INTERPRETATION UPDATE PAGES

To: J3
From: Malcolm Cohen
Subject: Interpretation Update Pages: Standing Document 018 (12-018r1)
Date: 2012/08/02

This document contains insertions for every interpretation edit that has been published as a corrigendum to ISO/IEC 1539-1:2010.

The following pages are intended for insertion into a loose-leaf binder version of 10-007r1. This document needs to be printed single-sided for this to work.

Most edits are followed by a "making the whole paragraph read" summary; in such summaries deleted text appears struck-out ~~like this~~ and new text is wavy-underlined like this. (NB: This has not been done when it might be more confusing than helpful.)

This version contains the soon-to-be-published corrigendum 1.

12-018r1 differs from 12-018 in that errors have been corrected on pages 24.1, 328.1, 329.1, and 392.1

Interp **F08/0046**, Status: *Corrigendum 1.*

Ref: Introduction, 2nd paragraph, 4th bullet, [xv]

At the end of the 4th bullet (beginning "Data declaration"),
Insert the following sentence:

> An array or an object with a nonconstant length type parameter can have the VALUE attribute.

Interp **F08/0051**, Status: *Corrigendum 1.*

Ref: Introduction, 2nd paragraph, last bullet, [xvi]

In the last bullet item (beginning “Programs and procedures”),
Before “An impure”
Insert the following sentence:

An argument to a pure procedure can have default INTENT if it has the VALUE attribute.

Interp **F08/0037**, Status: *Corrigendum 1.*

Ref: Introduction, 2nd paragraph, last bullet, [xvi]

In the last bullet item (beginning “Programs and procedures”),
Before “The FUNCTION and SUBROUTINE”
Insert the following sentence:

The PROTECTED attribute can be specified by the procedure declaration statement.

NOTE: This interp also has an edit on page 287.

These two edits make the last bullet paragraph read:

- Programs and procedures:
  An empty CONTAINS section is allowed. An internal procedure can be used as an actual argument or procedure pointer target. ALLOCATABLE and POINTER attributes are used in generic resolution. Procedureness of a dummy argument is used in generic resolution. An actual argument with the TARGET attribute can correspond to a dummy pointer. A null pointer or unallocated allocatable can be used to denote an absent nonallocatable nonpointer optional argument. An argument to a pure procedure can have default INTENT if it has the VALUE attribute. An impure elemental procedure processes array arguments in array element order. The PROTECTED attribute can be specified by the procedure declaration statement. The FUNCTION and SUBROUTINE keywords can be omitted from the END statement for a module or internal subprogram. A line in the program is permitted to begin with a semicolon.

Interp **F08/0011 and F08/0033**, Status: *Corrigendum 1.*

Ref: 1.6.2, 1st paragraph, 1st sentence, [24:9-10,11+]

NOTE: Interp F08/0033 also has an edit on pages 312 and 313.

Change the first paragraph as show below and insert new paragraphs afterwards, making the whole subclause read:

1.6.2 Fortran 2003 compatibility

1 ~~This~~Except as identified in this subclause, this part of ISO/IEC 1539 is an upward compatible extension to the preceding Fortran International Standard, ISO/IEC 1539-1:2004 (Fortran 2003). ~~Any~~Except as identified in this subclause, any standard-conforming Fortran 2003 program remains standard-conforming under this part of ISO/IEC 1539.

2 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.

3 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.

Interp **F08/0013**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 0^{th} and 9^{th} paragraphs, [76:10-,25-26]

Move paragraph 9 of the subclause and Note 4.49 to precede paragraph 1, with the following changes:

Change "the variable is"
to "if the variable is not an unallocated allocatable variable, it is",
and append new sentence to 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.".

Interp **F08/0013**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 1^{st} paragraph, [76:10]

After "it is finalized"
insert "unless it is the variable in an intrinsic assignment (7.2.1.3) or a component thereof".

Interp **F08/0011**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 5^{th} and 7^{th} paragraphs, [76:17-18,21-22]

Delete these paragraphs.

Interp **F03/0085 and F08/0034**, Status: *Corrigendum 1.*

Ref: 4.5.6.3, 8^{th} paragraph, [76:23-24]

Replace paragraph 8 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.

