when ADDRES> =>
    INVOKE SC-ADDRESS>
  when RECORD_REP> =>
    INVOKE SC_REC_REP>
  when SIMPLE_REP> =>
    INVOKE SC_SIMP_REP>
  when others =>
    INVOKE CL_DECL>
end case;
```

3.2.2.573.13 Diagnostic Messages Generated. - None

3.2.2.573.14 Examples of Data Structures. - None

3.2.2.574 Subprogram. - cl_dscrt_rng

3.2.2.574.1 Purpose. - Invoke a statement checking procedure for a node in the class DSCRT_RANGE.

3.2.2.574.2 Assumptions. - SUBTREE (the first parameter) is in the class DSCRT_RANGE.

3.2.2.574.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.574.4 Used Recursively. - This subprogram is used recursively.

3.2.2.574.5 Nested Within. - sc_class

3.2.2.574.6 Host Dependencies. - None

3.2.2.574.7 Target Dependencies. - None

3.2.2.574.8 Subprogram Visibility. - Outside Package.

3.2.2.574.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.574.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.574.11 Side-Effects. - None

3.2.2.574.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when CONSTRAINED> =>
    INVOKE SC_CONSTRND>
  when RANGE> =>
    INVOKE SC_RANGE>
  when INDEX> =>
    return;
end case;
```

3.2.2.574.13 Diagnostic Messages Generated. - None

3.2.2.574.14 Examples of Data Structures. - None

3.2.2.575 Subprogram. - cl_dscrt_r_v

3.2.2.575.1 Purpose. - Invoke a statement checking procedure for a node in the class DSCRT_RANGE_VOID.

3.2.2.575.2 Assumptions. - SUBTREE (the first parameter) is in the class DSCRT_RANGE_VOID.

3.2.2.575.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.575.4 Used Recursively. - This subprogram is used recursively.

3.2.2.575.5 Nested Within. - sc_class

3.2.2.575.6 Host Dependencies. - None

3.2.2.575.7 Target Dependencies. - None

3.2.2.575.8 Subprogram Visibility. - Outside Package.

3.2.2.575.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.575.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                            this node appears
```

3.2.2.575.11 Side-Effects. - None

3.2.2.575.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when VOID> =>
    return;
  when others =>
    INVOKE CL_DSCRT_RNG>
end case;
```

3.2.2.575.13 Diagnostic Messages Generated. - None

3.2.2.575.14 Examples of Data Structures. - None

3.2.2.576 Subprogram. - cl_exp

3.2.2.576.1 Purpose. - Invoke a statement checking procedure for a node in the class EXP.

3.2.2.576.2 Assumptions. - SUBTREE (the first parameter) is in the class EXP.

3.2.2.576.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.576.4 Used Recursively. - This subprogram is used recursively.

3.2.2.576.5 Nested within. - sc_class

3.2.2.576.6 Host Dependencies. - None

3.2.2.576.7 Target Dependencies. - None

3.2.2.576.8 Subprogram Visibility. - Outside Package.

3.2.2.576.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.576.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.576.11 Side-Effects. - None

3.2.2.576.12 Algorithm. -

```
if CONTEXT requires a static expression> then
  if NODE is not static>  then
     ERROR  17500>
  end if;
end if;
case NODE type of SUBTREE> is
  when AGGREGATE> =>
    INVOKE SC_AGGREGATE>
  when ALLOCATOR> =>
    INVOKE SC_ALLOCATOR>
  when BINARY> =>
    INVOKE SC_BINARY>
  when CONVERSION> =>
    INVOKE SC_TYPE_CONV>
  when MEMBERSHIP> =>
    INVOKE SC_MEMBERSHIP>
  when NULL_ACCESS> =>
    return;
  when NUMERIC_LITERAL> ! STRING_LITERAL> ! USED_CHAR> =>
    return;
  when PARENTHSIZED> =>
    INVOKE SC_PARENTHZD>
  when QUALIFIED> =>
    INVOKE SC_QUALIFIED>
  when others =>
    INVOKE CL_NAME>
end case;
```

3.2.2.576.13 Diagnostic Messages Generated. -

CODE	SEVERITY N.W.E.S.F	TEXT
17500	E	static expression required

3.2.2.576.14 Examples of Data Structures. - None

3.2.2.577 Subprogram. - ci_exp_void

3.2.2.577.1 Purpose. - Invoke a statement checking procedure for a node in the class EXP_VOID.

3.2.2.577.2 Assumptions. - SUBTREE (the first parameter) is in the class EXP_VOID.

3.2.2.577.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.577.4 Used Recursively. - This subprogram is used recursively.

3.2.2.577.5 Nested Within. - sc_class

3.2.2.577.6 Host Dependencies. - None

3.2.2.577.7 Target Dependencies. - None

3.2.2.577.8 Subprogram Visibility. - Outside Package.

3.2.2.577.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.577.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.577.11 Side-Effects. - None

3.2.2.577.12 Algorithm. -

```
if NODE type of SUBTREE> = VOID> then
  return;
else
  INVOKE CL_EXP>
end if;
```

3.2.2.577.13 Diagnostic Messages Generated. - None

3.2.2.577.14 Examples of Data Structures. - None

3.2.2.578 Subprogram. - cl_gen_parm

3.2.2.578.1 Purpose. - Invoke a statement checking procedure for a node in the class GENERIC_PARAM.

3.2.2.578.2 Assumptions. - SUBTREE (the first parameter) is in the class GENERIC_PARAM.

3.2.2.578.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.578.4 Used Recursively. - This subprogram is used recursively.

3.2.2.578.5 Nested Within. - sc_class

3.2.2.578.6 Host Dependencies. - None

3.2.2.578.7 Target Dependencies. - None

3.2.2.578.8 Subprogram Visibility. - Outside Package.

3.2.2.578.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.578.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node
context :  inout  statchk.context_rec; -- defines context in which
                                            this node appears
```

3.2.2.578.11 Side-Effects. - None

3.2.2.578.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when IN> =>
    INVOKE SC_IN>
  when IN_OUT> =>
    INVOKE SC_OUT>
  when SUBPROGRAM_DECL> =>
    INVOKE SC_SUBP_DECL>
  when TYPE> =>
    INVOKE SC_TYPE>
end case;
```

3.2.2.578.13 Diagnostic Messages Generated. - None

3.2.2.578.14 Examples of Data Structures. - None

3.2.2.579 Subprogram. - cl_header

3.2.2.579.1 Purpose. - Invoke a statement checking procedure for a node in the class HEADER.

3.2.2.579.2 Assumptions. - SUBTREE (the first parameter) is in the class HEADER.

3.2.2.579.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.579.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.579.5 Nested Within. - sc_class

