J3/08-177 INTERPRETATION UPDATE PAGES

To: J3
From: Malcolm Cohen
Subject: Interpretation Update Pages: Changes to Standing Document 018
Date: 2008/05/13

This document contains the new and updated pages from 08-018r1.

That means it contains insertions for every interpretation edit that will be published in Fortran 2003 Corrigendum 3.

The following pages are intended for insertion into a loose-leaf binder version of 04-007. This document needs to be printed single-sided for this to work.

Interp **F03/0094**, Status: *Corrigendum 3*.

Ref: 4.5.5, C473, [58:14]

After "The dummy argument shall not"
Change "be INTENT(OUT)"
To "have the INTENT(OUT) or VALUE attribute",
Making the whole constraint read:

C473 (R454) A *final-subroutine-name* shall be the name of a module procedure with exactly one dummy argument. That argument shall be nonoptional and shall be a nonpointer, nonallocatable, nonpolymorphic variable of the derived type being defined. All length type parameters of the dummy argument shall be assumed. The dummy argument shall not ~~be~~ have the INTENT(OUT) or VALUE attribute.

Interp **F03/0006**, Status: *Corrigendum 1.*

Ref: 7.4.1.3, 1^{st} paragraph, [139:17]

Replace "the evaluation of all operations in *expr* and *variable*"
By "the evaluation of *expr* and the evaluation of all expressions in *variable*",
Making the whole paragraph read:

> Execution of an intrinsic assignment causes, in effect, the evaluation of the expression *expr* and all expressions within *variable* (7.1.8), the possible conversion of *expr* to the type and type parameters of *variable* (Table 7.9), and the definition of *variable* with the resulting value. The execution of the assignment shall have the same effect as if the evaluation of ~~all operations in *expr* and~~ *expr* and the evaluation of all expressions in *variable* occurred before any portion of *variable* is defined by the assignment. The evaluation of expressions within *variable* shall neither affect nor be affected by the evaluation of *expr*. No value is assigned to *variable* if *variable* is of type character and zero length, or is an array of size zero.

Interp **F03/0093**, Status: *Corrigendum 3.*

Ref: 7.4.1.3, 3^{rd} paragraph, [139:22,25]

The paragraph begins "If *variable* is an allocated allocatable";
At beginning of paragraph
Insert new sentence "If *variable* is an unallocated allocatable array, *expr* shall have the same rank as *variable*.",
In the last sentence of the paragraph,
After "equal to the corresponding type"
Change "parameters" to "parameter",
Before "with the shape of *expr*"
Change "," to "." and
Insert "If *variable* is an array and *expr* is scalar it is allocated with the same bounds as before, otherwise it is allocated",
Making the whole paragraph read:

> If *variable* is an unallocated allocatable array, *expr* shall be an array of the same rank as *variable*. If *variable* is an allocated allocatable variable, it is deallocated if *expr* is an array of different shape or any of the corresponding length type parameter values of *variable* and *expr* differ. If *variable* is or becomes an unallocated allocatable variable, then it is allocated with each deferred type parameter equal to the corresponding type parameter~~s~~ of *expr*. If *variable* is an array and *expr* is scalar it is allocated with the same bounds as before, otherwise it is allocated with the shape of *expr*, and with each lower bound equal to the corresponding element of LBOUND(*expr*).

Interp **F03/0092**, Status: *Corrigendum 3*.

Ref: 7.4.2.2, penultimate paragraph, [145:5]

After "the same type"
Insert "or both be unlimited polymorphic",
Making the whole paragraph read:

> If *proc-target* and *proc-pointer-object* are functions, they shall have the same type or both be unlimited polymorphic; corresponding type parameters shall either both be deferred or both have the same value.

Interp **F03/0050**, Status: *Corrigendum 3*.

Ref: 9.5.3.4, after the 7^{th} paragraph, [196:29+]

Insert new paragraphs:

> During the execution of an output statement that specifies an internal file, no part of that internal file shall be referenced, defined, or become undefined as the result of evaluating any output list item.
>
> During the execution of an input statement that specifies an internal file, no part of that internal file shall be defined or become undefined as the result of transferring a value to any input list item.

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.8, 9.9.1.9, 9.9.1.12, [212:15,21,36]

NOTE: This interp also has edits on pages 213-216.

Edit subclauses 9.9.1.8, 9.9.1.9, and 9.9.1.12 as shown below.

9.9.1.8 DIRECT= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the DIRECT= specifier is assigned the value YES if DIRECT is included in the set of allowed access methods for the file, NO if DIRECT is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not DIRECT is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.9 ENCODING= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the ENCODING= specifier is assigned the value UTF-8 if the file is connected for formatted input/output with an encoding form of UTF-8, and is assigned the value UNDEFINED if the file is connected for unformatted input/output. If there is no connection, it is assigned the value UTF-8 if the processor is able to determine that the encoding form of the file is UTF-8. If the processor is unable to determine the encoding form of the file or if the unit specified by UNIT= is not connected to a file, the variable is assigned the value UNKNOWN.

