ISO/IEC JTC 1/SC 22 N 4342

2008-05-19

ISO/IEC JTC 1/SC 22
Programming Languages

Document Type:	**Defect Report**
Document Title:	**Defect Report 3 for ISO/IEC 1539-1:2004, Programming languages – Fortran**
Document Source:	**SC 22/WG 5 Convener**
Document Status:	**For information and review when voting on SC 22 N 4343.**
Action ID:	**FYI**
Due Date:	
No. of Pages:	**26**

ISO/IEC JTC 1/SC 22 N 4342 2008-05-19
ISO/IEC JTC1/SC22/WG5-N1730

Defect reports that led to Corrigendum 3 and responses to them

Stan Whitlock

```
*  W E F03/0050   Questions about internal files
*  W E F03/0079   Value of decimal exponent for a real zero value
*  W I F03/0080   Formatted output of a negative real zero value
*  W E F03/0086   Elemental and BIND(C)
*  W E F03/0088   Defined operations/assignments and
                  VOLATILE/ASYNCHRONOUS
*  W E F03/0089   Interoperability of non-BIND derived types
*  W E F03/0092   Procedure characteristics and unlimited
                  polymorphic
*  W E F03/0093   Allocatable array on intrinsic assignment with
                  scalar expr
*  W E F03/0094   Final subroutine and VALUE attribute
*  W I F03/0095   Bounds remapped pointer assignment and ASSOCIATED
*  W E F03/0097   Blanks as separators in NAMELIST input
*  W E F03/0101   Is UDDTIO output suitable for namelist and
                  list-directed input
*  W I F03/0104   Deallocation and finalization of bounds-remapped
                  pointers
*  W E F03/0106   Inquire by unit inconsistencies
*  W E F03/0107   Are the IEEE_* elemental routines required
*  W C F03/0108   Is IEEE_SUPPORT_NAN consistent with the other
                  IEEE_SUPPORT functions
```

NUMBER: F03/0050
TITLE: Questions about internal files
KEYWORDS: internal file, data transfer
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTIONS:

Question 1:

Fortran 2003 does not seem to prohibit this kind of recursive internal
input/output. Was this program intended to be standard-conforming?

If so, then what does the program print?

```
MODULE m1
  CHARACTER(20) :: ifile = ''
CONTAINS
  CHARACTER(3) FUNCTION foo()
    WRITE(ifile, *) 'QWERTY'
    foo = 'abc'
  END FUNCTION
END MODULE

PROGRAM ex1
  USE m1
  WRITE(ifile, *) 'xyz', foo(), 'zyx'
  PRINT *, ifile
END PROGRAM
```

Question 2:

Fortran 2003 does not seem to prohibit this kind of recursive internal input/output. Was this program intended to be standard-conforming? If so, then what does the program print?

```
MODULE m2
  CHARACTER(20) :: ifile = 'abc def ghi jkl mno '
  CHARACTER(3) :: char
CONTAINS
  CHARACTER(3) FUNCTION foo()
    READ(ifile, *) char
    foo = 'abc'
  END FUNCTION
END MODULE

PROGRAM ex2
  USE m2
  WRITE(ifile, *) 'xyz', foo(), 'zyx'
  PRINT *, ifile
  PRINT *, char
END PROGRAM
```

Question 3:

Fortran 2003 does not appear to prohibit modifying a character

variable when it is being used as an internal file in a data transfer statement that is currently executing. Was this program intended to be standard-conforming? If so, then what does the program print?

```
MODULE m3
  CHARACTER(20) :: ifile = ''
CONTAINS
  CHARACTER(3) FUNCTION foo()
    ifile = 'bad thing to do?'
    foo = 'abc'
  END FUNCTION
END MODULE

PROGRAM ex3
  USE m3
  WRITE(ifile, *) 'xyz', foo(), 'zyx'
  PRINT *, ifile
  PRINT *, flag
END PROGRAM
```

Question 4:

Fortran 2003 does not appear to prohibit referencing a character variable when it is being used as an internal file in a data transfer statement that is currently executing. Was this program intended to be standard-conforming? If so, then what does the program print?

```
MODULE m4
  CHARACTER(20) :: ifile = ''
  LOGICAL :: flag = .FALSE.
CONTAINS
  CHARACTER(3) FUNCTION foo()
    IF (ifile == ' xyz') THEN
      flag = .TRUE.
    END IF
    foo = 'abc'
  END FUNCTION
END MODULE

PROGRAM ex4
  USE m4
  WRITE(ifile, *) 'xyz', foo(), 'zyx'
  PRINT *, ifile
  PRINT *, flag
END PROGRAM
```

ANSWER:

All of these examples were intended to be prohibited.
Edits are provided to prohibit referencing or defining a variable used as an internal unit as a result of evaluating any output list items, or transferring values to any input list item.