3.2.2.579.6 Host Dependencies. - None

3.2.2.579.7 Target Dependencies. - None

3.2.2.579.8 Subprogram Visibility. - Outside Package.

3.2.2.579.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.579.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.579.11 Side-Effects. - None

3.2.2.579.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when FUNCTION> =>
    INVOKE SC_FUNCTION>
  when PROCEDURE> =>
    INVOKE SC_PROCEDURE>
  when ENTRY> =>
    INVOKE SC_ENTRY>
end case;
```

3.2.2.579.13 Diagnostic Messages Generated. - None

3.2.2.579.14 Examples of Data Structures. - None

3.2.2.580 Subprogram. - cl_item

3.2.2.580.1 Purpose. - Invoke a statement checking procedure for a node in the class ITEM.

3.2.2.580.2 Assumptions. - SUBTREE (the first parameter) is in the class ITEM.

3.2.2.580.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.580.4 Used Recursively. - This subprogram is used recursively.

3.2.2.580.5 Nested Within. - sc_class

3.2.2.580.6 Host Dependencies. - None

3.2.2.580.7 Target Dependencies. - None

3.2.2.580.8 Subprogram Visibility. - Outside Package.

3.2.2.580.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.580.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.580.11 Side-Effects. - None

3.2.2.580.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when PACKAGE_BODY> =>
    INVOKE SC_PACK_BODY>
  when SUBPROGRAM_BODY> =>
    INVOKE SC_SUBP_BODY>
  when TASK_BODY> =>
    INVOKE SC_TASK_BODY>
  when USE> =>
    return;
  when ADDRESS> =>
    INVOKE SC_ADDRESS>
  when RECORD_REP> =>
    INVOKE SC_REC_REP>
  when SINPLE_REP> =>
    INVOKE SC_SIMP_REP>
  when others =>
    INVOKE CL_DECL>
end case;
```

3.2.2.580.13 Diagnostic Messages Generated. - None

3.2.2.580.14 Examples of Data Structures. - None

3.2.2.581 Subprogram. - cl_iteration

3.2.2.581.1 Purpose. - Invoke a statement checking procedure for a node in the class ITERATION.

3.2.2.581.2 Assumptions. - SUBTREE (the first parameter) is in the class ITERATION.

3.2.2.581.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.581.4 Used Recursively. - This subprogram is used recursively.

3.2.2.581.5 Nested Within. - sc_class

3.2.2.581.6 Host Dependencies. - None

3.2.2.581.7 Target Dependencies. - None

3.2.2.581.8 Subprogram Visibility. - Outside Package.

3.2.2.581.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.581.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.581.11 Side-Effects. - None

3.2.2.581.12 Algorithm. -

```
case NODE of type SUBTREE> is
  when FOR> =>
    INVOKE SC_FOR> =>
  when REVERSE> =>
    INVOKE SC_REVERSE>
  when WHILE> =>
    INVOKE SC_WHILE>
  when VOID> =>
    return;
end case;
```

3.2.2.581.13 Diagnostic Messages Generated. - None

3.2.2.581.14 Examples of Data Structures. - None

3.2.2.582 Subprogram. - cl_name

3.2.2.582.1 Purpose. - Invoke a statement checking procedure for a node in the class NAME.

3.2.2.582.2 Assumptions. - SUBTREE (the first parameter) is in the class NAME.

3.2.2.582.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.582.4 Used Recursively. - This subprogram is used recursively.

3.2.2.582.5 Nested Within. - sc_class

3.2.2.582.6 Host Dependencies. - None

3.2.2.582.7 Target Dependencies. - None

3.2.2.582.8 Subprogram Visibility. - Outside Package.

3.2.2.582.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.582.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  in  stmtchk.context_rec; -- defines context in which
                                       this node appears
```

3.2.2.582.11 Side-Effects. - None

3.2.2.582.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when  ALL> =>
    return;
  when ATTRIBUTE> =>
    INVOKE SC_ATTRIBUTE>
  when ATTRIBUTE_CALL> =>
    INVOKE SC_ATTR_CALL>
  when FUNCTION_CALL>  =>
    INVOKE SC_FUNC_CALL>
  when INDEXED> =>
```

```
    INVOKE SC_INDEXED>
  when SELECTED> =>
    INVOKE SC_SELECTED>
  when SLICE> =>
    INVOKE SC_SLICE>
  when USED_OBJECT_ID> =>
    INVOKE SC_USED_OBID>
end case;
```

3.2.2.582.13 Diagnostic Messages Generated. - None

3.2.2.582.14 Examples of Data Structures. - None

3.2.2.583 Subprogram. - cl_name_void

3.2.2.583.1 Purpose. - Invoke a statement checking procedure for a node in the class NAME_VOID.

3.2.2.583.2 Assumptions. - SUBTREE (the first parameter) is in the class NAME_VOID.

3.2.2.583.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.583.4 Used Recursively. - This subprogram is used recursively.

3.2.2.583.5 Nested Within. - sc_class

3.2.2.583.6 Host Dependencies. - None

3.2.2.583.7 Target Dependencies. - None

3.2.2.583.8 Subprogram Visibility. - Outside Package.

3.2.2.583.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.583.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                             this node appears
```

3.2.2.583.11 Side-Effects. - None

3.2.2.583.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when VOID> =>
    return;
  when others =>
    INVOKE CL_NAME>
end case;
```

3.2.2.583.13 Diagnostic Messages Generated. - None

3.2.2.583.14 Examples of Data Structures. - None

3.2.2.584 Subprogram. - cl_pack_def

3.2.2.584.1 Purpose. - Invoke a statement checking procedure for a node in the class PACKAGE_DEF.

3.2.2.584.2 Assumptions. - SUBTREE (the first parameter) is in the class PACKAGE_DEF.

3.2.2.584.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.584.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.584.5 Nested Within. - sc_class

3.2.2.584.6 Host Dependencies. - None

3.2.2.584.7 Target Dependencies. - None

3.2.2.584.8 Subprogram Visibility. - Outside Package.

3.2.2.584.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.584.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_ref; -- defines context in which
                                          this node appears
```

3.2.2.584.11 Side-Effects. - None

3.2.2.584.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when PACKAGE_SPEC> =>
    INVOKE SC_PACK_SPEC>
  when RENAME> =>
    INVOKE SC_RENAME>
  when others =>
    return;
end case;
```

3.2.2.584.13 Diagnostic Messages Generated. - None

3.2.2.584.14 Examples of Data Structures. - None

3.2.2.585 Subprogram. - cl_param

3.2.2.585.1 Purpose. - Invoke a statement checking procedure for a node in the class PARAM.

3.2.2.585.2 Assumptions. - SUBTREE (the first parameter) is in the class PARAM.

3.2.2.585.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.585.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.585.5 Nested Within. - sc_class

