Annex P
(informative)
Syntax Summary

This Annex summarizes the complete syntax of the language. See 1.1.4 for a description of the notation used.

2.3:
```
identifier ::=
  identifier_start {identifier_start | identifier_extend}
```

2.3:
```
identifier_start ::=
    letter_uppercase
  | letter_lowercase
  | letter_titlecase
  | letter_modifier
  | letter_other
  | number_letter
```

2.3:
```
identifier_extend ::=
    mark_non_spacing
  | mark_spacing_combining
  | number_decimal
  | punctuation_connector
```

2.4:
```
numeric_literal ::= decimal_literal | based_literal
```

2.4.1:
```
decimal_literal ::= numeral [.numeral] [exponent]
```

2.4.1:
```
numeral ::= digit {[underline] digit}
```

2.4.1:
```
exponent ::= E [+] numeral | E – numeral
```

2.4.1:
```
digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

2.4.2:
```
based_literal ::=
  base # based_numeral [.based_numeral] # [exponent]
```

2.4.2:
```
base ::= numeral
```

2.4.2:
```
based_numeral ::=
  extended_digit {[underline] extended_digit}
```

2.4.2:
```
extended_digit ::= digit | A | B | C | D | E | F
```

2.5:
```
character_literal ::= 'graphic_character'
```

2.6:
```
string_literal ::= "{string_element}"
```

2.6:
```
string_element ::= "" | non_quotation_mark_graphic_character
```

2.7:
comment ::= --{*non_end_of_line*_character}

2.8:
pragma ::=
  pragma identifier [(pragma_argument_association {, pragma_argument_association})];

2.8:
pragma_argument_association ::=
    [*pragma_argument*_identifier =>] name
  | [*pragma_argument*_identifier =>] expression
  | *pragma_argument*_aspect_mark => name
  | *pragma_argument*_aspect_mark => expression

3.1:
basic_declaration ::=
    type_declaration | subtype_declaration
  | object_declaration | number_declaration
  | subprogram_declaration | abstract_subprogram_declaration
  | null_procedure_declaration | expression_function_declaration
  | package_declaration | renaming_declaration
  | exception_declaration | generic_declaration
  | generic_instantiation

3.1:
defining_identifier ::= identifier

3.2.1:
type_declaration ::= full_type_declaration
  | incomplete_type_declaration
  | private_type_declaration
  | private_extension_declaration

3.2.1:
full_type_declaration ::=
    type defining_identifier [known_discriminant_part] **is** type_definition
      [aspect_specification];
  | task_type_declaration
  | protected_type_declaration

3.2.1:
type_definition ::=
    enumeration_type_definition | integer_type_definition
  | real_type_definition | array_type_definition
  | record_type_definition | access_type_definition
  | derived_type_definition | interface_type_definition

3.2.2:
subtype_declaration ::=
  subtype defining_identifier **is** subtype_indication
      [aspect_specification];

3.2.2:
subtype_indication ::= [null_exclusion] subtype_mark [constraint]

3.2.2:
subtype_mark ::= *subtype*_name

3.2.2:
constraint ::= scalar_constraint | composite_constraint

3.2.2:
scalar_constraint ::=
    range_constraint | digits_constraint | delta_constraint

3.2.2:

composite_constraint ::=
    index_constraint | discriminant_constraint

3.3.1:
object_declaration ::=
   defining_identifier_list : [**aliased**] [**constant**] subtype_indication [:= expression]
      [aspect_specification];
 | defining_identifier_list : [**aliased**] [**constant**] access_definition [:= expression]
      [aspect_specification];
 | defining_identifier_list : [**aliased**] [**constant**] array_type_definition [:= expression]
      [aspect_specification];
 | single_task_declaration
 | single_protected_declaration

3.3.1:
defining_identifier_list ::=
 defining_identifier {, defining_identifier}

3.3.2:
number_declaration ::=
    defining_identifier_list : **constant** := *static_*expression;

3.4:
derived_type_definition ::=
   [**abstract**] [**limited**] **new** *parent_*subtype_indication [[**and** interface_list] record_extension_part]

3.5:
range_constraint ::= **range** range

3.5:
range ::= range_attribute_reference
  | simple_expression .. simple_expression

3.5.1:
enumeration_type_definition ::=
  (enumeration_literal_specification {, enumeration_literal_specification})

3.5.1:
enumeration_literal_specification ::= defining_identifier | defining_character_literal

3.5.1:
defining_character_literal ::= character_literal

3.5.4:
integer_type_definition ::= signed_integer_type_definition | modular_type_definition

3.5.4:
signed_integer_type_definition ::= **range** *static_*simple_expression .. *static_*simple_expression

3.5.4:
modular_type_definition ::= **mod** *static_*expression

3.5.6:
real_type_definition ::=
  floating_point_definition | fixed_point_definition

3.5.7:
floating_point_definition ::=
 digits *static_*expression [real_range_specification]

3.5.7:
real_range_specification ::=
 range *static_*simple_expression .. *static_*simple_expression

3.5.9:
fixed_point_definition ::= ordinary_fixed_point_definition | decimal_fixed_point_definition

3.5.9:

ordinary_fixed_point_definition ::=
  delta *static*_expression real_range_specification

3.5.9:
decimal_fixed_point_definition ::=
  delta *static*_expression **digits** *static*_expression [real_range_specification]

3.5.9:
digits_constraint ::=
  digits *static*_expression [range_constraint]

3.6:
array_type_definition ::=
  unconstrained_array_definition | constrained_array_definition

3.6:
unconstrained_array_definition ::=
  array(index_subtype_definition {, index_subtype_definition}) **of** component_definition

3.6:
index_subtype_definition ::= subtype_mark **range** <>

3.6:
constrained_array_definition ::=
  array (discrete_subtype_definition {, discrete_subtype_definition}) **of** component_definition

3.6:
