

**COMBINED TECHNICAL CORRIGENDA 1 TO 4 FOR FORTRAN 2008**

This document combines the edits from the WG5 versions of Technical Corrigenda 1, 2, 3 and 4 for Fortran 2008, that is N1902, N1957, N2002 and N2103. It is intended for use in WG5 only.

[xv] f08/0046 TC1

**Introduction**

At the end of the fourth item in the bulleted list (Data declaration), append the sentence: “An array or an object with a nonconstant length type parameter can have the VALUE attribute.”.

[xv] f08/0121 (N2062) TC4

In the second paragraph, append to the ‘Data declaration’ bullet point: “A *defined-operator* can be used in a specification expression.”.

[xv] f08/0042 TC2

Following the second sentence in the fifth item in the bulleted list (Data usage and computation), insert: “Multiple allocations are permitted in a single ALLOCATE statement with SOURCE=.”.

[xv] (new in N2102) TC4

In the second paragraph, append two new sentences to the ‘Data usage and computation’ bullet point: “All transformational functions from the intrinsic modules IEEE\_ARITHMETIC and IEEE\_EXCEPTIONS can be used in constant expressions. All transformational functions from the intrinsic modules IEEE\_ARITHMETIC, IEEE\_EXCEPTIONS, and ISO\_C\_BINDING can be used in specification expressions.”

[xvi] f08/0131 (N2062&N2102) TC4

In the second paragraph, in the ‘Intrinsic modules’ bullet point, before “The function C\_SIZEOF” insert the new sentence:

A contiguous array variable that is not interoperable but which has interoperable type and kind type parameter (if any), and a scalar character variable with length>1 and kind C\_CHAR, can be used as the argument of the function C\_LOC in the intrinsic module ISO\_C\_BINDING, provided the variable has the POINTER or TARGET attribute.

[xvi] f08/0094 TC3

In the last item in the main bulleted list (Programs and procedures), after “An empty CONTAINS section is allowed.”, insert new sentence: “A PROCEDURE statement can have a double colon before the first procedure name.”.

[xvi] f08/0051 TC1

In the last item in the main bulleted list (Programs and procedures), before “An impure” insert the new sentence: “An argument to a pure procedure can have default INTENT if it has the VALUE attribute.”.

[xvi] inadvertently omitted from f08/0037 TC1

In the same paragraph, before “The FUNCTION and SUBROUTINE” insert the new sentence: “The PROTECTED attribute can be specified by the procedure declaration statement.”

[xvi] f08/0127 (N2062&N2080&N2102) and f08/0139 (N2092) TC4

In the second paragraph, in the ‘Programs and procedures’ bullet point, replace the final sentence, “A line in the program is permitted to begin with a semicolon.” with “A free form continuation line can begin with zero or more blanks followed by a semicolon.”. Also, append the sentence, “The name of an external procedure that has a binding label is a local identifier and not a global identifier.”.

[6:7+] f08/0124 (N2062&N2102) TC4

**Subclause 1.3**

After the definition of **parent component** (1.3.33.2) insert a new term:

**1.3.33.2a****potential subobject component**

nonpointer component, or potential subobject component of a nonpointer component

**Subclause 1.3.77**

Following subclause 1.3.77 add new item:

**1.3.77a**

**function result**

entity that returns the value of a function

[15:31-33] *f03/0139 TC3*

**Subclause 1.3.121**

Delete term **1.3.121 result variable**.

[19:15-16] *f03/0053 TC3*

**Subclause 1.3.147.6**

Replace the definition of **extensible type** with:

type that may be extended using the EXTENDS clause (4.5.7.1)

[24:9] *f08/0011 TC1*

**Subclause 1.6.2**

In the first sentence of the first paragraph of the subclause change “This” to “Except as identified in this subclause, this”.

[24:10] *f08/0033 TC1*

In the second sentence of the first paragraph change “Any” to “Except as identified in this subclause, any”.

[24:11+] *f08/0011 and f08/0033 TC1*

Following the first paragraph of the subclause, add new paragraphs:

Fortran 2003 specified that array constructors and structure constructors of finalizable type are finalized. This part of ISO/IEC 1539 specifies that these constructors are not finalized.

Fortran 2003 permitted an INTENT(OUT) argument of a pure subroutine to be polymorphic; that is not permitted by this part of ISO/IEC 1539.

[24:11+] *f03/0120, f08/0032, f08/0054 and f08/0055 TC2*

Replace the two paragraphs added to the subclause in Technical Corrigendum 1 by the following six paragraphs:

Fortran 2003 permitted a sequence type to have type parameters; that is not permitted by this part of ISO/IEC 1539.

Fortran 2003 specified that array constructors and structure constructors of finalizable type are finalized. This part of ISO/IEC 1539 specifies that these constructors are not finalized.

The form produced by the G edit descriptor for some values and some I/O rounding modes differs from that specified by Fortran 2003.

Fortran 2003 required an explicit interface only for a procedure that was actually referenced in the scope, not merely passed as an actual argument. This part of ISO/IEC 1539 requires an explicit interface for a procedure under the conditions listed in 12.4.2.2, regardless of whether the procedure is referenced in the scope.

Fortran 2003 permitted the result variable of a pure function to be a polymorphic allocatable variable, or to be finalizable by an impure final subroutine. These are not permitted by this part of ISO/IEC 1539.

Fortran 2003 permitted an INTENT(OUT) argument of a pure subroutine to be polymorphic; that is not permitted by this part of ISO/IEC 1539.

After the six paragraphs added to the subclause in Technical Corrigendum 2, add the following new paragraphs:

Fortran 2003 interpreted assignment to an allocatable variable from a nonconformable array as intrinsic assignment, even when an elemental defined assignment was in scope; this part of ISO/IEC 1539 does not permit assignment from a nonconformable array in this context.

Fortran 2003 permitted a statement function to be of parameterized derived type; this part of ISO/IEC 1539 does not permit that.

[24:14] *f08/0087 TC3*

### **Subclause 1.6.3**

In the first paragraph of the subclause, replace “Any” by “Except as identified in this subclause, any”.

[24:15-16] *f08/0087 TC3*

Delete the final sentence of the first paragraph, “The following ...1539.” and insert two new paragraphs:

Fortran 95 permitted defined assignment between character strings of the same rank and different kinds. This part of ISO/IEC 1539 does not permit that if both of the different kinds are ASCII, ISO 10646, or default kind.

The following Fortran 95 features might have different interpretations in this part of ISO/IEC 1539.

[24:27+] *f08/0055 TC2*

Add the following item at the end of the bulleted list:

- The form produced by the G edit descriptor with  $d$  equal to zero differs from that specified by Fortran 95 for some values.

[25:2+] *f08/0087 TC3*

### **Subclause 1.6.4**

Following the third paragraph of the subclause, insert a new paragraph:

Fortran 90 permitted defined assignment between character strings of the same rank and different kinds. This part of ISO/IEC 1539 does not permit that if both of the different kinds are ASCII, ISO 10646, or default kind.

[25:6] *f08/0055 TC2*

In the fourth paragraph of the subclause, replace the full stop at the end of the third bulleted item by a semicolon and add a fourth item:

- the G edit descriptor with  $d$  equal to zero for some values.

[30:28] *f03/0139 TC3*

### **Subclause 2.2.3**

In the second paragraph of the subclause, after “data objects” insert “or procedure pointers”.

[33:36+] *f08/0093 TC3*

### **Subclause 2.3.5**

In the fifth paragraph of the subclause, before Note 2.7, insert new note:

#### **NOTE 2.6a**

If the processor supports the concept of a process exit status, it is recommended that error termination initiated other than by an ERROR STOP statement supplies a processor-dependent nonzero value as the process exit status.

[51:26+] f08/0097 TC3

**Subclause 4.3.1.1**

Following constraint C406, insert new constraint:

C406a (R403) In TYPE(*intrinsic-type-spec*) the *intrinsic-type-spec* shall not end with a comma.

[52:2] f03/0139 TC3

**Subclause 4.3.1.2**

In the second paragraph of the subclause, in the final sentence, change “function result variable” to “function result”.

[52:6+] f08/0129 (N2062&N2080&N2102) TC4

**Subclause 4.3.1.3**

After the first paragraph, insert the following new paragraph:

Where a data entity other than a component is declared explicitly using the CLASS specifier to be of derived type, the specified derived type shall have been defined previously. If the data entity is a function result, the derived type may be specified in the FUNCTION statement provided the derived type is defined within the body of the function or is accessible there by use or host association. If the derived type is specified in the FUNCTION statement and is defined within the body of the function, it is as if the function result variable were declared with that derived type immediately following its *derived-type-def*.

[54:18+] f08/0078 TC2

**Subclause 4.4.2.3**

In the third paragraph of the subclause, in Note 4.8 change “can distinguish” to “distinguishes”.