NOTE 9.62

The value assigned may be something other than UTF-8, UNDEFINED, or UNKNOWN if the processor supports other specific encoding forms (e.g. UTF-16BE).

9.9.1.12 FORMATTED= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the FORMATTED= specifier is assigned the value YES if FORMATTED is included in the set of allowed forms for the file, NO if FORMATTED is not included in the set of allowed forms for the file, and UNKNOWN if the processor is unable to determine whether or not FORMATTED is included in the set of allowed forms for the file or if the unit specified by UNIT= is not connected to a file.

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.16, 9.9.1.17, [213:15,16,20-21]

Edit subclauses 9.9.1.16 and 9.9.1.17 as shown below.

NOTE: This interp has other edits on pages 212-216.

9.9.1.16 NEXTREC= specifier in the INQUIRE statement

The *scalar-int-variable* in the NEXTREC= specifier is assigned the value $n + 1$, where n is the record number of the last record read from or written to the file connected for direct access. If the file is connected but no records have been read or written since the connection, the *scalar-int-variable* is assigned the value 1. If the file is not connected for direct access, ~~or~~ if the position of the file is indeterminate because of a previous error condition, or if the unit specified by UNIT= is not connected to a file, the *scalar-int-variable* becomes undefined. If there are pending data transfer operations for the specified unit, the value assigned is computed as if all the pending data transfers had already completed.

9.9.1.17 NUMBER= specifier in the INQUIRE statement

~~The *scalar-int-variable* in the NUMBER= specifier is assigned the value of the external unit number of the unit that is connected to the file. If there is no unit connected to the file, the value -1 is assigned.~~Execution of an INQUIRE by file statement causes the *scalar-int-variable* in the NUMBER= specifier to be assigned the value of the external unit number of the unit that is connected to the file. If there is no unit connected to the file, the value -1 is assigned. Execution of an INQUIRE by unit statement causes the *scalar-int-variable* to be assigned the value specified by UNIT=.

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.21, [214:19,20]

Edit subclause 9.9.1.21 as shown below.

NOTE: This interp has other edits on pages 212-216.

9.9.1.21 POS= specifier in the INQUIRE statement

The *scalar-int-variable* in the POS= specifier is assigned the number of the file storage unit immediately following the current position of a file connected for stream access. If the file is positioned at its terminal position, the variable is assigned a value one greater than the number of the highest-numbered file storage unit in the file. If the file is not connected for stream access ~~or~~, if the position of the file is indeterminate because of previous error conditions, or if the unit specified by UNIT= is not connected to a file, the variable becomes undefined.

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.23, 9.9.1.24, 9.9.1.27, 9.9.1.29, [215:2,7,26,34]

Edit subclauses 9.9.1.23, 9.9.1.24, 9.9.1.27 and 9.9.1.29 as shown below:

9.9.1.23 READ= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the READ= specifier is assigned the value YES if READ is included in the set of allowed actions for the file, NO if READ is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READ is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.24 READWRITE= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the READWRITE= specifier is assigned the value YES if READWRITE is included in the set of allowed actions for the file, NO if READWRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not READWRITE is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.27 SEQUENTIAL= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the SEQUENTIAL= specifier is assigned the value YES if SEQUENTIAL is included in the set of allowed access methods for the file, NO if SEQUENTIAL is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not SEQUENTIAL is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.29 SIZE= specifier in the INQUIRE statement

The *scalar-int-variable* in the SIZE= specifier is assigned the size of the file in file storage units. If the file size cannot be determined or if the unit specified by UNIT= is not connected to a file, the variable is assigned the value -1.

For a file that may be connected for stream access, the file size is the number of the highest-numbered file storage unit in the file.

For a file that may be connected for sequential or direct access, the file size may be different from the number of storage units implied by the data in the records; the exact relationship is processor-dependent.

NOTE: This interp has other edits on pages 212-216.

Interp **F03/0106**, Status: *Corrigendum 3*.

Ref: 9.9.1.30, 9.9.1.31, 9.9.1.32, [216:5,10,15]

Edit subclauses 9.9.1.30, 9.9.1.31 and 9.9.1.32 as shown below:

9.9.1.30 STREAM= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the STREAM= specifier is assigned the value YES if STREAM is included in the set of allowed access methods for the file, NO if STREAM is not included in the set of allowed access methods for the file, and UNKNOWN if the processor is unable to determine whether or not STREAM is included in the set of allowed access methods for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.31 UNFORMATTED= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the UNFORMATTED= specifier is assigned the value YES if UNFORMATTED is included in the set of allowed forms for the file, NO if UNFORMATTED is not included in the set of allowed forms for the file, and UNKNOWN if the processor is unable to determine whether or not UNFORMATTED is included in the set of allowed forms for the file or if the unit specified by UNIT= is not connected to a file.

9.9.1.32 WRITE= specifier in the INQUIRE statement

The *scalar-default-char-variable* in the WRITE= specifier is assigned the value YES if WRITE is included in the set of allowed actions for the file, NO if WRITE is not included in the set of allowed actions for the file, and UNKNOWN if the processor is unable to determine whether or not WRITE is included in the set of allowed actions for the file or if the unit specified by UNIT= is not connected to a file.