discrete_subtype_definition ::= *discrete*_subtype_indication | range

3.6:
component_definition ::=
  [**aliased**] subtype_indication
| [**aliased**] access_definition

3.6.1:
index_constraint ::= (discrete_range {, discrete_range})

3.6.1:
discrete_range ::= *discrete*_subtype_indication | range

3.7:
discriminant_part ::= unknown_discriminant_part | known_discriminant_part

3.7:
unknown_discriminant_part ::= (<>)

3.7:
known_discriminant_part ::=
  (discriminant_specification {; discriminant_specification})

3.7:
discriminant_specification ::=
  defining_identifier_list : [null_exclusion] subtype_mark [:= default_expression]
| defining_identifier_list : access_definition [:= default_expression]

3.7:
default_expression ::= expression

3.7.1:
discriminant_constraint ::=
  (discriminant_association {, discriminant_association})

3.7.1:
discriminant_association ::=
  [*discriminant*_selector_name {| *discriminant*_selector_name} =>] expression

3.8:
record_type_definition ::= [[**abstract**] **tagged**] [**limited**] record_definition

3.8:
```
record_definition ::=
   record
     component_list
   end record
 | null record
```

3.8:
```
component_list ::=
     component_item {component_item}
  | {component_item} variant_part
  |  null;
```

3.8:
```
component_item ::= component_declaration | aspect_clause
```

3.8:
```
component_declaration ::=
  defining_identifier_list : component_definition [:= default_expression]
      [aspect_specification];
```

3.8.1:
```
variant_part ::=
  case discriminant_direct_name is
     variant
     {variant}
  end case;
```

3.8.1:
```
variant ::=
  when discrete_choice_list =>
    component_list
```

3.8.1:
```
discrete_choice_list ::= discrete_choice {| discrete_choice}
```

3.8.1:
```
discrete_choice ::= choice_expression | discrete_subtype_indication | range | others
```

3.9.1:
```
record_extension_part ::= with record_definition
```

3.9.3:
```
abstract_subprogram_declaration ::=
   [overriding_indicator]
   subprogram_specification is abstract
      [aspect_specification];
```

3.9.4:
```
interface_type_definition ::=
   [limited | task | protected | synchronized] interface [and interface_list]
```

3.9.4:
```
interface_list ::= interface_subtype_mark {and interface_subtype_mark}
```

3.10:
```
access_type_definition ::=
   [null_exclusion] access_to_object_definition
 | [null_exclusion] access_to_subprogram_definition
```

3.10:
```
access_to_object_definition ::=
   access [general_access_modifier] subtype_indication
```

3.10:
```
general_access_modifier ::= all | constant
```

3.10:
access_to_subprogram_definition ::=
   access [**protected**] **procedure** parameter_profile
 | **access** [**protected**] **function** parameter_and_result_profile

3.10:
null_exclusion ::= **not null**

3.10:
access_definition ::=
   [null_exclusion] **access** [**constant**] subtype_mark
 | [null_exclusion] **access** [**protected**] **procedure** parameter_profile
 | [null_exclusion] **access** [**protected**] **function** parameter_and_result_profile

3.10.1:
incomplete_type_declaration ::= **type** defining_identifier [discriminant_part] [**is tagged**];

3.11:
declarative_part ::= {declarative_item}

3.11:
declarative_item ::=
   basic_declarative_item | body

3.11:
basic_declarative_item ::=
   basic_declaration | aspect_clause | use_clause

3.11:
body ::= proper_body | body_stub

3.11:
proper_body ::=
   subprogram_body | package_body | task_body | protected_body

4.1:
name ::=
    direct_name | explicit_dereference
  | indexed_component | slice
  | selected_component | attribute_reference
  | type_conversion | function_call
  | character_literal | qualified_expression
  | generalized_reference | generalized_indexing

4.1:
direct_name ::= identifier | operator_symbol

4.1:
prefix ::= name | implicit_dereference

4.1:
explicit_dereference ::= name.**all**

4.1:
implicit_dereference ::= name

4.1.1:
indexed_component ::= prefix(expression {, expression})

4.1.2:
slice ::= prefix(discrete_range)

4.1.3:
selected_component ::= prefix . selector_name

4.1.3:
selector_name ::= identifier | character_literal | operator_symbol

4.1.4:

attribute_reference ::= prefix'attribute_designator

4.1.4:
attribute_designator ::=
  identifier[(*static*_expression)]
 | Access | Delta | Digits | Mod

4.1.4:
range_attribute_reference ::= prefix'range_attribute_designator

4.1.4:
range_attribute_designator ::= Range[(*static*_expression)]

4.1.5:
generalized_reference ::= *reference_object*_name

4.1.6:
generalized_indexing ::= *indexable_container_object*_prefix actual_parameter_part

4.3:
aggregate ::= record_aggregate | extension_aggregate | array_aggregate

4.3.1:
record_aggregate ::= (record_component_association_list)

4.3.1:
record_component_association_list ::=
  record_component_association {, record_component_association}
 | **null record**

4.3.1:
record_component_association ::=
  [component_choice_list =>] expression
 | component_choice_list => <>

4.3.1:
component_choice_list ::=
  *component*_selector_name {| *component*_selector_name}
 | **others**

4.3.2:
extension_aggregate ::=
  (ancestor_part **with** record_component_association_list)

4.3.2:
ancestor_part ::= expression | subtype_mark

4.3.3:
array_aggregate ::=
 positional_array_aggregate | named_array_aggregate

4.3.3:
positional_array_aggregate ::=
  (expression, expression {, expression})
 | (expression {, expression}, **others** => expression)
 | (expression {, expression}, **others** => <>)

4.3.3:
named_array_aggregate ::=
  (array_component_association {, array_component_association})

4.3.3:
array_component_association ::=
  discrete_choice_list => expression
 | discrete_choice_list => <>

4.4:
expression ::=

