

Details of Language Change

Changes shown in **red font**. Deletions shown in ~~strikethrough red font~~. Comments shown in **green font**. Text that indicates an LCS dependency is shown in **blue font**.

Twiki has not been kind in regards to font colors and strikethrough. Word and PDF versions of these changes are attached as separate files to this LCS.

LRM 4.2.1 General (Subprogram declarations)

Page 19 near bottom

```
function_specification ::=  
  
[ pure | impure ] function designator  
  
subprogram_header  
  
[ [ parameter ] ( formal_parameter_list ) ] return [return_identifier :] type_mark
```

Page 20 first paragraph

The specification of a procedure specifies its designator, its generics (if any), and its formal parameters (if any). The specification of a function specifies its designator, its generics (if any), its formal parameters (if any), **the name that the function body may reference to retrieve attributes of return identifier (if any)**, the subtype of the returned value (the result subtype), and whether or not the function is pure. A function is impure if its specification contains the reserved word **impure**; otherwise, it is said to be pure. A procedure designator is always an identifier. A function designator is either an identifier or an operator symbol. A designator that is an operator symbol is used for the overloading of an operator (see 4.5.2). The sequence of characters represented by an operator symbol shall be an operator belonging to one of the classes of operators defined in 9.2. Extra spaces are not allowed in an operator symbol, and the case of letters is not significant.

Page 20 Add after the first paragraph

Add the following new material in entirety after the first paragraph (which ends with the sentence “Extra spaces are not allowed in an operator symbol, and the case of letters is not significant.”)

A function call with a return identifier shall have a return type which is either a scalar type or an array type. A function call with a return identifier shall appear only in one of the following contexts:

- a) **as the expression defining the initial value in a constant or variable declaration, where the corresponding subtype T fulfills the requirements below,**
- b) **as the expression defining the default value in a signal declaration, where the corresponding subtype T fulfills the requirements below,**
- c) **as a value expression in an assignment statement, where the target is a declared object (or a member thereof) and the subtype T of the target fulfills the requirements below,**
- d) **as an actual associated with a formal parameter, formal generic, or formal port (or member thereof), where the subtype T of the formal fulfills the requirements below,**

e) as the operand of a qualified expression, where the type mark T fulfills the requirements below.

The subtype T shall meet the following requirements:

- if the return type mark of the function is an array type, then T shall be an array subtype whose index ranges are defined by a constraint,
- if the return type mark of the function is a scalar type, then no further restrictions apply.

If the above conditions are met, then the target identifier is an alias for the subtype T. The target identifier shall be only used to determine attributes of the subtype T.

For an attribute name whose prefix is the return identifier and the return type mark is a scalar type or subtype, the attribute designator shall be one of BASE, LEFT, RIGHT, HIGH, LOW, ASCENDING, RANGE, REVERSE_RANGE, SUBTYPE defined in 16.2.2 (Predefined attributes of types and objects). It is an error to reference any other attributes. Reviewer's note: RANGE and REVERSE_RANGE are not currently in the LRM. They are being added by LCS-2016-018.

For an attribute name whose prefix is the return identifier and the return type mark is an array type or subtype, the attribute designator shall be one of the attributes defined in 16.2.3 (Predefined attributes of arrays).

LRM Annex C Syntax Summary Changes

Page 489 near middle of the page

Changes are shown in red font.

```
interface_function_specification ::= [§ 6.5.4]  
[ pure | impure ] function designator  
[ [ parameter ] ( formal_parameter_list ) ] return [return_identifier :] type_mark
```