3.2.2.585.6 Host Dependencies. - None

3.2.2.585.7 Target Dependencies. - None

3.2.2.585.8 Subprogram Visibility. - Outside Package.

3.2.2.585.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.585.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.585.11 Side-Effects. - None

3.2.2.585.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when IN> =>
    INVOKE SC_IN>
  when OUT> | IN_OUT> =>
    INVOKE SC_OUT>
end case;
```

3.2.2.585.13 Diagnostic Messages Generated. - None

3.2.2.585.14 Examples of Data Structures. - None

3.2.2.586 Subprogram. - cl_rng_void

3.2.2.586.1 Purpose. - Invoke a statement checking procedure for a node in the class RANGE_VOID.

3.2.2.586.2 Assumptions. - SUBTREE (the first parameter) is in the class RANGE_VOID.

3.2.2.586.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.586.4 Used Recursively. - This subprogram is used recursively.

3.2.2.586.5 Nested Within. - sc_class

3.2.2.586.6 Host Dependencies. - None

3.2.2.586.7 Target Dependencies. - None

3.2.2.586.8 Subprogram Visibility. - Outside Package.

3.2.2.586.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.586.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.586.11 Side-Effects. - None

3.2.2.586.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when VOID> =>
    return;
  when RANGE> =>
    INVOKE SC_RANGE>
end case;
```

3.2.2.586.13 Diagnostic Messages Generated. - None

3.2.2.586.14 Examples of Data Structures. - None

3.2.2.587 Subprogram. - cl_rep

3.2.2.587.1 Purpose. - Invoke a statement checking procedure for a node in the class REP.

3.2.2.587.2 Assumptions. - SUBTREE (the first parameter) is in the class REP.

3.2.2.587.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.587.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.587.5 Nested Within. - sc_class

3.2.2.587.6 Host Dependencies. - None

3.2.2.587.7 Target Dependencies. - None

3.2.2.587.8 Subprogram Visibility. - Outside Package.

3.2.2.587.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.587.10 Formal Parameters. -

```
subtree :  in   cdm_type.c_node_ref; -- reference to diana node
context :  inout stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.587.11 Side-Effects. - None

3.2.2.587.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when ADDRESS> =>
    INVOKE SC_ADDRESS>
  when RECORD_REP> =>
    INVOKE SC_REC_REP>
  when SIMPLE_REP> =>
    INVOKE SC_SIMP_REC>
end case;
```

3.2.2.587.13 Diagnostic Messages Generated. - None

3.2.2.587.14 Examples of Data Structures. - None

3.2.2.588 Subprogram. - cl_stm

3.2.2.588.1 Purpose. - Invoke a statement checking procedure for a node in the class STM.

3.2.2.588.2 Assumptions. - SUBTREE (the first parameter) is in the class STM.

3.2.2.588.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.588.4 Used Recursively. - This subprogram is used recursively.

3.2.2.588.5 Nested Within. - sc_class

3.2.2.588.6 Host Dependencies. - None

3.2.2.588.7 Target Dependencies. - None

3.2.2.588.8 Subprogram Visibility. - Outside Package.

3.2.2.588.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.588.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.588.11 Side-Effects. - None

3.2.2.588.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when LOOP> =>
    INVOKE SC_LOOP>
  when ABORT> =>
    INVOKE SC_ABORT>
  when ACCEPT> =>
    INVOKE SC_ACCEPT>
  when ASSIGN> =>
    INVOKE SC_ASSIGN>
  when BLOCK> =>
    INVOKE SC_BLOCK>
  when CASE> =>
    INVOKE SC_CASE>
  when CODE> =>
    INVOKE SC_CODE>
  when COND_ENTRY> =>
    INVOKE SC_COND_ENTR>
  when DELAY> =>
    INVOKE SC_DELAY>
  when ENTRY_CALL> =>
    INVOKE SC_ENTRY_CAL>
  when EXIT> =>
    INVOKE SC_EXIT>
  when GOTO> =>
    INVOKE SC_GOTO>
  when IF> =>
    INVOKE SC_IF>
  when LABELED>
    INVOKE SC_LABELED>
  when NAMED_STM>
    INVOKE SC_NAMED_STM>
  when NULL_STM>
    return;
  when PRAGMA>
    INVOKE SC_PRAGMA>
  when PROCEDURE_CALL>
    INVOKE SC_PROC_CALL>
  when RAISE>
    INVOKE SC_RAISE>
  when RETURN>
    INVOKE SC_RETURN>
  when SELECT>
    INVOKE SC_SELECT>
  when TERMINATE>
    INVOKE SC_TERMINATE>
  when TIMED_ENTRY>
    INVOKE SC_TIMED_ENTR>
end case;
```

3.2.2.588.13 Diagnostic Messages Generated. - None

3.2.2.588.14 Examples of Data Structures. - None

3.2.2.589 Subprogram. - cl_sbunt_body

3.2.2.589.1 Purpose. - Invoke a statement checking procedure for a node in the class SUBUNIT_BODY.

3.2.2.589.2 Assumptions. - SUBTREE (the first parameter) is in the class SUBUNIT_BODY.

3.2.2.589.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.589.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.589.5 Nested Within. - sc_class

3.2.2.589.6 Host Dependencies. - None

3.2.2.589.7 Target Dependencies. - None

3.2.2.589.8 Subprogram Visibility. - Outside Package.

3.2.2.589.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.589.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.589.11 Side-Effects. - None

3.2.2.589.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when SUBPROGRAM_BODY> =>
    INVOKE SC_SUBP_BODY>
  when PACKAGE_BODY> =>
    INVOKE SC_PACK_BODY>
  when TASK_BODY> =>
    INVOKE SC_TASK_BODY>
end case;
```

3.2.2.589.13 Diagnostic Messages Generated. - None

3.2.2.589.14 Examples of Data Structures. - None

3.2.2.590 Subprogram. - cl_type_rng

3.2.2.590.1 Purpose. - Invoke a statement checking procedure for a node in the class TYPE_RANGE.

3.2.2.590.2 Assumptions. - SUBTREE (the first parameter) is in the class TYPE_RANGE.

3.2.2.590.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.590.4 Used Recursively. - This subprogram is used recursively.