```
    relation {and relation}    | relation {and then relation}
  | relation {or relation}     | relation {or else relation}
  | relation {xor relation}
```

4.4:
```
choice_expression ::=
    choice_relation {and choice_relation}
  | choice_relation {or choice_relation}
  | choice_relation {xor choice_relation}
  | choice_relation {and then choice_relation}
  | choice_relation {or else choice_relation}
```

4.4:
```
choice_relation ::=
    simple_expression [relational_operator simple_expression]
```

4.4:
```
relation ::=
    simple_expression [relational_operator simple_expression]
  | simple_expression [not] in membership_choice_list
```

4.4:
```
membership_choice_list ::= membership_choice {| membership_choice}
```

4.4:
```
membership_choice ::= choice_expression | range | subtype_mark
```

4.4:
```
simple_expression ::= [unary_adding_operator] term {binary_adding_operator term}
```

4.4:
```
term ::= factor {multiplying_operator factor}
```

4.4:
```
factor ::= primary [** primary] | abs primary | not primary
```

4.4:
```
primary ::=
   numeric_literal | null | string_literal | aggregate
 | name | allocator | (expression)
 | (conditional_expression) | (quantified_expression)
```

4.5:
```
logical_operator ::=                    and | or | xor
```

4.5:
```
relational_operator ::=                 = | /= | < | <= | > | >=
```

4.5:
```
binary_adding_operator ::=              + | – | &
```

4.5:
```
unary_adding_operator ::=               + | –
```

4.5:
```
multiplying_operator ::=                * | / | mod | rem
```

4.5:
```
highest_precedence_operator ::=         ** | abs | not
```

4.5.7:
```
conditional_expression ::= if_expression | case_expression
```

4.5.7:
```
if_expression ::=
   if condition then dependent_expression
   {elsif condition then dependent_expression}
   [else dependent_expression]
```

4.5.7:
condition ::= *boolean*_expression

4.5.7:
case_expression ::=
   case *selecting*_expression **is**
   case_expression_alternative {,
   case_expression_alternative}

4.5.7:
case_expression_alternative ::=
   when discrete_choice_list =>
      *dependent*_expression

4.5.8:
quantified_expression ::= **for** quantifier loop_parameter_specification => predicate
 | **for** quantifier iterator_specification => predicate

4.5.8:
quantifier ::= **all** | **some**

4.5.8:
predicate ::= *boolean*_expression

4.6:
type_conversion ::=
   subtype_mark(expression)
 | subtype_mark(name)

4.7:
qualified_expression ::=
  subtype_mark'(expression) | subtype_mark'aggregate

4.8:
allocator ::=
  new [subpool_specification] subtype_indication
 | **new** [subpool_specification] qualified_expression

4.8:
subpool_specification ::= (*subpool_handle*_name)

5.1:
sequence_of_statements ::= statement {statement} {label}

5.1:
statement ::=
  {label} simple_statement | {label} compound_statement

5.1:
simple_statement ::= null_statement
  | assignment_statement | exit_statement
  | goto_statement | procedure_call_statement
  | simple_return_statement | entry_call_statement
  | requeue_statement | delay_statement
  | abort_statement | raise_statement
  | code_statement

5.1:
compound_statement ::=
    if_statement | case_statement
  | loop_statement | block_statement
  | extended_return_statement
  | accept_statement | select_statement

5.1:
null_statement ::= **null**;

5.1:
label ::= <<*label*_statement_identifier>>

5.1:
statement_identifier ::= direct_name

5.2:
assignment_statement ::=
  *variable*_name := expression;

5.3:
if_statement ::=
   if condition **then**
     sequence_of_statements
  {**elsif** condition **then**
     sequence_of_statements}
  [**else**
     sequence_of_statements]
   end if;

5.4:
case_statement ::=
  case *selecting*_expression **is**
      case_statement_alternative
     {case_statement_alternative}
  end case;

5.4:
case_statement_alternative ::=
  when discrete_choice_list =>
     sequence_of_statements

5.5:
loop_statement ::=
  [*loop*_statement_identifier:]
     [iteration_scheme] **loop**
        sequence_of_statements
     end loop [*loop*_identifier];

5.5:
iteration_scheme ::= **while** condition
  | **for** loop_parameter_specification
  | **for** iterator_specification

5.5:
loop_parameter_specification ::=
  defining_identifier **in** [**reverse**] discrete_subtype_definition

5.5.2:
iterator_specification ::=
    defining_identifier **in** [**reverse**] *iterator*_name
  | defining_identifier [: subtype_indication] **of** [**reverse**] *iterable*_name

5.6:
block_statement ::=
  [*block*_statement_identifier:]
      [**declare**
          declarative_part]
       begin
          handled_sequence_of_statements
      end [*block*_identifier];

5.7:
exit_statement ::=
  exit [*loop*_name] [**when** condition];

5.8:
goto_statement ::= **goto** *label*_name;

6.1:
subprogram_declaration ::=
   [overriding_indicator]
   subprogram_specification
      [aspect_specification];

6.1:
subprogram_specification ::=
   procedure_specification
 | function_specification

6.1:
procedure_specification ::= **procedure** defining_program_unit_name parameter_profile

6.1:
function_specification ::= **function** defining_designator parameter_and_result_profile

6.1:
designator ::= [parent_unit_name .]identifier | operator_symbol

6.1:
defining_designator ::= defining_program_unit_name | defining_operator_symbol

6.1:
defining_program_unit_name ::= [parent_unit_name .]defining_identifier

6.1:
operator_symbol ::= string_literal

6.1:
defining_operator_symbol ::= operator_symbol

6.1:
parameter_profile ::= [formal_part]

6.1:
parameter_and_result_profile ::=
   [formal_part] **return** [null_exclusion] subtype_mark
 | [formal_part] **return** access_definition

6.1:
formal_part ::=
  (parameter_specification {; parameter_specification})

6.1:
parameter_specification ::=
   defining_identifier_list : [**aliased**] mode [null_exclusion] subtype_mark [:= default_expression]
 | defining_identifier_list : access_definition [:= default_expression]

6.1:
mode ::= [**in**] | **in out** | **out**

6.3:
subprogram_body ::=
   [overriding_indicator]
   subprogram_specification
     [aspect_specification] **is**
     declarative_part
   begin
      handled_sequence_of_statements
   end [designator];

6.4:
procedure_call_statement ::=
   *procedure*_name;

| *procedure_*prefix actual_parameter_part;

6.4:
function_call ::=
  *function_*name
| *function_*prefix actual_parameter_part

6.4:
actual_parameter_part ::=
  (parameter_association {, parameter_association})

6.4:
parameter_association ::=
  [*formal_parameter_*selector_name =>] explicit_actual_parameter

6.4:
explicit_actual_parameter ::= expression | *variable_*name

6.5:
simple_return_statement ::= **return** [expression];

6.5:
extended_return_object_declaration ::=
  defining_identifier : [**aliased**][**constant**] return_subtype_indication [:= expression]

6.5:
extended_return_statement ::=
  return extended_return_object_declaration [**do**
    handled_sequence_of_statements
  end return];

6.5:
return_subtype_indication ::= subtype_indication | access_definition

6.7:
null_procedure_declaration ::=
  [overriding_indicator]
  procedure_specification **is null**
    [aspect_specification];

6.8:
expression_function_declaration ::=
  [overriding_indicator]
  function_specification **is**
    (expression)
    [aspect_specification];

7.1:
package_declaration ::= package_specification;

7.1:
package_specification ::=
  package defining_program_unit_name
    [aspect_specification] **is**
    {basic_declarative_item}
  [**private**
    {basic_declarative_item}]
  end [[parent_unit_name.]identifier]

7.2:
package_body ::=
  package body defining_program_unit_name
    [aspect_specification] **is**
    declarative_part
  [**begin**
    handled_sequence_of_statements]

