To: J3
From: interop
Subject: Enumeration types
Date: November 13, 1998
References: 98-165r1, 98-171r1, 98-194r1

The specifications were adopted in the february 1998 meeting. The syntax was adopted in the november 1998 meeting. The edits are in the form of a J3 note.

1. Specifications

It is in the context of interoperability of Fortran and C that a correspondance for the enums allowed in C is needed in Fortran. The facility in Fortran shall be used to pass C enum values in the variable list of a C function, both from C into Fortran and from Fortran into C. The C enum facility is described in the C9X committee draft, section 6.7.2.2. This facility will also be available as a alternative method of specifying integer named constants.

2. Syntax

To declare enumeration types and their literals, the ENUM statement is introduced:

enum-def-stmt	**is** ENUM, BIND(C) :: *enum-def-list* **or** ENUM [*kind-selector*] [::] *enum-def-list*
enum-def	**is** *type-alias-name (enumerator-list)*
enumerator	**is** *named-constant* [= *scalar-int-initialization-expr*]

The *type-alias-name* is used as type name in the definition of an enumeration type. It defines a type alias for an integer type with a kind type parameter as specified below.

The *named-constant*s in the *enumerator-list* are declared of type integer. The kind of integer used to represent the *named-constant* and the integer type of *type-alias-name* shall be one of the following:

- If BIND(C) is specified, then the kind is selected for the *named-constant*s by the processor, and will be of a kind that interoperates with the C integer type that the target C processor would use to represent an enum type containing the same values. The integer type of *type-alias-name* shall have the same kind type parameter.
- If *kind-selector* is specified, then the kind type parameter specified in the *kind-selector* is used to represent the *named-constant* and the integer type of *type-alias-name*.
- If neither BIND(C), nor *kind-selector* is specified, then the kind is selected for the *named-constant*s by the processor. A kind, c.q. representation method will be chosen, such that the largest *scalar-int-initialization-expr* fits in it.

An enumerator with = defines an integer constant as the value of the corresponding named

constant. If the first enumerator has no =, then the value of its named constant is 0. Each subsequent enumerator with no = defines its named constant as the value of the constant expression obtained by adding 1 to the value of the previous named constant.

The use of enumerators with = may produce named constants with values that duplicate other values in the same enumeration.

3. Edits

All references are to J3/98-007r3.

- After [10:12], add to R207:

  or *enum-def-stmt*

- After the following subsection before [56:14], and renumber subsequent subsections.

4.6 Enumerators

begin J3 note

Enumerators provide a method of defining integer named constants. It is possible to pass C enum values in the argument list of a C function, both from C into Fortran and from Fortran into C.

R4xx *enum-def-stmt*	**is** ENUM, BIND(C) :: *enum-def-list* **or** ENUM [*kind-selector*] [::] *enum-def-list*
R4xx *enum-def*	**is** *type-alias-name (enumerator-list)*
R4xx *enumerator*	**is** *named-constant* [= *scalar-int-initialization-expr*]

The *type-alias-name* is used as type name in the definition of an enumeration type. It defines a type alias for an integer type with a kind type parameter as specified below.

The *named-constant*s in the *enumerator-list* are declared of type integer. The kind of integer used to represent the *named-constant* and the integer type of *type-alias-name* shall be one of the following:

- If BIND(C) is specified, then the kind is selected for the *named-constant*s by the processor, and shall be of a kind that interoperates with the C integer type that the target C processor would use to represent an enum type containing the same values. The integer type of *type-alias-name* shall have the same kind type parameter.
- If *kind-selector* is specified, then the kind type parameter specified in the *kind-selector* is used to represent the *named-constant* and the integer type of *type-alias-name*.
- If neither BIND(C), nor *kind-selector* is specified, then the kind is selected for the *named-constant*s by the processor. A kind, c.q. representation method will be chosen, such that the largest *scalar-int-initialization-expr* fits in it.

An enumerator with = defines an integer constant as the value of the corresponding named constant. If the first enumerator has no =, then the value of its named constant is 0. Each sub-

sequent enumerator with no = defines its named constant as the value of the constant expression obtained by adding 1 to the value of the previous named constant.

begin note
The use of enumerators with = may produce named constants with values that duplicate other values in the same enumeration.
end note

end J3 note

4. Example

Fortran main:

```
program main
   use iso_c_types
   implicit none
   integer (c_int) :: l
   enum, bind(c) :: xen(red=1, green, blue=30)
   type(xen) :: color
   interface block
      bind(c, name='Csub') subroutine c_sub(l, color)
      use iso_c_types
      integer(c_int) :: l
      enum, bind(c) :: xen(red=1, green, blue=30)
      type (xen) :: color
      end subroutine c_sub
   end interface
   .
   .
   call c_sub(l, color)
   if (color.eq.red) <do something>
   if (color.eq.green) <do something else>
   .
   .
end program main
```

C sub:

```
enum couleur {red=1, green, blue=30};
void Csub(l, couleur)
int l;
{
.
.
}
```