Interp **F03/0085, F08/0011, F08/0013, F08/0034**, Status: *Corrigendum 1.*

Ref: Entire subclause 4.5.6.3

Edit subclause 4.5.6.3 as shown below. Note that the old paragraph 9 (with its NOTE) has been moved to become paragraph 1, thus renumbering paragraphs 1-8 to become 2-9.

4.5.6.3 When finalization occurs

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 4.1

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 statement (7.2.1.3) or a component thereof.

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 an executable construct references a structure constructor or array constructor, the entity created by the constructor is finalized after execution of the innermost executable construct containing the reference.~~

7 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.

8 ~~If a specification expression in a scoping unit references a structure constructor or array constructor, the entity created by the constructor is finalized before execution of the executable constructs in the scoping unit.~~

9 When a procedure is invoked, an object that becomes argument associated with a nonpointer, nonallocatable ~~object that is an actual argument corresponding to an~~ 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.

10 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.

Interp **F08/0052**, Status: *Corrigendum 1.*

Ref: 4.5.7.3, 1^{st} paragraph, [78:4]

Change “as a type-bound”
to “as an accessible type-bound”,
making the whole paragraph read

> If a specific type-bound procedure specified in a type definition has the same binding name as an accessible type-bound procedure from the parent type then the binding specified in the type definition overrides the one from the parent type.

Interp **F03/0123 and F08/0015**, Status: *Corrigendum 1.*

Ref: 5.5, 4th paragraph, [109:21-23]

Delete “The mapping may ... scoping unit.”,
change “in the outermost inclusive scope in which it appears”
to “; 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”,
making the whole paragraph read

Any data entity that is not explicitly declared by a type declaration, is not an intrinsic function, is not a component, and is not accessed by use or host association is declared implicitly to be of the type (and type parameters) mapped from the first letter of its name, provided the mapping is not null. The mapping for the first letter of the data entity shall either have been established by a prior IMPLICIT statement or be the default mapping for the letter. ~~The mapping may be to a derived type that is inaccessible in the local scope if the derived type is accessible in the host scoping unit.~~ The data entity is treated as if it were declared in an explicit type declaration ~~in~~; 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. An explicit type specification in a FUNCTION statement overrides an IMPLICIT statement for the name of the result variable of that function subprogram.

Interp **F08/0002**, Status: *Corrigendum 1.*

Ref: 5.6, 5th paragraph, [111:19-20]

Change “type parameters, and shape”
to “kind type parameters, and rank”,
making the whole paragraph read

A namelist group object shall either be accessed by use or host association or shall have its type, kind type parameters, and ~~shape~~rank specified by previous specification statements or the procedure heading in the same scoping unit or by the implicit typing rules in effect for the scoping unit. If a namelist group object is typed by the implicit typing rules, its appearance in any subsequent type declaration statement shall confirm the implied type and type parameters.

Interp **F08/0014 and F08/0016**, Status: *Corrigendum 1.*

Ref: 6.5.3.3.2, 2nd paragraph, [124:4-7]

Replace the bullet list with "finalized by a nonelemental final subroutine.",
making the whole paragraph read:

> An array section with a vector subscript shall not be finalized by a nonelemental final subroutine.
>
> - ~~argument associated with a dummy array that is defined or redefined,~~
> - ~~the *data-target* in a pointer assignment statement, or~~
> - ~~an internal file.~~

NOTE: Interp F08/0014 also has an edit on page 295.

Interp **F08/0039**, Status: *Corrigendum 1.*

Ref: 6.5.4.4.2, 3rd paragraph, [124:9]

Edit the paragraph as follows:

> If a vector subscript has two or more elements with the same value, an array section with that vector subscript ~~shall not appear in a variable definition context (16.6.7)~~ is not definable and shall not be defined or become undefined.

Interp **F08/0010**, Status: *Corrigendum 1.*

Ref: 6.7.3.2, 1st paragraph, [130:23]

Append new sentence to the end of the 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.

NOTE: This interp also has an edit on page 131.

Interp **F08/0010**, Status: *Corrigendum 1.*

Ref: 6.7.3.3, 1st paragraph, [131:27]

Append new sentence to the end of the 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.

NOTE: This interp also has an edit on page 130.

Interp **F08/0050**, Status: *Corrigendum 1.*

Ref: 7.1.11, 9th paragraph, [151:13-15]

Edit the paragraph as follows:

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

NOTE: This interp also has an edit on page 152.

