ISO/IEC JTC1/SC22/WG5 N1299

From: Michael Hennecke
To: WG5 and J3
Subject: Interoperability PDTR - C enum support
Date: 1997/07/23

This document provides the specifications to add C enum support to WG5/N1277. If accepted, it could replace section 3.3.3 of the PDTR.

DISCUSSION:

Enum support has been repeatedly asked for, and some APIs like X windows heavily use enumerations. So supporting them is useful. It has been proposed to introduce new kind parameter names like C_SHORT_ENUM which designates the type a C procressor chooses for all enumerations whose constants fall within the range of C short integers. This has two shortcomings:

(1) it is very difficult for a user to tell the correct kind parameter name, taht is to find out if a given value like 33333 is in the range of C short. This would need comparison of the largest enumeration constant to the SHRT_MAX limit and changing the kind parameter from C_SHORT_ENUM to C_LONG_ENUM, say, all within a <kind-selector>.

(2) it is not guaranteed that the C processor maps all enums in the C short range to one and the same integer type

It has also been proposed to introduce a C_SELECTED_ENUM_KIND procedure which takes a LOW_ENUM and HIGH_ENUM argument. This is more user-friendly but still dangerous:

(1) If on system A an enumeration RGB is

```
  enum RGB { Red=1, Green, Blue } ;
```

and on system B that enum is

```
  enum RGB { Red=16, Green=8, Blue=0 } ;
```

the selected kind type for the enum on system A is

```
  C_SELECTED_ENUM_KIND(LOW_ENUM=Red, HIGH_ENUM=Blue)
```

whereas on system B it is

```
  C_SELECTED_ENUM_KIND(LOW_ENUM=Blue, HIGH_ENUM=Red)
```

which may be a cause of trouble if overlooked. It is preferable that the definition of the enumeration constants is the only place which needs change when moving from system A to system B.

(2) it is not guaranteed that the C processor bases its choice only on the lowest and highest value of the enumeration constants.

The text below circumvents all of these problems.

UNRESOLVED ISSUE:

How do we ensure that C_ENUM_KIND results can be used as initialization expression in kind selectors? Probably the only way is to add C_ENUM_KIND to (5) in the list under "initialization expression" in 7.1.6.1 of F95 (page 94)?

SUGGESTED NEW PDTR TEXT:

3.3.3 C enumerated types

Fortran does not support enumerated types. But since C enumeration constants and types have C integral types, they can be mapped to Fortran integer types of suitable kind type parameters.
This section provides the means to bind C enumerated types to Fortran integer types.

All C ``enumerators'' (the <enumeration constants>) are constants of the C type #int# and shall be mapped to Fortran constants of type INTEGER(C_INT) which are initialized with the same values as the respective C enumerators.

Note 3.13
For example, if a C enumeration is declared as

```
enum RGB { Red=1, Green, Blue } ;
```

the enumeration constants may be declared in Fortran as

```
INTEGER(C_INT), PARAMETER :: Red=1, Green=2, Blue=3
```

Note that C enumerator names do not have their own name classes and must be distinct from all other enumerator names and other local names. So there is no name class problem in using individual integer constants on the Fortran side.

The C integer type chosen for a given enumeration type is implementation-defined. It need not be #int# but only conformable to any C integer type which is capable of representing the value of that enumeration type's enumerators.
The module ISO_C shall make available a function C_ENUM_KIND which can be used to inquire the implementation-defined integer kind type parameter for a given enumeration.

3.3.3.1 C_ENUM_KIND (ENUM)

Description.
Returns the implementation-defined integer kind type parameter that corresponds to a given C enumeration type.

Class.
Transformational function.

Argument.

ENUM	shall be a rank-1 array of type INTEGER(C_INT). It is an INTENT(IN) argument. It contains the list of enumerator values of a C enumeration type.

Result Characteristics.
The result is a scalar of type default integer.

Result Value.
If the C type chosen for an enumeration type whose enumerators have the same values as the elements of ENUM is
(i) #signed char# or #unsigned char#, the result value is C_SCHAR.
(ii) #short# or #unsigned short#, the result value is C_SHRT.
(iii) #int# or #unsigned int#, the result value is C_INT.
(iv) #long# or #unsigned long#, the result value is C_LONG.

Example.
Given a C enumerated type

```
  enum Bool { False=0, True } ;
```

and the Fortran constants

```
  INTEGER(c_int), PARAMETER :: False=0, True=1
```

the result of C_ENUM_KIND((/ False, True /)) is the integer kind type parameter which identifies the C type chosen for #enum Bool#.

A type alias statement (3.3.9) may be used to make a type alias name for a C enumeration type available. This can but need not be identical to the tag of the C enumeration.

Note 3.13a
For example, a type alias name corresponding to #enum RGB# of Note 3.13 may be estalished as

```
  ALIAS :: RGB => INTEGER(C_ENUM_KIND((/ Red,Green,Blue /)))
```

using the Fortran constants declared in Note 3.13.