Information technology – Programming languages – Fortran

TECHNICAL CORRIGENDUM 2

Technical corrigendum 2 to international Standard ISO/IEC 1539-1:1991 (E) was prepared by Joint Technical Committee ISO/IEC JTC1, Information technology.

Page 21

Clause 3.3

Change the first sentence to: “A Fortran program unit is a sequence of one or more lines, organized as Fortran statements, comments, and INCLUDE lines.”

Page 22

Subclause 3.3.1

In line 1 of the second paragraph, change “... character context.” to “... character context or in a format specification.”.

Page 27

Clause 4.3.1

Delete the last sentence: “In this standard ...”.

Pages 40 and 41

Clause 5.1

Change the thirteenth constraint of page 40 to:

Constraint: The SAVE attribute must not be specified for an object that is in a common block, a dummy argument, a procedure, a function result, an automatic data object, or an object with the PARAMETER attribute.

Insert the following paragraph after the second paragraph following the constraints on page 40:

A function result may be declared to have the POINTER attribute.

Add the following to the end of the second paragraph of page 41: “If the variable is an array, it must have its shape specified either in the type declaration statement or a previous attribute specification statement in the same scoping unit.”

Change the first sentence of the third paragraph of page 41 to “The presence of = *initialization-expr* implies that *object-name* is a saved object, except for an *object-name* in a named common block or an *object-name* with the PARAMETER attribute.

Page 43

Subclause 5.1.1.5

Change the paragraph following item (3) to:

The length specified for a character-valued statement function or statement function dummy argument of type character must be a constant specification expression.

Subclause 5.1.1.7

Add after the last sentence of the first paragraph: “If the data entity is a function result, the derived type can be specified on the FUNCTION statement providing the derived type is defined within the body of the function or is accessible there by use or host association.”

Page 46

Subclause 5.1.2.4.3

Change line 2 of the seventh paragraph from “an array ... intrinsic” to: “an array may be referenced or defined; however, the array may appear as an argument to an intrinsic”.

Delete the penultimate paragraph: “A pointer dummy argument ... argument.”.

Delete the final paragraph: “A function ... attribute.”.

Page 47

Subclause 5.1.2.5

Remove the penultimate paragraph: “The SAVE attribute ... result, or an automatic data object.”.

Page 50

Subclause 5.2.4

Change the first constraint to:

Constraint: An *object-name* must not be the name of an object in a common block, a dummy argument name, a procedure name, a function result name, an automatic data object name, or the name of an object with the PARAMETER attribute.

Pages 52 and 53

Subclause 5.2.9

Add a new constraint after the second constraint:

Constraint: The *scalar-structure-component* must contain at least one *part-ref* that contains a *subscript-list*.

Add a new constraint after the second constraint following R537:

Constraint: If a DATA statement constant value is a named constant or a structure constructor, the named constant or derived type must have been declared previously in the scoping unit or made accessible by use or host association.

Lines 4-5 of page 53, delete the sentence “Each such value ... host association.”.

Line 6 of page 53, add “value” after “following constant”.

Line 11 of page 53, add “value” after “constant”.

Line 17 of page 53, add “value” after “constant”.

Lines 18-19 of page 53, change “value of the constant” to “constant value” (twice).

Page 53

Subclause 5.2.10

Change the paragraph at the bottom of page 53 to:

> The named constant must have its type, type parameters, and shape specified in a prior specification of the *specification-part* or declared implicitly (5.3). If the named constant is typed by the implicit typing rules, its appearance in any subsequent specification of the *specification-part* must confirm this implied type and the values of any implied type parameters.

Page 54

Clause 5.3

In line 3 of the paragraph that starts with “Any data entity that is” after ”not null.” and before the Corrigendum 1 addition, add: “The mapping for the first letter of the data entity must either have been established by a prior IMPLICIT statement or be the default mapping for the letter.”

Page 56

Clause 5.4

Add to the end of the second constraint “or have private components”.

In line 3 of the penultimate paragraph, delete “currently” and in lines 4-5 replace “this implied type” by “the implied type and type parameters”.

