To: /B subgroup
Subject: Comments on section 9
From: Van Snyder

1 Edits

Edits refer to 01-007r2. 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 immediately following text, while a page and line number followed by + (-) indicates that immediately following text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

[There's no verb after the semicolon, we don't need an outdated example, and even if we keep it, it ought to be a note. Editor: Delete "; ... device".] 161:39-40

[The note, as written, seems not to be germane at this point. Editor: Replace with:] 162:30-32

NOTE 9.3

An example of a processor-dependent set of allowed actions is that for a printer, the set would include the write action but not the read action.

[Is it really true that the name of a named file *is* a character string? Editor: Insert "represented" before "a character"; insert "value" after "string".] 162:34

[Editor: First "the file" ⇒ "a file".] 163:14

[Record numbers are defined only for direct access files; they are not defined for sequential files. Editor: "the orders ... numbers" ⇒ "order".] 163:17-18

[Editor: To avoid confusion about the meaning of "unit" insert "file storage" after "preceeding".] 164:23

[9.2.2.3 appears to require that it always be possible to position a stream file (though the description of POS= later on says that this is not so). Fix the misstatement for unformatted streams. Editor: "File" ⇒ "If it is possible to position the file, the file".] 164:24

[Ditto formatted streams. Editor: "The" ⇒ "If it is possible to position the file, the".] 164:46-47

[The terms used here are not the names of the specifiers for the changeable modes, so it takes some investigation to discover that they're all covered. The initial values of changeable modes are the same that would result by not specifying the mode in an OPEN statement. It's simpler to say that, and it would cover all changeable modes, even if we add a few later. By specifying how padding works, the present wording appears to preclude the possibility that a changeable mode might be changed during a READ or WRITE statement.] 167:37-42

(8) The initial value of a connection mode (9.4.1) is the value that would be implied by an initial OPEN statement without the corresponding keyword.

[Editor: Re-number subsequent items (I presume Frame does this automatically).]

[This restriction isn't quite right. One can imagine, for example, creating a file name by writing to an internal file, and then using the variable that is the internal file in a FILE= specifier in an INQUIRE or OPEN statement. Editor: After "specified" insert "as the unit".] 167:45-46

[The phrases "for an existing file" and "for a new file" are used in the discussion of several specifiers. It's not clear how this interacts with a STATUS= specifier with the value REPLACE.] 172:1–

In the following subclauses, the term "existing file" refers to a file that exists at the moment the connection is established, and the term "new file" refers to a file that does not exist at the moment the connection is established.

NOTE 9.18$\frac{1}{3}$

A file may exist at the moment the OPEN statement begins execution, but cease to exist before the connection is established. This may occur, for example, if a STATUS= specifier appears and has the value REPLACE.

[Why does this say "A file that did not exist previously (a new file, either specified explicitly or by default)" while elsewhere it just says "A new file"? Editor: "A file ... default)" ⇒ "A new file".] 173:19-20

[What else might an asterisk signify here? Editor: Delete "specifying ... input/output".] 176:42-43

[Editor: "be specified" ⇒ "appear".] 180:4

[Editor: insert "if and only" after "list".] 181:11

[Editor: "allocatables" ⇒ "allocatable" twice.] 181:37,43

[Editor: Insert "or subobject thereof" after "list item" at [182:9], then move to [181:37+] to be with the rest of unformatted UDDTIO.] 182:8-20

[Editor: Insert "is" after "record".] 182:35

[Editor: It appears that there's an extra blank after "either".] 184:33

[Editor: "unit or internal file" ⇒ "or internal unit" twice.] 185:8,13

[Editor: Delete "The next effective item to be processed is called the next effective item" because it doesn't say anything.] 185:31-32

[User-defined derived-type input/output does not alter the processing of derived-type effective items – it completely replaces it. Editor: "alter" ⇒ "override".] 187:33

[Editor: "as" ⇒ "that is".] 187:34

[Editor: Move [191:39-42] to here. 191:23+

[Editor: "values" ⇒ "a value".] 191:27

["there is no current record and no preceding record" is just a fancy way of saying "the file is at its initial point". Editor: replace the former by the latter.] 195:4

[Editor: "variable" ⇒ "variables"; "specifier" ⇒ "and IOMSG= specifiers".] 197:25

[How do you read "on" a file? Editor: "or written on" ⇒ "from or written to".] 199:39

[Since an inquire by output list form "includes only an IOLENGTH= specifier" there's no point in saying that it "does not include a FILE= or UNIT= specifier." Editor: "does not include a FILE= or UNIT= specifier, and includes" ⇒ "contains".] 202:15-16

---

2 Subclauses 9.5.3 and 9.9 ought to be combined

Subclauses 9.5.3 and 9.9 contain some duplicative material, some contradictory material and some incomplete material (mostly incomplete with respect to wait operations). There is no subclause not in 9.5 that addresses error conditions, but error conditions can occur in other than data transfer statements. To address these problems, subclauses 9.5.3 and 9.9 ought to be combined.