[58:23] f03/0139 TC3

**Subclause 4.4.3.2**

In the fifth paragraph of the subclause, in the fifth bulleted item in the list change “result variable in the function” to “function result”.

[61:19+] f08/0092 TC3

**Subclause 4.5.2.1**

After constraint C427 insert new constraint:

C427a (R426) The same *type-param-name* shall not appear more than once in a *derived-type-stmt*.

[62:19] f03/0120 TC2

**Subclause 4.5.2.3**

Replace constraint C436 by:

C436 (R425) If SEQUENCE appears, each data component shall be declared to be of an intrinsic type or of a sequence type, the derived type shall not have type parameters, and a *type-bound-procedure-part* shall not appear.

[62:19] f08/0091 TC3

In constraint C436, after “appears,” insert “the type shall have at least one component,”.

[63:9] f03/0120 TC2

**Subclause 4.5.2.4**

In the second sentence of the second paragraph of the subclause, delete “type parameters and”.

[64:8] f08/0092 TC3

**Subclause 4.5.3.1**

In constraint C438, after “shall appear” insert “exactly once”.

**Subclause 4.5.4.6**

In the first sentence of constraint C461, insert “, noncoindexed” after “nonallocatable” so that the sentence reads:

C461 (R443) The designator shall designate a nonallocatable, noncoindexed variable that has the TARGET and SAVE attributes and does not have a vector subscript.

[75:10] f08/0072 TC2

**Subclause 4.5.6.1**

In the second sentence of constraint C480, insert “noncoarray,” before “nonpointer”.

[76:10-, 25-26] f08/0013 TC1

**Subclause 4.5.6.3**

Move paragraph 9 of the subclause and Note 4.49 to precede paragraph 1. In addition, edit the paragraph by changing “the variable is” to “if the variable is not an unallocated allocatable variable, it is” and by appending a new sentence at the end of the paragraph: “If the variable is an allocated allocatable that would be deallocated by intrinsic assignment, the finalization occurs before the deallocation.”.

[76:10] f08/0013 TC1

In paragraph 1 of the subclause, after “it is finalized” insert “unless it is the variable in an intrinsic assignment (7.2.1.3) or a component thereof”

[76:10] f08/0081 TC2

To the second paragraph of the subclause (which was paragraph 1 prior to the edits of Technical Corrigendum 1), append the new sentence:

If an error condition occurs during deallocation, it is processor dependent whether finalization occurs.

[76:17-18,21-22] f08/0011 TC1

Delete paragraphs 5 and 7 of the subclause.

[76:23-24] f03/0085 and f08/0034 TC1

Replace paragraph 8 of the subclause with:

When a procedure is invoked, an object that becomes argument associated with a nonpointer, nonallocatable INTENT(OUT) dummy argument of that procedure is finalized. The finalization caused by INTENT(OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT(OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.

[76:23-24] f08/0070 TC2

Replace the seventh paragraph of the subclause (which was paragraph 8 prior to the edits of Technical Corrigendum 1) by the following new paragraph. This supersedes the substitute paragraph given in Technical Corrigendum 1.

When a procedure is invoked, a nonpointer, nonallocatable INTENT(OUT) dummy argument of that procedure is finalized before it becomes undefined. The finalization caused by INTENT(OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT(OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.

---

*The combined edits to subclause 4.5.6.3 result in the following complete replacement:*

- 1 When an intrinsic assignment statement is executed, if the variable is not an unallocated allocatable variable, it is finalized after evaluation of *expr* and before the definition of the variable. If the variable is an allocated allocatable that would be deallocated by intrinsic assignment, the finalization occurs before the deallocation.

NOTE 4.49

If finalization is used for storage management, it often needs to be combined with defined assignment.

- 2 When a pointer is deallocated its target is finalized. When an allocatable entity is deallocated, it is finalized unless it is the variable in an intrinsic assignment (7.2.1.3) or a component thereof. If an error condition occurs during deallocation, it is processor dependent whether finalization occurs.
- 3 A nonpointer, nonallocatable object that is not a dummy argument or function result is finalized immediately before it would become undefined due to execution of a RETURN or END statement (16.6.6, item (3)).
- 4 A nonpointer nonallocatable local variable of a BLOCK construct is finalized immediately before it would become undefined due to termination of the BLOCK construct (16.6.6, item (22)).
- 5 If an executable construct references a function, the result is finalized after execution of the innermost executable construct containing the reference.
- 6 If a specification expression in a scoping unit references a function, the result is finalized before execution of the executable constructs in the scoping unit.
- 7 When a procedure is invoked, a nonpointer, nonallocatable INTENT(OUT) dummy argument of that procedure is finalized before it becomes undefined. The finalization caused by INTENT(OUT) is considered to occur within the invoked procedure; so for elemental procedures, an INTENT(OUT) argument will be finalized only if a scalar or elemental final subroutine is available, regardless of the rank of the actual argument.
- 8 If an object is allocated via pointer allocation and later becomes unreachable due to all pointers associated with that object having their pointer association status changed, it is processor dependent whether it is finalized. If it is finalized, it is processor dependent as to when the final subroutines are called.

---

---

[77:3]f03/0053 TC3

**Subclause 4.5.7.1**

In the first paragraph of the subclause, after “A derived type” insert “, other than the type C\_PTR or C\_FUNPTR from the intrinsic module ISO\_C\_BINDING”.

[78:4]f08/0052 TC1

**Subclause 4.5.7.3**

In the first paragraph of the subclause, change “as a type-bound” to “as an accessible type-bound”.

[85:8-9]f08/0080 TC2

**Subclause 4.8**

In constraint C4105 in the first paragraph of the subclause, change “all *ac-value* expressions in the *array-constructor* shall be of that derived type and” to “the declared type of each *ac-value* expression in the *array-constructor* shall be that derived type and”.

[85:10+]f08/0080 TC2

After constraint C4106, insert the following new constraint:

C4106a (R472) The declared type of an *ac-value* shall not be abstract.

[85:13-14]f08/0080 TC2

In the second paragraph of the subclause, change “each *ac-value* expression in the array constructor shall have the same length type parameters;” to “corresponding length type parameters of the declared type of each *ac-value* expression shall have the same value;”.

[85:18] f08/0080 TC2

In the third paragraph of the subclause, after “Each value is converted to the” insert “type and”.

[87:9] f03/0139 TC3

#### Subclause 5.1

In the second paragraph of the subclause, change “its result variable” to “the function result”.

[88:14] f08/0085 TC3

#### Subclause 5.2.1

In the second paragraph of the subclause, replace constraint C507 by:

C507 (R501) If the PARAMETER keyword appears, *initialization* shall appear in each *entity-decl*.

Add new constraint:

C507a An expression that specifies a length type parameter or array bound of a named constant shall be a constant expression.

[88:30+] f08/0090 TC3

Following the final paragraph of the subclause, insert a new paragraph:

- If *initialization* appears for a nonpointer entity,
- its type and type parameters shall conform as specified for intrinsic assignment (7.2.1.2);
  - if the entity has implied shape, the rank of *initialization* shall be the same as the rank of the entity;
  - if the entity does not have implied shape, *initialization* shall either be scalar or have the same shape as the entity.

[90:15] f08/0115 (N2062&N2102) TC4

#### Subclause 5.3.4

In the first bullet point of the second paragraph, after “the variable” insert “is a dummy argument or” and insert a comma after “scoping unit”.

[93:7-8] f08/0061 TC2

#### Subclause 5.3.7

In the first paragraph of the subclause, change “can only be argument associated with a contiguous effective argument” to “is contiguous”.

[94:10] f08/0086 TC3

#### Subclause 5.3.8.1

In syntax rule R515, change “*implied-shape-spec-list*” to “*implied-shape-spec*” and insert new production:

or *implied-shape-or-assumed-size-spec*

[95:32] f08/0086 TC3

#### Subclause 5.3.8.5

In the first paragraph of the subclause, replace the final sentence “An assumed-size array is declared with an *assumed-size-spec*.” with “A dummy argument is declared to be an assumed-size array by an *assumed-size-spec* or an *implied-shape-or-assumed-size-spec*.”.

[95:33-] f08/0086 TC3

Before syntax rule R521 insert new BNF term:

R520a *assumed-implied-spec* is [ *lower-bound* : ] \*

[95:33] f08/0086 TC3

Replace syntax rule R521 with:

R521 *assumed-size-spec* is *explicit-shape-spec-list*, *assumed-implied-spec*

[95:37+] f08/0086 TC3

Following constraint C534 insert new syntax rule and constraint:

R521a *implied-shape-or-assumed-size-spec* **is** *assumed-implied-spec*

C534a An object whose array bounds are specified by an *implied-shape-or-assumed-size-spec* shall be a dummy data object or a named constant.