```
   end [[parent_unit_name.]identifier];
```

7.3:

```
private_type_declaration ::=
  type defining_identifier [discriminant_part] is [[abstract] tagged] [limited] private
    [aspect_specification];
```

7.3:

```
private_extension_declaration ::=
  type defining_identifier [discriminant_part] is
    [abstract] [limited | synchronized] new ancestor_subtype_indication
    [and interface_list] with private
      [aspect_specification];
```

8.3.1:

```
overriding_indicator ::= [not] overriding
```

8.4:

```
use_clause ::= use_package_clause | use_type_clause
```

8.4:

```
use_package_clause ::= use package_name {, package_name};
```

8.4:

```
use_type_clause ::= use [all] type subtype_mark {, subtype_mark};
```

8.5:

```
renaming_declaration ::=
     object_renaming_declaration
   | exception_renaming_declaration
   | package_renaming_declaration
   | subprogram_renaming_declaration
   | generic_renaming_declaration
```

8.5.1:

```
object_renaming_declaration ::=
    defining_identifier : [null_exclusion] subtype_mark renames object_name
       [aspect_specification];
  | defining_identifier : access_definition renames object_name
       [aspect_specification];
```

8.5.2:

```
exception_renaming_declaration ::= defining_identifier : exception renames exception_name
  [aspect_specification];
```

8.5.3:

```
package_renaming_declaration ::= package defining_program_unit_name renames package_name
  [aspect_specification];
```

8.5.4:

```
subprogram_renaming_declaration ::=
    [overriding_indicator]
    subprogram_specification renames callable_entity_name
       [aspect_specification];
```

8.5.5:

