DRAFT TECHNICAL CORRIGENDUM 1 - PART TWO

Note for WG5: The edits in this document are drafted on the assumption that the relevant interpretations in N1622 are approved. Those which ***are*** *approved by WG5 will be added to those in N1620 to form a new document for eventual submission to SC22; those which are not approved will be withdrawn for further consideration.*

[80:9] *f03/045*

Subclause 5.1.2.5.4
In constraint C544, insert "a finalizable type or" before "a type".

[98:20] *f03/046*

Subclause 5.5.2
In constraint C588, insert "a polymorphic pointer," before "an allocatable variable".

[266:8] *f03/044*

Subclause 12.3.2.5
In line 3 of the subclause, change "referenced" to "invoked".

[268:17] *f03/043*

Subclause 12.4.1.1
In line 1 of the subclause, after "procedure" insert "or a procedure pointer component".

[292:18+] *f03/047*

Subclause 13.2.2
[Note for WG5: The N1622 interpretation edit added a new subclause 13.2 (presumably 13.3 was intended) with renumbering of subsequent subclauses. However such large-scale renumbering is impractical in a corrigendum. Therefore it is suggested that the new subclause be 13.2.3.]

Following subclause 13.2.2, add new subclause 13.2.3:

13.2.3 Polymorphic intrinsic function arguments and results

Table 13.1 specifies the intrinsic functions that are allowed to have polymorphic arguments, and the arguments that are allowed to be polymorphic.

Table 13.1: **Polymorphic intrinsic function arguments**

Function name	Arguments permitted to be polymorphic
ALLOCATED	ARRAY, SCALAR
ASSOCIATED	POINTER, TARGET
CSHIFT	ARRAY
EOSHIFT	ARRAY, BOUNDARY
EXTENDS_TYPE_OF	A, MOLD
LBOUND	ARRAY
MERGE	TSOURCE, FSOURCE
MOVE_ALLOC	FROM, TO
PACK	ARRAY, VECTOR
RESHAPE	SOURCE, PAD
SAME_TYPE_AS	A, B
SHAPE	SOURCE
SIZE	ARRAY
SPREAD	SOURCE
TRANSFER	SOURCE
TRANSPOSE	MATRIX
UBOUND	ARRAY
UNPACK	VECTOR, FIELD

The intrinsic functions shown in table 13.2 have a polymorphic result if and only if the specified argument is polymorphic. Where the result is specified to have the same type and type parameters as the argument specified in table 13.2, the result has the same dynamic type as the specified argument. If the specified argument is unlimited polymorphic the result is unlimited polymorphic; otherwise it has the same declared type as the specified argument. If another argument is required to have the same type as the specified argument, it shall be polymorphic if and only if the specified argument is polymorphic, and have the same dynamic type as the specified argument. If the specified argument is unlimited polymorphic, the other argument shall also be unlimited polymorphic; otherwise, it shall have the same declared type as the specified argument.

Table 13.2: **Polymorphic intrinsic function results**

Function name	Argument that determines result characteristics
CSHIFT	ARRAY
EOSHIFT	ARRAY
MERGE	TSOURCE
PACK	ARRAY
RESHAPE	SOURCE
SPREAD	SOURCE
TRANSPOSE	MATRIX
UNPACK	VECTOR

[316:5-6] *f03/054*

Subclause 13.7.37
In lines 1 and 2 of the Result Value paragraph of the subclause, replace "model ... X" by "representation for the value of X in the model (13.4) that has the radix of X but no limits on exponent values".

[317:8] *f03/054*

Subclause 13.7.40
In line 2 of the Result Value paragraph of the subclause, replace "model ... X" by "representation for the value of X in the model that has the radix of X but no limits on exponent values".

[347:22] *f03/055*

Subclause 13.7.100
In line 2 of the Result Value paragraph of the subclause, replace "model ... X" by "value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken".

[351:5] *f03/054*

Subclause 13.7.107
In line 2 of the Result Value paragraph of the subclause, replace "model ... X" by "representation for the value of X in the model that has the radix of X but no limits on exponent values".

[353:9] *f03/055*

Subclause 13.7.113
In line 2 of the Result Value paragraph of the subclause, replace "model ... X" by "value nearest to X in the model for real values whose kind type parameter is that of X; if there are two such values, the value of greater absolute value is taken".

[365:13-15] *f03/030*

Subclause 14.2
In each of line 1 and line 3 of the IEEE_OVERFLOW paragraph of the subclause, after "assignment" add "with finite operands".

[365:18] *f03/030*

In line 1 of the IEEE_DIVIDE_BY_ZERO paragraph of the subclause, insert "finite" before "nonzero numerator".

[376:15] *f03/034*

Subclause 14.10.12

In line 2 of the Result Value paragraph of the subclause, after "Note:" insert "if X is normal,".

[376:17+] *f03/034*

In the Result Value paragraph of the subclause, add an extra case:

Case (iii): If IEEE_SUPPORT_INF(X) is true and X is infinite, the result is +infinity.

[389:0+11] *f03/039*

Subclause 14.11

In line 2 of the final paragraph of Note 14.17, insert "overflow or underflow" before "exception".

[389:0+15+] *f03/039*

At the end of the final paragraph of Note 14.17, add

This HYPOT function does not handle infinite arguments in the same way that the hypot function in the C International Standard does.

[397:3] *f03/053*

Subclause 15.2.2

At the end of the first paragraph of the subclause, add

Each has the BIND attribute but is not interoperable with any C struct type.