Subject: Comments on Clause 8
From: Van Snyder

1 Edits

Edits refer to 06-007. 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.

[The paragraph contributes nothing that the syntax rules don't already specify. Editor: Delete it.] 175:19-21

If a statement contains a construct name, it **belongs** to that construct. Otherwise if it is not a CYCLE or EXIT statement it belongs to the innermost construct in which it appears. Otherwise it belongs to the innermost DO construct in which it appears. 175:22-23

[Editor: put the "construct" subclauses into alphabetical order, with the EXIT suclause after them.] 176-193

[Subclause **8.1.2.1 Executable constructs in blocks** says nothing that is not said better by the syntax rules. Editor: Delete it.] 176:2-3

[A RETURN statement causes transfer of control, so if this paragraph is to be believed, one cannot return from a procedure that is invoked within the construct. Editor: "Transfer of control" ⇒ "Branching".] 176:5

[The paragraph overlooks EXIT, CYCLE, RETURN and STOP statements. This is a good place to cover all the bases. Editor: Replace "or . . . place" by ", when a branch (8.2) within the block that has a branch target outside the block occurs, when an EXIT or CYCLE statement that belongs to the construct that contains the block or to a construct that contains that construct is executed, when a STOP statement anywhere within the program is executed, or when execution of the program is terminated for any other reason".] 176:10-11

[**8.1.3 IF construct** needs to start with 8.1.3.1. Editor: Replace the subclause heading by "**8.1.3 IF construct and statement**". Then add another subclause heading "**8.1.3.1 IF construct**". Then move "The **IF statement. . . .**" to [177:10], replacing the paragraph there.] 176:12-15

[Editor: "this" ⇒ "completion of execution of this block". This covers EXIT, CYCLE, RETURN and STOP statements.] 177:1

[The branch targets are enumerated at [193:7-10], which is the correct place. Editor: Delete "An ELSE IF . . . statement."] 177:5

[Editor: For stylistic parallelism with the IF construct, insert the following sentences at the beginning of the paragraph:] 178:37

At most one of the blocks in the CASE construct is executed. If there is a CASE DEFAULT statement in the construct, exactly one of the blocks in the construct is executed.

[Editor: "This" ⇒ "Completion of execution of this block". This covers EXIT, CYCLE, RETURN and STOP statements.] 179:11

[Editor: In light of the edit for [178:37], delete this paragraph.] 179:13

[The branch targets are enumerated at [193:7-10], which is the correct place. Editor: Delete "A CASE . . . statement."] 179:14

[Editor: Before "During" insert "Completion of execution of its block completes execution of the construct."] 181:14

[Editor: Before "If the associating entity . . . " insert "If the selector is allocatable it shall be allocated; if it is a pointer it shall be associated with a target and the associating entity becomes associated with that target." This may need to be in a corrigendum. See section 3 below.] 181:27

[Constraint C808 and C812 say that a *variable* with a vector subscript shall not appear in a variable definition context. Editor: Delete "or is an array with a vector subscript,".] 181:30-31

[Editor: For stylistic parallelism with the IF construct, Replace "The SELECT TYPE ... The selection" by "At most one of the blocks in the SELECT TYPE construct is executed. If there is a CLASS DEFAULT statement in the construct, exactly one of the blocks in the construct is executed. The selection of the block to execute".] 182:2

[Editor: Before "A name" insert "Completion of execution of this block completes execution of the construct." This covers EXIT, CYCLE, RETURN and STOP statements.] 182:3

C811a (R822) An associate name shall not be the same as another associate name in the same *select-type-stmt*. 182:12+

[This depends upon the definition of "associate name" in the ordinary normative text in the first paragraph after C819 [183:9-10]. This may need to be in a corrigendum. See section 2 below.]

