INTERNATIONAL STANDARD ISO/IEC 1539-1:2018
TECHNICAL CORRIGENDUM 2

Published 2023-03

Information technology — Programming languages — Fortran — Part 1: Base language

TECHNICAL CORRIGENDUM 2

Technologies de l'information — Langages de programmation — Fortran — Partie 1: Langage de base

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 1539-1:2018 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ICS 35.060 **Ref. No. ISO/IEC 1539-1:2018/Cor.2:2023(E)**

Blank page

Information technology — Programming languages — Fortran — Part 1: Base language

TECHNICAL CORRIGENDUM 2

Introduction
In the second paragraph, in the tenth sentence of bullet point “Intrinsic procedures and modules”, after “C_F_POINTER” add “and C_F_PROCPOINTER”.

In the second paragraph, in the last sentence of bullet point “Program units and procedures”, after “dummy argument” add “, or a coarray ultimate component of a dummy argument,”.

5.4.7
Append a new sentence to the second paragraph:

> “If a coarray is an unsaved local variable of a recursive procedure, its corresponding coarrays are the ones at the same depth of recursion of that procedure on each image.”

9.7.1.2
Delete the last sentence in the third paragraph, that is “If the coarray … on those images.”, and insert the following three sentences:

> “If the coarray is a dummy argument, the ultimate arguments (15.5.2.3) on those images shall be corresponding coarrays. If the coarray is an ultimate component of a dummy argument, the ultimate arguments on those images shall be declared with the same name in the same scoping unit. If the coarray is an unsaved local variable of a recursive procedure, the execution of the ALLOCATE statement shall be at the same depth of recursion of that procedure on every active image in the current team.”

10.1.11
At the end of the sixth paragraph, add the sentence:

> “If a specification inquiry depends on the type of an object of derived type, that type shall be previously defined.”

11.1.7.2
In the first sentence of constraint C1128, after “of finalizable type,” insert “shall not have an allocatable ultimate component,”

12.6.2.1
After constraint C1213 insert a new constraint:

“C1213a A SIZE= specifier shall not appear in a list-directed or namelist input statement.”

13.7.2.3.3
In table 13.1:
change row 1, column 1 from “E*w*.*d*” to “E*w*.*d* with *w* > 0”;
change row 3, column 1 from “E*w*.*d* E0” to “E*w*.*d* E0 or E0.*d*”;

change row 4, column 1 from "D*w.d*" to "D*w.d* with *w* > 0";
add new row 5 with cells:
column 1: "D0.*d*"
column 2: "any"
column 3: "D$\pm z_1 z_{2\ldots} z_s$ or E$\pm z_1 z_{2\ldots} z_s$"

13.7.2.3.4
In Table 13.2:
change row 1, column 1 from "EN*w.d*" to "EN*w.d* with *w* > 0";
change row 3, column 1 from "EN*w.d* E0" to "EN*w.d* E0 or EN0.*d*";

13.7.2.3.5
In Table 13.3:
change row 1, column 1 from "ES*w.d*" to "ES*w.d* with *w* > 0";
change row 3, column 1 from "ES*w.d* E0" to "ES*w.d* E0 or ES0.*d*";

15.4.3.4.2
In the final sentence of the first paragraph, after "(10.1.5)" insert ", treating a CLASS(*) dummy argument as not differing in type or kind".

15.5.2.11
In the second paragraph of the subclause delete the second and third sentences, that is "If the dummy argument … array element order". Insert a new (third) paragraph:

"If the dummy argument is not of type character with default or C character kind:
- if the actual argument is an array expression, the element sequence consists of the elements in array element order;
- if the actual argument is an array element designator of a simply contiguous array, the element sequence consists of that array element and each element that follows it in array element order;
- otherwise, if the actual argument is scalar, the element sequence consists of that scalar."

In the second bullet point of the third (now fourth) paragraph, after "substring designator" insert "of a simply contiguous array". In the third bullet point change "if the actual" to "otherwise, if the actual" and delete "and not an array ... designator".

15.5.2.13
In the first paragraph, at the end of item (3) (c) delete "or".
At the end of item (3) (d) replace "image." by "image, or
(e) the dummy argument has a coarray ultimate component and the action is a coindexed definition of the corresponding coarray by a different image.".