```
generic_renaming_declaration ::=
    generic package          defining_program_unit_name renames generic_package_name
       [aspect_specification];
  | generic procedure        defining_program_unit_name renames generic_procedure_name
       [aspect_specification];
  | generic function         defining_program_unit_name renames generic_function_name
       [aspect_specification];
```

9.1:

```
task_type_declaration ::=
```

```
   task type defining_identifier [known_discriminant_part]
       [aspect_specification] [is
     [new interface_list with]
     task_definition];
```

9.1:

```
single_task_declaration ::=
   task defining_identifier
       [aspect_specification][is
     [new interface_list with]
     task_definition];
```

9.1:

```
task_definition ::=
     {task_item}
  [ private
     {task_item}]
  end [task_identifier]
```

9.1:

```
task_item ::= entry_declaration | aspect_clause
```

9.1:

```
task_body ::=
   task body defining_identifier
       [aspect_specification] is
     declarative_part
   begin
     handled_sequence_of_statements
   end [task_identifier];
```

9.4:

```
protected_type_declaration ::=
  protected type defining_identifier [known_discriminant_part]
       [aspect_specification] is
     [new interface_list with]
     protected_definition;
```

9.4:

```
single_protected_declaration ::=
  protected defining_identifier
       [aspect_specification] is
     [new interface_list with]
     protected_definition;
```

9.4:

```
protected_definition ::=
    { protected_operation_declaration }
[ private
    { protected_element_declaration } ]
  end [protected_identifier]
```

9.4:

```
protected_operation_declaration ::= subprogram_declaration
     | entry_declaration
     | aspect_clause
```

9.4:

```
protected_element_declaration ::= protected_operation_declaration
     | component_declaration
```

9.4:

```
protected_body ::=
  protected body defining_identifier
       [aspect_specification] is
```

```
   { protected_operation_item }
 end [protected_identifier];
```

9.4:

```
protected_operation_item ::= subprogram_declaration
    | subprogram_body
    | entry_body
    | aspect_clause
```

9.5:

```
synchronization_kind ::= By_Entry | By_Protected_Procedure | Optional
```

9.5.2:

```
entry_declaration ::=
  [overriding_indicator]
  entry defining_identifier [(discrete_subtype_definition)] parameter_profile
    [aspect_specification];
```

9.5.2:

```
accept_statement ::=
  accept entry_direct_name [(entry_index)] parameter_profile [do
    handled_sequence_of_statements
  end [entry_identifier]];
```

9.5.2:

```
entry_index ::= expression
```

9.5.2:

```
entry_body ::=
 entry defining_identifier  entry_body_formal_part  entry_barrier is
   declarative_part
 begin
   handled_sequence_of_statements
 end [entry_identifier];
```

9.5.2:

```
entry_body_formal_part ::= [(entry_index_specification)] parameter_profile
```

9.5.2:

```
entry_barrier ::= when condition
```

9.5.2:

```
entry_index_specification ::= for defining_identifier in discrete_subtype_definition
```

9.5.3:

```
entry_call_statement ::= entry_name [actual_parameter_part];
```

9.5.4:

```
requeue_statement ::= requeue procedure_or_entry_name [with abort];
```

9.6:

```
delay_statement ::= delay_until_statement | delay_relative_statement
```

9.6:

```
delay_until_statement ::= delay until delay_expression;
```

9.6:

```
delay_relative_statement ::= delay delay_expression;
```

9.7:

```
select_statement ::=
  selective_accept
 | timed_entry_call
 | conditional_entry_call
 | asynchronous_select
```

9.7.1:

```
selective_accept ::=
```

```
 select
  [guard]
   select_alternative
{ or
  [guard]
   select_alternative }
[ else
  sequence_of_statements ]
 end select;
```

9.7.1:
```
guard ::= when condition =>
```

9.7.1:
```
select_alternative ::=
   accept_alternative
 | delay_alternative
 | terminate_alternative
```

9.7.1:
```
accept_alternative ::=
 accept_statement [sequence_of_statements]
```

9.7.1:
```
delay_alternative ::=
 delay_statement [sequence_of_statements]
```

9.7.1:
```
terminate_alternative ::= terminate;
```

9.7.2:
```
timed_entry_call ::=
 select
  entry_call_alternative
 or
  delay_alternative
 end select;
```

9.7.2:
```
entry_call_alternative ::=
 procedure_or_entry_call [sequence_of_statements]
```

9.7.2:
```
procedure_or_entry_call ::=
 procedure_call_statement | entry_call_statement
```

9.7.3:
```
conditional_entry_call ::=
 select
  entry_call_alternative
 else
  sequence_of_statements
 end select;
```

9.7.4:
```
asynchronous_select ::=
 select
  triggering_alternative
 then abort
  abortable_part
 end select;
```

9.7.4:
```
triggering_alternative ::= triggering_statement [sequence_of_statements]
```

9.7.4:

triggering_statement ::= procedure_or_entry_call | delay_statement

9.7.4:
abortable_part ::= sequence_of_statements

9.8:
abort_statement ::= **abort** *task*_name {, *task*_name};

10.1.1:
compilation ::= {compilation_unit}

10.1.1:
compilation_unit ::=
   context_clause library_item
 | context_clause subunit

10.1.1:
library_item ::= [**private**] library_unit_declaration
 | library_unit_body
 | [**private**] library_unit_renaming_declaration

10.1.1:
library_unit_declaration ::=
    subprogram_declaration | package_declaration
  | generic_declaration | generic_instantiation

10.1.1:
library_unit_renaming_declaration ::=
  package_renaming_declaration
| generic_renaming_declaration
| subprogram_renaming_declaration

10.1.1:
library_unit_body ::= subprogram_body | package_body

10.1.1:
parent_unit_name ::= name

10.1.2:
context_clause ::= {context_item}

10.1.2:
context_item ::= with_clause | use_clause

10.1.2:
with_clause ::= limited_with_clause | nonlimited_with_clause

10.1.2:
limited_with_clause ::= **limited** [**private**] **with** *library_unit*_name {, *library_unit*_name};

10.1.2:
nonlimited_with_clause ::= [**private**] **with** *library_unit*_name {, *library_unit*_name};

10.1.3:
body_stub ::= subprogram_body_stub | package_body_stub | task_body_stub | protected_body_stub

10.1.3:
subprogram_body_stub ::=
  [overriding_indicator]
  subprogram_specification **is separate**
    [aspect_specification];

10.1.3:
package_body_stub ::=
  package body defining_identifier **is separate**
    [aspect_specification];

10.1.3:
task_body_stub ::=

