J3/16-247

Fortran 2015 Changes

SEPTEMBER 27, 2016

Revision History

Rev.	Dates	Reasons for change
0.1	Sep 27, 2016	Original Draft, Steve Lionel

Contents

1 Introduction

This document attempts to collect information about changes from Fortran 2008 to Fortran 2015, including references to the standard and J3 documents. It is based on the list of changes from the introduction to the draft standard 16-007r2.

Standing document 010 (currently 15-010r1) has much of this information in abbreviated form, but misses some additional items and lacks explanatory detail.

2 Reference

In this specification, references to the standard if not otherwise specified are to Fortran 2015 and draft document 16-007r2. Papers with N prefixes are WG5 papers.

3 Data Declaration

3.1 Constant properties of an object declared in its entity-decl can be used in its initialization.

See 15-194

In F2008, a constant expression that depended on a constant property of another entity required that the entity appear in a “prior specification”.

F2008 7.1.12p2

> If a constant expression includes a specification inquiry that depends on a type parameter or an array bound of an entity specified in the same specification-part , the type parameter or array bound shall be specified in a prior specification of the specification-part . The prior specification may be to the left of the specification inquiry in the same statement, but shall not be within the same entity-decl .

F2015 allows the use of constant properties of the same entity-decl in an initialization:

F2015 10.1.12p2

> …The prior specification may be to the left of the specification inquiry in the same statement, but shall not be within the same *entity-decl* unless the specification inquiry appears within an *initialization*.

4 Data Usage and Computation

4.1 The declared type of the value supplied for a polymorphic allocatable component in a structure constructor is no longer required to be the same as the declared type of the component.

See 16-233r1

F2008 4.5.10p6

> If a component of a derived type is allocatable, the corresponding constructor expression shall either be a reference to the intrinsic function NULL with no arguments, an allocatable entity of the same rank, or shall evaluate to an entity of the same rank. If the expression is a reference to the intrinsic function NULL, the corresponding component of the constructor has a status of unallocated. If the expression is an allocatable entity, the corresponding component of the constructor has the same allocation status as that allocatable entity and, if it is allocated, the same dynamic type, bounds, and value; if a length parameter of the component is deferred, its value is the same as the corresponding parameter of the expression. Otherwise the corresponding component of the constructor has an allocation status of allocated and has the same bounds and value as the expression.

F2015 7.5.10p6-8

> If a component of a derived type is allocatable, the corresponding constructor expression shall be a reference to the intrinsic function NULL with no arguments, an allocatable entity of the same rank, or shall evaluate to an entity of the same rank. If the expression is a reference to the intrinsic function NULL, the corresponding component of the constructor has a status of unallocated.
>
> If the component is allocatable and the expression is an allocatable entity, the corresponding component of the constructor has the same allocation status as that allocatable entity. If it is allocated, it has the same bounds; if a length parameter of the component is deferred, its value is the same as the corresponding parameter of the expression. If the component is polymorphic, it has the same dynamic type and value; otherwise, it has the value converted, if necessary, to the declared type of the component.
>
> If the component is allocatable and the expression is not an allocatable entity, the component has an allocation status of allocated and the same bounds as the expression; if a length parameter of the component is deferred, its value is the same as the corresponding parameter of the expression. If the component is polymorphic, it has the same dynamic type and value; otherwise, it has the value converted, if necessary, to the declared type of the component.

4.2 Labeled DO loops have been redundant since Fortran 90 and are now specified to be obsolescent.

13-320r2, B.3.10

4.3 The arithmetic IF statement has been deleted.

13-316r2, B.2

4.4 The EQUIVALENCE and COMMON statements and the block data program unit have beenredundant since Fortran 90 and are now specified to be obsolescent.

13-322r1, B.3.11

4.5 The nonblock DO construct has been deleted.

13-317r2, B.2
The nonblock forms of the do loop are deleted. This includes the shared termination forms of the do loop.

4.6 FORALL is now specified to be obsolescent.

14-129r1, B.3.13

Both the construct and statement are obsolescent. I'm not sure if we have references in documentation or error messages, but the standard terms starting with "forall-" have been replaced. For example, "forall-header" is now "concurrent-header", but see the paper for details.

4.7 The type and kind of an implied DO variable in an array constructor or DATA statement can be specified within the constructor or statement.