In the first paragraph, at the end of item (4) (c) delete "or".
At the end of item (4) (d) replace "image." by "image, or
(e) the dummy argument has a coarray ultimate component and the reference is a coindexed reference of the corresponding coarray by a different image.".

Replace the first sentence of NOTE 5 by:

> "The exceptions to the aliasing restrictions for dummy arguments that are coarrays or have coarray ultimate components enable cross-image access while the procedure is executing."

15.7
In the second paragraph, following NOTE 1 and before constraint C1590, add a new constraint:

> C1589a A named local entity or construct entity of a pure subprogram shall not be of a type that has default initialization of a data pointer component to a target at any level of component selection.

In the second paragraph, following constraint C1599, add a new constraint:

> C1599a A reference to the function C_FUNLOC from the intrinsic module ISO_C_BINDING shall not appear in a pure subprogram if its argument is impure.

16.9.46
In paragraph 3, **Arguments,** in the first sentence of the description for argument A delete "dynamic".

In the second sentence, after "It shall not be" insert "polymorphic or".

In the third paragraph, at the end of the final sentence of the description for argument A add: ", including (re)allocation of any allocatable ultimate component, and setting the dynamic type of any polymorphic allocatable ultimate component".

16.9.49
In paragraph 3, **Arguments,** after the first sentence of the description for argument A add the new sentence:

> "It shall not be of a type with an ultimate component that is allocatable or a pointer."

In the same paragraph, in the first sentence of the description for argument `OPERATION` after "nonallocatable, " add "noncoarray, ".

16.9.144
Add a new sentence to the end of the sixth paragraph:

> "If the context of the reference to NULL is an actual argument corresponding to an assumed-rank dummy argument, MOLD shall be present."

16.9.161
In paragraph 3, **Arguments,** in the first sentence of the description for argument `OPERATION` before "nonpointer, " add "noncoarray, ".

17.10
In the third paragraph change the description of ES to read:

> "ES indicates that the procedure is a pure elemental subroutine"

17.11.5
In paragraph 2, **Class**, change "Elemental" to "Pure elemental".

17.11.6
In paragraph 2, **Class**, change "Elemental" to "Pure elemental".

18.2.3.1
In the second sentence, change "C_F_POINTER subroutine is" to "C_F_POINTER and C_F_PROCPOINTER subroutines are".

18.2.3.4
In paragraph 2, **Class**, change "Pure subroutine" to "Subroutine".

18.2.3.7
Replace paragraph 3, **Argument,** by:

> **Argument.** `X` shall be a data entity with interoperable type and type parameters, and shall not be an assumed-size array, an assumed-rank array that is associated with an assumed-size array, an unallocated allocatable variable, or a pointer that is not associated.

18.5.5.9
In paragraph 2, **Formal Parameters**, in the description of `source`, second sentence, delete "`elem_len`, " and delete the comma after "`rank`".

After the same sentence, add a new sentence:

> "If `source` is not a null pointer and the C descriptor with the address `result` does not describe a deferred length character pointer, the corresponding values of the `elem_len` member shall be the same in the C descriptors with the addresses `source` and `result`."

In paragraph 3, **Description,** first sentence, replace "`base_addr` and `dim`" by "`base_addr`, `dim`, and possibly `elem_len`".
At the end of the second bullet point of paragraph 3, **Description,** add the new sentence:

> "If the C descriptor with the address `result` describes a character pointer of deferred length, the value of its `elem_len` member is set to `source->elem_len`."

C.6.8
In the second paragraph replace the entire sample program, that is:

```
   PROGRAM ... END PROGRAM possibly_recoverable_simulation
```

by the following:

```
PROGRAM possibly_recoverable_simulation
  USE, INTRINSIC :: ISO_FORTRAN_ENV, ONLY:TEAM_TYPE, STAT_FAILED_IMAGE
  IMPLICIT NONE
  INTEGER, ALLOCATABLE :: failures (:) ! Indices of the failed images.
  INTEGER, ALLOCATABLE :: old_failures(:) ! Previous failures.
  INTEGER, ALLOCATABLE :: map(:) ! For each spare image k in use,
             ! map(k) holds the index of the failed image it replaces.
  INTEGER :: images_spare ! No. spare images.
                          ! Not altered in main loop.
  INTEGER :: images_used [*] ! On image 1, max index of image in use.
  INTEGER :: failed ! Index of a failed image.
  INTEGER :: i, j, k ! Temporaries
  INTEGER :: status ! stat= value
  INTEGER :: team_number [*] ! 1 if in working team; 2 otherwise.
  INTEGER :: local_index [*] ! Index of the image in the team.
  TYPE (TEAM_TYPE) :: simulation_team
  LOGICAL :: done [*] ! True if computation finished on the image.

  ! Keep 1% spare images if we have a lot, just 1 if 10-199 images,
  !                                             0 if <10.
  images_spare = MAX(NUM_IMAGES()/100,0,MIN(NUM_IMAGES()-9,1))
  images_used = NUM_IMAGES () - images_spare
  ALLOCATE ( old_failures(0), map(images_used+1:NUM_IMAGES()) )
  SYNC ALL (STAT=status)
  local_index = THIS_IMAGE ()
  team_number = MERGE (1, 2, local_index<=images_used[1])
  SYNC ALL (STAT = status)

  outer : DO
    IF (status/=0 .AND. status/=STAT_FAILED_IMAGE) EXIT outer
    IF (IMAGE_STATUS (1) == STAT_FAILED_IMAGE) &
        ERROR STOP "cannot recover"
    IF (THIS_IMAGE () == 1) THEN
    ! For each newly failed image in team 1, move into team 1 a
    ! non-failed image of team 2.
       failures = FAILED_IMAGES () ! Note that the values
                   ! returned by FAILED_IMAGES increase monotonically.
       k = images_used
       j = 1
       DO i = 1, SIZE (failures)
          IF (failures(i) > images_used) EXIT ! This failed image and
          ! all further failed images are in team 2 and do not matter.
          failed = failures(i)
          ! Check whether this is an old failed image.
          IF (j <= SIZE (old_failures)) THEN
             IF (failed == old_failures(j)) THEN
                j = j+1
                CYCLE ! No action needed for old failed image.
             END IF
          END IF
          ! Allow for the failed image being a replacement image.
          IF (failed > NUM_IMAGES()-images_spare) failed = map(failed)
          ! Seek a non-failed image
```

```
         DO k = k+1, NUM_IMAGES ()
           IF (IMAGE_STATUS (k) == 0) EXIT
         END DO
         IF (k > NUM_IMAGES ()) ERROR STOP "cannot recover"
         local_index [k] = failed
         team_number [k] = 1
         map(k) = failed
      END DO
      old_failures = failures
      images_used = k
      ! Find the local indices of team 2
      j = 0
      DO k = k+1, NUM_IMAGES ()
           IF (IMAGE_STATUS (k) == 0) THEN
           j = j+1
           local_index[k] = j
         END IF
      END DO
   END IF
   SYNC ALL (STAT = status)
   IF (status/=0 .AND. status/=STAT_FAILED_IMAGE) EXIT outer
   !
   ! Set up a simulation team of constant size.
   ! Team 2 is the set of spares, so does not participate.
   FORM TEAM (team_number, simulation_team, NEW_INDEX=local_index, &
              STAT=status)
   IF (status/=0 .AND. status/=STAT_FAILED_IMAGE) EXIT outer
   simulation : CHANGE TEAM (simulation_team, STAT=status)
     IF (status == STAT_FAILED_IMAGE) EXIT simulation
     IF (team_number == 1) THEN
        iter : DO
          CALL simulation_procedure (status, done)
          ! The simulation_procedure:
          !  - sets up and performs some part of the simulation;
          !  - starts from checkpoint data if these are available;
          !  - stores checkpoint data for all images from time to
          !  - time and always before return;
          !  - sets status from its internal synchronizations;
          !  - sets done to .TRUE. when the simulation has completed.
          IF (status == STAT_FAILED_IMAGE) THEN
             EXIT simulation
          ELSE IF (done) THEN
             EXIT iter
          END IF
        END DO iter
     END IF
   END TEAM (STAT=status) simulation

   SYNC ALL (STAT=status)
   IF (team_number == 2) done = done[1]
   IF (done) EXIT outer
 END DO outer
 IF (status/=0 .AND. status/=STAT_FAILED_IMAGE) &
   PRINT *,'Unexpected failure',status
END PROGRAM possibly_recoverable_simulation
```