```
  task body defining_identifier is separate
    [aspect_specification];
```

10.1.3:

```
protected_body_stub ::=
  protected body defining_identifier is separate
    [aspect_specification];
```

10.1.3:

```
subunit ::= separate (parent_unit_name) proper_body
```

11.1:

```
exception_declaration ::= defining_identifier_list : exception
  [aspect_specification];
```

11.2:

```
handled_sequence_of_statements ::=
    sequence_of_statements
 [exception
    exception_handler
   {exception_handler}]
```

11.2:

```
exception_handler ::=
  when [choice_parameter_specification:] exception_choice {| exception_choice} =>
    sequence_of_statements
```

11.2:

```
choice_parameter_specification ::= defining_identifier
```

11.2:

```
exception_choice ::= exception_name | others
```

11.3:

```
raise_statement ::= raise;
     | raise exception_name [with string_expression];
```

12.1:

```
generic_declaration ::= generic_subprogram_declaration | generic_package_declaration
```

12.1:

```
generic_subprogram_declaration ::=
    generic_formal_part  subprogram_specification
      [aspect_specification];
```

12.1:

```
generic_package_declaration ::=
    generic_formal_part  package_specification;
```

12.1:

```
generic_formal_part ::= generic {generic_formal_parameter_declaration | use_clause}
```

12.1:

```
generic_formal_parameter_declaration ::=
     formal_object_declaration
   | formal_type_declaration
   | formal_subprogram_declaration
   | formal_package_declaration
```

12.3:

```
generic_instantiation ::=
    package defining_program_unit_name is
      new generic_package_name [generic_actual_part]
        [aspect_specification];
  | [overriding_indicator]
    procedure defining_program_unit_name is
      new generic_procedure_name [generic_actual_part]
```

```
        [aspect_specification];
  | [overriding_indicator]
    function defining_designator is
      new generic_function_name [generic_actual_part]
        [aspect_specification];
```

12.3:

```
generic_actual_part ::=
  (generic_association {, generic_association})
```

12.3:

```
generic_association ::=
  [generic_formal_parameter_selector_name =>] explicit_generic_actual_parameter
```

12.3:

```
explicit_generic_actual_parameter ::= expression | variable_name
  | subprogram_name | entry_name | subtype_mark
  | package_instance_name
```

12.4:

```
formal_object_declaration ::=
   defining_identifier_list : mode [null_exclusion] subtype_mark [:= default_expression]
      [aspect_specification];
 |  defining_identifier_list : mode access_definition [:= default_expression]
      [aspect_specification];
```

12.5:

```
formal_type_declaration ::=
     formal_complete_type_declaration
   | formal_incomplete_type_declaration
```

12.5:

```
formal_complete_type_declaration ::=
   type defining_identifier[discriminant_part] is formal_type_definition
      [aspect_specification];
```

12.5:

```
formal_incomplete_type_declaration ::=
   type defining_identifier[discriminant_part] [is tagged];
```

12.5:

```
formal_type_definition ::=
     formal_private_type_definition
   | formal_derived_type_definition
   | formal_discrete_type_definition
   | formal_signed_integer_type_definition
   | formal_modular_type_definition
   | formal_floating_point_definition
   | formal_ordinary_fixed_point_definition
   | formal_decimal_fixed_point_definition
   | formal_array_type_definition
   | formal_access_type_definition
   | formal_interface_type_definition
```

12.5.1:

```
formal_private_type_definition ::= [[abstract] tagged] [limited] private
```

12.5.1:

```
formal_derived_type_definition ::=
   [abstract] [limited | synchronized] new subtype_mark [[and interface_list]with private]
```

12.5.2:

```
formal_discrete_type_definition ::= (<>)
```

12.5.2:

