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 a scalar object or an unconstrained array object, s~~S~~uch 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 that is associated with V when the subprogram is called, not the attribute of V which is unconstrained.

Reference to 'section 16.2.2' in the added text below refers to the LRM section as redefined by LCS-2016-018.

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.

For a nonforeign subprogram having a parameter of class constant or variable, if the actual meets any of the conditions of 'Determinable index range conditions', the attributes of the parameter shall be assigned from the corresponding attributes of the actual. If the actual does not meet any of the conditions of 'Determinable index range conditions', the object attribute (see section 16.2.2) of the actual of each of the parameters shall be assigned as:

- BASE, SUBTYPE: Copy from the BASE, SUBTYPE type attributes of the interface formal parameter
- LEFT, RIGHT, HIGH, LOW: Set to the value of the object attributes of the actual
- ASCENDING: Set to TRUE (Or I suppose it could be "Set to the 'ASCENDING object attribute of the formal parameter)

Reviewer's note: The list of 'Determinable index range conditions' listed here are copied from section 9.3.3 which defines the conditions under which "The index range of an array aggregate that has an others choice shall be determinable from the context". The following changes are intended to create a list of conditions under which "The index range of a scalar object that shall be determinable from the context"
- Change "fully constrained array subtype" to "scalar subtype with an explicit range constraint"
- Delete bullets j) and k) which have to do with aggregates which have no mapping to scalars

Determinable index range conditions
a) As an actual associated with a formal parameter, formal generic, or formal port (or member thereof), where the formal (or the member) is declared to be a scalar subtype with an explicit range constraint
b) As the default expression defining the default initial value of a port declared to be a scalar subtype with an explicit range constraint
c) As the default expression for a generic constant declared to be a scalar subtype with an explicit range constraint
d) As the result expression of a function, where the corresponding function result type is a fully constrained subtype
e) As a value expression in an assignment statement, where the target is a declared object (or member thereof), and either the subtype of the target is a fully constrained subtype
f) As the expression defining the initial value of a constant or variable object, where that object is declared to be a scalar subtype with an explicit range constraint
g) As the expression defining the default values of variables in a variable declaration, where the corresponding subtype is a fully constrained subtype
h) As the expression defining the value of an attribute in an attribute specification, where that attribute is declared to be a scalar subtype with an explicit range constraint
i) As the operand of a qualified expression whose type mark denotes a fully constrained subtype

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. A reference to an attribute of the formal signal parameter of a scalar object is equivalent to a reference to the object attribute of the actual scalar object signal.