14-101r1, 7.8 and 8.6.7

4.8 The locality of a variable used in a DO CONCURRENT construct can be explicitly specified.

15-150r2, 11.1.7.2

4.9 The SELECT RANK construct provides structured access to the elements of an assumed-rank array.

15-142r2, 11.10.10

5 Input/Output

5.1 The SIZE= specifier can be used with advancing input.

13-208r1, 134-218r1, 13-244r1

F2008 9.6.2.1
C923 (R913) If an EOR= or SIZE= specifier appears, an ADVANCE= specifier also shall appear.

2 If an EOR= or SIZE= specifier appears, an ADVANCE= specifier with the value NO shall also appear.

F2015 12.6.2.1
C1222 (R1213) If an EOR= specifier appears, an ADVANCE= specifier also shall appear.

2 If an EOR= specifier appears, an ADVANCE= specifier with the value NO shall also appear.

Give standards warning for F08 and earlier if SIZE= and not also ADVANCE=. Make sure RTL can handle this combination. It counts only characters transferred in this specific input statement.

5.2 It is no longer prohibited to open a file on more than one unit.

16-120r3

Changes the behavior from non-standard to implementation-defined – just terminology.

5.3 The value assigned by the RECL= specifier in an INQUIRE statement has been standardized.

13-330r1

F2008 9.10.2.26
If there is no connection, or if the connection is for stream access, the scalar-int-variable becomes undefined.

F2015 12.10.2.26
If there is no connection, the scalar-int-variable is assigned the value −1, and if the connection is for stream access the scalar-int-variable is assigned the value −2.

5.4 The values assigned by the POS= and SIZE= specifiers in an INQUIRE statement for a unit that has pending asynchronous operations have been standardized.

16-162r1, 12.10.2.22, 12.10.2.30

For both POS= and SIZE= in INQUIRE, the following text was added:

"If there are pending data transfer operations for the specified unit,
the value assigned is computed as if all the pending data transfers
had already completed."

5.5 The G0.d edit descriptor can be used for list items of type Integer, Logical, and Character.

13-309r2

The G0.d edit descriptor shall be permitted to be used for list items of type Integer, Logical, or Character. In all these cases, the .d value shall be ignored.

5.6 The D, E, EN, and ES edit descriptors can have a field width of zero, analogous to the F edit descriptor.

13-351r2

New edit descriptor forms D0.d, E0.d, E0.dEe, EN0.d, EN0.dEe, ES0.d, ES0.dEe.

5.7 The exponent width e in a data edit descriptor can be zero, analogous to a field width of zero.

14-176r2
Permit the "e" in the "Ee" part of the E, EN, ES, and G edit descriptors to have the value zero. E0 requests the exponent width to be minimal.

5.8 Floating-point formatted input accepts hexadecimal-significand numbers that conform to ISO/IEC/IEEE 60559:2011.

14-198r1

Accept for numeric input values of the form 0x……

5.9 The EX edit descriptor provides hexadecimal-significand formatted output conforming to ISO/IEC/IEEE 60559:2011.

14-198r1

New EX format edit descriptor

5.10 An error condition occurs if unacceptable characters are presented for logical or numeric editing during execution of a formatted input statement.

14-270, 14-175r3

If during formatted input a character input for numeric or logical editing is not acceptable to the processor, an error condition occurs.

The standard didn’t previously specify that this was an error condition.

6 Execution Control

6.1 The stop code in a STOP or ERROR STOP statement can be nonconstant.

15-192r2

In F2008 the stop code could be only a constant. In F2015 it can be any scalar integer or character expression.

6.2 Output of the stop code and exception summary from the STOP and ERROR STOP statements can be controlled.

15-192r2

Adds optional QUIET= specifier to STOP and ERROR STOP to suppress output of things such as “FORTRAN STOP”.

7 Intrinsic Procedures and Modules

7.1 In references to the intrinsic functions ALL, ANY, FINDLOC, IALL, IANY, IPARITY, MAXLOC, MAXVAL, MINLOC, MINVAL, NORM2, PARITY, PRODUCT, SUM, and THIS_IMAGE, the actual argument for DIM can be a present optional dummy argument.

13-307r2

7.2 In a reference to the intrinsic function CMPLX with an actual argument of type complex, no keyword is needed for a KIND argument.