[96:24-25] f08/0086 TC3

#### **Subclause 5.3.8.6**

In the first paragraph of the subclause, replace the sentence “An implied-shape array is declared ... *assumed-implied-spec-list*.” with “A named constant is declared to be an implied-shape array with an *array-spec* that is an *implied-shape-or-assumed-size-spec* or an *implied-shape-spec*.”.

[96:26] f08/0086 TC3

Replace syntax rule R522 by:

R522 *implied-shape-spec* **is** *assumed-implied-spec, assumed-implied-spec-list*

[96:28] f08/0086 TC3

Replace the second paragraph of the subclause, “The rank ... *implied-shape-spec-list*”, by:

The rank of an implied-shape array is the number of *assumed-implied-specs* in its *array-spec*.

[97:13] f08/0040 TC2

#### **Subclause 5.3.10**

In constraint C541 change “An entity” to “A dummy argument of a nonintrinsic procedure”.

[102:9] f08/0122 (N2062) TC4

#### **Subclause 5.3.19**

In constraint C560, after “for a coarray” insert “, or a variable with a coarray ultimate component,”.

[102:11] f08/0122 (N2062) TC4

In constraint C561, after “for a coarray” insert “, or a variable with a coarray ultimate component,”.

[104:26-27] f08/0077 TC2

#### **Subclause 5.4.7**

In the fourth paragraph of the subclause, replace constraint C566 by:

C566 (R536) A *data-stmt-object* that is a *variable* shall be a *designator*. Each subscript, section subscript, substring starting point, and substring ending point in the *variable* shall be a constant expression.

[107:11] f08/0086 TC3

#### **Subclause 5.4.11**

In the second paragraph of the subclause, in the final sentence change “shape” to “rank”.

[107:12+] f08/0090 TC3

Following that paragraph, insert a new paragraph:

The constant expression that corresponds to a named constant shall have type and type parameters that conform with the named constant as specified for intrinsic assignment (7.2.1.2). If the named constant has implied shape, the expression shall have the same rank as the named constant; otherwise, the expression shall either be scalar or have the same rank as the named constant.

[109:16] f08/0074 TC2

#### **Subclause 5.5**

In the final sentence of the third paragraph of the subclause, change “an internal or module procedure” to “a BLOCK construct, internal subprogram, or module subprogram”.

[109:21-23] f03/0123 and f08/0015 TC1

In the fourth paragraph of the subclause, delete the sentence “The mapping may ... scoping unit.” and replace “in the outermost inclusive scope in which it appears” by “; if the outermost inclusive scope in which it appears is not a type definition, it is declared in that scope, otherwise it is declared in the host of that scope”.

[109:24] f03/0139 TC3

In the fourth paragraph of the subclause, in the final sentence change “name of the result variable of that function subprogram” to “result of that function”.

[111:13-14] f08/0101 (N2042) TC4

#### **Subclause 5.6**

Replace the second paragraph, by:

The order in which the values appear on output is the same as the order of the *namelist-group-objects* in the namelist group object list; if a variable appears more than once as a *namelist-group-object* for the same namelist group, its value appears once for each occurrence.

[111:19-20] f08/0002 TC1

In the first sentence of the fifth paragraph, replace “type parameters, and shape” by “kind type parameters, and rank”.

[111:19-20] f08/0079 TC2

In the fifth paragraph of the subclause, change what was originally “type, type parameters, and shape” but which was changed by Technical Corrigendum 1 to “type, kind type parameters, and rank” to “declared type, kind type parameters of the declared type, and rank”.

[112:15] f03/0139 TC3

#### **Subclause 5.7.1.1**

In the second paragraph of the subclause, in constraint C587 change “result variable” to “function result”.

[114:22] f03/0139 TC3

#### **Subclause 5.7.2.1**

In the second paragraph of the subclause, in constraint C5100 change “result variable” to “function result”.

[117:13, 15] f08/0075 TC3

#### **Subclause 6.2**

In syntax rule R602, change “*expr*” to “*function-reference*” and replace constraint C602 by:

C602 (R602) *function-reference* shall have a data pointer result.

[119:13] f08/0124 (N2062) TC4

#### **Subclause 6.4.2**

In constraint C617, replace “subcomponent” with “potential subobject component”.

[124:4-7] f08/0014 and f08/0016 TC1

#### **Subclause 6.5.3.3.2**

Replace the second paragraph of the subclause by:

A vector-subscripted array section shall not be finalized by a nonelemental final subroutine.

[124:9] f08/0039 TC1

In the third paragraph of the subclause, replace “shall ... (16.6.7)” with “is not definable and shall not be defined or become undefined”.

[126:31-33] f08/0042 TC2

#### **Subclause 6.7.1.1**

Replace constraint C633 by:

C633 (R626) If an *allocate-object* is an array, either *allocate-shape-spec-list* shall appear in its *allocation*, or *source-expr* shall appear in the ALLOCATE statement and have the same rank as the *allocate-object*.

C633a (R631) If *allocate-object* is scalar, *allocate-shape-spec-list* shall not appear.

[127:5]f08/0042 TC2

Replace constraint C639 by:

C639 (R626) If *source-expr* appears, the kind type parameters of each *allocate-object* shall have the same values as the corresponding type parameters of *source-expr*.

[127:8-9]f08/0109 (N2086) TC4

In constraint C642, change “C\_PTR,” to “C\_PTR or” and delete “, LOCK\_TYPE ... LOCK\_TYPE”.

[127:9+]f08/0109 (N2086&N2087) TC4

Following constraint C642, add a new constraint:

C642a (R627) If SOURCE= appears, the declared type of *source-expr* shall not be LOCK\_TYPE or have a potential subobject component of type LOCK\_TYPE.

[127:18-19]f08/0042 TC2

Replace the fourth paragraph of the subclause by:

If an *allocate-object* is a coarray, the ALLOCATE statement shall not have a *source-expr* with a dynamic type of C\_PTR, C\_FUNPTR, or LOCK\_TYPE, or which has a subcomponent whose dynamic type is LOCK\_TYPE.

[127:18-19]f08/0109 (N2086) TC4

Instead of the edit in Technical Corrigendum 2, which replaced the entire fourth paragraph of the subclause, make the following change to this paragraph: replace “If *allocate-object* is” by “If an ALLOCATE statement has a SOURCE= specifier and an *allocate-object* that is”. The edited paragraph thus reads:

If an ALLOCATE statement has a SOURCE= specifier and an *allocate-object* that is a coarray, *source-expr* shall not have a dynamic type of C\_PTR, C\_FUNPTR, or LOCK\_TYPE, or have a subcomponent whose dynamic type is LOCK\_TYPE.

[128:15-17]f08/0130 (N2062&N2094&N2102) TC4

#### **Subclause 6.7.1.2**

In the second sentence of the fourth paragraph, change “On each image” to “If no error condition other than STAT\_STOPPED\_IMAGE occurs” and change “all other images” to “all non-stopped images”. Append a new sentence so that the entire paragraph reads:

When an ALLOCATE statement is executed for which an *allocate-object* is a coarray, there is an implicit synchronization of all images. If no error condition other than STAT\_STOPPED\_IMAGE occurs, execution of the segment (8.5.2) following the statement is delayed until all non-stopped images have executed the same statement the same number of times. The coarray shall not become allocated on an image unless it is successfully allocated on all non-stopped images.

[128:24-26]f08/0056 TC2

In the seventh paragraph of the subclause, before “On successful”, insert the new sentence:

If an *allocate-object* is not polymorphic and the *source-expr* is polymorphic with a dynamic type that differs from its declared type, the value provided for that *allocate-object* is the ancestor component of the *source-expr* that has the type of the *allocate-object*; otherwise, the value provided is the value of the *source-expr*.

In the sentence beginning “On successful”, replace “that of *source-expr*” with “the value provided”, twice.

[128:26]f08/0042 TC2

At the end of the seventh paragraph append the new sentence:

The *source-expr* is evaluated exactly once for each execution of an ALLOCATE statement.

**Subclause 6.7.1.3**

In the second bulleted item of the first paragraph, fourth sentence, insert “nonoptional” before “nonallocatable dummy argument”.

[130:23]f08/0010 TC1

**Subclause 6.7.3.2**

Add the following sentence to the end of the first paragraph: “An allocatable variable shall not be deallocated if it or any subobject of it is argument associated with a dummy argument or construct associated with an associate name.”.

[130:26]f03/0139 TC3

In the second paragraph of the subclause, after “function result” delete “variable”.

[131:12]f08/0081 TC2

Append the following new sentence to the eighth paragraph of the subclause:

If an error condition occurs during deallocation, it is processor dependent whether an allocated allocatable subobject is deallocated.

[131:17-19]f08/0130 (N2062&N2094&N2102) TC4

In the second sentence of the eleventh paragraph, change “On each image” to “If no error condition other than STAT\_STOPPED\_IMAGE occurs” and change “all other images” to “all non-stopped images”. Append a new sentence so that the entire paragraph reads:

When a DEALLOCATE statement is executed for which an *allocate-object* is a coarray, there is an implicit synchronization of all images. If no error condition other than STAT\_STOPPED\_IMAGE occurs, execution of the segment (8.5.2) following the statement is delayed until all non-stopped images have executed the same statement the same number of times. If the coarray is a dummy argument, its ultimate argument (12.5.2.3) shall be the same coarray on every image. The coarray shall not become deallocated on an image unless it is successfully deallocated on all non-stopped images.

[131:27]f08/0010 TC1

**Subclause 6.7.3.3**

Add the following sentence to the end of the first paragraph: “A pointer shall not be deallocated if its target or any subobject thereof is argument associated with a dummy argument or construct associated with an associate name.”.

[132:4]f08/0112 (N2042) TC4

**Subclause 6.7.4**

Append to the first paragraph the sentence: “The *stat-variable* shall not depend on the value of the *errmsg-variable*.”.

[132:22]f08/0112 (N2042) TC4

**Subclause 6.7.5**

Append to the first paragraph the sentence: “The *errmsg-variable* shall not depend on the value of the *stat-variable*.”.

[133:26+]f03/0139 TC3

**Subclause 7.1.2.2**

Following constraint C702, add new constraint:

C702a (R701) The *expr* shall not be a function reference that returns a procedure pointer.

[150:24]f08/0095 TC3

**Subclause 7.1.11**

In the second paragraph of the subclause, in list item (9)(b), after “variable” insert “, that is not an optional dummy argument,”.

Before item (10) insert two new list items:

- (9a) a specification inquiry that is a constant expression,
- (9b) a reference to the intrinsic function PRESENT,

[150:28+] f08/0104 (N2042&N2047&N2102) TC4

In the second paragraph after item (10), insert a new item:

- (10a) a reference to a transformational function from the intrinsic module IEEE\_ARITHMETIC (14), IEEE\_EXCEPTIONS (14), or ISO\_C\_BINDING (15.2), where each argument is a restricted expression,

[150:37] f08/0095 TC3

In the fourth paragraph of the subclause, in list item (1), after “intrinsic inquiry function” insert “other than PRESENT”.

[151:7-8] f08/0126 (N2062&N2102) TC4

In the eighth paragraph, replace “a type parameter or an array bound” with “a type parameter, array bound, or cobound” and replace “the type parameter, or array bound” with “the type parameter, array bound, or cobound”.

[151:13-15] f08/0050 TC1

Replace the ninth paragraph of the subclause by:

A generic entity referenced in a specification expression in the *specification-part* of a scoping unit shall have no specific procedures defined in that scoping unit, or its host scoping unit, subsequent to the specification expression.

[152:4] f08/0066 TC2

#### **Subclause 7.1.12**

In the first paragraph of the subclause, in item (6) of the numbered list, after “THIS\_IMAGE” insert “, or TRANSFER”.

[152:6+] f08/0066 TC2

After item (7) of the numbered list, insert new item:

- (7a) A reference to the intrinsic function TRANSFER where each argument is a constant expression and each ultimate pointer component of the SOURCE argument is disassociated.

[152:7-8] f08/0104 (N2042&N2102) TC4

In the first paragraph, replace item (8) by:

- (8) a reference to a transformational function from the intrinsic module IEEE\_ARITHMETIC or IEEE\_EXCEPTIONS (14), where each argument is a constant expression,

[152:9] f08/0083 TC3

In the first paragraph of the subclause, replace item (9) in the list by:

- (9) a previously declared kind type parameter of the type being defined,

[152:26-28] f08/0050 TC1

Replace the third paragraph of the subclause by:

A generic entity referenced in a constant expression in the *specification-part* of a scoping unit shall have no specific procedures defined in that scoping unit, or its host scoping unit, subsequent to the constant expression.

**Subclause 7.2.1.2**

In the second paragraph, after “coindexed object,” delete “the variable”. Before each “shall not” insert “the variable” and before “shall have” insert “of the variable” so that the second paragraph reads:

If *variable* is a coindexed object,

- the variable shall not be polymorphic,
- the variable shall not have an allocatable ultimate component, and
- each deferred length type parameter of the variable shall have the same value as the corresponding type parameter of *expr*.

[157:14]f08/0147 (N2086) TC4

**Subclause 7.2.1.4**

In item (5) (b) of the second paragraph, change “ $x_1$  and  $x_2$  are conformable” to “ $x_2$  is scalar or has the same rank as  $x_1$ ”.

[157:16]f08/0147 (N2086) TC4

In the third paragraph of the subclause, append a new sentence:

If the subroutine is elemental,  $x_2$  shall have the same shape as  $x_1$ .

[158:18+]f08/0075 TC3

**Subclause 7.2.2.2**

In syntax rule R737, add new production:

**or** *expr*

[158:19]f08/0071 TC3

In constraint C724, replace “(R737) A *variable*” by “A variable that is a pointer target”.

[158:20+]f08/0075 TC3

Following constraint C724, add new constraint:

C724a (R737) An *expr* shall be a reference to a function that has a data pointer result.

[158:33-159:2]f08/0060 TC2

In constraint C729 replace “an external ... bullet (•)” with “a specific intrinsic function listed in 13.6 and not marked with a bullet (•), or an external procedure that is accessed by use or host association, referenced in the scoping unit as a procedure, or that has the EXTERNAL attribute”.

[170:19]f08/0118 (N2062&N2080) TC4

**Subclause 8.1.3.1**

In constraint C801, change “*associate-name* shall not appear” to “neither the *associate-name* nor any subobject thereof shall appear”.

[170:23+]f03/0139 TC3

Following constraint C804, add new constraint:

C804a (R805) The *expr* shall not be a function reference that returns a procedure pointer.

[171:12]f08/0118 (N2062&N2080) TC4

**Subclause 8.1.3.3**

In the second paragraph, change “the *associate name* shall not appear” to “neither the *associate name* nor any subobject thereof shall appear”.

[172:13+]f08/0119 (N2062) TC4

**Subclause 8.1.4**

Following the third paragraph, and before NOTE 8.5, insert a new paragraph:

It is permissible to branch to an *end-block-stmt* only from within its BLOCK construct.

**Subclause 8.1.5**

Following the third paragraph, and before NOTE 8.6, insert a new paragraph:

It is permissible to branch to an *end-critical-stmt* only from within its CRITICAL construct.

[177:28-29]f08/0028 TC1

**Subclause 8.1.6.6.4**

In the first paragraph of the subclause replace the fourth item in the bulleted list with the following:

- A branch occurs within the range of a DO construct and the branch target statement is neither the *end-do* nor within the range of the same DO construct.

[178:8-9]f08/0023 TC1

**Subclause 8.1.6.7**

In the first paragraph of the subclause, in the second item in the bulleted list replace the first sentence by: “A pointer that is used in an iteration other than as the pointer in pointer assignment, allocation, or nullification, either shall be previously pointer-assigned, allocated, or nullified in that iteration or shall not have its pointer association changed during any iteration.”.

[178:13-14]f08/0025 TC1

In the third item in the bulleted list replace the second sentence by: “An allocatable object that is referenced, defined, deallocated, or has its allocation status, dynamic type, or a deferred type parameter value inquired about, in any iteration, either shall be previously allocated in that iteration or shall not be allocated or deallocated in any other iteration.”.

[178:15-16]f08/0022 TC1

Replace the fourth item in the bulleted list (“An input/output ... iteration.”) by:

- If data are written to a file record or position in one iteration, that record or position in that file shall not be read from or written to in a different iteration.

[178:16+]f08/0144 (N2092&N2102) TC4

In the first paragraph, after the fourth bullet point, add new bullet point:

- A DO CONCURRENT construct shall not contain an input/output statement that has an ADVANCE= specifier.

[178:17-18]f08/0026 TC1

Delete the fifth item in the bulleted list (“Records ... order.”).

[178:18+]f08/0026 TC1

At the end of the first paragraph, and before Note 8.9, add the new paragraph:

If records are written to a file connected for sequential access by more than one iteration, the ordering between records written by different iterations is indeterminate.

[184:13]f08/0118 (N2062&N2080) TC4

**Subclause 8.1.9.1**

In constraint C836, change “*associate-name* shall not appear” to “neither the *associate-name* nor any subobject thereof shall appear”.

[188:10+10]f08/0093 TC3

**Subclause 8.4**

In the second paragraph of Note 8.30, before “is of type character or does not appear” insert “in a STOP statement”.

At the end of Note 8.30, insert new paragraph:

If the *stop-code* in an ERROR STOP statement is of type character or does not appear, it is recommended that a processor-dependent nonzero value be supplied as the process exit status, if the processor supports that concept.

[188:23+]f08/0040 TC2

#### Subclause 8.5.1

In the bulleted list in the second paragraph of the subclause, add the following new item before the STOP statement item:

- a CALL statement that invokes the intrinsic subroutine MOVE\_ALLOC with coarray arguments;

[190:5+]f08/0134 (N2062) TC4

#### Subclause 8.5.3

Following constraint C851, add new constraint:

C851a (R859) A *stat-variable* or *errmsg-variable* in a *sync-stat* shall not be a coindexed object.

[190:16-]f08/0113 (N2042) TC4

#### Subclause 8.5.4

Following constraint C852 and before the first paragraph, insert a new paragraph:

The value of *image-set* shall not depend on the value of *stat-variable* or *errmsg-variable*.

[194:2+]f08/0098 TC3

#### Subclause 8.5.6

After syntax rule R864, insert new constraint:

C852a No specifier shall appear more than once in a given *lock-stat-list*.

[194:6-]f08/0113 (N2042&amp;N2047) TC4

Following constraint C853 and before the first paragraph, insert a new paragraph:

The *lock-variable* shall not depend on the value of *stat-variable*, *errmsg-variable*, or the *scalar-logical-variable* in the ACQUIRED\_LOCK= specifier. The *scalar-logical-variable* shall not depend on the value of the *lock-variable*, *stat-variable*, or *errmsg-variable*.

[195:2-]f08/0113 (N2042) TC4

#### Subclause 8.5.7

Before the first paragraph, insert a new paragraph:

The *stat-variable* shall not depend on the value of the *errmsg-variable*, *lock-variable*, or the *scalar-logical-variable* in the ACQUIRED\_LOCK= specifier. The *errmsg-variable* shall not depend on the value of the *stat-variable*, *lock-variable*, or the *scalar-logical-variable* in the ACQUIRED\_LOCK= specifier.

[227:15]f03/0048 TC1

#### Subclause 9.6.4.8.3

In the twenty-fifth paragraph of the subclause, delete “record positioning”.

[227:17-18]f03/0048 TC1

In the twenty-sixth paragraph, replace “A record positioning edit descriptor, such as TL and TR,” by “The edit descriptors T and TL” and replace “record position” by “file position” twice.

**Subclause 9.12**

Replace the fifth paragraph of the subclause by:

The value of a specifier in an input/output statement shall not depend on the definition or evaluation of any other specifier in the *io-control-spec-list* or *inquire-spec-list* in that statement. The value of an *internal-file-variable* or of a FMT=, ID=, IOMSG=, IOSTAT= or SIZE= specifier shall not depend on the values of any *input-item* or *io-implied-do do-variable* in the same statement.

[246:15+] f08/0030 TC1

**Subclause 10.3.1**

After constraint C1002, add a new constraint:

C1002A (R1005) An *unlimited-format-item* shall contain at least one data edit descriptor.

[249:11+] f08/0030 TC1

**Subclause 10.4**

After the seventh paragraph of the subclause, insert a new paragraph:

If format control encounters the rightmost parenthesis of an unlimited format item, format control reverts to the leftmost parenthesis of that unlimited format item. This reversion of format control has no effect on the changeable modes (9.5.2).

[249:19-20] f08/0030 TC1

In the last sentence of the eighth paragraph of the subclause, change “If format control reverts ... , the” to “The”.

[252:33-34] f03/0100 TC3

**Subclause 10.7.2.3.2**

In the seventh paragraph of the subclause, replace the final sentence (“If *w* is ... produced.”) by

“The minimum field width required for output of the form 'Inf' is 3 if no sign is produced, and 4 otherwise. If *w* is greater than zero but less than the minimum required, the field is filled with asterisks. The minimum field width for output of the form 'Infinity' is 8 if no sign is produced and 9 otherwise. If *w* is greater than or equal to the minimum required for the form 'Infinity', the form 'Infinity' is output. If *w* is zero or *w* is less than the minimum required for the form 'Infinity' and greater than or equal to the minimum required for the form 'Inf', the form 'Inf' is output. Otherwise, the field is filled with asterisks.”.

[252:37] f03/0100 TC3

In the eighth paragraph of the subclause, replace the final sentence (“If *w* is ... asterisks.”) by “If *w* is greater than zero and less than 3, the field is filled with asterisks. If *w* is zero, the output field is 'NaN'.”.

[258:14-] f08/0055 TC2

**Subclause 10.7.5.2.2**

Following the third paragraph of the subclause, add a new paragraph:

If *d* is zero, *kPEw.0* or *kPEw.0Ee* editing is used for *Gw.0* editing or *Gw.0Ee* editing respectively.

[258:15-20] f08/0055 TC2

In the original fourth paragraph of the subclause replace the second and subsequent sentences, including the two tables, by:

If the internal value is a zero value, let *s* be one. If the internal value is a number other than zero, let *N* be the decimal value that is the result of converting the internal value to *d* significant digits according to the I/O rounding mode and let *s* be the integer such that  $10^{s-1} \leq N < 10^s$ . If  $s < 0$  or  $s > d$ , *kPEw.d* or *kPEw.dEe* editing is used for *Gw.d* editing or *Gw.dEe* editing respectively, where *k* is the scale factor (10.8.5). If  $0 \leq s \leq d$ , the scale factor has no effect and *F(w-n).(d-s),n('b')* editing is used where *b* is a blank and *n* is 4 for *Gw.d* editing and *e+2* for *Gw.dEe* editing.

**Subclause 11.2.3**

In constraint C1113, after “shall be the name of a nonintrinsic module” insert “that declares a separate module procedure”.

[278:11] f03/0139 TC3

**Subclause 12.3.1**

Change “result value” to “function result”.

[279:19] f08/0054 TC2

**Subclause 12.4.2.2**

At the beginning of the subclause, replace “A procedure ... and” with “Within the scope of a procedure identifier, the procedure shall have an explicit interface if it is not a statement function and”.

[281:11-12] f03/0019 TC2

**Subclause 12.4.3.2**

Replace constraint C1209 by:

C1209 (R1201) An *interface-specification* in a generic interface block shall not specify a procedure that is specified previously in any accessible interface with the same generic identifier.

[281:25-28] f08/0132 (N2062&N2102) TC4

At the end of the first sentence of the fifth paragraph, change “or a dummy procedure” to “, dummy procedure, or procedure pointer”.

In the second sentence of the same paragraph, after “interface body, the procedure is a dummy procedure” change “; otherwise” to “. If the procedure has the POINTER attribute, it is a procedure pointer. If it is not a dummy procedure or procedure pointer”.

This makes that whole paragraph read:

An interface body in a generic or specific interface block specifies the EXTERNAL attribute and an explicit specific interface for an external procedure, dummy procedure, or procedure pointer. If the name of the declared procedure is that of a dummy argument in the subprogram containing the interface body, the procedure is a dummy procedure. If the procedure has the POINTER attribute, it is a procedure pointer. If it is not a dummy procedure or procedure pointer, it is an external procedure.

[282:7] f08/0100 (N2042) TC4

**Subclause 12.4.3.3**

In the first paragraph, after “imported in this manner and is” change “defined” to “declared”.

[282:14] f08/0100 (N2042) TC4

In the second paragraph, after “is accessed by host association and is” change “defined” to “declared”.

[286:4] f08/0001 TC1

**Subclause 12.4.3.4.5**

In the third paragraph, in the third item in the bulleted list, after “the other has the POINTER attribute”, insert “and not the INTENT(IN) attribute”.

[286:12-13] f08/0053 TC1

In the third paragraph of the subclause, in constraint C1214 replace “two ... identifier” by “if two procedures have the same generic identifier, their **dtv** arguments (9.6.4.8.3)”.

[286:12-13] f08/0082 TC2

In the third paragraph of the subclause, in constraint C1214 as amended in Technical Corrigendum 1 replace “the same” by “that”.

[286:38] f08/0053 TC1

In the fifth paragraph of the subclause, replace “applies to” by “is consistent with”.

**Subclause 12.4.3.6**

In rule R1213 in the first paragraph, following the line “**or POINTER**”, add the new line

**or PROTECTED**

[288:3] f03/0064 TC3

Append the following new sentence to the second paragraph of the subclause, “The interface specified by *interface-name* shall not depend on any characteristic of a procedure identified by a *procedure-entity-name* in the *proc-decl-list* of the same procedure declaration statement.”.

[292:15-16] f08/0068 TC2

**Subclause 12.5.2.3**

Replace the second paragraph of the subclause by:

If a nonpointer dummy argument without the VALUE attribute corresponds to a pointer actual argument that is pointer associated with a target,

- if the dummy argument is polymorphic, it becomes argument associated with that target;
- if the dummy argument is nonpolymorphic, it becomes argument associated with the declared type part of that target.

[292:17-18] f08/0068 TC2

Replace the third paragraph of the subclause by:

If a present nonpointer dummy argument without the VALUE attribute corresponds to a nonpointer actual argument,

- if the dummy argument is polymorphic it becomes argument associated with that actual argument;
- if the dummy argument is nonpolymorphic, it becomes argument associated with the declared type part of that actual argument.

[293:5] f08/0067 TC2

**Subclause 12.5.2.4**

Append to the second paragraph of the subclause the sentence:

If the actual argument is a polymorphic assumed-size array, the dummy argument shall be polymorphic.

[293:6] f03/0103 TC2

In the third paragraph of the subclause, add the following sentence at the start of the paragraph:

The kind type parameter values of the actual argument shall agree with the corresponding ones of the dummy argument.

In the original first sentence of the third paragraph change “The type parameter values of the actual argument” to “The length type parameter values of a present actual argument”.

[293:10] f03/0103 TC2

In the fourth paragraph of the subclause, before “scalar dummy argument” insert “present”.

[294:40, 294:42-295:2] f08/0069 TC2

In the second sentence of the seventeenth paragraph of the subclause, after “has INTENT (OUT),” change “the actual argument” to “the effective argument” and delete the last sentence of the paragraph (“If ... undefined.”).

[295:3] f08/0014 TC1

In paragraph 18 of the subclause, after “If” insert “the procedure is nonelemental and”.

[295:3] f08/0135 (N2062&N2080) TC4

In the eighteenth paragraph, after applying the changes in Technical Corrigendum 1, between “is nonelemental” and “and the actual argument”, insert “, the dummy argument does not have the VALUE attribute.”. This makes the whole sentence read:

If the procedure is nonelemental, the dummy argument does not have the VALUE attribute, and the actual argument is an array section having a vector subscript, the dummy argument is not

definable and shall not have the ASYNCHRONOUS, INTENT (OUT), INTENT (INOUT), or VOLATILE attributes.

[295:4+] f08/0122 (N2062&N2080) TC4

After the eighteenth paragraph, add the following new paragraph before NOTE 12.24:

If the dummy argument has a coarray ultimate component, the corresponding actual argument shall have the VOLATILE attribute if and only if the dummy argument has the VOLATILE attribute.

[295:6] f08/0136 (N2062) TC4

In constraint C1238, append at the end of the sentence: “, unless the dummy argument has the VALUE attribute”.

[295:9] f08/0136 (N2062) TC4

In constraint C1239, after “ASYNCHRONOUS attribute” insert: “, but does not have the VALUE attribute”.

[295:13] f08/0136 (N2062) TC4

In constraint C1240, after “ASYNCHRONOUS attribute” insert: “, but does not have the VALUE attribute”.

[295:16-17] f08/0059 TC2

#### **Subclause 12.5.2.5**

Replace the first paragraph of the subclause by:

The requirements in this subclause apply to an actual argument with the ALLOCATABLE or POINTER attribute that corresponds to a dummy argument with the same attribute.

[296:4-5] f08/0059 TC2

Delete the fourth paragraph of the subclause, that is “The values of assumed type parameters ... effective argument.”.

[296:12+] f08/0059 TC2

#### **Subclause 12.5.2.6**

Following the third paragraph of the subclause, add the new paragraph:

The values of assumed type parameters of a dummy argument are assumed from the corresponding type parameters of its effective argument.

[296:35] f08/0059 TC2

#### **Subclause 12.5.2.7**

Add the following sentence at the end of the third paragraph of the subclause:

The values of assumed type parameters of a dummy argument are assumed from the corresponding type parameters of its effective argument.

[297:5] f08/0048 TC2

#### **Subclause 12.5.2.8**

In the second paragraph of the subclause, add at the end of the sentence, “or an element of a simply contiguous array”.

[300:14] f08/0117 (N2062&N2102) TC4

#### **Subclause 12.5.2.13**

In item (3)(b) of the first paragraph, after “target other than” insert “a coindexed object or”.

[300:22] f08/0117 (N2062) TC4

In item (4)(b) of the first paragraph, after “target other than” insert “a coindexed object or”.

[306:31] f08/0096 TC3

#### **Subclause 12.6.2.2**

In the first paragraph of the subclause, in constraint C1255, after “(15.3.5, 15.3.6)” insert “that is not an array with the VALUE attribute,”.

[307:5,9] f03/0139 TC3

In the third paragraph of the subclause, change the two occurrences of “result variable” to “function result”.

[307:12-20] f03/0139 TC3

In the fourth paragraph of the subclause, in the first two sentences, change the three occurrences of “result variable” to “function result”. Delete the third sentence: “The characteristics ... result variable”. In each of the final four sentences change “result variable” to “function result”.

Further, in the fifth sentence (before the deletion above) change “If the function result is a pointer” to “If the function result is a data pointer”.

[307:20+2, 20+4, 20+5] f03/0139 TC3

In Note 12.41 replace the first sentence with “The function result is similar to any other entity (variable or procedure pointer) local to the function subprogram.”. Also change “this variable” to “this entity” and change “that variable” to “that entity”.

[309:23,24] f03/0139 TC3

#### **Subclause 12.6.2.5**

In the third paragraph of the subclause, replace the two occurrences of “result variable name” by “name of the function result”.

[310:2,3] f03/0139 TC3

#### **Subclause 12.6.2.6**

In the third paragraph of the subclause, after “name of its result” delete “variable”, and delete the second sentence “The characteristics ... the result variable.”.

[310:5,6] f03/0139 TC3

In the same paragraph, in the penultimate sentence replace “result variables identify the same variable, although their names need not be the same” with “result names identify the same entity”. In the final sentence, replace “scalars” with “scalar variables”.

[310:20] f08/0058 TC2

In the eighth paragraph of the subclause append the sentence:

A name that appears as a *result-name* in an ENTRY statement shall not appear in any executable statement that precedes the first RESULT clause with that name.

[310:23] f08/0058 TC2

In the ninth paragraph of the subclause append the sentence:

A name that appears as a *result-name* in an ENTRY statement shall not appear in the expression of a statement function that precedes the first RESULT clause with that name unless the name is also a dummy argument of that statement function.

[311:34+] f08/0141 (N2092) TC4

#### **Subclause 12.6.4**

In the first paragraph, following constraint C1275 add new constraint:

C1275a A statement function shall not be of a parameterized derived type.

[312:12+] f08/0065 TC2

#### **Subclause 12.7**

In the first paragraph of the subclause, insert as the second item in the bulleted list:

- a module procedure in an intrinsic module, if it is specified to be pure,

[312:19+] f08/0032 TC2

In the second paragraph of the subclause, following constraint C1276 add:

C1276a The result variable of a pure function shall not be such that finalization of a reference to the function would reference an impure procedure.

C1276b A pure function shall not have a polymorphic allocatable result variable.

[312:21+] f08/0031 TC2

and following constraint C1277 add:

- C1277a An INTENT(OUT) argument of a pure procedure shall not be such that finalization of the actual argument would reference an impure procedure.

[312:23+] f08/0033 TC1

Following constraint C1278 and Note 12.47, insert new constraint:

- C1278a An INTENT(OUT) dummy argument of a pure procedure shall not be polymorphic.

[312:23+] f08/0143 (N2092&N2093) TC4

In constraint C1278a which was added by Technical Corrigendum 1, after the word “polymorphic” insert “or have a polymorphic allocatable ultimate component”.

[312:31] f08/0084 TC3

In the second paragraph of the subclause, in constraint C1283, after “association” insert “, is a dummy argument of a pure function”.

[312:35] f08/0148 (N2086) TC4

In the second paragraph, replace list item (3) in constraint C1283 by:

- (3) as the *expr* corresponding to a component in a *structure-constructor* if the component has the POINTER attribute or has a pointer component at any level of component selection,

[312:37] f08/0088 TC3

In constraint C1283, in list item (4) delete “or” and insert new list item:

- (4a) as the *source-expr* in a SOURCE= clause if the designator is of a derived type that has an ultimate pointer component, or

[313:4+] f08/0033 TC1

Following constraint C1284, insert new constraint and new note:

- C1284a A statement that might result in the deallocation of a polymorphic entity is not permitted in a pure procedure.

**NOTE 12.48x**

Apart from the DEALLOCATE statement, this includes intrinsic assignment if the variable has a polymorphic allocatable component at any level of component selection that does not involve a pointer component but which might involve one or more allocatable components.

[314:3] f03/0139 TC3

**Subclause 12.8.1**

In constraint C1290, after “The result” delete “variable”.

[314:4-5] f08/0049 TC1

In constraint C1290, delete “, and shall not ... constant expression”.