EDITS:

In section 9.5.3.4, after the seventh paragraph:
"If an internal file has been specified, an input/output list item shall not be in the file or associated with the file."

add these paragraphs [196:29+]:

"During the execution of an output statement that specifies an internal file, no part of that internal file shall be referenced, defined, or become undefined as the result of evaluating any output list item.

During the execution of an input statement that specifies an internal file, no part of that internal file shall be defined or become undefined as the result of transferring a value to any input list item."

SUBMITTED BY: Rob James

```
HISTORY: 05-141    m171  F03/0050 Submitted
         06-368    m178  Passed by J3 meeting
         07-272    m181  Passed as changed by J3 letter ballot #13
         08-155    m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0079
TITLE: Value of decimal exponent for a real zero value
KEYWORDS: Data edit descriptors, Numeric editing, decimal exponent, zero value
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

In formatted output, what is the value of the
decimal exponent produced for a real zero value
under the D, E, EN, ES, and G edit descriptors?

ANSWER:

In such a case, the decimal exponent should have
the value zero whether or not a nonzero scale factor
is in effect. Edits are supplied to make this clear.

DISCUSSION:

The Fortran 2003 standard does not specify what the value of the decimal exponent of a real zero value should be under formatted output. Every implementation of which Sun is aware uses the value zero for the decimal exponent unless a nonzero scale factor is in effect. Different implementations format real zeros differently under nonzero scale factors, but the difference is mainly in the form of the mantissa and not the exponent.

EDITS:

[227:15+] At the end of the numbered list in 10.6.1 "Numeric
editing", add:

"(7) On output of a real zero value, the digits in the
exponent field shall all be zero."

SUBMITTED BY: Michael Ingrassia

```
HISTORY: 06-125    m175  F03/0079 Submitted
         07-281r2  m182  Passed by J3 meeting
         08-133r2  m183  Passed by letter ballot #15 08-101
         08-164    m184  Passed by WG5 ballot N1722-N1726
```

NUMBER: F03/0080
TITLE: Formatted output of a negative real zero value
KEYWORDS: formatted output, negative zero, IEEE

DEFECT TYPE: Interpretation
STATUS: Passed by WG5 ballot

QUESTION:

Suppose a Fortran processor's representation of the real zero value is signed. When a negative real zero value is written using formatted output, does the Fortran 2003 standard require the representation of the zero value in the output field to be prefixed with a minus sign?

ANSWER:

Yes, the negative sign is required to appear in formatted output of a negative zero value. In subclause 10.6.1, list item (3) at [227:3-4] says "The representation of a negative internal value in the field shall be prefixed with a minus sign." For a processor that distinguishes between positive and negative zero, there is no exemption for output at [38:1-6]. For the case of IEEE reals, the IEEE_IS_NEGATIVE function at [375:25] explicitly says that -0.0 is "negative".

EDITS:

None.

SUBMITTED BY: Michael Ingrassia

```
HISTORY: 06-126    m175  F03/0080 Submitted
         07-282r1  m182  Passed by J3 meeting
         08-133r2  m183  Passed by letter ballot #15 08-101
         08-164    m184  Passed by WG5 ballot N1722-N1726
```

NUMBER: F03/0086
TITLE: Elemental and BIND(C)
KEYWORDS: Elemental, BIND(C), ENTRY
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

Is it allowed for a procedure to have both the BIND(C) and elemental attributes?

Constraint C1242 disallows trivial ways of writing an elemental BIND(C) procedure. However, the following example achieves the effect for sub_c without violating the syntactic constraint.

```
  elemental subroutine sub(x)
    entry sub_c(x) bind(c)
  end subroutine sub
```

ANSWER:

No, it is not allowed. Constraint C1242 was intended to disallow the combination of elemental and BIND(C), but it inadvertently failed to cover the case shown in the above example.

EDITS

Replace C1242 in subclause 12.5.2.1 with
[280:6-7]
"C1242 An elemental procedure shall not have the BIND attribute.".

SUBMITTED BY: Richard Maine

```
HISTORY: 07-101    m179  Submitted F03/0086
         07-101    m179  Passed by J3 meeting
         07-272    m181  Passed as changed by J3 letter ballot #13
         08-155    m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0088
TITLE: Defined operations/assignments and VOLATILE/ASYNCHRONOUS
KEYWORDS: Defined operations, defined assignment, VOLATILE,
          ASYNCHRONOUS
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

PROBLEM:

Fortran 2008 Unresolved Technical issue 097 asked a question that also affects Fortran 2003. Consider this example:

```
       INTERFACE ASSIGNMENT(=)
         SUBROUTINE s(a,b)
           REAL,INTENT(OUT),VOLATILE :: a(1,*)
           REAL,INTENT(IN) :: b(:)
         END SUBROUTINE
       END
       REAL,POINTER :: p(:,:),q(:)
       ...
       CALL s(p,q)    ! Violation of constraint C1233 [271:9-11],
                      !  associating P with A
       p = q          ! No constraint violation because
                      !  <actual-arg> syntax is not being used
```

QUESTION:

Did Fortran 2003 intend to enforce constraints on <actual-arg> in defined assignment?

ANSWER:

Yes, the <actual-arg> constraints and restrictions should be enforced in defined assignment and in defined operator evaluation.

Edits are provided below to do this.

EDITS:

[262:16] add at the end of the paragraph
"All restrictions and constraints that apply to actual arguments in a reference to the function also apply to the corresponding operands in the expression as if they were used as actual arguments."

[263:12] insert after "the second argument."
"All restrictions and constraints that apply to actual arguments in a reference to the subroutine also apply to the left-hand side and to the right-hand side enclosed in parentheses as if they were used as actual arguments."

SUBMITTED BY: Stan Whitlock

HISTORY: 07-172   m179 Submitted F03/0088 {see 07-171 for F08 fix}

07-172 m179 Passed by J3 meeting
07-272 m181 Passed as changed by J3 letter ballot #13
08-155 m184 Passed by WG5 ballot #4 N1711-N1721

NUMBER: F03/0089
TITLE: Interoperability of non-BIND derived types
KEYWORDS: Interoperability, derived type
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

INTRODUCTION

Subclause 15.2.3 of 04-007 says [398:9-12]:

"A Fortran derived type is interoperable with a C struct type if the derived-type definition of the Fortran type specifies BIND(C) (4.5.1), the Fortran derived type and the C struct type have the same number of components, and the components of the Fortran derived type have types and type parameters that are interoperable with the types of the corresponding components of the struct type."

QUESTIONS

1. Is a Fortran derived type for which BIND(C) is not specified interoperable with any C struct type?

2. Does a Fortran derived type interoperate with a C struct type that has a different number of components?

3. Does a Fortran derived type interoperate with a C struct type that specifies the same components in a different order?

4. Does a Fortran derived type with a pointer or allocatable component that has interoperable type and type parameters interoperate with any C struct type?

ANSWERS:

None of these Fortran derived types are interoperable with any C struct type.

EDITS:

[398:9] Replace "if" by "if and only if".

SUBMITTED BY: Van Snyder

```
HISTORY: 07-213    m180  Submitted F03/0089
         07-213r2  m180  Passed by J3 meeting
         07-272    m181  Passed by J3 letter ballot #13
         08-155    m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0092
TITLE: Procedure characteristics and unlimited polymorphic
KEYWORDS: Procedure, unlimited polymorphic
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

Consider

```
   abstract interface
      function foo (x)
         class(*) x
         class(*), pointer :: foo
      end function
   end interface

   procedure (foo), pointer :: proc_ptr
   procedure (foo),         :: proc_tgt

   proc_ptr => proc_tgt
```

According to the rules of procedure pointer assignment at [144:39-41], proc_ptr and proc_tgt are required to have the same interface characteristics. However because an unlimited polymorphic entity is not considered to have a declared type, the rules for characteristics of dummy data objects [256:26-32] and characteristics of function results [257:2-8] are not applicable. In addition, rules at [145:5-6] require that proc_ptr and proc_tgt have the same function return type. This also does not apply to unlimited polymorphic data.

Is the example intended to be standard-conforming?

ANSWER:

Yes, the example was intended to be standard-conforming.
An edit is provided to clarify this.

The characteristics however are adequately defined. FOO, and thus
both PROC_PTR and PROC_TGT have no type, but are polymorphic; this
precisely characterises an unlimited polymorphic entity. Only the
requirement of type matching in 7.4.2.2 is incorrect.

EDITS to 04-007:

[145:5] After "the same type"
 insert " or both be unlimited polymorphic".

SUBMITTED BY: Jim Xia

```
HISTORY: 07-247      m181  F03/0092 Submitted
         07-247r1    m181  Passed by J3 meeting
         07-279/321  m182  Passed as changed by J3 letter ballot #14
         08-155      m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0093
TITLE: Allocatable array on intrinsic assignment with scalar expr
KEYWORDS: allocatable array, intrinsic assignment
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

Consider

```
   CHARACTER(:), ALLOCATABLE :: str(:)
   ALLOCATE (CHARACTER(1) :: str(0:9))
   str = 'reallocate?'
```

According to the third paragraph of 7.4.1.3, the variable STR should
be deallocated on this assignment because it has a deferred length

type parameter different from the <expr> ('reallocate?'); it should then be allocated with its length type parameter the same as that of the <expr> and with the shape and bounds of <expr>. But the STR cannot be allocated with the shape and bounds of the <expr> since it is a scalar.

