To: **X3J3**
From: **/interop**
Subject: **Edits for ENUMs**

Add type alias, enumerations, and procedure declarations to table 2.1, "Requirements on statement ordering".
Page 13, table 2.1 add to the block starting on line 38 with the words "Derived-type definitions:

"type alias definitions,
enumeration declarations,
procedure declarations,"

Add type alias definitions, enumeration definitions, and procedure definition to table 2.2, "Statements allowed in scoping units".
Page 14, line 30 after "statement, " add

" type alias statements, enum statements, procedure declaration statements, "

Replace section 4.7
Delete page 56 line 20 through page 57 line 30 and replace with:
"4.7 Enumerations and enumerators

An enumeration is a type alias name for an integer type. An enumerator is a named integer constant. An enumeration definition specifies a set of enumerators associated with the type alias name of the enumeration. Enumerations and enumerators may interoperate (16.2) with the enumeration types and enumerators of the companion processor (2.5.10).

R453 *enum-def-stmt* **is** ENUM, BIND(C) :: *enum-def-list*
or ENUM [*kind-selector*] [::] *enum-def-list*

R454 *enum-def* **is** *type-alias-name* (*enumerator-list*)

R455 *enumerator* **is** *named-constant* [= *scalar-int-initialization-expr*]

The enumeration is treated as if it were explicitly declared in a type alias statement as a type alias for an integer whose kind parameter is determined as follows:

(1) If BIND(C) is present, the type kind is that which interoperates with the companion processor's integer type which the companion processor uses to represent the enumeration type containing the same values.

(2) If *kind-selector* is specified, the kind of the enumeration and the enumerators is the kind type specified by the *kind-selector*.

(3) If neither BIND(C) nor *kind-selector* is specified, the kind type of the enumeration

is processor dependent. The value of the kind type chosen by the processor shall characterize a representation method that can represent the values of all the enumerators in *enumerator-list.*

An enumerators is treated as if explicitly declared with type *type-alias-name* and the PARAMETER attribute. The enumerator is defined in accordance with the rules of intrinsic assignment (7.5.1.4) with the value determined as follows:

(1) If *scalar-int-initialization-expr* is present, the value of the enumerator is the result of *scalar-int-initialization-expr.*

(2) If *scalar-int-initialization-expr* is not present and the enumerator is the first enumerator in *enumerator-list* the enumerator has the value 0.

(3) If *scalar-int-initialization-expr* is not present and the enumerator is not the first enumerator in *enumerator-list*, it has the value of the constant expression formed by adding 1 to the value of the enumerator which immediately precedes it in *enumerator-list.*

NOTE 4.51
The declarations

```
ENUM (SELECTED_INT_KIND(1)) :: DIGITS (ZERO, ONE, TWO)
ENUM :: PRIMARY_COLORS (RED=4, BLUE=9, YELLOW)
TYPE (DIGITS)  :: X
```

are equivalent to the declarations

```
TYPEALIAS :: DIGITS => INTEGER (SELECTED_INT_KIND(1))
TYPE (DIGITS), PARAMETER :: ZERO = 0, ONE = 1, TWO =2
TYPE (DIGITS) :: X

! Note that the kind type parameter for PRIMARY_COLORS is processor dependent, but
! the processor must select a kind type sufficient to represent the value of YELLOW, the
! enumerator whose value (10) is the largest of  all enumerators associated with
! PRIMARY_COLORS.  The following declaration is one possibility for PRIMARY_COLORS.
TYPEALIAS :: PRIMARY_COLORS => INTEGER (SELECTED_INT_KIND(2))
TYPE (PRIMARY_COLORS), PARAMETER :: RED = 4, BLUE = 9, YELLOW = 10        "
```