3.2.2.590.5 Nested Within. - sc_class

3.2.2.590.6 Host Dependencies. - None

3.2.2.590.7 Target Dependencies. - None

3.2.2.590.8 Subprogram Visibility. - Outside Package.

3.2.2.590.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.590.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                           this node appears
```

3.2.2.590.11 Side-Effects. - None

3.2.2.590.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when RANGE> =>
    INVOKE SC_RANGE>
  when others =>
    INVOKE SC_CONTRNO>
end case;
```

3.2.2.590.13 Diagnostic Messages Generated. - None

3.2.2.590.14 Examples of Data Structures. - None

3.2.2.591 Subprogram. - cl_type_spec

3.2.2.591.1 Purpose. - Invoke a statement procedure for a node in the class TYPE_SPEC.

3.2.2.591.2 Assumptions. - SUBTREE (the first parameter) is in the class TYPE_SPEC.

3.2.2.591.3 Implementation Language. - This Subprogram is written in Ada.

3.2.2.591.4 Used Recursively. - This subprogram is not used recursively.

3.2.2.591.5 Nested Within. - sc_class

3.2.2.591.6 Host Dependencies. - None

3.2.2.591.7 Target Dependencies. - None

3.2.2.591.8 Subprogram Visibility. - Outside Package.

3.2.2.591.9 Function/Procedure. - This subprogram is a Procedure.

3.2.2.591.10 Formal Parameters. -

```
subtree :  in  cdm_type.c_node_ref; -- reference to diana node
context :  inout  stmtchk.context_rec; -- defines context in which
                                          this node appears
```

3.2.2.591.11 Side-Effects. - None

3.2.2.591.12 Algorithm. -

```
case NODE type of SUBTREE> is
  when CONSTRAINED> =>
    INVOKE SC_CONSTRND>
  when ACCESS> =>
    INVOKE SC_ACCESS>
  when ARRAY> =>
    INVOKE sc_array>
  when DERIVED> =>
    INVOKE SC_DERIVED>
  when FIXED> =>
    INVOKE SC_FIXED>
  when FLOAT> =>
    INVOKE SC_FLOAT>
  when INTEGER> =>
    INVOKE SC_INTEGER>
  when 1-PRIVATE> ! PRIVATE>
    INVOKE SC_PRIVATE>
  when RECORD> =>
    INVOKE SC-RECORD>
  when TASK_SPEC> =>
    INVOKE SC_TASK_SPEC>
  when others =>
    return;
end case;
```

3.2.2.591.13 Diagnostic Messages Generated. - None

3.2.2.591.14 Examples of Data Structures. - None

APPENDIX 10

INDEX OF PACKAGE AND SUBPROGRAM DESCRIPTIONS

Subprogram	Section	Page	Package
abort_stm	3.2.2.132	3-420	stms
abs	3.2.2.668	3-1309	con_man
accept_stm	3.2.2.128	3-414	stms
access_specf	3.2.2.211	3-560	types
access_type_assoc	3.2.2.198	3-540	gen_ren
add	3.2.2.109	3-385	withtab
add	3.2.2.667	3-1307	con_man
add_ancestry	3.2.2.85	3-341	context
add_context_withs	3.2.2.87	3-345	context
add_current	3.2.2.86	3-344	context
add_dnt_entry	3.2.2.181	3-509	vistree
add_dnt_entry_s	3.2.2.182	3-511	vistree
add_large_integers	3.2.2.673	3-1316	con_man
add_variants	3.2.2.88	3-347	context
add_withs	3.2.2.89	3-348	context
address	3.2.2.239	3-604	reps
aggregate	3.2.2.172	3-493	search
aggregate	3.2.2.152	3-454	exps
aggregate	3.2.2.437	3-928	static_exp
alll	3.2.2.438	3-930	static_exp
all_out_formals	3.2.2.619	3-1220	inline_expansion
allocator	3.2.2.439	3-931	static_exp
allocator	3.2.2.163	3-475	exps
apply	3.2.2.160	3-469	exps
array_check	3.2.2.457	3-958	static_exp
array_specf	3.2.2.208	3-555	types
array_type_assoc	3.2.2.191	3-525	gen_ren
as_comp_rep_s	3.2.2.237	3-601	reps
as_range	3.2.2.149	3-450	exps
asg_mod	3.2.2.703	3-1369	data_col_p
asg_stm_eval	3.2.2.748	3-1446	exp_eval_cc_p
asg_usg_info	3.2.2.788	3-1509	usg_prcs
assign	3.2.2.119	3-399	stms
associated_actual	3.2.2.618	3-1218	inline_expansion
associated_actual	3.2.2.626	3-1232	generic_expansion
ast	3.2.2.817	3-1554	buildtree
at_fork	3.2.2.728	3-1412	f_data_flow_p
at_join	3.2.2.729	3-1414	f_data_flow_p
attribute	3.2.2.282	3-675	prag_attr
attribute	3.2.2.259	3-637	prag_attr
attribute	3.2.2.270	3-655	prag_attr
attribute	3.2.2.306	3-715	prag_attr
attribute	3.2.2.294	3-695	prag_attr
b_branch_end	3.2.2.777	3-1494	b_data_flow
b_branch_top	3.2.2.778	3-1495	b_data_flow
b_fork	3.2.2.776	3-1493	b_data_flow
b_join	3.2.2.775	3-1491	b_data_flow
b_loop_end	3.2.2.779	3-1497	b_data_flow
b_loop_top	3.2.2.780	3-1498	b_data_flow
b_opt_driver;	3.2.2.770	3-1483	b_opt_control

b_subprg_call	3.2.2.781	3-1499	b_data_flow
b_worst_case	3.2.2.774	3-1490	b_data_flow
backend	3.2.2.18	3-213	crl_main
backup	3.2.2.632	3-1244	formatter
backward_opt	3.2.2.769	3-1481	b_opt_control
binary	3.2.2.440	3-933	static_exp
binary	3.2.2.155	3-459	exps
binary_walk	3.2.2.756	3-1459	cse_p
bk_decl_part	3.2.2.709	3-1379	data_col_p
block	3.2.2.118	3-397	stms
block_check	3.2.2.708	3-1378	data_col_p
branch_end	3.2.2.731	3-1417	f_data_flow_p
branch_stm_opt	3.2.2.768	3-1480	b_opt_control
branch_top	3.2.2.730	3-1415	f_data_flow_p
break_comment	3.2.2.633	3-1245	formatter
build_list	3.2.2.168	3-485	search
call_parms	3.2.2.701	3-1365	data_col_p
case_delete	3.2.2.737	3-1428	exp_eval_cc_p
case_stm	3.2.2.139	3-432	stms
check_option	3.2.2.835	3-1611	use_opt
check_trace	3.2.2.836	3-1612	use_opt
check_unit_id	3.2.2.90	3-350	context
check_use	3.2.2.837	3-1614	use_opt
check_user	3.2.2.26	3-225	crl_esp
checks	3.2.2.19	3-215	crl_main
choice	3.2.2.145	3-442	exps
choice_s	3.2.2.146	3-444	exps
cl_acc_const	3.2.2.567	3-1139	sc_class
cl_acc_const	3.2.2.408	3-883	ovp_clas
cl_blk_stub	3.2.2.409	3-884	ovp_clas
cl_blk_stub	3.2.2.568	3-1140	sc_class
cl_choice	3.2.2.569	3-1142	sc_class
cl_choice	3.2.2.410	3-886	ovp_clas
cl_cnst_void	3.2.2.411	3-887	ovp_clas
cl_cnst_void	3.2.2.570	3-1143	sc_class
cl_comp	3.2.2.597	3-1186	sc_class
cl_comp_assc	3.2.2.593	3-1180	sc_class
cl_constrnt	3.2.2.571	3-1145	sc_class
cl_constrnt	3.2.2.412	3-889	ovp_clas
cl_decl	3.2.2.413	3-891	ovp_clas
cl_decl	3.2.2.572	3-1146	sc_class
cl_decl_rep	3.2.2.573	3-1148	sc_class
cl_decl_rep	3.2.2.414	3-892	ovp_clas
cl_dscrt_r_v	3.2.2.416	3-895	ovp_clas
cl_dscrt_r_v	3.2.2.575	3-1151	sc_class
cl_dscrt_rng	3.2.2.574	3-1150	sc_class
cl_dscrt_rng	3.2.2.415	3-894	ovp_clas
cl_error	3.2.2.434	3-923	ovp_clas
cl_exp	3.2.2.417	3-897	ovp_clas
cl_exp	3.2.2.576	3-1153	sc_class
cl_exp_void	3.2.2.577	3-1155	sc_class
cl_exp_void	3.2.2.418	3-899	ovp_clas
cl_gen_hdr	3.2.2.594	3-1182	sc_class

cl_gen_parm	3.2.2.578	3-1155	sc_class
cl_gen_parm	3.2.2.419	3-900	ovp_clas
cl_header	3.2.2.420	3-902	ovp_clas
cl_header	3.2.2.579	3-1158	sc_class
cl_item	3.2.2.580	3-1159	sc_class
cl_item	3.2.2.421	3-903	ovp_clas
cl_iteration	3.2.2.422	3-905	ovp_clas
cl_iteration	3.2.2.581	3-1161	sc_class
cl_name	3.2.2.582	3-1162	sc_class
cl_name	3.2.2.423	3-906	ovp_clas
cl_name_void	3.2.2.424	3-908	ovp_clas
cl_name_void	3.2.2.583	3-1164	sc_class
cl_pack_def	3.2.2.584	3-1165	sc_class
cl_pack_def	3.2.2.425	3-909	ovp_clas
cl_param	3.2.2.426	3-911	ovp_clas
cl_param	3.2.2.585	3-1167	sc_class
cl_rep	3.2.2.587	3-1170	sc_class
cl_rep	3.2.2.428	3-914	ovp_clas
cl_rng_void	3.2.2.427	3-912	ovp_clas
cl_rng_void	3.2.2.586	3-1168	sc_class
cl_sbunt_body	3.2.2.589	3-1174	sc_class
cl_sbunt_body	3.2.2.430	3-917	ovp_clas
cl_stm	3.2.2.429	3-915	ovp_clas
cl_stm	3.2.2.588	3-1171	sc_class
cl_subp_def	3.2.2.595	3-1183	sc_class
cl_task_spec	3.2.2.596	3-1185	sc_class
cl_type_rng	3.2.2.590	3-1175	sc_class
cl_type_rng	3.2.2.431	3-919	ovp_clas
cl_type_spec	3.2.2.432	3-920	ovp_clas
cl_type_spec	3.2.2.591	3-1177	sc_class
cl_unit_body	3.2.2.592	3-1179	sc_class
cl_unit_body	3.2.2.433	3-922	ovp_clas
clean_up_old_compila	3.2.2.39	3-245	lexer
cleanup	3.2.2.27	3-227	crl_esp
code_stm	3.2.2.140	3-433	stms
compare	3.2.2.682	3-1333	con_man
compilation_unit	3.2.2.604	3-1196	expansion
compile	3.2.2.20	3-216	crl_main
component	3.2.2.210	3-558	types
concatenation	3.2.2.689	3-1345	con_man
cond_clause	3.2.2.137	3-428	stms
cond_timed_entry	3.2.2.131	3-418	stms
const_redef	3.2.2.245	3-614	redefs
constant_var	3.2.2.231	3-591	objects
constrained	3.2.2.148	3-447	exps
constraint	3.2.2.150	3-451	exps
constraint_check	3.2.2.455	3-955	static_exp
constraint_check_del	3.2.2.746	3-1443	exp_eval_cc_p
context	3.2.2.101	3-371	context
controlled	3.2.2.300	3-706	prag_attr
controlled	3.2.2.275	3-664	prag_attr
controlled	3.2.2.288	3-686	prag_attr
controlled	3.2.2.264	3-646	prag_attr

controlled	3.2.2.252	3-625	prag_attr
conv_numeric_literal	3.2.2.691	3-1348	con_man
conv_to_large_intege	3.2.2.672	3-1315	con_man
conversion	3.2.2.441	3-934	static_exp
copy3	3.2.2.608	3-1202	copy_subtree
copyspec	3.2.2.21	3-218	crl_main
creat_cse_temp	3.2.2.751	3-1450	cse_p
create_derived_subpr	3.2.2.176	3-500	derived
create_rec_block	3.2.2.698	3-1360	data_col_p
create_subprogs	3.2.2.175	3-498	derived
create_vis_node	3.2.2.180	3-508	vistree
cross_ref	3.2.2.665	3-1301	xref
cse	3.2.2.759	3-1466	cse_p
cse_exp_walk	3.2.2.758	3-1462	cse_p
cse_match	3.2.2.754	3-1455	cse_p
cse_store	3.2.2.752	3-1452	cse_p
cse_substitute	3.2.2.753	3-1453	cse_p
data_collection	3.2.2.712	3-1384	data_col_p
dead_cd_check	3.2.2.791	3-1513	dc_delete
decls	3.2.2.142	3-437	decls
delay_stm	3.2.2.129	3-415	stms
delete_container	3.2.2.28	3-228	crl_esp
derived	3.2.2.214	3-564	types
determine_ancestry	3.2.2.91	3-352	context
diag_stats	3.2.2.652	3-1284	diagnose
discrete_type_assoc	3.2.2.197	3-538	gen_ren
discriminant_check	3.2.2.458	3-960	static_exp
discrims	3.2.2.216	3-568	types
divide_large_integer	3.2.2.676	3-1321	con_man
dscrt_range	3.2.2.151	3-453	exps
dscrt_range_s	3.2.2.164	3-478	exps
endlinecheck	3.2.2.634	3-1247	formatter
entry_decl	3.2.2.221	3-576	tasks
enum_literal_redef	3.2.2.246	3-616	redefs
enum_literals	3.2.2.207	3-553	types
error_check	3.2.2.801	3-1528	argparser
error_handler	3.2.2.53	3-265	parser
errorfound	3.2.2.653	3-1285	diagnose
escape	3.2.2.29	3-230	crl_esp
establish	3.2.2.22	3-219	crl_main
ev_binary_walk	3.2.2.744	3-1439	exp_eval_cc_p
ev_unary_walk	3.2.2.743	3-1438	exp_eval_cc_p
eval	3.2.2.436	3-926	static_exp
exception_case_alter	3.2.2.138	3-429	stms
exception_dec	3.2.2.232	3-593	objects
exception_renaming	3.2.2.187	3-519	gen_ren
exit_stm	3.2.2.125	3-408	stms
exp	3.2.2.144	3-440	exps
exp_delete	3.2.2.722	3-1401	f_common
exp_eval_walk	3.2.2.747	3-1445	exp_eval_cc_p
exp_evaluation	3.2.2.740	3-1433	exp_eval_cc_p
exp_used	3.2.2.702	3-1367	data_col_p
exp_usg_info	3.2.2.787	3-1508	usg_prcs

expand	3.2.2.614	3-1211	inline_expansion
expand	3.2.2.605	3-1198	expansion
expand_package	3.2.2.624	3-1228	generic_expansion
expand_routine	3.2.2.623	3-1226	generic_expansion
exponentiation	3.2.2.681	3-1331	con_man
f_opt_driver	3.2.2.719	3-1395	f_control
fill_buffer	3.2.2.40	3-246	lexer
fill_dnt	3.2.2.93	3-355	context
fill_dnt_block	3.2.2.92	3-354	context
fill_dnt_items	3.2.2.94	3-357	context
find	3.2.2.110	3-386	withtab
find_def	3.2.2.95	3-360	context
find_dnt_entry	3.2.2.178	3-505	vistree
find_previous_token	3.2.2.818	3-1586	buildtree
find_red	3.2.2.54	3-298	parser
find_routine_calls	3.2.2.599	3-1188	operator_replacement
find_stub	3.2.2.96	3-361	context
find_stubs	3.2.2.16	3-210	basic_units
find_tran	3.2.2.55	3-300	parser
first_generic	3.2.2.204	3-550	gen_ren
first_module_body	3.2.2.12	3-204	basic_units
first_subprogram_bod	3.2.2.8	3-197	basic_units
fixed_float	3.2.2.147	3-445	exps
flag_is_set	3.2.2.838	3-1615	use_opt
float_type	3.2.2.461	3-965	target_type
for_loop	3.2.2.123	3-405	stms
formal_subprogram_as	3.2.2.199	3-541	gen_ren
formal_subprogram_de	3.2.2.201	3-545	gen_ren
frontend	3.2.2.23	3-221	crl_main
function_call	3.2.2.442	3-936	static_exp
g_assoc	3.2.2.193	3-529	gen_ren
general_assoc_s	3.2.2.161	3-472	exps
general_pragma	3.2.2.255	3-630	prag_attr
general_pragma	3.2.2.291	3-690	prag_attr
general_pragma	3.2.2.267	3-650	prag_attr
general_pragma	3.2.2.278	3-668	prag_attr
general_pragma	3.2.2.303	3-710	prag_attr
generic_assoc_s	3.2.2.194	3-530	gen_ren
generic_dec	3.2.2.205	3-551	gen_ren
generic_header	3.2.2.203	3-548	gen_ren
generic_param_s	3.2.2.202	3-547	gen_ren
get_arg	3.2.2.802	3-1529	argparser
get_char	3.2.2.41	3-248	lexer
get_nx_token	3.2.2.635	3-1249	formatter
get_stmt	3.2.2.819	3-1587	buildtree
get_token	3.2.2.42	3-250	lexer
goto_srch	3.2.2.170	3-489	search
goto_stm	3.2.2.127	3-412	stms
hash	3.2.2.111	3-388	withtab
hash	3.2.2.43	3-251	lexer
hash_dnt	3.2.2.179	3-506	vistree
id_s	3.2.2.227	3-585	objects
ident	3.2.2.166	3-480	search

main	3.2.2.24	3-223	crl_main
mainence	3.2.2.636	3-1250	formatter
maintenance_tracing	3.2.2.637	3-1252	formatter
make_apply_node	3.2.2.822	3-1592	buildtree
make_arg_name	3.2.2.803	3-1531	argparser
make_block_node	3.2.2.823	3-1593	buildtree
make_id_node	3.2.2.824	3-1595	buildtree
make_id_s	3.2.2.825	3-1596	buildtree
match_and_transform_	3.2.2.616	3-1215	inline_expansion
match_and_transform_	3.2.2.625	3-1230	generic_expansion
match_id_s	3.2.2.99	3-368	context
match_separate_claus	3.2.2.100	3-369	context
membership	3.2.2.154	3-458	exps
membership	3.2.2.444	3-939	static_exp
minus	3.2.2.670	3-1311	con_man
mod	3.2.2.680	3-1329	con_man
modify_inpointing_se	3.2.2.611	3-1207	copy_subtree
modify_pointer	3.2.2.612	3-1209	copy_subtree
module_body_redef	3.2.2.244	3-613	redefs
multiply	3.2.2.671	3-1313	con_man
multiply_large_integ	3.2.2.675	3-1320	con_man
name_resolution	3.2.2.114	3-391	name_res
named_stm	3.2.2.121	3-403	stms
new_compilation	3.2.2.47	3-258	lexer
new_spec	3.2.2.3	3-187	basic_units
newfile	3.2.2.46	3-256	lexer
newline	3.2.2.638	3-1254	formatter
no_side_effect	3.2.2.790	3-1512	dc_delete
not_operator	3.2.2.686	3-1340	con_man
null_access	3.2.2.445	3-941	static_exp
number	3.2.2.226	3-583	objects
numeric_literal	3.2.2.446	3-942	static_exp
obj_init	3.2.2.228	3-586	objects
object_assoc	3.2.2.192	3-527	gen_ren
only_zeros_remaining	3.2.2.685	3-1338	con_man
operator_def	3.2.2.4	3-189	basic_units
operator_kind	3.2.2.602	3-1194	operator_replacement
opt_driver	3.2.2.694	3-1353	gop_driver_p
opt_exp	3.2.2.717	3-1391	f_control
opt_sta_driver	3.2.2.799	3-1524	opt_statistics
opt_stms	3.2.2.718	3-1392	f_control
optimize	3.2.2.292	3-692	prag_attr
optimize	3.2.2.279	3-670	prag_attr
optimize	3.2.2.268	3-652	prag_attr
optimize	3.2.2.304	3-712	prag_attr
optimize	3.2.2.257	3-634	prag_attr
original_def	3.2.2.601	3-1192	operator_replacement
other_nodes	3.2.2.447	3-943	static_exp
ov_abort	3.2.2.387	3-853	ovp_trav
ov_accept	3.2.2.383	3-848	ovp_trav
ov_access	3.2.2.365	3-824	ovp_trav
ov_address	3.2.2.394	3-863	ovp_trav
ov_aggregate	3.2.2.323	3-752	ovp_expr

ov_all	3.2.2.316	3-735	ovp_expr
ov_allocator	3.2.2.331	3-770	ovp_expr
ov_apply	3.2.2.332	3-773	ovp_expr
ov_apply	3.2.2.358	3-813	ovp_ctxt
ov_array	3.2.2.361	3-819	ovp_trav
ov_assign	3.2.2.345	3-793	ovp_ctxt
ov_attr_call	3.2.2.318	3-740	ovp_expr
ov_attribute	3.2.2.317	3-737	ovp_expr
ov_binary	3.2.2.324	3-755	ovp_expr
ov_block	3.2.2.370	3-831	ovp_trav
ov_case	3.2.2.347	3-797	ovp_ctxt
ov_code	3.2.2.395	3-865	ovp_trav
ov_comp_rep	3.2.2.393	3-862	ovp_trav
ov_comp_unit	3.2.2.389	3-856	ovp_trav
ov_cond_cls	3.2.2.346	3-795	ovp_ctxt
ov_cond_entr	3.2.2.385	3-851	ovp_trav
ov_constant	3.2.2.335	3-778	ovp_ctxt
ov_constrnd	3.2.2.338	3-782	ovp_ctxt
ov_convers	3.2.2.327	3-761	ovp_expr
ov_dcl_rep_s	3.2.2.381	3-845	ovp_trav
ov_decl_s	3.2.2.380	3-844	ovp_trav
ov_delay	3.2.2.355	3-808	ovp_ctxt
ov_derived	3.2.2.360	3-817	ovp_trav
ov_dscr_agg	3.2.2.311	3-724	ovp_expr
ov_dscrtrngs	3.2.2.362	3-820	ovp_trav
ov_entry	3.2.2.353	3-805	ovp_ctxt
ov_entry_cal	3.2.2.354	3-807	ovp_ctxt
ov_exit	3.2.2.349	3-800	ovp_ctxt
ov_fixedcon	3.2.2.343	3-790	ovp_ctxt
ov_fixedtype	3.2.2.341	3-787	ovp_ctxt
ov_floatcon	3.2.2.342	3-789	ovp_ctxt
ov_floattype	3.2.2.340	3-785	ovp_ctxt
ov_for_rev	3.2.2.348	3-798	ovp_ctxt
ov_func_call	3.2.2.329	3-765	ovp_expr
ov_function	3.2.2.373	3-835	ovp_trav
ov_gen_asscs	3.2.2.391	3-859	ovp_trav
ov_if	3.2.2.367	3-827	ovp_trav
ov_in	3.2.2.351	3-803	ovp_ctxt
ov_in_out	3.2.2.375	3-837	ovp_trav
ov_indexed	3.2.2.313	3-728	ovp_expr
ov_integer	3.2.2.339	3-783	ovp_ctxt
ov_labeled	3.2.2.400	3-871	ovp_trav
ov_loop	3.2.2.368	3-828	ovp_trav
ov_membershp	3.2.2.325	3-757	ovp_expr
ov_named_stm	3.2.2.401	3-873	ovp_trav
ov_null_acc	3.2.2.321	3-747	ovp_expr
ov_num_lit	3.2.2.319	3-742	ovp_expr
ov_number	3.2.2.337	3-781	ovp_ctxt
ov_out	3.2.2.376	3-839	ovp_trav
ov_pack_body	3.2.2.382	3-847	ovp_trav
ov_pack_dcl	3.2.2.378	3-841	ovp_trav
ov_pack_spec	3.2.2.379	3-843	ovp_trav
ov_param_s	3.2.2.374	3-836	ovp_trav

sc_exit	3.2.2.537	3-1091	scp_stmt
sc_fcn_call	3.2.2.523	3-1068	scp_expr
sc_fixed	3.2.2.475	3-990	scp_decl
sc_floatc	3.2.2.474	3-989	scp_decl
sc_floatt	3.2.2.473	3-987	scp_decl
sc_for_rev	3.2.2.535	3-1088	scp_stmt
sc_function	3.2.2.500	3-1031	scp_decl
sc_gen_parms	3.2.2.489	3-1012	scp_decl
sc_generic	3.2.2.488	3-1010	scp_decl
sc_goto	3.2.2.539	3-1094	scp_stmt
sc_if	3.2.2.530	3-1079	scp_stmt
sc_in	3.2.2.502	3-1034	scp_decl
sc_indexed	3.2.2.524	3-1070	scp_expr
sc_integer	3.2.2.472	3-985	scp_decl
sc_item_s	3.2.2.498	3-1028	scp_decl
sc_labeled	3.2.2.528	3-1076	scp_stmt
sc_loop	3.2.2.534	3-1086	scp_stmt
sc_mem_size	3.2.2.554	3-1120	scp_prag
sc_membershp	3.2.2.514	3-1053	scp_expr
sc_named_stm	3.2.2.533	3-1085	scp_stmt
sc_number	3.2.2.466	3-976	scp_decl
sc_optimize	3.2.2.555	3-1121	scp_prag
sc_out	3.2.2.503	3-1036	scp_decl
sc_pack_body	3.2.2.481	3-1000	scp_decl
sc_pack_decl	3.2.2.505	3-1039	scp_decl
sc_pack_spec	3.2.2.506	3-1041	scp_decl
sc_param_s	3.2.2.501	3-1033	scp_decl
sc_pragma	3.2.2.552	3-1116	scp_prag
sc_pragma_s	3.2.2.561	3-1130	scp_comp
sc_priority	3.2.2.556	3-1123	scp_prag
sc_private	3.2.2.482	3-1001	scp_decl
sc_proc_call	3.2.2.550	3-1113	scp_stmt
sc_procedure	3.2.2.499	3-1030	scp_decl
sc_qualified	3.2.2.516	3-1057	scp_expr
sc_raise	3.2.2.548	3-1110	scp_stmt
sc_range	3.2.2.471	3-984	scp_decl
sc_rec_rep	3.2.2.491	3-1015	scp_decl
sc_record	3.2.2.478	3-995	scp_decl
sc_rename	3.2.2.509	3-1045	scp_decl
sc_return	3.2.2.538	3-1093	scp_stmt
sc_select	3.2.2.543	3-1101	scp_stmt
sc_selected	3.2.2.519	3-1062	scp_expr
sc_simp_rep	3.2.2.490	3-1013	scp_decl
sc_slice	3.2.2.518	3-1060	scp_expr
sc_stm_s	3.2.2.527	3-1074	scp_stmt
sc_str_lit	3.2.2.522	3-1066	scp_expr
sc_strg_unit	3.2.2.557	3-1125	scp_prag
sc_subp_body	3.2.2.504	3-1038	scp_decl
sc_subp_decl	3.2.2.507	3-1042	scp_decl
sc_subtype	3.2.2.468	3-979	scp_decl
sc_subunit	3.2.2.562	3-1132	scp_comp
sc_suppress	3.2.2.564	3-1134	scp_tdep
sc_suppress	3.2.2.565	3-1136	scp_tdep

sc_system	3.2.2.558	3-1126	scp_prag
sc_task_body	3.2.2.485	3-1006	scp_decl
sc_task_decl	3.2.2.483	3-1003	scp_decl
sc_task_spec	3.2.2.484	3-1004	scp_decl
sc_terminate	3.2.2.549	3-1112	scp_stmt
sc_timed_ent	3.2.2.545	3-1105	scp_stmt
sc_title	3.2.2.553	3-1118	scp_prag
sc_type	3.2.2.467	3-978	scp_decl
sc_type_conv	3.2.2.515	3-1055	scp_expr
sc_used_obj	3.2.2.525	3-1071	scp_expr
sc_var	3.2.2.465	3-974	scp_decl
sc_variant	3.2.2.496	3-1023	scp_decl
sc_variant_p	3.2.2.479	3-997	scp_decl
sc_variant_s	3.2.2.495	3-1022	scp_decl
scalar_logical_opera	3.2.2.688	3-1343	con_man
scan	3.2.2.647	3-1276	formatter
scan	3.2.2.49	3-260	lexer
scanner	3.2.2.60	3-307	parser
search_used_dnts	3.2.2.169	3-487	search
seqx_walk	3.2.2.755	3-1457	cse_p
select_stm	3.2.2.130	3-417	stms
selected	3.2.2.450	3-948	static_exp
selection	3.2.2.157	3-463	exps
selection_check	3.2.2.742	3-1436	exp_eval_cc_p
set_cv_id_s	3.2.2.229	3-588	objects
set_dc_id_s	3.2.2.230	3-590	objects
set_depends	3.2.2.30	3-232	crl_esp
set_flag_option	3.2.2.809	3-1541	argparser
set_flags	3.2.2.840	3-1618	use_opt
set_option	3.2.2.810	3-1542	argparser
set_range_option	3.2.2.811	3-1544	argparser
set_save_option	3.2.2.812	3-1546	argparser
set_stmt_option	3.2.2.813	3-1548	argparser
set_use_option	3.2.2.814	3-1549	argparser
simple_rep	3.2.2.235	3-596	reps
slice	3.2.2.451	3-950	static_exp
spec_exp_eval	3.2.2.741	3-1435	exp_eval_cc_p
src_finish	3.2.2.831	3-1605	buildtree
src_start	3.2.2.832	3-1606	buildtree
stat_gath	3.2.2.663	3-1299	stats
stm	3.2.2.117	3-395	stms
stm_s	3.2.2.116	3-394	stms
stms_prcs	3.2.2.704	3-1371	data_col_p
store	3.2.2.36	3-239	rec_adv
store_data	3.2.2.34	3-237	crl_data
store_diags	3.2.2.657	3-1292	diagnose
store_options	3.2.2.815	3-1551	argparser
string_literal	3.2.2.452	3-951	static_exp
sub_info_proc	3.2.2.706	3-1375	data_col_p
subprg_items	3.2.2.711	3-1383	data_col_p
subprog_match	3.2.2.200	3-543	gen_ren
subprog_redefinition	3.2.2.242	3-609	redefs
subprog_top	3.2.2.733	3-1420	f_data_flow_p

subprogram_body	3.2.2.6	3-192	basic_units
subprogram_call	3.2.2.732	3-1418	f_data_flow_p
subprogram_decl	3.2.2.2	3-186	basic_units
subprogram_def	3.2.2.5	3-191	basic_units
subprogram_formal	3.2.2.628	3-1236	generic_expansion
subprogram_renaming	3.2.2.190	3-523	gen_ren
subscripting	3.2.2.690	3-1347	con_man
subtract	3.2.2.669	3-1310	con_man
subtract_large_integ	3.2.2.674	3-1318	con_man
subtype_dec	3.2.2.218	3-571	types
suppress	3.2.2.305	3-714	prag_attr
suppress	3.2.2.280	3-672	prag_attr
suppress	3.2.2.269	3-654	prag_attr
suppress	3.2.2.293	3-694	prag_attr
suppress	3.2.2.258	3-635	prag_attr
system	3.2.2.298	3-702	prag_attr
system	3.2.2.286	3-682	prag_attr
system	3.2.2.281	3-673	prag_attr
system	3.2.2.250	3-622	prag_attr
system	3.2.2.262	3-642	prag_attr
take_snapshot	3.2.2.841	3-1619	use_opt
task_body	3.2.2.224	3-580	tasks
task_decl	3.2.2.222	3-577	tasks
task_renaming	3.2.2.189	3-522	gen_ren
task_spec	3.2.2.223	3-579	tasks
token_number	3.2.2.73	3-326	par_tabs
trace	3.2.2.107	3-380	maint_c
trace	3.2.2.309	3-720	maint_n
transform_return_sta	3.2.2.620	3-1222	inline_expansion
translate	3.2.2.61	3-309	parser
traverse2_stmt	3.2.2.105	3-378	context
traverse_body	3.2.2.102	3-373	context
type_conversion	3.2.2.162	3-474	exps
type_dec	3.2.2.219	3-573	types
type_formal	3.2.2.629	3-1237	generic_expansion
type_redef	3.2.2.243	3-611	redefs
type_specf	3.2.2.213	3-563	types
unary_walk	3.2.2.757	3-1460	cse_p
union_set	3.2.2.714	3-1387	tran_clo
union_stacks	3.2.2.773	3-1489	b_data_flow
unit_total	3.2.2.658	3-1293	diagnose
unmark_formals	3.2.2.607	3-1200	copy_subtree
up_stmt	3.2.2.833	3-1607	buildtree
update	3.2.2.174	3-497	derived
use_clause	3.2.2.103	3-374	context
use_clause	3.2.2.136	3-426	stms
use_obj_info	3.2.2.786	3-1506	usg_prcs
used_char	3.2.2.159	3-467	exps
used_char	3.2.2.453	3-953	static_exp
used_id	3.2.2.156	3-461	exps
used_object_id	3.2.2.454	3-954	static_exp
used_string	3.2.2.158	3-465	exps
usg_attaching	3.2.2.784	3-1503	usg_prcs