```
formal_signed_integer_type_definition ::= range <>
```

12.5.2:
formal_modular_type_definition ::= **mod** <>

12.5.2:
formal_floating_point_definition ::= **digits** <>

12.5.2:
formal_ordinary_fixed_point_definition ::= **delta** <>

12.5.2:
formal_decimal_fixed_point_definition ::= **delta** <> **digits** <>

12.5.3:
formal_array_type_definition ::= array_type_definition

12.5.4:
formal_access_type_definition ::= access_type_definition

12.5.5:
formal_interface_type_definition ::= interface_type_definition

12.6:
formal_subprogram_declaration ::= formal_concrete_subprogram_declaration
  | formal_abstract_subprogram_declaration

12.6:
formal_concrete_subprogram_declaration ::=
  with subprogram_specification [**is** subprogram_default]
    [aspect_specification];

12.6:
formal_abstract_subprogram_declaration ::=
  with subprogram_specification **is abstract** [subprogram_default]
    [aspect_specification];

12.6:
subprogram_default ::= default_name | <> | **null**

12.6:
default_name ::= name

12.7:
formal_package_declaration ::=
  with package defining_identifier **is new** *generic_package_*name formal_package_actual_part
    [aspect_specification];

12.7:
formal_package_actual_part ::=
  ([**others** =>] <>)
 | [generic_actual_part]
 | (formal_package_association {, formal_package_association} [, **others** => <>])

12.7:
formal_package_association ::=
  generic_association
 | *generic_formal_parameter_*selector_name => <>

13.1:
aspect_clause ::= attribute_definition_clause
    | enumeration_representation_clause
    | record_representation_clause
    | at_clause

13.1:
local_name ::= direct_name
    | direct_name'attribute_designator
    | *library_unit_*name

13.1.1:
```
aspect_specification ::=
  with aspect_mark [=> aspect_definition] {,
       aspect_mark [=> aspect_definition] }
```

13.1.1:
```
aspect_mark ::= aspect_identifier['Class]
```

13.1.1:
```
aspect_definition ::= name | expression | identifier
```

13.3:
```
attribute_definition_clause ::=
    for local_name'attribute_designator use expression;
  | for local_name'attribute_designator use name;
```

13.4:
```
enumeration_representation_clause ::=
   for first_subtype_local_name use enumeration_aggregate;
```

13.4:
```
enumeration_aggregate ::= array_aggregate
```

13.5.1:
```
record_representation_clause ::=
   for first_subtype_local_name use
    record [mod_clause]
      {component_clause}
    end record;
```

13.5.1:
```
component_clause ::=
   component_local_name at position range first_bit .. last_bit;
```

13.5.1:
```
position ::= static_expression
```

13.5.1:
```
first_bit ::= static_simple_expression
```

13.5.1:
```
last_bit ::= static_simple_expression
```

13.8:
```
code_statement ::= qualified_expression;
```

13.11.3:
```
storage_pool_indicator ::= storage_pool_name | null
```

13.12:
```
restriction ::= restriction_identifier
   | restriction_parameter_identifier => restriction_parameter_argument
```

13.12:
```
restriction_parameter_argument ::= name | expression
```

J.3:
```
delta_constraint ::= delta static_expression [range_constraint]
```

J.7:
```
at_clause ::= for direct_name use at expression;
```

J.8:
```
mod_clause ::= at mod static_expression;
```

Syntax Cross Reference

1/3 In the following syntax cross reference, each syntactic category is followed by the subclause number where it is defined. In addition, each syntactic category *S* is followed by a list of the categories that use *S* in their definitions. For example, the first listing below shows that abort_statement appears in the definition of simple_statement.

Annex Q
(informative)
Language-Defined Entities

This annex lists the language-defined entities of the language. A list of language-defined library units can be found in Annex A, “Predefined Language Environment”. 1/2

Q.1 Language-Defined Packages

This subclause lists all language-defined packages. 1/3

Ada A.2(2)
Address_To_Access_Conversions
child of System 13.7.2(2)
Arithmetic
child of Ada.Calendar 9.6.1(8/2)
ASCII
in Standard A.1(36.3/2)
Assertions
child of Ada 11.4.2(12/2)
Asynchronous_Task_Control
child of Ada D.11(3/2)
Bounded
child of Ada.Strings A.4.4(3)
Bounded_IO
child of Ada.Text_IO A.10.11(3/2)
child of Ada.Wide_Text_IO A.11(4/3)
child of Ada.Wide_Wide_Text_IO A.11(4/3)
Bounded_Priority_Queues
child of Ada.Containers A.18.31(2/3)
Bounded_Synchronized_Queues
child of Ada.Containers A.18.29(2/3)
C
child of Interfaces B.3(4)
Calendar
child of Ada 9.6(10)
Characters
child of Ada A.3.1(2)
COBOL
child of Interfaces B.4(7)
Command_Line
child of Ada A.15(3)
Complex_Arrays
child of Ada.Numerics G.3.2(53/2)
Complex_Elementary_Functions
child of Ada.Numerics G.1.2(9/1)
Complex_Text_IO
child of Ada G.1.3(9.1/2)
Complex_Types
child of Ada.Numerics G.1.1(25/1)
Complex_IO
child of Ada.Text_IO G.1.3(3)
child of Ada.Wide_Text_IO G.1.4(1)
child of Ada.Wide_Wide_Text_IO G.1.5(1/2)
Constants
child of Ada.Strings.Maps A.4.6(3/2)
Containers
child of Ada A.18.1(3/2)
Conversions
child of Ada.Characters A.3.4(2/2)
child of Ada.Strings.UTF_Encoding A.4.11(15/3)
Decimal
child of Ada F.2(2)
Decimal_Conversions
in Interfaces.COBOL B.4(31)
Decimal_IO
in Ada.Text_IO A.10.1(73)
Decimal_Output
in Ada.Text_IO.Editing F.3.3(11)
Direct_IO
child of Ada A.8.4(2)
Directories
child of Ada A.16(3/2)
Discrete_Random
child of Ada.Numerics A.5.2(17)
Dispatching
child of Ada D.2.1(1.2/3)
Dispatching_Domains
child of System.Multiprocessors D.16.1(3/3)
Doubly_Linked_Lists
child of Ada.Containers A.18.3(5/3)
Dynamic_Priorities
child of Ada D.5.1(3/2)
EDF
child of Ada.Dispatching D.2.6(9/2)
child of Ada.Synchronous_Task_Control D.10(5.2/3)
Editing
child of Ada.Text_IO F.3.3(3)
child of Ada.Wide_Text_IO F.3.4(1)
child of Ada.Wide_Wide_Text_IO F.3.5(1/2)
Elementary_Functions
child of Ada.Numerics A.5.1(9/1)
Enumeration_IO
in Ada.Text_IO A.10.1(79)

Wide_Wide_Maps
  child of Ada.Strings A.4.8(3/2)
Wide_Wide_Strings
  child of Ada.Strings.UTF_Encoding A.4.11(38/3)
Wide_Wide_Unbounded
  child of Ada.Strings A.4.8(1/3)

Q.2 Language-Defined Types and Subtypes

1/3 This subclause lists all language-defined types and subtypes.

Address
  in System 13.7(12)
Alignment
  in Ada.Strings A.4.1(6)
Alphanumeric
  in Interfaces.COBOL B.4(16/3)
Any_Priority *subtype of* Integer
  in System 13.7(16)
Attribute_Handle
  in Ada.Task_Attributes C.7.2(3)
Barrier_Limit *subtype of* Positive
  in Ada.Synchronous_Barriers D.10.1(4/3)
Binary
  in Interfaces.COBOL B.4(10)
Binary_Format
  in Interfaces.COBOL B.4(24)
Bit_Order
  in System 13.7(15/2)
Boolean
  in Standard A.1(5)
Bounded_String
  in Ada.Strings.Bounded A.4.4(6)
Buffer_Type *subtype of* Storage_Array
  in Ada.Storage_IO A.9(4)
Byte
  in Interfaces.COBOL B.4(29/3)
Byte_Array
  in Interfaces.COBOL B.4(29/3)
C_float
  in Interfaces.C B.3(15)
Cause_Of_Termination
  in Ada.Task_Termination C.7.3(3/2)
char
  in Interfaces.C B.3(19)
char16_array
  in Interfaces.C B.3(39.5/3)
char16_t
  in Interfaces.C B.3(39.2/2)
char32_array
  in Interfaces.C B.3(39.14/3)
char32_t
  in Interfaces.C B.3(39.11/2)
char_array
  in Interfaces.C B.3(23/3)
char_array_access
  in Interfaces.C.Strings B.3.1(4)
Character
  in Standard A.1(35/3)
Character_Mapping
  in Ada.Strings.Maps A.4.2(20/2)
Character_Mapping_Function
  in Ada.Strings.Maps A.4.2(25)
Character_Range
  in Ada.Strings.Maps A.4.2(6)
Character_Ranges
  in Ada.Strings.Maps A.4.2(7)
Character_Sequence *subtype of* String
  in Ada.Strings.Maps A.4.2(16)
Character_Set
  in Ada.Strings.Maps A.4.2(4/2)
  in Interfaces.Fortran B.5(11)
chars_ptr
  in Interfaces.C.Strings B.3.1(5/2)
chars_ptr_array
  in Interfaces.C.Strings B.3.1(6/2)
COBOL_Character
  in Interfaces.COBOL B.4(13)
Complex
  in Ada.Numerics.Generic_Complex_Types G.1.1(3)
  in Interfaces.Fortran B.5(9)
Complex_Matrix
  in Ada.Numerics.Generic_Complex_Arrays G.3.2(4/2)
Complex_Vector
  in Ada.Numerics.Generic_Complex_Arrays G.3.2(4/2)
Constant_Reference_Type
  in Ada.Containers.Indefinite_Holders A.18.18(16/3)
  in Ada.Containers.Multiway_Trees A.18.10(28/3)
Controlled
  in Ada.Finalization 7.6(5/2)
Count
  in Ada.Direct_IO A.8.4(4)
  in Ada.Streams.Stream_IO A.12.1(7)
  in Ada.Text_IO A.10.1(5)
Count_Type
  in Ada.Containers A.18.1(5/2)
Country_Code
  in Ada.Locales A.19(4/3)
CPU *subtype of* CPU_Range
  in System.Multiprocessors D.16(4/3)
CPU_Range
  in System.Multiprocessors D.16(4/3)
CPU_Time
  in Ada.Execution_Time D.14(4/2)

Q.3 Language-Defined Subprograms

1/3 This subclause lists all language-defined subprograms.

Q.4 Language-Defined Exceptions

This subclause lists all language-defined exceptions. 1/3

Q.5 Language-Defined Objects

1/3 This subclause lists all language-defined constants, variables, named numbers, and enumeration literals.

Index

Index entries are given by paragraph number.

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y