Interp **F08/0050**, Status: *Corrigendum 1.*

Ref: 7.1.12, 3rd paragraph, [152:26-28]

Edit the paragraph as follows:

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

NOTE: This interp also has an edit on page 151.

Interp **F08/0028**, Status: *Corrigendum 1.*

Ref: 8.1.6.6.4, 1st paragraph, [177:28-29]

In the 4th bullet item,
change "Control is transferred from"
to "A branch occurs",
making that bullet item:

- ~~Control is transferred from~~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.

Interp **F08/0023**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1st paragraph, [178:8-9]

Edit the 2nd bullet item as follows:

- A pointer that is ~~referenced~~used in an iteration other than as the pointer in pointer assignment, allocation, or nullification, either shall be previously ~~pointer associated during~~pointer-assigned, allocated, or nullified in that iteration~~,~~ or shall not have its pointer association changed during any iteration. A pointer that has its pointer association changed in more than one iteration has an association status of undefined when the construct terminates.

Interp **F08/0025**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1st paragraph, [178:13-14]

Edit the 3rd bullet item (replacing the 2nd sentence) as follows:

- An allocatable object that is allocated in more than one iteration shall be subsequently deallocated during the same iteration in which it was allocated. ~~An object that is allocated or deallocated in only one iteration shall not be deallocated, allocated, referenced, defined, or become undefined in a different iteration.~~ 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.

Interp **F08/0022**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1st paragraph, [178:15-16]

Edit the 4th bullet item as follows:

- ~~An input/output statement shall not write data to a file record or position in one iteration and read from the same record or position in a different iteration.~~ 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.

Interp **F08/0022**, Status: *Corrigendum 1.*

Ref: 8.1.6.7, 1st paragraph, [178:17-18+]

Delete the 5th bullet item ("Records written ... order.") and make a new paragraph after the list, as follows:

~~• Records written by output statements in the range of the loop to a sequential access file appear in the file in an indeterminate order.~~

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.

Interp **F03/0048**, Status: *Corrigendum 1.*

Ref: 9.6.4.8, 25th and 26th paragraphs, [227:15,17-18]

NOTE: This interp also has an edit on page 487.

In the 25th paragraph, delete “record positioning”.

In the 26th paragraph,
change “A record positioning edit descriptor, such as TL and TR,”
to “The edit descriptors T and TL”, and
change “record position” to “file position” twice,
making those two paragraphs read:

> Because a child data transfer statement does not position the file prior to data transfer, the child data transfer statement starts transferring data from where the file was positioned by the parent data transfer statement’s most recently processed effective item or ~~record positioning~~ edit descriptor. This is not necessarily at the beginning of a record.
>
> ~~A record positioning edit descriptor, such as~~The edit descriptors TL and TR~~,~~ used on `unit` by a child data transfer statement shall not cause the ~~record~~file position to be positioned before the ~~record~~file position at the time the defined input/output procedure was invoked.

Interp **F08/0030**, Status: *Corrigendum 1.*

Ref: 10.3.1, [246:15+]

After constraint C1002, add a new constraint:

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

NOTE: This interp also has an edit on page 249.

Interp **F08/0030**, Status: *Corrigendum 1.*

Ref: 10.4, 7th and 8th paragraphs, [249:11+,19-20]

Between the 7th and 8th paragraphs, insert a new paragraph 7a:

> 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).

In the last sentence of the 8th paragraph, change "If ..., the" to "The", making the whole paragraph read:

> If format control encounters the rightmost parenthesis of a complete format specification and another effective item is not specified, format control terminates. However, if another effective item is specified, format control then reverts to the beginning of the format item terminated by the last preceding right parenthesis that is not part of a DT edit descriptor. If there is no such preceding right parenthesis, format control reverts to the first left parenthesis of the format specification. If any reversion occurs, the reused portion of the format specification shall contain at least one data edit descriptor. If format control reverts to a parenthesis that is preceded by a repeat specification, the repeat specification is reused. Reversion of format control, of itself, has no effect on the changeable modes (9.5.2). ~~If format control reverts to a parenthesis that is not the beginning of an *unlimited-format-item*, the~~The file is positioned in a manner identical to the way it is positioned when a slash edit descriptor is processed (10.8.2).

NOTE: This interp also has an edit on page 246.

Interp **F08/0001**, Status: *Corrigendum 1.*

Ref: 12.4.3.4.5, 3rd paragraph, 3rd bullet item, [286:4]

After "the other has the POINTER attribute"
insert "and not the INTENT (IN) attribute",
making the whole paragraph (excluding the constraints that follow it) read:

Two dummy arguments are distinguishable if

- one is a procedure and the other is a data object,
- they are both data objects or known to be functions, and neither is TKR compatible with the other,
- one has the ALLOCATABLE attribute and the other has the POINTER attribute and not the INTENT (IN) attribute, or
- one is a function with nonzero rank and the other is not known to be a function.

Interp **F08/0053**, Status: *Corrigendum 1.*

Ref: 12.4.3.4.5, constraint C1214 and 5th paragraph, [286:12-13,38]

In constraint C1214, change "two ... identifier"
to "if two procedures have the same generic identifier, their `dtv` arguments (9.6.4.8.3)",
making the whole constraint read:

C1214 Within the scope of a *defined-io-generic-spec*, if two procedures ~~with that~~have the same generic identifier, their `dtv` arguments (9.6.4.8.3) shall be distinguishable.

Insert new UTI after the constraint:

Unresolved Technical Issue C1-F08/0053

This means that in any scope where defined i/o is available, two procedures that have the generic identifier e.g. `fred` will need to have `dtv` arguments that are distinguishable, regardless of the rest of their argument lists. This would seem to be a serious problem with backwards compatibility.

In the 5*th* paragraph, change "applies to" to "is consistent with",
making the whole paragraph read:

Within the scope of a generic name that is the same as the generic name of an intrinsic procedure, the intrinsic procedure is not accessible by its generic name if the procedures in the interface and the intrinsic procedure are not all functions or not all subroutines. If a generic invocation ~~applies to~~is consistent with both a specific procedure from an interface and an accessible intrinsic procedure, it is the specific procedure from the interface that is referenced.

Interp **F08/0037**, Status: *Corrigendum 1.*

Ref: 12.4.3.6, BNF rule R1213, [287:15+]

After the production "**or** POINTER"
insert a new production "**or** PROTECTED",
making the whole rule read:

R1213	*proc-attr-spec*	**is**	*access-spec*
		or	*proc-language-binding-spec*
		or	INTENT (*intent-spec*)
		or	OPTIONAL
		or	POINTER
		or	PROTECTED
		or	SAVE

NOTE: This interp also has an edit on page 16.

Interp **F08/0014**, Status: *Corrigendum 1.*

Ref: 12.5.2.4, 18th paragraph, [295:3]

Between “If” and “the actual argument is an array section having a vector subscript”,
insert “the procedure is nonelemental and”,
making the paragraph (excluding the notes and constraints that follow it) read:

> If the procedure is nonelemental 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.

NOTE: This interp also has an edit on page 124.

Interp **F08/0033**, Status: *Corrigendum 1.*

Ref: 12.7, after Note 12.47, [312:23+]

Insert new constraint

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

NOTE: This interp also has edits on pages 24 and 313.

Interp **F08/0033**, Status: *Corrigendum 1.*

Ref: 12.7, after constraint C1284, [313:4+]

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.

NOTE: This interp also has edits on pages 24 and 312.

Interp **F08/0049**, Status: *Corrigendum 1.*

Ref: 12.8.1, constraint C1290, [314:4-5]

In C1290, delete ", and shall not ... expression", making the whole constraint read:

C1290 The result variable of an elemental function shall be scalar~~,~~ and shall not have the POINTER or ALLOCATABLE attribute~~, and shall not have a type parameter that is defined by an expression that is not a constant expression~~.

NOTE: The editor fixed the grammar in the preceding by changing a comma to "and".

Interp **F08/0024 and F08/0049**, Status: *Corrigendum 1.*

Ref: 12.8.1, after constraint C1290, [314:5+]

Insert new constraints as follows.

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.

Interp **F08/0018**, Status: *Corrigendum 1.*

Ref: 12.8.1, 12.8.2, 12.8.3

Insert new paragraph at the end of 12.8.1 [314:5+]:

> 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.

In 12.8.2 [314:9-10], delete the sentence beginning "For those elemental,
making the whole paragraph read:

> If a generic name or a specific name is used to reference an elemental function, the shape of the result is the same as the shape of the actual argument with the greatest rank. If there are no actual arguments or the actual arguments are all scalar, the result is scalar. ~~For those elemental functions that have more than one argument, all actual arguments shall be conformable.~~ In the array case, the values of the elements, if any, of the result are the same as would have been obtained if the scalar function had been applied separately, in array element order, to corresponding elements of each array actual argument.