[The branch targets are enumerated at [193:7-10], which is the correct place. Editor: Delete "A type guard statement ... statement."] 184:2

[The quoted "DO forever" is a poorly defined term. Editor: Replace it by "DO without *loop-control*". At least use opening and closing quotes instead of closing quotes on both sides.] 185:6

[The phrase "anywhere in the DO construct" is inaccurate: consider the case of an EXIT statement that belongs to a construct that is within the subject DO construct. The paragraph also overlooks a CYCLE statement that belongs to a containing constuct, as well as RETURN, STOP and branching statements. Why say all this again when it's already said in subclause 8.1.7.5.4? Editor: "an EXIT ... immediately" ⇒ "the loop can be terminated immediately (8.1.7.5.4)".] 185:7-8

[Editor: "action-stmt" ⇒ "*action-stmt*" twice.] 186:8,21

[Editor: "are said to" doesn't contribute anything. Delete it.] 186:29

[A RETURN statement causes transfer of control, so if this paragraph is to be believed, one cannot return from a procedure that is invoked within the construct. Editor: "transfer of control" ⇒ "branching".] 186:32

[A RETURN statement causes transfer of control, so if this paragraph is to be believed, one cannot return from a procedure that is invoked within the construct. Editor: "Transfer of control" ⇒ "Branching".] 186:37

[The terms "range of the loop" and "loop range" are used without definition. Editor: Insert the following new paragraph:] 186:39+

The range of a loop is the range of the DO construct that defines the loop.

["DO CONCURRENT" isn't a noun. Editor "DO CONCURRENT" ⇒ "construct".] 187:23

[It appears that the range of the loop is not executed if there's no iteration count. Editor: "If ... nonzero" ⇒ "If the loop does not terminate".] 187:36

[The appearance of a *do-construct-name* in a CYCLE statement is not a reference. Editor: "a *cycle-stmt* refers to a *do-construct-name*, it" ⇒ "*do-construct-name* appears, the CYCLE statement".] 188:10

[C829 and C831 ought to be stylistically parallel. Editor: "an outer construct" ⇒ "a construct that contains that DO CONCURRENT construct".] 188:13

[The syntax rules do not permit a *do-term-shared-stmt* to be a construct. The word "itself" doesn't contribute anything. Editor: Delete "or construct itself".] 188:25

[Procedure references are transfers of control, so it seems that if the *do-term-shared-stmt* executes a procedure reference, step (3) of the execution sequence is not executed. Editor: "a further transfer of control results" ⇒ "execution of the *do-term-action-stmt* or *do-term-shared-stmt* causes a branch".]

[Procedure references are transfers of control, so it seems that if a procedure is executed within the range 188:35

of a DO construct then execution of the loop is terminated. Editor: "Control ... to a statement" ⇒ "A branch occurs within the range of the DO construct and it has a branch target".]

[I couldn't find constraints against RETURN or STOP statements appearing within the range of a DO CONCURRENT construct. Do they terminate its execution? If so, after "execution" insert ", when a RETURN or STOP statement within the range of the DO construct is executed, or when execution of the program is terminated for any other reason". Otherwise, constraints are needed on the RETURN and STOP statements. In any case the "when execution of the program is terminated for any other reason" is still needed (so the construct doesn't live on after the program is dead).] 189:4-5

[Editor: overfull hbox.] 189:28-29

[The appearance of a *do-construct-name* in an EXIT statement is not a reference. Editor: "an *exit-stmt* refers to a *do-construct-name*, it" ⇒ "*do-construct-name* appears, the EXIT statement".] 191:4

[A BLOCK construct isn't a block, it's a construct. Editor: Delete "is a block which".] 191:16

[Procedure references are transfers of control, so it seems that if a procedure is invoked within a BLOCK construct, execution of the construct terminates. Also, it seems that execution of a STOP, EXIT or CYCLE statement cannot terminate execution of the construct. This is all covered by the edit for [176:10-11]. Editor: Replace the paragraph by the following:] 191:31-32

Execution of the BLOCK construct is completed when execution of its block is completed.