14-204, 16.9.45

Split the template for the intrinsic function CMPLX into two templates: one where the argument X is of type complex, and a second where the argument X is of type real or integer or is a <boz-literal-constant>. The template where the argument X is of type complex omits the dummy argument Y.
As a consequence, the requirement that no actual argument shall correspond to Y if the argument X is of type complex is not needed.

7.3 The new intrinsic function COSHAPE returns the coshape of a coarray.

14-181r3, 16.9.55

7.4 The new intrinsic function OUT_OF_RANGE tests whether a numeric value can be safely converted to a different type or kind.

14-183r2, 16.9.146

Add a new intrinsic function or functions which can check whether a REAL or INTEGER value can be converted to a different type (or kind) without error,where "without error" means no integer or real overflow will occur, or for the case where the value ia a NaN, that the target type can represent NaN.

7.5 The new intrinsic subroutine RANDOM_INIT establishes the initial state of the pseudorandom number generator used by RANDOM_NUMBER.

14-184r4, 16.9.155

7.6 The new intrinsic function REDUCE performs user-specified array reductions.

13-329r2, 16.9.161

7.7 A processor is required to report use of a nonstandard intrinsic procedure, use of a nonstandard intrinsic module, and use of a nonstandard procedure from a standard intrinsic module.

13-310r3

7.8 Integer and logical arguments to intrinsic procedures and intrinsic module procedures that were previously required to be of default kind no longer have that requirement, except for RANDOM_SEED.

14-168r4

7.9 Specific names for intrinsic functions are now deemed obsolescent.

13-319r1

7.10 All standard procedures in the intrinsic module ISO_C_BINDING, other than C_F_POINTER, are now pure.

14-237r2

7.11 The arguments to the intrinsic function SIGN can be of different kind.

15-202, 16.9.176

7.12 Nonpolymorphic pointer arguments to the intrinsic functions EXTENDS_TYPE_OF and SAME_TYPE_AS need not have defined pointer association status.

15-111r1

7.13 The effects of invoking the intrinsic procedures COMMAND_ARGUMENT_COUNT, GET_COMMAND, and

GET_COMMAND_ARGUMENT, on images other than image one, are no longer processor dependent.

15-204r1

7.14 Access to error messages from the intrinsic subroutines GET_COMMAND, GET_COMMAND_ARGUMENT, and GET_ENVIRONMENT_VARIABLE is provided by an optional ERRMSG argument.

15-230r2

8 Program units and procedures:

8.1 The IMPORT statement can appear in a contained subprogram or BLOCK construct, and can restrict access via host association; diagnosis of violation of the IMPORT restrictions is required.

13-304r1, 15.4.3.4

(a) To be able to specify that an entity is host-associated.
(b) To be able to limit host association to a specific list of names; this shall include the empty set.
(c) To be able to prevent inadvertent "shadowing" of host names.
(d) No renaming.

8.2 The GENERIC statement can be used to declare generic interfaces.

14-177r1, 13-209, 15.4.3.3

Allow a GENERIC statement as an alternative to an interface block, with syntax similar to its use within a type definition.

8.3 The ERROR STOP statement can appear in a pure subprogram.

13-331, 11.4

8.4 The number of procedure arguments is used in generic resolution.

13-332

8.5 In a module, the default accessibility of entities accessed from another module can be controlled separately from the default accessibility of entities declared in the using module.

13-327r3

8.6 An IMPLICIT NONE statement can require explicit declaration of the EXTERNAL attribute throughout a scoping unit and its contained scoping units.

13-312r4, 8.7

8.7 A defined operation need not specify INTENT (IN) for a dummy argument with the VALUE attribute.

14-178r1

8.8 A defined assignment need not specify INTENT (IN) for the second dummy argument if it has the VALUE attribute.

14-178r1

8.9 Procedures, including elemental procedures, can be invoked recursively by default; the RECURSIVE keyword is advisory only.

14-179r2

8.10 The NON_RECURSIVE keyword specifies that a procedure is not recursive.

14-179r2

8.11 A dummy argument of a pure function is permitted in a variable definition context, if it has the VALUE attribute.

14-237r2

9 Features previously described by ISO/IEC TS 29113:2012

All in N1942.

9.1 A dummy data object can assume its rank from its effective argument.

9.2 A dummy data object can assume the type from its effective argument, without having the ability to perform type selection.