In 12.8.3 [314:14-17] delete the sentence beginning "In a reference",
making the whole paragraph read:

> An elemental subroutine has only scalar dummy arguments, but may have array actual arguments. ~~In a reference to an elemental subroutine, either all actual arguments shall be scalar, or all actual arguments corresponding to INTENT (OUT) and INTENT (INOUT) dummy arguments shall be arrays of the same shape and the remaining actual arguments shall be conformable with them.~~ In the case that the actual arguments corresponding to INTENT (OUT) and INTENT (INOUT) dummy arguments are arrays, the values of the elements, if any, of the results are the same as would be obtained if the subroutine had been applied separately, in array element order, to corresponding elements of each array actual argument.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.2.4, 1st sentence, 2nd sentence, [316:24-25]

Change “an optional” to “a”,
and change “, if present, specifies” to “specify”,
making the whole paragraph read:

> Some array intrinsic functions are “reduction” functions; that is, they reduce the rank of an array by collapsing one dimension (or all dimensions, usually producing a scalar result). These functions have ~~an optional~~ a DIM argument that~~, if present,~~ can specifies the dimension to be reduced. The DIM argument of a reduction function is not permitted to be an optional dummy argument.

NOTE: This interp also has edits on pages 319, 322, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [319]

In the table lines for ALL and ANY,
change "(MASK [, DIM])" to "(MASK) or (MASK, DIM)",
making those lines of the table read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
ALL	(MASK) or (MASK, DIM)	T	Reduce logical array by AND operation.
...			
ANY	(MASK) or (MASK, DIM)	T	Reduce logical array with OR operation.
...			

NOTE: This interp also has edits on pages 316, 322, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [322]

In the table line for NORM2,
change “(X [, DIM])” to “(X) or (X, DIM)”,
and in the table line for PARITY,
change “(MASK [, DIM])” to “(MASK) or (MASK, DIM)”,
making those lines of the table read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
NORM2	(X) or (X, DIM)	T	L_2 norm of an array.
...			
PARITY	(MASK) or (MASK, DIM)	T	Reduce array with .NEQV. operation.
...			

NOTE: This interp also has edits on pages 316, 319, 323, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.5, Table 13.1, [323]

In the second table line for THIS_IMAGE,
change "(COARRAY [, DIM])" to "(COARRAY) or (COARRAY, DIM)",
making the line of the table for that function read:

Table 13.1: **Standard generic intrinsic procedure summary**

Procedure	Arguments	Class	Description
...			
THIS_IMAGE	( )	T	Index of the invoking image.
THIS_IMAGE	(COARRAY) or (COARRAY, DIM)	T	Cosubscript(s) for this image.
...			

NOTE: This interp also has edits on pages 316, 319, 322, 328, 329, 338, 360, 374, 377, 392, 394 and 395.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.10, [328:2,7,10]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 329, 338, 360, 374, 377, 392, 394 and 395.

Change the subclause heading to "ALL (MASK, DIM) or ALL (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.10 ALL (MASK, DIM) or ALL (MASK)

1 **Description.** Reduce logical array by AND operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK	shall be a logical array.
DIM	shall be an integer scalar with value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear or $n = 1$; otherwise, the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of ALL (MASK) has the value true if all elements of MASK are true or if MASK has size zero, and the result has value false if any element of MASK is false.

Case (ii): If MASK has rank one, ALL (MASK, DIM) is equal to ALL (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of ALL (MASK, DIM) is equal to ALL (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of ALL ([.TRUE., .FALSE., .TRUE.]) is false.

Case (ii): If B is the array $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ and C is the array $\begin{bmatrix} 0 & 3 & 5 \\ 7 & 4 & 8 \end{bmatrix}$ then ALL (B /= C, DIM = 1) is [true, false, false] and ALL (B /= C, DIM = 2) is [false, false].

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.13, [329:6,11,14]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 338, 360, 374, 377, 392, 394 and 395.

Change the subclause heading to "ANY (MASK, DIM) or ANY (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.13 ANY (MASK, DIM) or ANY (MASK)

1 **Description.** Reduce logical array with OR operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK	shall a logical array.
DIM	shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear or $n = 1$; otherwise, the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of ANY (MASK) has the value true if any element of MASK is true and has the value false if no elements are true or if MASK has size zero.

Case (ii): If MASK has rank one, ANY (MASK, DIM) is equal to ANY (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of ANY (MASK, DIM) is equal to ANY (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of ANY ([.TRUE., .FALSE., .TRUE.]) is true.

Case (ii): If B is the array $\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$ and C is the array $\begin{bmatrix} 0 & 3 & 5 \\ 7 & 4 & 8 \end{bmatrix}$ then ANY (B /= C, DIM = 1) is [true, false, true] and ANY (B /= C, DIM = 2) is [true, true].

Interp **F08/0027**, Status: *Corrigendum 1.*

Ref: 13.7.21, 4th paragraph, [332:25]

Change "CALL ATOMIC_REF (I [3], VAL)"
to "CALL ATOMIC_REF (VAL, I [3])",
making the whole paragraph read:

Example. CALL ATOMIC_REF (VAL, I [3]) causes VAL to become defined with the value of I on image 3.

Interp **F08/0019**, Status: *Corrigendum 1.*

Ref: 13.7.24, 3rd paragraph, [333:12-14]

NOTE: This interp also has an edit on page 334.

For the arguments N1 and N2,
change “ot type integer and nonnegative” to “an integer scalar with a nonnegative value”,
and for argument X,
after “real” insert “; if the function is transformational, X shall be scalar”,
making the whole paragraph read:

Arguments.

N	shall be of type integer and nonnegative.
N1	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
N2	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
X	shall be of type real; if the function is transformational, X shall be scalar.

Interp **F08/0019**, Status: *Corrigendum 1.*

Ref: 13.7.27, 3rd paragraph, [334:12-14]

NOTE: This interp also has an edit on page 333.

For the arguments N1 and N2,
change "ot type integer and nonnegative" to "an integer scalar with a nonnegative value",
and for argument X,
after "real" insert "; if the function is transformational, X shall be scalar",
making the whole paragraph read:

Arguments.

N	shall be of type integer and nonnegative.
N1	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
N2	shall be ~~of type~~ an integer scalar with a ~~and~~ nonnegative value.
X	shall be of type real; if the function is transformational, X shall be scalar.

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.41, 3rd paragraph, DIM argument, [338:31]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 360, 374, 377, 392, 394 and 395.

After "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
making the whole paragraph read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F08/0020**, Status: *Corrigendum 1.*

Ref: 13.7.61, 3^{rd} paragraph, VALUE argument, [347:31-32]

Change “relational ... 7.1.5.5.2)” to “the operator == or the operator .EQV.”,
making the whole paragraph read:

VALUE	shall be scalar and in type conformance with ARRAY, as specified in Table 7.2 for ~~relational intrinsic operations (7.1.5.5.2)~~ the operator == or the operator .EQV..

Interp **F08/0003**, Status: *Corrigendum 1.*

Ref: 13.7.90 and 13.7.91, 3rd paragraph of each, DIM argument, [360:4,25]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 374, 377, 392, 394 and 395.

In both subclauses,
after "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
this makes the paragraph in 13.7.90 read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of ARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

and makes the paragraph in 13.7.91 read:

DIM (optional) shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of COARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.123, heading and 3rd and 4th paragraphs, [374:24,29,31]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 377, 392, 394 and 395.

Change "NORM2 (X [, DIM])" to "NORM2 (X, DIM) or NORM2 (X)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.123 NORM2 (X, DIM) or NORM2 (X)

1 **Description.** L_2 norm of an array.

2 **Class.** Transformational function.

3 **Arguments.**

X	shall be a real array.
DIM	shall be an integer scalar with a value in the range $1 \le \text{DIM} \le n$, where n is the rank of X. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of the same type and type parameters as X. It is scalar if DIM does not appear; otherwise the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM-1}}, d_{\text{DIM+1}}, \ldots, d_n]$, where n is the rank of X and $[d_1, d_2, \ldots, d_n]$ is the shape of X.

5 **Result Value.**

Case (i):	The result of NORM2 (X) has a value equal to a processor-dependent approximation to the generalized L_2 norm of X, which is the square root of the sum of the squares of the elements of X.
Case (ii):	The result of NORM2 (X, DIM=DIM) has a value equal to that of NORM2 (X) if X has rank one. Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots\ s_n)$ of the result is equal to NORM2 (X$(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots\ s_n)$).

6 It is recommended that the processor compute the result without undue overflow or underflow.

7 **Example.** The value of NORM2 ([3.0, 4.0]) is 5.0 (approximately). If X has the value $\begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix}$ then the value of NORM2 (X, DIM=1) is [3.162, 4.472] (approximately) and the value of NORM2 (X, DIM=2) is [2.236, 5.0] (approximately).

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.128, heading and 3^{rd} and 4^{th} paragraphs, [377:20,25,28]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 392, 394 and 395.

Change "PARITY (MASK [, DIM])" to "PARITY (MASK, DIM) or PARITY (MASK)",
in paragraph 3, DIM argument, delete "(optional)",
in paragraph 4, change "is absent" to "does not appear",
making the whole subclause read:

13.7.128 PARITY (MASK, DIM) or PARITY (MASK)

1 **Description.** Reduce array with .NEQV. operation.

2 **Class.** Transformational function.

3 **Arguments.**

MASK	shall be a logical array.
DIM	shall be an integer scalar with a value in the range $1 \leq \text{DIM} \leq n$, where n is the rank of MASK. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** The result is of type logical with the same kind type parameter as MASK. It is scalar if DIM does not appear; otherwise, the result has rank $n - 1$ and shape $[d_1, d_2, \ldots, d_{\text{DIM}-1}, d_{\text{DIM}+1}, \ldots, d_n]$ where $[d_1, d_2, \ldots, d_n]$ is the shape of MASK.

5 **Result Value.**

Case (i): The result of PARITY (MASK) has the value true if an odd number of the elements of MASK are true, and false otherwise.

Case (ii): If MASK has rank one, PARITY (MASK, DIM) is equal to PARITY (MASK). Otherwise, the value of element $(s_1, s_2, \ldots, s_{\text{DIM}-1}, s_{\text{DIM}+1}, \ldots, s_n)$ of PARITY (MASK, DIM) is equal to PARITY (MASK $(s_1, s_2, \ldots, s_{\text{DIM}-1}, :, s_{\text{DIM}+1}, \ldots, s_n)$).

6 **Examples.**

Case (i): The value of PARITY ([T, T, T, F]) is true if T has the value true and F has the value false.

Case (ii): If B is the array $\begin{bmatrix} T & T & F \\ T & T & T \end{bmatrix}$, where T has the value true and F has the value false, then PARITY (B, DIM=1) has the value [F, F, T] and PARITY (B, DIM=2) has the value [F, T].

Interp **F08/0021**, Status: *Corrigendum 1.*

Ref: 13.7.160, 3rd paragraph, [390:6]

Around “has any deferred type parameters” insert “is unlimited polymorphic or” and a comma, making the whole paragraph read:

Arguments.

A	shall be a scalar or array of any type. If it is polymorphic it shall not be an undefined pointer. If it is unlimited polymorphic or has any deferred type parameters, it shall not be an unallocated allocatable variable or a disassociated or undefined pointer.
KIND (optional)	shall be a scalar integer constant expression.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.165, heading and 3rd paragraph, [392:6,11]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 394 and 395.

Change "THIS_IMAGE (COARRAY [, DIM])"
to "THIS_IMAGE (COARRAY) or THIS_IMAGE(COARRAY, DIM)",
in paragraph 3, DIM argument, delete "(optional)",
the result is shown below; **the editor has made additional changes:**
— deleted the mistaken paragraph 7 marker, — added commas to the heading.

13.7.165 THIS_IMAGE ( ), THIS_IMAGE (COARRAY), or THIS_IMAGE (COARRAY, DIM)

1 **Description.** Cosubscript(s) for this image.

2 **Class.** Transformational function.

3 **Arguments.**

COARRAY	shall be a coarray of any type. If it is allocatable it shall be allocated.
DIM	shall be a default integer scalar. Its value shall be in the range $1 \leq \text{DIM} \leq n$, where n is the corank of COARRAY. The corresponding actual argument shall not be an optional dummy argument.

4 **Result Characteristics.** Default integer. It is scalar if COARRAY does not appear or DIM is present; otherwise, the result has rank one and its size is equal to the corank of COARRAY.