[314:5+] f08/0024 and f08/0049 TC1

Following constraint C1290 insert two new constraints:

- C1290a The *specification-part* of an elemental subprogram shall specify the intents of all of its dummy arguments that do not have the VALUE attribute.

- C1290b In the *specification-expr* that specifies a type parameter value of the result of an elemental function, an object designator with a dummy argument of the function as the base object shall appear only as the subject of a specification inquiry, and that specification inquiry shall not depend on a property that is deferred.

At the end of the subclause, insert the new paragraph:

In a reference to an elemental procedure, if any argument is an array, all actual arguments that correspond to INTENT (OUT) or INTENT (INOUT) dummy arguments shall be arrays. All actual arguments shall be conformable.

[314:9-10] f08/0018 TC1

### Subclause 12.8.2

In the first paragraph of the subclause delete the sentence “For those elemental ... conformable.”.

[314:14-17] f08/0018 TC1

### Subclause 12.8.3

Delete the sentence “In a reference ... conformable with them.”.

[316:12+] f03/0139 TC3

### Subclause 13.2.1

Following the sixth paragraph of the subclause, add the new paragraph:

An argument to an intrinsic procedure other than ASSOCIATED, NULL, or PRESENT shall be a data object.

[316:24-25] f08/0003 TC1

### Subclause 13.2.4

In the second sentence of the first paragraph of the subclause, replace “an optional” by “a” and replace “, if present, specifies” by “can specify”.

[319, 322, 323] f08/0003 TC1

### Subclause 13.5

In Table 13.1 replace

“ALL (MASK [, DIM])”	by	“ALL (MASK) or (MASK, DIM)”;
“ANY (MASK [, DIM])”	by	“ANY (MASK) or (MASK, DIM)”;
“NORM2 (X [, DIM])”	by	“NORM2 (X) or (X, DIM)”;
“PARITY (MASK [, DIM])”	by	“PARITY (MASK) or (MASK, DIM)”;
“THIS_IMAGE (COARRAY[, DIM])”	by	“THIS_IMAGE (COARRAY) or (COARRAY, DIM)”.

[325:7-12] f08/0008 TC2

### Subclause 13.7.1

In the second paragraph of the subclause, replace the fourth to sixth sentences (“A program ... invoked.”) by:

A program shall not invoke an intrinsic procedure under circumstances where a value to be assigned to a subroutine argument or returned as a function result is not representable by objects of the specified type and type parameters.

[325:12+] f08/0008 TC2

Add the following as the third paragraph of the subclause:

If an IEEE infinity is assigned or returned by an intrinsic procedure, the intrinsic module IEEE\_ARITHMETIC is accessible, and the actual arguments were finite numbers, the flag IEEE\_OVERFLOW or IEEE\_DIVIDE\_BY\_ZERO shall signal. If an IEEE NaN is assigned or returned, the actual arguments were finite numbers, the intrinsic module IEEE\_ARITHMETIC is accessible, and the exception IEEE\_INVALID is supported, the flag IEEE\_INVALID shall signal. If no IEEE infinity or NaN is assigned or returned, these flags shall have the same status as when the intrinsic procedure was invoked.

[328:2] f08/0003 TC1

**Subclause 13.7.10**

Replace the subclause heading by “ALL (MASK, DIM) or ALL (MASK)”.

[328:7] f08/0003 TC1

In the description of the DIM argument, delete “(optional)”.

[328:10] f08/0003 TC1

In the description of Result Characteristics, replace “is absent” by “does not appear”.

[329:6] f08/0003 TC1

**Subclause 13.7.13**

Replace the subclause heading by “ANY (MASK, DIM) or ANY (MASK)”.

[329:11] f08/0003 TC1

In the description of the DIM argument, delete “(optional)”.

[329:14] f08/0003 TC1

In the description of Result Characteristics, replace “is absent” by “does not appear”.

[330:20] f08/0103 (N2042) TC4

**Subclause 13.7.16**

In the fifth paragraph (Result Value) Case (ii), after “with TARGET” insert: “and, if TARGET is an internal procedure, they have the same host instance”.

[330:22] f08/0103 (N2042) TC4

In the same paragraph Case (iii), after “same procedure” insert: “and, if the procedure is an internal procedure, they have the same host instance”.

[330:36+] f08/0004 TC2

**Subclause 13.7.16**

Following the fifth paragraph of the subclause, insert the following note:

**NOTE 13.8a**

The references to TARGET in the above cases are referring to properties that might be possessed by the actual argument, so the case of TARGET being a disassociated pointer will be covered by case (iii), (vi), or (vii).

[332:25] f08/0027 TC1

**Subclause 13.7.21**

In the fourth paragraph of the subclause, change “CALL ATOMIC\_REF (I [3], VAL)” to “CALL ATOMIC\_REF (VAL, I [3])”.

[333:12-14] f08/0019 TC1

**Subclause 13.7.24**

In the third paragraph of the subclause, in the lines beginning N1 and N2, replace “of type integer and nonnegative” by “an integer scalar with a nonnegative value” and in the line beginning X, after “real” insert “; if the function is transformational, X shall be scalar”.

[334:12-14] f08/0019 TC1

**Subclause 13.7.27**

In the third paragraph of the subclause, in the lines beginning N1 and N2, replace “of type integer and nonnegative” by “an integer scalar with a nonnegative value” and in the line beginning X, after “real” insert “; if the function is transformational, X shall be scalar”.

[338:31] f08/0003 TC1

**Subclause 13.7.41**

In the description of the DIM argument, after “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”.

[347:31-32] f08/0020 TC1

**Subclause 13.7.61**

In the third paragraph of the subclause, for the VALUE argument, replace “for ... 7.1.5.5.2)” by “for the operator == or the operator .EQV.”.

[351:18] f08/0064 TC2

**Subclause 13.7.67**

In the third paragraph of the subclause, in the description of the STATUS argument, after “either has no value” change “or” to a comma. After “assigned to VALUE,” insert “or the VALUE argument is not present.”.

[360:4] f08/0003 TC1

**Subclause 13.7.90**

In the description of the DIM argument, after “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”.

[360:25] f08/0003 TC1

**Subclause 13.7.91**

In the description of the DIM argument, after “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”.

[368:26] f08/0102 (N2042) TC4

**Subclause 13.7.110**

In the fourth paragraph (Result Characteristics), replace “Same as TSOURCE.” by:

Same type and type parameters as TSOURCE. Because TSOURCE and FSOURCE are required to have the same type and type parameters (for both the declared and dynamic types), the result is polymorphic if and only if both TSOURCE and FSOURCE are polymorphic.

[372:18,19] f08/0040 TC2

**Subclause 13.7.118**

In the third paragraph of the subclause, in the description of the FROM argument, change “type and rank” to “type, rank, and corank”.

In the description of the TO argument, after “same rank” insert “and corank”.

[372:18] f08/0106 (N2042) TC4

In the third paragraph (Arguments) in the description of FROM, after “It shall be allocatable” add “and shall not be a coindexed object”.

[372:19] f08/0106 (N2042) TC4

In the same paragraph in the description of TO, after “It shall be allocatable” add “and shall not be a coindexed object”.

[372:29+] f08/0040 TC2

Before the seventh paragraph of the subclause (**Example.**) insert the following new paragraph:

When a reference to MOVE\_ALLOC is executed for which the FROM argument is a coarray, there is an implicit synchronization of all images. On each image, execution of the segment (8.5.2) following the CALL statement is delayed until all other images have executed the same statement the same number of times.

[374:24] f08/0003 TC1

**Subclause 13.7.123**

Replace the subclause heading by “**NORM2 (X, DIM) or NORM2 (X)**”.

[374:29] f08/0003 TC1

In the description of the DIM argument, delete “(optional)”.

[374:31] f08/0003 TC1

In the description of Result Characteristics, replace “is absent” by “does not appear”.

[377:20] f08/0003 TC1

**Subclause 13.7.128**

Replace the subclause heading by “**PARITY (MASK, DIM) or PARITY (MASK)**”.

[377:25] f08/0003 TC1

In the description of the DIM argument, delete “(optional)”.

[377:28] f08/0003 TC1

In the description of Result Characteristics, replace “is absent” by “does not appear”.

[387:32] f08/0078 TC2

**Subclause 13.7.153**

In the fifth paragraph of the subclause, in Case (iv), change “cannot distinguish” to “does not distinguish”.

**Subclause 13.7.157**

In the fifth paragraph (Result Value), change “ $\max(e-p, e_{\min}-1)$ ” to “ $e-p$ ”. After “that of X” replace “; if there are two such values” by “, provided this result is representable; otherwise the result is the same as that of TINY (X). If there are two extended model values equally near to X,”

This makes the whole paragraph read:

If X does not have the value zero and is not an IEEE infinity or NaN, the result has the value  $b^{e-p}$ , where  $b$ ,  $e$ , and  $p$  are as defined in 13.4 for the value nearest to X in the model for real values whose kind type parameter is that of X, provided this result is representable; otherwise the result is the same as that of TINY (X). If there are two extended model values equally near to X, the value of greater absolute value is taken. If X has the value zero, the result is the same as that of TINY (X). If X is an IEEE infinity, the result is an IEEE NaN. If X is an IEEE NaN, the result is that NaN.