Page 57

Subclause 5.5.1

After the final constraint add:

Constraint: An *equivalence-object* must not have the TARGET attribute.

Page 59

Subclause 5.5.2.1

First line, after “For each common block”, insert “in a scoping unit”.

At the end, add the paragraph:

> Only COMMON statements and EQUIVALENCE statements appearing in the scoping unit contribute to common block storage sequences formed in that unit. Variables, in common blocks, made accessible by use association or host association do not contribute.

Subclause 5.5.2.3

At the end of the first paragraph, add the sentence: “Use association or host association may cause these associated objects to be accessible in the same scoping unit.”

Page 60

Subclause 5.5.2.5

Delete the last two sentences: “A common ... association”.

Page 63

Subclause 6.2.1

In the second paragraph, change “the name of a constant expression (5.1.2.1 and 5.2.10)” to “a named constant (5.1.2.1 and 5.2.10)”.

Page 76

Subclause 7.1.4.1

Change the first sentence of the last paragraph to:

> If a pointer appears as one of the following, the associated target object is referenced:
>
> (1) A primary in an intrinsic or defined operation,
>
> (2) As the *expr* of a parenthesized primary, or
>
> (3) As the only primary on the right-hand side of an intrinsic assignment statement.

In the final sentence, insert before the terminating period “, or as the target in a pointer assignment statement”.

Pages 77 and 78

Subclause 7.1.6.1

Change item (6), in the first list, to:

(6) A reference to an intrinsic function which is:

> (a) an array inquiry function (13.10.15) other than ALLOCATED,
>
> (b) the bit inquiry function BIT_SIZE,
>
> (c) the character inquiry function LEN,
>
> (d) the kind inquiry function KIND, or
>
> (e) a numeric inquiry function (13.10.8)

and where each argument of the function is

> (a) a constant expression, or
>
> (b) a variable whose properties inquired about are not:
>
> 1. assumed,

2. defined by an expression that is not a constant expression, or
3. definable by an ALLOCATE or POINTER assignment statement, or

Change item (6), in the second list, to:

(6) A reference to an intrinsic function which is:

(a) an array inquiry function (13.10.15) other than ALLOCATED,
(b) the bit inquiry function BIT_SIZE,
(c) the character inquiry function LEN,
(d) the kind inquiry function KIND, or
(e) a numeric inquiry function (13.10.8)

and where each argument of the function is

(a) an initialization expression, or
(b) a variable whose properties inquired about are not:

1. assumed,
2. defined by an expression that is not an initialization expression, or
3. definable by an ALLOCATE or POINTER assignment statement, or

Page 79

Subclause 7.1.6.2

Change item (9), in the list, to:

(9) A reference to an intrinsic function which is:

(a) an array inquiry function (13.10.15) other than ALLOCATED,
(b) the bit inquiry function BIT_SIZE,
(c) the character inquiry function LEN,
(d) the kind inquiry function KIND, or
(e) a numeric inquiry function (13.10.8)

and where each argument of the function is

(a) a restricted expression, or
(b) a variable whose properties inquired about are not:

1. dependent on the upper bound of the last dimension of an assumed-size array,
2. defined by an expression that is not a restricted expression, or
3. definable by an ALLOCATE or POINTER assignment statement, or

In line 2 of the paragraph after the constraint, delete “currently”.

Page 85

Subclause 7.2.3

On the first line, change “operator” to “operation”.

Page 92

Subclause 7.5.2

In the third constraint, change “, type parameters,” to “, kind type parameters,”.

Add at the beginning of the paragraph following the constraints, “The target must have the same type parameters as the pointer.”.

Page 93

Subclause 7.5.3.2

Following the third paragraph, insert the paragraph:

> If an array constructor appears in an *assignment-stmt*, the array constructor is evaluated without any masked control by the *mask-expr* and then the *assignment-stmt* is evaluated.

Pages 124 and 125

Subclause 9.4.3

Last line of page 124, change “any implied-DO variables” to “if the input/output statement contains any implied-DOs, all of the implied-DO variables in the statement”.