5 **Result Value.**

Case (i): The result of THIS_IMAGE ( ) is a scalar with a value equal to the index of the invoking image.

Case (ii): The result of THIS_IMAGE (COARRAY) is the sequence of cosubscript values for COARRAY that would specify the invoking image.

Case (iii): The result of THIS_IMAGE (COARRAY, DIM) is the value of cosubscript DIM in the sequence of cosubscript values for COARRAY that would specify the invoking image.

6 **Examples.** If A is declared by the statement

```
REAL A (10, 20) [10, 0:9, 0:*]
```

then on image 5, THIS_IMAGE ( ) has the value 5 and THIS_IMAGE (A) has the value [5, 0, 0]. For the same coarray on image 213, THIS_IMAGE (A) has the value [3, 1, 2].

The following code uses image 1 to read data. The other images then copy the data.

```
IF (THIS_IMAGE()==1) READ (*,*) P
SYNC ALL
P = P[1]
```

NOTE 13.2

For an example of a module that implements a function similar to the intrinsic function THIS_IMAGE, see subclause C.10.1.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.171, 3rd paragraph, DIM argument, [394:27]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 392 and 395.

After “dummy argument” insert “, a disassociated pointer, or an unallocated allocatable”, making the whole paragraph read:

DIM ~~(optional)~~ shall be an integer scalar with a value in the range $1 \leq \mathrm{DIM} \leq n$, where n is the rank of ARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F03/0003**, Status: *Corrigendum 1.*

Ref: 13.7.172, 3rd paragraph, DIM argument, [395:11]

NOTE: This interp also has edits on pages 316, 319, 322, 323, 328, 329, 338, 360, 374, 377, 392 and 394.

After "dummy argument" insert ", a disassociated pointer, or an unallocated allocatable",
making the whole paragraph read:

DIM ~~(optional)~~ shall be an integer scalar with a value in the range $1 \leq$ DIM $\leq n$, where n is the corank of COARRAY. The corresponding actual argument shall not be an optional dummy argument, a disassociated pointer, or an unallocated allocatable.

Interp **F08/0009**, Status: *Corrigendum 1.*

Ref: 14.9, 1st paragraph, [406:15+]

Add a new item after the second item of the bulleted list,
making the whole paragraph read:

The inquiry function IEEE_SUPPORT_DATATYPE can be used to inquire whether IEEE arithmetic is supported for a particular kind of real. Complete conformance with IEC 60559:1989 is not required, but

- the normal numbers shall be exactly those of an IEC 60559:1989 floating-point format,
- for at least one rounding mode, the intrinsic operations of addition, subtraction and multiplication shall conform whenever the operands and result specified by IEC 60559:1989 are normal numbers,
- the IEEE function abs shall be provided by the intrinsic function ABS,
- the IEEE operation rem shall be provided by the function IEEE_REM, and
- the IEEE functions copysign, scalb, logb, nextafter, and unordered shall be provided by the functions IEEE_COPY_SIGN, IEEE_SCALB, IEEE_LOGB, IEEE_NEXT_AFTER, and IEEE_UNORDERED, respectively,

for that kind of real.

Interp **F03/0124**, Status: *Corrigendum 1.*

Ref: 16.6.6, 1st paragraph, [455:4-10]

Replace the entire item (1) 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.

Interp **F03/0048**, Status: *Corrigendum 1.*

Ref: C.6.2, 1^{st} paragraph, [487:28]

NOTE: This interp also has an edit on page 227.

Delete “record positioning”, making the whole paragraph read:

Data transfer statements affect the positioning of an external file. In FORTRAN 77, if no error or end-of-file condition exists, the file is positioned after the record just read or written and that record becomes the preceding record. This part of ISO/IEC 1539 contains the ~~record positioning~~ ADVANCE= specifier in a data transfer statement that provides the capability of maintaining a position within the current record from one formatted data transfer statement to the next data transfer statement. The value NO provides this capability. The value YES positions the file after the record just read or written. The default is YES.

Interp **F08/0036**, Status: *Corrigendum 1.*

Ref: C.13.3.6, 3^{rd} paragraph, [527:18]

Insert a superscript "2" to square the absolute value of X_i,
making the whole paragraph read:

The L^2-norm of vector X, defined as $\sqrt{\sum_{i=1}^{n} |X_i|^2}$, can be formed using the Fortran expression NORM2 (X).