9.3 An interoperable procedure can have dummy arguments that are assumed-type and/or assumed-rank.

9.4 An interoperable procedure can have dummy data objects that are allocatable, assumed-shape, optional, or pointers.

9.5 The character length of a dummy data object of an interoperable procedure can be assumed.

9.6 The argument to C_LOC can be a noninteroperable array.

9.7 The FPTR argument to C_F_POINTER can be a noninteroperable array pointer.

9.8 The argument to C_FUNLOC can be a noninteroperable procedure. The FPTR argument to C_F_PROCPOINTER can be a noninteroperable procedure pointer.

10 Changes to the intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS, and IEEE_FEATURES for conformance with ISO/IEC/IEEE 60559:2011:

All in 13-356, 14-196r1, 14-198r1

10.1 There is a new, optional, rounding mode IEEE_AWAY. The new type IEEE_MODES_TYPE encapsulates all floating-point modes. Features associated with subnormal numbers can be accessed

with functions and types named . . .SUBNORMAL.. . (the old . . .DENORMAL.. . names remain).

10.2 The standard intrinsic relational operations on IEEE numbers provide the compareSignaling{relation} operations.

10.3 The new function IEEE_FMA performs fused multiply-add operations.

10.4 The function IEEE_INT performs rounded conversions to integer type.

10.5 The new functions IEEE_MAX_NUM, IEEE_MAX_NUM_MAG, IEEE_MIN_NUM, and IEEE_MIN_NUM_MAG calculate maximum and minimum numeric values.

10.6 The new functions IEEE_NEXT_DOWN and IEEE_NEXT_UP return the adjacent machine numbers.

10.7 The new functions IEEE_QUIET_EQ, IEEE_QUIET_GE, IEEE_QUIET_GT, IEEE_QUIET_LE, IEEE_QUIET_LT, and IEEE_QUIET_NE perform quiet comparisons.

10.8 The decimal rounding mode can be inquired and set independently of the binary rounding mode, using the RADIX argument to IEEE_GET_ROUNDING_MODE and IEEE_SET_ROUNDING_MODE.

10.9 The new function IEEE_REAL performs rounded conversions to real type.

10.10 The function IEEE_REM now requires its arguments to have the same radix.

10.11 The function IEEE_RINT now has a ROUND argument to perform specific rounding.

10.12 The new function IEEE_SIGNBIT tests the sign bit of an IEEE number.

11 Features previously described by ISO/IEC TS 18508:2015

All in N2074

11.1 The CRITICAL statement has optional ERRMSG= and STAT= specifiers.

11.2 The intrinsic subroutines ATOMIC_DEFINE and ATOMIC_REF have an optional STAT argument.

11.3 The new intrinsic subroutines ATOMIC_ADD, ATOMIC_AND, ATOMIC_CAS, ATOMIC_FETCH_ADD, ATOMIC_FETCH_AND, ATOMIC_FETCH_OR, ATOMIC_FETCH_XOR, ATOMIC_OR, and ATOMIC_XOR perform atomic operations.

11.4 The new intrinsic functions FAILED_IMAGES and STOPPED_IMAGES return indices of images known to have failed or stopped respectively.

11.5 The new intrinsic function IMAGE_STATUS returns the image execution status of an image. The intrinsic subroutine MOVE_ALLOC has optional ERRMSG and STAT arguments.

11.6 The intrinsic functions IMAGE_INDEX and NUM_IMAGES have additional forms with a TEAM or TEAM_NUMBER argument.

11.7 The intrinsic function THIS_IMAGE has an optional TEAM argument.

11.8 The EVENT POST and EVENT WAIT statements, the intrinsic subroutine EVENT_QUERY, and the type EVENT_TYPE provide an event facility for one-sided segment ordering.

11.9 The CHANGE TEAM construct, derived type TEAM_TYPE, FORM TEAM and SYNC TEAM statements, the intrinsic functions GET_TEAM and TEAM_NUMBER, and the STAT= and TEAM= specifiers on image selectors, provide a team facility for a subset

of the program’s images to act in concert as if it were the set of all images.

11.10 The new intrinsic subroutines CO_BROADCAST, CO_MAX, CO_MIN, CO_REDUCE, and CO_SUM perform collective reduction operations on the images of the current team.

11.11 The concept of failed images, the FAIL IMAGE statement, and the named constant STAT_FAILED_IMAGE provide support for fault-tolerant parallel execution.