ISO/IEC 1539-1:2010 - TECHNICAL CORRIGENDUM 3

Introduction
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.".

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

Subclause 1.3.121
Delete term **1.3.121 result variable**.

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)

Subclause 1.6.3
In the first paragraph of the subclause, replace "Any" by "Except as identified in this subclause, any".

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.

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.

Subclause 2.2.3
In the second paragraph of the subclause, after "data objects" insert "or procedure pointers".

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.

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.

Subclause 4.3.1.2
In the second paragraph of the subclause, in the final sentence, change “function result variable” to “function result”.

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”.

Subclause 4.5.2.1
After constraint C427 insert new constraint:

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

Subclause 4.5.2.3
In constraint C436, after “appears,” insert “the type shall have at least one component,”.

Subclause 4.5.3.1
In constraint C438, after “shall appear” insert “exactly once”.

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,”.

Subclause 5.1
In the second paragraph of the subclause, change “its result variable” to “the function result”.

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.

Subclause 5.2.1
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.

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*

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*.”.

Before syntax rule R521 insert new BNF term:

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

Replace syntax rule R521 with:

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

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.

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*.".

Replace syntax rule R522 by:
R522 *implied-shape-spec* **is** *assumed-implied-spec*, *assumed-implied-spec-list*

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-spec*s in its *array-spec*.

Subclause 5.4.11
In the second paragraph of the subclause, in the final sentence change "shape" to "rank".

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.

Subclause 5.5
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".

Subclause 5.7.1.1
In the second paragraph of the subclause, in constraint C587 change "result variable" to "function result".

Subclause 5.7.2.1
In the second paragraph of the subclause, in constraint C5100 change "result variable" to "function result".

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.

Subclause 6.7.3.2
In the second paragraph of the subclause, after "function result" delete "variable".

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.

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,

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

Subclause 7.1.12
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,

Subclause 7.2.2.2
In syntax rule R737, add new production:

or *expr*

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

Following constraint C724, add new constraint:

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

Subclause 8.1.3.1
Following constraint C804, add new constraint:

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

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.

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*.

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.".

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'.".

Subclause 12.3.1
Change "result value" to "function result".

Subclause 12.4.3.6
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.".

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,".

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

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".

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".

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".

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.".

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".

Subclause 12.7
In the second paragraph of the subclause, in constraint C1283, after "association" insert ", is a dummy argument of a pure function".

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

Subclause 12.8.1
In constraint C1290, after "The result" delete "variable".

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.

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.

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.”.

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

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”.

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”.

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

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”.

Subclause 16.3.3
Replace the three occurrences of “result variable” by “function result”.

Subclause 16.5.3.1
Replace “result variables” with “function results that are variables”.

Subclause 16.5.3.4
In the sixth paragraph of the subclause, replace “result variables” by “function results that are variables”.

Subclause 16.6.6
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”.

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);

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;