1. **Clause 2,**

 last line. Do not underline "ISO IEC IEEE 60559-2011" and change "ISO IEC IEEE" to "ISO/IEC/IEEE".

1. **Replace Clause 4.7 by**

Fortran is an inherently parallel programming language, with program execution consisting of one or more asynchronously executing replications, called *images*, of the program. The standard makes no requirements of how many images exist for any program, nor of the mechanism of inter-image communication. Inquiry intrinsic procedures are defined to allow a program to detect the number of images in use, and which replication a particular image represents. Synchronization statements are defined to allow a program to synchronize its images; the sync all statement provides a barrier for all images and the sync images statement provides a barrier for specified images. The critical construct defines a scope in which only one image at a time may execute.

A data object may be declared as a *coarray* which allows it to be accessed from another image by using cosubscripts in square brackets to indicate the image.

*Teams* are sets of images that execute independently. The set of all images forms the *initial team*. The form team statement subdivides a team into a set of new teams. The change team construct defines a scope in which the new teams execute.

Any statement that implies ordering between the execution of statements on different images is known as an *image control statement*. One each image, the set of statements executed between two image control statements is known as a *segment*. The orderings imposed by the image control statements imply a partial ordering of all the segments on all the images. Unless a coarray is atomic (next paragraph), if its value (or of part of it) is altered in a segment, it must not be referenced in other segment unless the two segments are ordered.

There is an exception for the segment ordering rule for integers of kind atomic\_int\_kind and logicals of kind atomic\_logical\_kind. These may be referenced and defined in unordered segments by intrinsic subroutines including atomic\_define, atomic\_ref, and atomic\_or. The system insures that for each variable all such actions occur sequentially.

There are several intrinsic subroutines are that *collective* in the sense that the images of the current team collaborate to perform an action, such as summation, that effects data on them all.

1. **Clause 5, item 8 in table.**

At end change ".." to ".".

1. **Clause 6.1, final line.**

Double space after "6.2".

1. **Clause 6.2.1, Ugly spacing in code. I suggest**

 real( kind=rkp ) :: a

and

 real( i, kind=rkp )

1. **Section 6.3.1**

replace paras 3-5 (13 lines) by

Fortran allows constants to be defined by binary, octal, or hexadecimal digits, collectively called BOZ constants. BOZ constants can be used only to initialize variables and as arguments to intrinsic functions that perform bit operations or convert to the numeric types.

The intrinsic int converts to integer type; it pads on the left with zero bits if the bit model for the integer result has more bits, but truncates from the left if the result has fewer bits. The intrinsic real converts to real type; it pads on the left with zero bits if values of the real result have more bits and truncates from the left if they have fewer bits. The intrinsic cmplx converts to complex type; it accepts two arguments for the real and imaginary parts of the result and converts each in the same way as the intrinsic real .

1. **Section 6.6.1, para 6,**

correct double space before "to ISO" and change "kinds" to "kind" after "10646".

1. **Section 6.6.1, end of para 7, add the sentence**

"What happens on input when a value cannot be represented is processor dependent but an error condition should be expected."

1. **Section 6.6.1, para 9,**

change "global state IOSTAT" to "IOSTAT variable" then move the para to be the final sentence of para 7.

1. **Section 6.6.1,**

example in para 10, correct the indenting and line spacing to

 type centigrade

 real :: temp

 end type

 type fahrenheit

 real :: temp

 end type

and

 type (centigrade) function FtoC(t)

 type (fahrenheit) :: t

 FtoC%temp = (t%temp-32.0)/1.8

 end function

1. **Section 6.6.2,**

change final bullet to

 • Include an IOSTAT variable in each IO statement and check its value to ensure that no errors occurred.

1. **Section 6.7.**

At end delete "[SM – What about in C compatibility mode?]"

1. **Section 6.9.2. Add new first bullet**

• Follow the guidance of ISO/IEC 24772-1 clause 6.9.5.

1. **Section 6.16.2.**

Remove first bullet

• Follow the guidance of ISO/IEC 24772-1 clause 6.16.5.

Reasons:

The first two bullets of 24772-1 clause 6.16.5 are

• Determine applicable upper and lower bounds for the range of all variables and use language mechanisms or static analysis to determine that values are confined to the proper range; and

• Analyze the software using static analysis to identify unexpected consequences of shift operations;

This is ridiculous advice in the context of our bullets 2 and 3. Basically,

we are saying don't ever do this unnecessary thing.

Bullet 3 of ISO/IEC 24772-1 clause 6.16.5 is essentially the same as our bullet 3.