[390:6]f08/0021 TC1

**Subclause 13.7.160**

In the third paragraph of the subclause, change “has any deferred type parameters” to “is unlimited polymorphic or has any deferred type parameters,”.

[392:6]f08/0003 TC1

**Subclause 13.7.165**

In the subclause heading replace “or THIS\_IMAGE (COARRAY[, DIM])” by “, THIS\_IMAGE (COARRAY) or THIS\_IMAGE (COARRAY, DIM)”.

[392:11]f08/0003 TC1

In the description of the DIM argument, delete “(optional)”.

[393:18]f08/0137 (N2062) TC4

**Subclause 13.7.168**

In the third paragraph (Arguments), to the definition of MOLD, append: “If the storage size of SOURCE is greater than zero and MOLD is an array, a scalar with the type and type parameters of MOLD shall not have a storage size equal to zero.”

[394:27]f08/0003 TC1

**Subclause 13.7.171**

In the description of the DIM argument, after “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”.

[395:11]f08/0003 TC1

**Subclause 13.7.172**

In the description of the DIM argument, after “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”.

[397:7]f08/0065 TC2

**Subclause 13.8.2.1**

Append the following sentence to the second paragraph of the subclause:

The module procedures described in 13.8.2 are pure.

[399:17]f08/0109 (N2086&amp;N2102) TC4

**Subclause 13.8.2.16**

In the second paragraph, in constraint C1302, replace “variable of type LOCK\_TYPE” by “variable with declared type LOCK\_TYPE”.

[403:7-11]f03/0030 TC3

**Subclause 14.3**

In the first paragraph of the subclause, replace the first two bulleted items in the list by:

- IEEE\_OVERFLOW occurs in an intrinsic real addition, subtraction, multiplication, division, or conversion by the intrinsic function REAL, as specified by IEC 60559:1989 if

IEEE\_SUPPORT\_DATATYPE is true for the operands of the operation or conversion, and as determined by the processor otherwise. It occurs in an intrinsic real exponentiation as determined by the processor. It occurs in a complex operation, or conversion by the intrinsic function CMPLX, if it is caused by the calculation of the real or imaginary part of the result.

- IEEE\_DIVIDE\_BY\_ZERO occurs in a real division as specified by IEC 60559:1989 if IEEE\_SUPPORT\_DATATYPE is true for the operands of the division, and as determined by the processor otherwise. It is processor-dependent whether it occurs in a real exponentiation with a negative exponent. It occurs in a complex division if it is caused by the calculation of the real or imaginary part of the result.

[406:15+] f08/0009 TC1

#### **Subclause 14.9**

In the first paragraph, add a new item after the second item of the bulleted list:

- the IEEE function abs shall be provided by the intrinsic function ABS,

[408:1-] f08/0104 (N2042&N2047&N2094) TC4

#### **Subclause 14.10**

In the third paragraph, Table 14.1, for procedure IEEE\_SUPPORT\_ROUNDING change the “Class” column entry from “I” to “T”.

In the same paragraph, Table 14.2, for procedures IEEE\_SUPPORT\_FLAG and IEEE\_SUPPORT\_HALTING change the “Class” column entries from “I” to “T”.

[418:16] f08/0104 (N2042) TC4

#### **Subclause 14.11.27**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[418:32] f08/0104 (N2042) TC4

#### **Subclause 14.11.28**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[420:4] f08/0104 (N2042) TC4

#### **Subclause 14.11.32**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[426:19] f08/0104 (N2042) TC4

#### **Subclause 15.2.3.2**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[428:9] f08/0104 (N2042) TC4

#### **Subclause 15.2.3.5**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[428:21] f08/0104 (N2042) TC4

#### **Subclause 15.2.3.6**

In the second paragraph (Class), change “Inquiry function” to “Transformational function”.

[431:6] f03/0053 TC3

#### **Subclause 15.3.4**

In the first paragraph of the subclause, replace the first sentence by: “Interoperability between derived types in Fortran and struct types in C is provided by the BIND attribute on the Fortran type.”.

[431:11-] f08/0057 TC2

In the first paragraph of the subclause, before C1505 add a new constraint:

C1504a (R425) A derived type with the BIND attribute shall have at least one component.

[431:12+2] f03/0053 TC3

In the first paragraph of the subclause, in Note 5.11 after “is interoperable” insert “with a C struct type”.

[431:13-18] *f03/0053 TC3*

In the second paragraph of the subclause, change the four occurrences of “Fortran derived type” to “derived type” and change the single occurrence of “Fortran type” to “derived type”.

[433:7] *f03/0139 TC3*

#### **Subclause 15.3.7**

In the second paragraph of the subclause, in item (2) (a) of the list, replace “result variable is a scalar” by “result is a scalar variable”.

[433:12] *f08/0096 TC3*

In item (4) of the list, after “any” insert “scalar”.

[436:15] *f08/0116 (N2062) TC4*

#### **Subclause 15.5.1**

Append the following sentence to the first paragraph: “A C function that has an inline definition and no external definition is not considered to be defined in this sense.”

[436:16-19] *f08/0116 (N2062) TC4*

Replace the second paragraph by:

If the procedure is defined by means other than Fortran,

- it shall be describable by a C prototype that is interoperable with the interface, and
- if it is accessed using its binding label, it shall
  - have a name that has external linkage as defined by 6.2.2 of ISO/IEC 9899:1999, and
  - have the same binding label as the interface.

[440:4] *f08/0120 (N2062&N2102) TC4*

#### **Subclause 16.3.1**

In item (1) of the first paragraph, after “named constants,” insert “named procedure pointers,”.

[441:7,10] *f03/0139 TC3*

#### **Subclause 16.3.1**

In the fourth paragraph of the subclause, in each of the second and third bulleted items in the list, replace “result variable” by “function result”.

[441:18-20] *f03/0139 TC3*

#### **Subclause 16.3.3**

Replace the three occurrences of “result variable” by “function result”.

[449:3,4] *f03/0139 TC3*

#### **Subclause 16.5.3.1**

Replace “result variables” with “function results that are variables”.

[450:20] *f03/0139 TC3*

#### **Subclause 16.5.3.4**

In the sixth paragraph of the subclause, replace “result variables” by “function results that are variables”.

[455:4-10] *f03/0124 TC1*

#### **Subclause 16.6.6**

In the first paragraph replace item (1) entirely by:

- (1) When a scalar variable of intrinsic type becomes defined, all totally associated variables of different type become undefined.  
When a double precision scalar variable becomes defined, all partially associated scalar variables become undefined.  
When a scalar variable becomes defined, all partially associated double precision scalar variables become undefined.

[456:11] *f03/0139 TC3*

In item (15)(e) of the list, replace “the result variable of a function” by “a variable that is the function result of that procedure”.

[459:17+] *f08/0093 TC3*

### **Subclause A.2**

After the bullet item “how soon an image terminates if another image initiates error termination (2.3.5);” insert new bullet point:

- the recommended process exit status when error termination is initiated other than by an ERROR STOP statement with an integer *stop-code* (2.3.5);

[459:36+] *f08/0081 TC2*

After “whether and when an object is finalized ... (4.5.6.3);” insert a new bullet point:

- whether an object is finalized by a deallocation in which an error condition occurs (4.5.6.3);

[460:5+] *f08/0081 TC2*

After “the order ... event described in 6.7.3.2;” insert a new bullet point:

- whether an allocated allocatable subobject is deallocated when an error condition occurs in the deallocation of an object (6.7.3.2);

[462:24+] *f03/0030 TC3*

After the fifth bullet from the end of the clause “the extent to which a processor supports IEEE arithmetic (14);”, insert new bullet points:

- the conditions under which IEEE\_OVERFLOW is raised in a calculation involving non-IEC 60559:1989 floating-point data;
- the conditions under which IEEE\_OVERFLOW and IEEE\_DIVIDE\_BY\_ZERO are raised in a floating-point exponentiation operation;
- the conditions under which IEEE\_DIVIDE\_BY\_ZERO is raised in a calculation involving non-IEC 60559:1989 floating-point data;

[487:28] *f03/0048 TC1*

### **Subclause C.6.2**

In the third sentence of the first paragraph, delete “record positioning”.

[527:18] *f08/0036 TC1*

### **Subclause C.13.3.6**

In the third paragraph of the subclause, replace “ $|X_i|$ ” by “ $|X_i|^2$ ”.