Second-to-last paragraph, change “any implied-DO variables” to “if the input statement contains any implied-DOs, all of the implied-DO variables in the statement”.

Page 128

Subclause 9.4.6

Item 3, delete “or namelist input reaches the end of a record after having processed a name-value subsequence for every item in the *namelist-group-object-list*”.

Pages 139 and 140

Subclause 10.5.1.1

Paragraph 3, change “be in the form of an optionally signed integer constant” to “be a *signed-digit-string* (R401)”.

Paragraph 4, change “in the form of an unsigned integer constant” to “as a *digit-string*” and change “Note that an integer constant” to “Note that a *digit-string*”.

Line 6 of page 140, change “the unsigned integer constant” to “the *digit-string*”.

Page 140

Subclause 10.5.1.2.1

Item (1), change “Explicitly signed integer constant” to “A *sign* followed by a *digit-string*”.

Item (2), change “an optionally signed integer constant” to “a *signed-digit-string*”.

Item (3), change “an optionally signed integer constant” to “a *signed-digit-string*”.

Page 143

Subclause 10.5.2

Second paragraph, after “additional characters in the field” add “, which are ignored”.

Page 144

Subclause 10.5.3

First paragraph, change the last sentence from “All characters ... list.” to “The kind type parameter of all characters transferred and converted under control of one A or G edit descriptor is implied by the kind of the corresponding list item.”.

Page 148

Clause 10.8

Second paragraph, change “constant with no kind type parameter specified.” to “constant. Neither *c* nor *r* may have kind type parameters specified. The constant *c* is interpreted as though it had the same kind type parameter as the corresponding list item.”

Page 149

Subclause 10.8.1

In lines 1 and 2 of page 149, change “character literal constant of the same kind as” to “possibly delimited sequence of zero or more *rep-char*s whose kind type parameter is implied by the kind of”.

Page 151

Subclause 10.8.2

In line 7 of page 151, delete “possibly are preceded by a *kind-param* and an underscore,”.

In line 11 of page 151, delete “possibly are preceded by a *kind-param* and an underscore,”.

Clause 10.9

Third paragraph, change “constant with no kind type parameter specified.” to “constant. Neither *c* nor *r* may have kind type parameters specified. The constant *c* is interpreted as though it had the same kind type parameter as the corresponding list item.”

Clause 10.9.1

Replace item (2) by:

(2) The character & followed immediately by the *namelist-group-name* as specified in the NAMELIST statement,

Page 152

Subclause 10.9.1

Item (5), delete “statement”.

Add at the end of the second paragraph the sentence “The optional qualification, if any, must not contain a vector subscript.”.

Subclause 10.9.1.2

First line, after “ *c* ” add “(10.9)”.

Page 153

Subclause 10.9.1.3

Fifth paragraph, change “character literal constant of the same kind as” to “delimited sequence of zero or more *rep-char*s whose kind type parameter is implied by the kind of”.

Page 155

Subclause 10.9.2.1

In line 12 of page 155, delete “possibly are preceded by a *kind-param* and an underscore,”.

In line 16 of page 155, delete “possibly are preceded by a *kind-param* and an underscore,”.

Page 158

Subclause 11.3.2

In the twelfth line after the constraints, which starts "If two or more", change "the same name" to "the same local name".

Page 161

Subclause 11.3.3.7

In the second sentence, change "C.11.5" to "C.11.4".

Page 164

Subclause 12.1.2.2.1

In the line after the list of items, replace "that has this as its nongeneric name is inaccessible." with "that has this as its nongeneric name is inaccessible by that name by host association.".

Page 167

Subclause 12.3.2.1

Change the second constraint to: "The MODULE PROCEDURE specification is allowed only if the *interface-block* has a *generic-spec* and is contained in a scoping unit where each *procedure-name* is accessible as a module procedure."

Add the following constraint after the present constraints:

Constraint: A *procedure-name* in a *module-procedure-stmt* must not be one which previously had been specified in any *module-procedure-stmt* with the same generic identifier in the same specification part.