[Editor: Delete.] 183:1-23

[Editor: “9.5.3” ⇒ “9.9” twice.] 166:11,12

[Editor: “9.5.3” ⇒ “9.9”.] 166:27

[Editor: “9.5.3” ⇒ “9.9”.] 183:43

[Editor: “9.5.3” ⇒ “9.9”.] 184:21

[Editor: “9.5.3” ⇒ “9.9”.] 187:18

[Editor: “9.5.3” ⇒ “9.9” twice.] 191:25,37

[Editor: “9.5.3” ⇒ “9.9”.] 193:8

[Editor: Move [182:31-183:7] to replace the subclause heading and this paragraph. Make sure to do the edit specified above for [182:35] first. This paragraph is useless and repetitive introductory waffle, so it’s not a loss to just replace it.] 202:26-30

[Editor: Delete “(9.5.3)” twice.] 203:3,18

[Editor: Delete “(9.5.3)” twice.] 203:30

[Editor: Delete “(9.5.3)” thrice.] 203:41-42

9.9.3 Error conditions and the ERR= specifier 203:17-19

If an error condition occurs during execution of a data transfer the position of the file becomes indeterminate. [This is here because it occurs independently of whether the statement contains an ERR= or an IOSTAT= specifier. The lack of “statement” after “transfer” is intentional, to cover the case of an error condition occurring during a wait operation.]

If an error condition occurs during execution of an input/output statement that contains neither an ERR= specifier nor an IOSTAT= specifier, execution of the program is terminated. If an error condition occurs during execution of an input/output statement that contains either an ERR= specifier or an IOSTAT= specifier

[Editor: Insert “, if any,” after “list”.] 203:20

(2) If the statement is a data transfer statement or the error condition occurs during a wait operation, all implied DO variables in the statement that initiated the transfer become undefined. 203:21

($5\frac{1}{2}$) If the statement is a READ statement or the error condition occurs in a wait operation for a transfer initiated by a READ statement, all input list items or namelist group objects in the statement that initiated the transfer become undefined. [Is the “in” part good enough to cover namelist group objects?] 203:27+

[Editor: “Execution” ⇒ “If an ERR= specifier appears, execution”.] 203:28

9.9.4 End-of-file conditions and the END= specifier 203:29-31

If an end-of-file condition occurs during execution of an input/output statement that contains neither an END= specifier nor an IOSTAT= specifier, execution of the program is terminated. Otherwise if an error condition does not occur [we don’t need the “If an end-of-file condition occurs during execution of an input/output statement...” part that we needed for error conditions, because end-of-file conditions can only occur during execution of input/output statements]

($1\frac{1}{3}$) All implied DO variables in the statement that initiated the transfer become undefined. 203:32+

($1\frac{2}{3}$) If the statement is a READ statement or the end-of-file condition occurs in a wait operation for a transfer initiated by a READ statement, all input list items or namelist group objects in the statement that initiated the transfer become undefined.

[Editor: "input" ⇒ "input/output" twice to cover the case of an end-of-file condition occurring during a wait operation.] 203:35,37

[Editor: "Execution" ⇒ "If an END= specifier appears, execution".] 203:39

9.9.5 End-of-record conditions and the EOR= specifier 203:40-42

[This needs to be checked carefully because the material from which it is taken at [183:8-13] and [183:22-23] is difficult to follow, and may be contradictory.]

If an end-of-record condition occurs during a data transfer initiated by a nonadvancing input statement, the statement in which the condition is detected has neither an EOR= nor an IOSTAT= specifier, and the pad mode has the value NO, execution of the program is terminated. Otherwise if neither an error nor an end-of-file condition occurs

---

($2\frac{1}{2}$) All implied DO variables become undefined. 204:4+

[Editor: "input" ⇒ "input/output" twice to cover the case of an end-of-record condition occurring during a wait operation.] 204:6,8

[Editor: "Execution" ⇒ "If an EOR= specifier appears, execution".] 203:39

[Editor: Move [202:31-203:16] to here.] 204:12+

[Editor: "9.5.3" ⇒ "9.9".] 385:24

---

3 Don't know what to do

The term "permanently" is a bit extreme. 169:17-18

This constraint is rather fatuous, given that one can do a nonadvancing asynchronous read, and then put an EOR= specifier in a WAIT statement. 176:47

The term "base object" is defined in 6.1.2. It's not obvious that it's defined for anything other than structures. 178:35

The explanation that a variable may be a pending input/output storage sequence affector in some situations and not others is schizophrenic. First it discusses different scoping units, and then different instants of time. Which is it? 179:1-3

Appears to contradict [183:8-13]. This paragraph is deleted by edits proposed in section 2. 183:22-23

Is there a set of specifiers that is allowed for inquire by name but not inquire by unit, or vice-versa? If so, a summary here would be useful. 197:26+

---

4 Very MTE

[There is no good reason why backspacing over records written using list-directed or namelist formatting is prohibited. Records are perfectly well defined in these cases. The problem is that one might not know where a record boundary is. But this also applies in the case of "ordinary" formatted output that has slash edit descriptors or in which format reversion takes place.] 195:11

NOTE 9.49$\frac{1}{2}$

Backspacing over records written by list-directed or namelist formatting should only be done with care. List-directed or namelist formatting can put record boundaries in unpredictable places.