NOTE: This interp also has other edits on pages 212-215.

Interp **F04/0079**, Status: *Corrigendum 3*.

Ref: 10.6.1, numbered list, [227:15+]

Add new list item:

(7) On output of a real zero value, the digits in the exponent field shall all be zero.

Interp **F03/0101**, Status: *Corrigendum 3*.

Ref: 10.9.2, 1^{st} paragraph, [241:5]

Append new sentences to paragraph:

> The form of the values produced by a user-defined derived type output routine invoked during list-directed output is specified by the invoked routine. This form need not be compatible with list-directed input.

NOTE: This interp also has edits on page 246.

Interp **F03/0097**, Status: *Corrigendum 3*.

Ref: 10.10, paragraph before 10.10.1, [243:5]

Replace paragraph in its entirety,
Making it read:

> A value separator for namelist formatting is ~~the same as~~ a value separator for list-directed formatting (10.9), or one or more contiguous blanks between a nonblank value and the following object designator or "!" comment initiator.

Interp **F03/0101**, Status: *Corrigendum 3*.

Ref: 10.10.2, 1^{st} paragraph, [246:4,7]

After "and logical values"
Insert ", and output produced by user-defined derived-type output",
Add new sentences to end of paragraph,
Making the whole paragraph read:

> The form of the output produced is the same as that required for input, except for the forms of real, character, and logical values, and output produced by user-defined derived-type output. The name in the output is in upper case. With the exception of adjacent undelimited character values, the values are separated by one or more blanks or by a comma, or a semicolon if the decimal edit mode is COMMA, optionally preceded by one or more blanks and optionally followed by one or more blanks. The form of the output produced by a user-defined derived-type output routine invoked during namelist output is specified by the invoked routine. This form need not be compatible with namelist input.

NOTE: This interp also has an edit on page 241.

Interp **F03/0088**, Status: *Corrigendum 3*.

Ref: 12.3.2.1.1, 1^{st} paragraph, [262:12]

After "the second argument."
Append new sentence to paragraph:

> All restrictions and constraints that apply to actual arguments in a reference to the function also apply to the corresponding operands in the expression as if they were used as actual arguments.

NOTE: This interp also has an edit on p263.

Interp **F03/0069**, Status: *Corrigendum 2*.

Ref: 12.3.2.1.2, 2^{nd} paragraph, 2^{nd} sentence, [263:6]

Replace entire sentence "Each argument shall be nonoptional."
By "The dummy arguments shall be nonoptional dummy data objects.",
(See below for resulting paragraph.)

Interp **F03/0088**, Status: *Corrigendum 3*.

Ref: 12.3.2.1.2, 2^{nd} paragraph, [263:12]

After "the second argument."
Insert the following new sentence:

> All restrictions and constraints that apply to actual arguments in a reference to the subroutine also apply to the left-hand-side and to the right-hand-side enclosed in parentheses as if they were used as actual arguments.

Together with the previous interp on this page, making the whole paragraph read:

> Each of these subroutines shall have exactly two dummy arguments. ~~Each argument shall be nonoptional.~~ The dummy arguments shall be nonoptional dummy data objects. The first argument shall have INTENT (OUT) or INTENT (INOUT) and the second argument shall have INTENT (IN). Either the second argument shall be an array whose rank differs from that of the first argument, the declared types and kind type parameters of the arguments shall not conform as specified in Table 7.8, or the first argument shall be of derived type. A defined assignment is treated as a reference to the subroutine, with the left-hand side as the first argument and the right-hand side enclosed in parentheses as the second argument. All restrictions and constraints that apply to actual arguments in a reference to the subroutine also apply to the left-hand-side and to the right-hand-side enclosed in parentheses as if they were used as actual arguments. The ASSIGNMENT generic specification specifies that assignment is extended or redefined.

Interp **F03/0086**, Status: *Corrigendum 3*.

Ref: 12.5.2.1, constraint C1242, [280:6-7]

Replace constraint with:

C1242 An elemental procedure shall not have the BIND attribute.

Interp **F03/0107**, Status: *Corrigendum 3*.

Ref: 14.9.2, 1^{st} paragraph, [370:8-9]

Edit paragraph as shown below:

> The module IEEE_ARITHMETIC contains the following elemental functions for all reals X and Y ~~for which IEEE_SUPPORT_DATATYPE(X) and IEEE_SUPPORT_DATATYPE(Y) are true~~:

Interp **F03/0089**, Status: *Corrigendum 3.*

Ref: 15.2.3, 2^{nd} paragraph, [398:9]

After "A Fortran derived type is interoperable with a C struct type if"
Insert "and only if",
Making the whole paragraph:

> A Fortran derived type is interoperable with a C struct type if and only if the derived-type definition of the Fortran type specifies BIND(C) (4.5.1), the Fortran derived type and the C struct type have the same number of components, and the components of the Fortran derived type have types and type parameters that are interoperable with the types of the corresponding components of the struct type. A component of a Fortran derived type and a component of a C struct type correspond if they are declared in the same relative position in their respective type definitions.