Details of Language Change

Changes shown in red font. Deletions shown in strikethrough red font. Comments shown in green 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 4.2.2.1 Formal parameter lists

Page 21 Immediately before section 4.2.2.2

Reviewer’s note: I think the striking of the first sentence “Attributes of an actual...” and the addition of the phrase “or an unconstrained array object” in the following paragraph should be needed anyway. Currently, inside of a subprogram, if a variable array ‘V’ is defined as an unconstrained array, then references to V’left return the attribute of the actual that is associated with V when the subprogram is called, not the attribute of V which is unconstrained and, I think, undefined.

NOTE-- Attributes of an actual are never passed into a subprogram. References to an attribute of a formal parameter are legal only if that formal has such an attribute. For an unconstrained array object, or certain attributes of a scalar object, such references retrieve the value of the attribute associated with the actual, otherwise such references retrieve the value of the attribute associated with the formal.

LRM 4.2.2.2 Constant and variable parameters

Page 21, near middle

Reviewer’s note: I think the addition of “and attributes” in the following paragraph should be needed anyway. Currently, inside of a subprogram, if a variable array ‘V’ is defined as an unconstrained array, then references to V’left return the attribute of the actual designator that is associated with V when the subprogram is called, not the attribute of V which is unconstrained.

For parameters of class constant or variable, only the values and attributes of the actual or formal are transferred into or out of the subprogram call. The manner of such transfers, and the accompanying access privileges that are granted for constant and variable parameters, are described in this subclause.

Reviewer’s notes: Each subprogram parameter has a subtype indication which may include an optional range constraint. The new term ‘explicit_constraint_subtype_indication’ that is about to be defined here is to be exactly the same thing as the currently defined ‘subtype_indication’ but for the subset that includes an explicit constraint. The intention is to distinguish between the following two types of parameter definitions:

- Function foo(a: integer) return integer; -- Parameter ‘a’ does not meet conditions for explicit_constraint_subtype_indication
- Function foo(a: integer range 1 to 10) return integer; -- Parameter ‘a’ does meet conditions for explicit_constraint_subtype_indication
Current subtype indication definition is:

```
subtype indication ::= [§ 6.3] [resolution_indication] type_mark [constraint ]
```

```
subtype_indication_with_explicit_constraint ::= [ resolution_indication ] type_mark constraint
```

For a nonforeign subprogram having a parameter of class `constant` or `variable` and a scalar subtype:

If the subtype indication of the interface declaration of the formal parameter is not a subtype indication with an explicit constraint:

- If the actual is a declared object, or a subelement thereof, the LEFT, RIGHT, HIGH, LOW, RANGE and REVERSE_RANGE object attributes of the parameter shall be assigned from the corresponding attributes of the subtype of the actual.
- If the actual is not a declared object, or a subelement thereof:
  - LEFT, RIGHT, HIGH, LOW object attributes of the parameter shall be assigned from the value of the actual designator.
  - RANGE attribute shall be assigned an ascending range
  - REVERSE_RANGE shall be assigned a descending range
- All attributes other than LEFT, RIGHT, HIGH, LOW, RANGE and REVERSE_RANGE are set to the corresponding attributes of the formal.

If the subtype indication of the interface declaration of the formal parameter is a subtype indication with an explicit constraint:

- All object attributes of the parameter shall be assigned from the corresponding attributes of the subtype of the formal

---

**LRM 4.2.2.3 Signal parameters**

**Page 22 near middle**

For a signal parameter of mode in or inout, the actual signal is associated with the corresponding formal signal parameter at the start of each call. Thereafter, during the execution of the subprogram body, a reference to the formal signal parameter within an expression is equivalent to a reference to the actual signal.

For a nonforeign subprogram having a parameter of class `signal` and a scalar subtype:

If the subtype indication of the interface declaration of the formal parameter is not a subtype indication with an explicit constraint:

- If the actual is a declared object, or a subelement thereof, the LEFT, RIGHT, HIGH, LOW, RANGE and REVERSE_RANGE object attributes of the parameter shall be assigned from the corresponding attributes of the subtype of the actual.
- If the actual is not a declared object, or a subelement thereof:
  - LEFT, RIGHT, HIGH, LOW object attributes of the parameter shall be assigned from the value of the actual designator.
  - RANGE attribute shall be assigned an ascending range
  - REVERSE_RANGE shall be assigned a descending range
- All attributes other than LEFT, RIGHT, HIGH, LOW, RANGE and REVERSE_RANGE are set to the corresponding attributes of the formal.

If the subtype indication of the interface declaration of the formal parameter is a subtype indication with an explicit constraint:
- All object attributes of the parameter shall be assigned from the corresponding attributes of the subtype of the formal