Date: 1999/06/10
To: J3
From: interop
Subject: Resolution of unresolved issues 147, 150, 152, 156, 158
References: J3/98-165r1, J3/98-195r1, J3/98-196r2, J3/98-239

Edits refer to J3/99-007R1

1. [10:36] Add to R214

   or *bind-stmt*

2. [69:40] Add to R504

   or BIND(C [, BINDNAME = *scalar-char-initialization-expr*])

3. [70:9-11] Delete J3 note 147

4. [72:4] Add constraints

Constraint: If a BIND(C) attribute is specified, the variable shall be declared in the specification part of a module.

Constraint: A *function-name* may not be given the BIND(C) attribute in a *type-declaration-stmt*

5. [85:43] Add new paragraph and note and renumber following notes.

5.1.2.15 BIND attribute

The BIND attribute specifies that the variable interoperates with a C variable with external linkage. Only one variable that is associated with a particular C variable with external linkage is permitted to be declared within a program.

NOTE 5.22

The BINDNAME= *bind-spec* is meant to allow the user to specify a companion processor name that is not valid a Fortran name, and provides a mechanism through which the processor can distinguish between upper and lower case. The name is a (potentially) mangled name, rather than the name that is actually specified in the companion processor code.

END NOTE 5.22

6. Add after [93:11] and renumber following sections and syntax rules

5.3.13 BIND statement

R539 *bind-stmt* **is** BIND(C [, BINDNAME= n
n *scalar-char-initialization-expr*) [::] *bind-entity-list*

R540 *bind-entity* **is** *object-name*
or / *common-block-name* /

The BIND statement specifies the BIND attribute (5.1.2.15) for all objects named in the bind entity list. Only one variable that is associated with a particular C variable with external linkage is permitted to be declared within a program.

NOTE 5.33

It is not allowed to have two or more variables equivalenced through the use of several BIND(C) statement. For example the following is not allowed:

BIND (C, BINDNAME = '_alpha_') alpha
BIND (C, BINDNAME = '_alpha_') beta

END NOTE 5.33

7. [285:14] Change constraint to:

Constraint: A NAME= *bind-spec* or BINDNAME= *bind-spec* shall not be specified in the *function-stmt* or *subroutine-stmt* of an abstract interface body (12.3.2.1) or an interface body for a dummy procedure.

8. [285:16-20] Delete J3 note 150

9. [285:32] Change "specfied" to "specified"

10. [285:33-37] Delete J3 note 152

11. [290:17] Add the following sentence:

At most one NAME= specifier is permitted to appear in a *bind-spec-list*.

An additional BINDNAME= specifier may also appear, followed by a scalar default initialization expression of type default character. More than one BINDNAME= specifier may appear in a *bind-spec-list* for a subprogram, but not in a *bind-spec-list* for a *function-stmt* or *subroutine-stmt* in an interface body. The value of the BINDNAME= specifier shall be the name by which a procedure defined by Fortran may be referenced from C.

Any leading and trailing blanks in the value of a NAME= specifier are ignored. The value of the NAME= specifier in the *bind-spec-list* for a *function-stmt* or *subroutine-stmt* in an interface body must correspond to some C function with the same name.

12. [291:24-34] Delete J3 note 156

13. [291:35-44] Replace Note 12.36 by

NOTE 12.36

The intent here is that NAME= allows the user to specify C names that are not valid Fortran names, and provides a mechanism through which the processor can distinguish between upper and lower case.

A processor shall give a unique label, often referred to as a binder name, to each external procedure in a program. The label is derived in some way from the name of the external procedure and need not be the same as the binding label.

A processor may permit a procedure defined by means of Fortran to be known by more than one binder name if it needs to be referenced from more than one companion processor, each with a different way of transforming an external name to a binder name.

The value of the BINDNAME= specifier is intended to specify one or more alternative names by which a procedure defined by Fortran may be referenced from C, when a user wants to build a library that supports multiple C processors at once. The name is a (potentially) mangled name, rather than the name that is actually specified in the C code.

This is not the only possible meaning of the BINDNAME= specifier; nor is the processor required to ascribe such a meaning to the specifier.

END NOTE 12.36

14. [292:1-11] Remove J3 note 158

15. [362:11] Add the following note and renumber following notes.

NOTE 14.2

Two external procedures might have the same name, but will still be distinct entities, because the values specified by NAME= specifiers might be different. For example,

```
program p
   interface
      bind(c,name='CSub') subroutine c_sub
      end subroutine c_sub
   end interface
   ....
   call f_sub
   ....
end program p

subroutine f_sub
   interface
      bind(c,name='CSub2') subroutine c_sub
      end subroutine c_sub
   end interface
```

```
   ....
end subroutine f_sub
```

END NOTE 14.2

16. [411:40] Add paragraph

16.2.7 Interoperation with C global variables

A C variable with external linkage interoperates with a variable declared in the scope of a module or with a common block.

The BIND(C) attribute shall only be specified for a variable if it is declared in the scope of a module. The variable shall interoperate with a C variable that has external linkage. The variable shall not be explicitly initialized, it shall not have the POINTER attribute, the ALLOCATABLE attribute, appear in an EQUIVALENCE statement or be a member of a common block.

If a common block is given the BIND(C) attribute, it shall be given the BIND(C) attribute in all scoping units in which it is declared. A C variable with external linkage interoperates with a common block that has the BIND(C) attribute, if the C variable is of a struct type and the variables that are members of the common block interoperate with corresponding components of the struct type, or if the common block contains a single variable, and the variable interoperates with the C variable.

A variable in a common block with the BIND(C) attribute shall not be explicitly initialized and it shall not be the parent object of an *equivalence-object* in an EQUIVALENCE statement (5.6.1).

If a variable or common block has the BIND(C) attribute, it has the SAVE attribute as well.

A variable with the BIND(C) attribute is a global entity of a program (14.1.1). Such an entity shall not be declared in more than one scoping unit of the program.

NOTE 16.15

The following is an example of the usage of bind(c) for variables and a common block:

```
module example_1
    integer, bind(c) :: i
    integer :: j, k
    bind(c) :: j
end module example1

program example_2
    common /com/ k
    bind(c) :: /com/
    ...
end program example2
```

END NOTE 16.15