Page 172

Subclause 12.4.1

In the penultimate constraint, ahead of "(12.3.2.1, 13.1)", insert "unless it is also a specific name".

Page 173

Subclause 12.4.1.1

Add at the end of the second paragraph of page 173 "If the dummy argument has the TARGET attribute and the actual argument has the TARGET attribute but is not an array section with a vector subscript, the dummy and actual arguments must have the same shape.".

In the last sentence of the third paragraph of page 173, delete "with a dummy argument of the procedure that has the TARGET attribute or".

Add the following new paragraph following the third paragraph of page 173:

> If the dummy argument is not a pointer and the corresponding actual argument is, the actual argument must be currently associated with a target and the dummy argument becomes argument associated with that target.

Replace the fourth paragraph of page 173 with:

> If the dummy argument does not have the TARGET or POINTER attribute, any pointers associated with the actual argument do not become associated with the corresponding dummy argument on invocation of the procedure.
>
> If the dummy argument has the TARGET attribute and the corresponding actual argument has the TARGET attribute but is not an array section with a vector subscript:

(1) Any pointers associated with the actual argument become associated with the corresponding dummy argument on invocation of the procedure.

(2) When execution of the procedure completes, any pointers associated with the dummy argument remain associated with the actual argument.

If the dummy argument has the TARGET attribute and the corresponding actual argument does not have the TARGET attribute or is an array section with a vector subscript, any pointers associated with the dummy argument become undefined when execution of the procedure completes.

Page 184

Subclause 13.5.4

Replace the second sentence “The value of ...” with “If the argument to this function consists of a single primary (7.1.1.1) that is a variable name, the variable need not be defined.”.

Page 185

Subclause 13.5.7

In the paragraph beginning “An inquiry ...”, replace the second sentence “The value of ... ” with “If the argument to this function consists of a single primary (7.1.1.1) that is a variable name, the variable need not be defined, if a pointer it may have undefined or disassociated association status, and if allocatable it need not be allocated.”.

Page 186

Subclause 13.7.2

Change the second sentence to: “If the argument to these functions consists of a single primary (7.1.1.1) that is a variable name, the variable need not be defined, if a pointer it may have undefined or disassociated association status, and if allocatable it need not be allocated.”.

Page 187

Subclause 13.8.5

Replace the second paragraph “The values of ...” with

If an array argument to these functions consists of a single primary (7.1.1.1) that is a variable name, the variable need not be defined.

Page 210

Subclause 13.13.42

Result Value, change “.... bit POS right-adjusted” to “.... bit POS, right-adjusted”.

Page 214

Subclause 13.13.52

Replace the text of case (i) with:

For an array section or for an array expression other than a whole array or array structure component, LBOUND(ARRAY,DIM) has the value 1. For a whole array or array structure component, LBOUND(ARRAY,DIM) has the value:

(a) equal to the lower bound for subscript DIM of ARRAY if dimension DIM of ARRAY does not have extent zero or if ARRAY is an assumed-size array of rank DIM, or

(b) 1 otherwise.

Page 236

Subclause 13.13.104

COUNT_RATE, change “the number” to “a processor-dependent approximation to the number”.

Page 245

Subclause 14.1.2.5

In the second sentence, change “It” to “Outside the type definition, it”.

Subclause 14.1.2.6

Add a new paragraph after the current paragraph:

> A dummy argument name in an intrinsic procedure has a scope as an argument keyword of the scoping unit making reference to it. As an argument keyword, it may appear only in a procedure reference for the procedure of which it is a dummy argument.

Page 248

Subclause 14.6.3.3

Last line of page 248, change “COMMON, EQUIVALENCE, or ENTRY” to “COMMON or EQUIVALENCE”.

Page 250

Subclause 14.7.5

Item (8), change “input/output IOSTAT= specifier” to “IOSTAT= specifier”.

Page 251

Subclause 14.7.6

Item (6), change “statement, some or all of the implied-DO variables may become” to “statement and the statement contains any implied-DOs, all of the implied-DO variables in the statement become”.