[Is the description of a critical construct adequate? Is this an improvement on the first paragraph of **8.1.10 CRITICAL construct**? This depends upon the multi-thread model introduced in 177-wvs-005.] 192:2

A **critical construct** does not allow an execution sequence to enter it if one has entered it but not completed execution of it. The execution sequence that is prevented from entering is not terminated; its entry is simply delayed until the execution sequence that is executing the construct completes execution of it. If several execution sequences simultaneously attempt to enter a critical construct, exactly one of them enters it and the others are delayed; which one enters it is processor dependent. If several execution sequences attempt to enter a critical construct while another execution sequence is executing it, which one proceeds when the execution sequence that is executing it completes executing it is processor dependent.

[The description of termination of a critical construct is defective. It allows a branch within a procedure called within the construct to terminate the construct's execution (since its target is not within the block), and it ignores EXIT, CYCLE, STOP and RETURN statements. This is all covered by the edit for [176:10-11]. Editor: Replace the paragraph by the following:] 192:13-14

Execution of the CRITICAL construct is completed when execution of its block is completed.

[What is the reason to use underscores in the SYNC statements? No other statement keyword has underscores, and the only attribute keyword that has underscores is NON_OVERRIDABLE. Suggestion: Replace the underscores by optional blanks (optional only to truckle to fixed form).] Subclause 8.5

[Although Note 8.36 explains why SYNC_IMAGES(*) might not be the same as SYNC_ALL, it doesn't explain the difference, if any, between all images executing SYNC_ALL and all images executing SYNC_IMAGES(*). If there is no difference, why do we have both?] 202 Note 8.36

[It is confusing to state events in the opposite order from which they occur. Editor: "increases ... $N_{T\rightarrow M}$" ⇒ "compares a record of the number of times, $Q_{M\leftarrow T}$, image M executed such a QUERY statement, with $N_{T\rightarrow M}$, and then increases the value of $Q_{M\leftarrow T}$ by 1"] 203:8-10

[Editor: "<=″ ⇒ "`\leq`".] 203:12

2 If the constraint on SELECT TYPE needs an interp ...

NUMBER: TBD
TITLE: Duplicate associate names in SELECT TYPE
KEYWORDS: ASSOCIATE NAME, SELECT TYPE

DEFECT TYPE: Erratum
STATUS:

QUESTION:

does the following SELECT TYPE statement conform to the 2003 Fortran
standard?

```
  SELECT TYPE ( A, A => B, A => C )
```

ANSWER:

The statement does not conform to the 2003 Fortran standard. It was an
oversight that a constraint parallel to C809, which prohibits duplicate
associate names in ASSOCIATE statements, does not apply to the SELECT TYPE
statement. Edits are supplied to correct this oversight.

EDITS:

After constraint C811 [182:12+] insert the following constraint:

"C811a (R822) An associate name shall not be the same as another associate
name in the same <select-type-stmt>."

This depends upon the definition of "associate name" in the ordinary
normative text in the first paragraph after C819 [163:9-10].

SUBMITTED BY: Van Snyder

HISTORY:

3 If the allocatable or pointer selector needs an interp . . .

NUMBER: TBD
TITLE: Duplicate associate names in SELECT TYPE
KEYWORDS: ASSOCIATE NAME, SELECT TYPE
DEFECT TYPE: Erratum
STATUS:

QUESTION:

Assuming B is an allocated allocatable and D is a pointer, does the
following fragment conform to the 2003 Fortran standard?

```
  DEALLOCATE ( B )
  NULLIFY ( D )
  ASSOCIATE ( A => B, C => D )
```

ANSWER:

The fragment does not conform to the 2003 Fortran standard. It was an
oversight that the selector was not required to be allocated if
allocatable, or associated with a target if a pointer. Edits are supplied
to correct this oversight.

EDITS:

Within the first paragraph of subclause 8.1.5.3 Attributes of associate names, before "If the associating entity is polymorphic" insert "If the selector is allocatable it shall be allocated; if it is a pointer it shall be associated with a target and the associating entity becomes associated with that target."

SUBMITTED BY: Van Snyder

HISTORY: