Subject: Comments on Clause 7
From: Van Snyder

1 Edits

Edits refer to 07-007r1. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

142:11-22

[It is confusing and verbose to say that the operation involves the specified list of operators, and then say that the operator is the operator in the operation. Why not define the operators first, as at [142:23-30]? Editor: Replace the paragraphs:]

A **numeric intrinsic operator** is +, –, *, /, or **. A **numeric intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is a numeric intrinsic operator and the operands are of numeric type.

The **character intrinsic operator** is //. The **character intrinsic operation** is the intrinsic operation for which the *intrinsic-operator* is the character intrinsic operator and the operands are of type chaacter.

A **logical intrinsic operator** is .AND., .OR., .XOR., .NOT., .EQV., or .NEQV.. A **logical intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is a logical intrinsic operator and both operands are of type logical.

A **bits intrinsic operator** is //, .AND., .OR., .XOR., .NOT., .EQV., or .NEQV.. A **bits intrinsic operation** is an intrinsic operation for which the *intrinsic-operator* is a bits intrinsic operator and at least one operand is of type bits.

144:2-3

[Doesn't belong here; probably shouldn't even be normative. Editor: Delete the paragraph.]

144:4-12

[Why is [144:4-9] so long winded? Editor: Replace the paragraph:]

If both operands of a division operation are integers the result q is the integer such that $x_1/x_2 = q + r$ where r is an integer such that $0 \leq |r| < |x_2|$ and the sign of r is the same as the sign of q.

[Then move 7.1.5.2.2 and 7.1.5.2.3 to [145:3-].]

145: Note 7.19

[To make up for deleting [144:2-3] insert the following after "A / 5.0":]

A ** B	1/(A ** (-B))
A ** B	(1/A) ** (-B)

For integer A and B with nonzero A and negative B, either of the last two alternative forms show that the result is zero. The final alternative form is not recommended for real A if B is large and sufficiently negative that $\log_2 |B|$ is greater than the number of guard digits.

146:7-9

[Paragraph concerns evaluation, not interpretation, and it's confusing. Editor: Move "with the value of x_2" before "concatenated", than move the paragraph to [146:10+].]

147:1, Table 7.8

[Paragraph and Table 7.8 concerns evaluation, not interpretation, and it's confusing. Editor: Move [147:2] to [147:1-].]

148:1-5, Table 7.10

[Paragraphs and Table 7.19 concern evaluation, not interpretation, and it's confusing. Editor: Move [148:6] to [148:1-].]

149:10-11

[Paragraph is about evaluation, not interpretation, notwithstanding that it says "interpreted. . . ." Editor: Replace "is interpreted as having" by "has", then move the paragraph to [150:15+]

149:16-17

[Paragraph is about evaluation, not interpretation, notwithstanding that it says "interpreted. . . ." Editor: Replace "is interpreted as having" by "has", then move the paragraph to [150:15+]

150:10-14

[Paragraphs are about evaluation, not interpretation, notwithstanding that they say "interpreted. . . ."

Editor: Replace "is interpreted as having" by "has", then move the paragraphs to [150:15+]	
[Editor: Replace "often" by "obviously".]	152: Note 7.30
[Needs an ISO-mandated subclause heading.]	153:1+
[Editor: Replace "type parameters and the declared and dynamic types" by "declared and dynamic type and type parameters"; replace "type parameters and the declared and dynamic type" by "declared and dynamic type and type parameters".]	153:6-8
[Editor: Move to [120:4+], where it belongs.]	153:34-154:2
[Editor: Insert "(6.2.2.2.2)" after "subscript".]	154:3
[Editor: Insert "an initialization expression or" before "an expression", insert "or defined by a specification function" after "intrinsic" (this is a little bit of feature creep), delete item (1) from the list.]	155:20-21
[Can't reference both functions at once. Editor: Replace "functions" by "function".]	156:18
[Creating a new instance while construction of one is in progress shouldn't really be a problem. The real problem is that the recursion can't stop. Editor: Replace "The prohibition ... progress" by "Recursion would not terminate and therefore is prohibited.".]	156: Note 7.34
[A module procedure is in a modle, so this applies to module procedures, too. Editor: Insert "the *specification-part* of" before "a module".]	156:35
[One might momentarily wonder what is "a reference to an initialization target?" Editor: Exchange "a reference to the intrinsic function NULL" and "an initialization target".]	157:10-11
[Editor: Move item (8) and its subitems to [157:12+] to make the structure parallel to the list in 7.1.11.]	157:23-28
[A module procedure is in a modle, so this applies to module procedures, too. Editor: Insert "the *specification-part* of" before "a module".]	158:12
[Replace "explicit-shaped" by "explicit-shape" in the last line of the note.]	158: Note 7.36
[Needs an ISO-mandated subclause heading.]	158:16+
[The subclause is about assignment statements, not variables; the assignment statement ought to be the subject. Editor: Replace the paragraph:]	158:17
Execution of an assignment statement can define or redefine the value of a variable.	
[not "defines or redefines" because zero-size arrays or zero-length strings don't get defined or redefined.]	
[Editor: Replace "*variable*" by "the variable".]	159:7
[Editor: In he antepenultimate line of Note 7.40 replace "will cause" by "causes"; in the penultimate line replace "will be" by "is".]	161: Note 7.40
[Every character kind has a blank, so the blank padding character is kind dependent, not procesor dependent. Editor: Delete the first sentence.]	161: Note 7.42
[Editor: Replace "operation" by "assignment".]	163:22
[In dozens of places in the last several pages, we were able to get by with "the variable" instead of "the variable in the assignment." Editor: Delete "in the assignment".]	163:26
[Editor: Insert "(6.2.2.2.2)" after "subscript".]	165:3
[Editor: Delete "no" and insert "not" before "specified".]	166:16
[Editor: Insert "control" after "pending".]	169:16
[Editor: Replace "variable that" by "variable. If *type-spec* appears the variable is an integer of the specified kind. Otherwise it".]	172:1

[Make the note more illustrative: Replace "`INTEGER :: X = -1`" by "`REAL :: X = -1.5`", insert "`INTEGER ::`" before "`X`" in the `FORALL` statement, replace "-1" by "-1.5" in the text after the code.] 172:Note 7.58

[Editor: Replace "will be" by "are" in the text between the input and output.] 174: Note 7.61

2 Comments and questions without edits

What happens when an object is converted to type bits? Where is "converted" defined? Perhaps "converted to" should be "interpreted as if it were of". 149:4

It is confusing to specify some of the material on type, type parameters and shape here, some in 7.1.5.*, and some in both places. 154:18-155:6

The term "variable" and the syntex term "*variable*" appear to be used randomly. Do we want to be consistent? 7.2

Do we need to say something about intrinsic assignment of derived-type objects from one image to another if the type has pointer components? Either "it's prohibited" or "pointer components become undefined?" 163:2+

There's no normative mention here of pointer assignment arising from intrinsic derived-type assignment. In light of C725 at [164:29] it appears that it's impossible for a pointer and target to be on different images. How does it come about? 165:25

Should these paragraphs be constraints, say C733a and C733b? 166:29-33

What is the point of C746? What's the problem with a left-hand function in a *forall-assignment-stmt*, so long as it's pure? Isn't that already covered by C744? 171:12

There's no requirement that if functions appear in the *subscript*s or *stride* they shall be pure, or that one shall not depend upon another, so how can we get away with evaluating them in any order? 172:11

Shouldn't this be a constraint? 176:4-5

3 Does this need an interp?

The requirement ought to be a constraint. Either way, it prevents using length type parameters within type definitions. Conflicts with C453 [66:1-3]. C540 [94:11-12] also prevents using specification expressions for component bounds. 155:15-17

Do we need "and *forall-triplet-spec*" after "*scalar-mask-expr*"? 171:2

It appears to be permitted to have identical *index-name*s in the same FORALL statement. Shouldn't there be a constraint against it? 171:12+