Pages 255 to 258

Annex A

construct; change “CASE” to “SELECT CASE”.

entity; change “a *named variable*, an *expression*, a *component* of a *structure*, a *named constant*” to “a *data entity*”.

host association; change “11.2.2” to “12.1.2.2.1”.

main program; change “*subprogram*” to “*external subprogram*”.

Page 266

Subclause C.4.3

On line 17, change “KIND” to “kind”.

Page 273

Subclause C.7.4

In the first example, change “ENDTYPE” to “END TYPE”.

Page 281

Subclause C.10.1

In line 3, change “an apostrophe edit descriptor” to “a character constant edit descriptor delimited with apostrophes”.

In line 4, change “the apostrophe edit descriptor” to “the edit descriptor”.

In line 5, change “the apostrophe edit descriptor” to “the edit descriptor”.

Pages 286 and 287

Subclause C.11.4

In function ELEMENT, change

```
    INTEGER X
```

to

```
    INTEGER, INTENT(IN) :: X
```

and change

```
    TYPE (SET) A
```

to

```
    TYPE (SET), INTENT(IN) :: A
```

In function UNION, change

```
    TYPE (SET) A, B, UNION
```

to

```
    TYPE (SET) UNION
    TYPE (SET), INTENT(IN) :: A, B
```

In function DIFFERENCE, change

```
    TYPE (SET) A, B, DIFFERENCE
```

to

```
    TYPE (SET) DIFFERENCE
    TYPE (SET), INTENT(IN) :: A, B
```

In function INTERSECTION, change

```
    TYPE (SET) A, B, INTERSECTION
```

to

```
    TYPE (SET) INTERSECTION
    TYPE (SET), INTENT(IN) :: A, B
```

In function SUBSET, change

```
    TYPE (SET) A, B
```

to

```
    TYPE (SET), INTENT(IN) :: A, B
```

Page 292

Subclause C.12.8

Replace the second paragraph through the end of the section “Since it is ... element of B” with

> When execution of a procedure completes, any pointer that remains defined and that is associated with a dummy argument that has the TARGET attribute, remains associated with the corresponding actual argument if the actual argument has the TARGET attribute and is not an array section with a vector subscript.

```
 REAL, POINTER      :: PBEST
 REAL, TARGET       :: B (10000)
 CALL BEST (PBEST, B)            ! Upon return PBEST is associated
 ...                             ! with the "best" element of B
 CONTAINS
   SUBROUTINE BEST (P, A)
     REAL, POINTER     :: P
     REAL, TARGET      :: A (:)
     ...                         ! Find the "best" element A(I)
     P => A (I)
    RETURN
  END SUBROUTINE
END
```

When the procedure BEST completes, the pointer PBEST is associated with an element of B.

An actual argument without the TARGET attribute can become associated with a dummy argument with the TARGET attribute. This permits pointers to become associated with the dummy argument during execution of the procedure that contains the dummy argument. For example:

```
 INTEGER LARGE(100,100)
 CALL SUB(LARGE)
 ...
 CALL SUB()
 CONTAINS
   SUBROUTINE SUB(ARG)
     INTEGER, TARGET, OPTIONAL :: ARG(100,100)
     INTEGER, POINTER, DIMENSION(:,:) :: PARG
     IF (PRESENT(ARG)) THEN
       PARG => ARG
     ELSE
       ALLOCATE (PARG(100,100))
       PARG = 0
     ENDIF
     ...  ! Code with lots of references to PARG
     IF (.NOT. PRESENT(ARG)) DEALLOCATE(PARG)
  END SUBROUTINE SUB
END
```

Within subroutine SUB the pointer PARG is either associated with the dummy argument ARG or it is associated with an allocated target. The bulk of the code can reference PARG without further calls to the PRESENT intrinsic.

Page 294

Subclause C.13.1.6.3

Delete word “Declared” (4 times) and capitalize the first word after each deletion.

Page 339

Subclause D.2.3

After the line

' R408 R409 R410 R420

add the line

" R408 R409 R410 R420