The standard, however, provides a possible interpretation for the shape of <expr> two paragraphs later where it says
"If <expr> is a scalar and <variable> is an array, the <expr> is treated as if it were an array of the same shape as <variable> with every element of the array equal to the scalar value of <expr>."

Q(1). Should the variable STR be reallocated in this case?

Q(2). If so, what are the values of its length type parameter, shape and bounds?

ANSWER:

(1) Yes, STR should be reallocated - that is the purpose of the combination of ALLOCATABLE and deferred type parameters. If the user does not wish for automatic reallocation he can use "str(:) = 'do not reallocate'" instead.

(2) The length parameter of str after the assignment is 11 (the value returned by LEN('reallocate?')). The shape and bounds should be unchanged. An edit is provided to clarify this.

Note that the standard does not forbid, but does not specify semantics for,

```
  str = 'oops'
```

when STR is an unallocated array with a deferred length parameter. An edit is supplied to make it clear that this is not allowed.

Note also that this applies to parameterized derived types with deferred type parameters.

EDITS:

[139:22-] Insert new sentence at beginning of paragraph

"If <variable> is an unallocated allocatable array, <expr> shall
have the same rank as <variable>."

[139:25] Change "corresponding type parameters of <expr>,"
to "corresponding type parameter of <expr>."

[139:25] Before ", with the shape of <expr>"
Insert ". If <variable> is an array and <expr> is scalar it
is allocated with the same bounds as before,
otherwise it is allocated".

SUBMITTED BY: Jim Xia

HISTORY: 07-248 m181 F03/0093 Submitted
07-248r2 m181 Passed by J3 meeting
07-279/321 m182 Passed as changed by J3 letter ballot #14
08-155 m184 Passed by WG5 ballot #4 N1711-N1721
N1727 m184 Note edit changes in F2003 Corrigendum 3

The second [139:25] edit leaves a "," after the insertion. The edit
should read:

[139:25] Replace ", with" with
". If <variable> is an array and <expr> is scalar it
is allocated with the same bounds as before,
otherwise it is allocated with".

N1727 combines the 2 edits on [139:25] above as

In the second sentence of the third paragraph of the subclause,
change "corresponding type parameters of <expr>," to "corresponding
type parameter of <expr>. If variable is an array and <expr> is scalar
it is allocated with the same bounds as before, otherwise it is
allocated".

NUMBER: F03/0094
TITLE: Final subroutine and VALUE attribute
KEYWORDS: Final subroutine, VALUE
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

Currently, the F03 standard allows the VALUE attribute to be specified for the dummy argument of a final subroutine. This seems to defeat the purpose of final subroutine, which is intended to apply to the finalizable entity (the actual argument) itself.

Should the dummy argument of a final subroutine be allowed to have the VALUE attribute?

ANSWER:

No, the VALUE attribute should not be allowed.
An edit is provided to correct this oversight.

EDITS to 04-007:

[58:14] In the last sentence of C473 in 4.5.5 "Final subroutines",
replace "not be INTENT(OUT)"
with "not have the INTENT(OUT) or VALUE attribute".

SUBMITTED BY: Jim Xia

```
HISTORY: 07-249      m181  F03/0094 Submitted
         07-249r1    m181  Passed by J3 meeting
         07-279/321  m182  Passed by J3 letter ballot #14
         08-155      m184  Passed by WG5 ballot #4 N1711-N1721
```

--

NUMBER: F03/0095
TITLE: Bounds remapped pointer assignment and ASSOCIATED
KEYWORDS: pointer assignment, bounds-remapping, ASSOCIATED
DEFECT TYPE: Interpretation
STATUS: Passed by WG5 ballot

QUESTION:

Case (v) of intrinsic inquiry function ASSOCIATED [305:5-9] says

> **If TARGET is present and is an array target, the result is true if the target associated with POINTER and TARGET have the same shape, are neither of size zero nor arrays whose elements are**

zero-sized storage sequences, and occupy the same storage units
in array element order. Otherwise, the result is false. If
POINTER is disassociated, the result is false.

This will cause the intrinsic to return false if the POINTER is
pointer assigned to the TARGET with bounds-remapping (POINTER and
TARGET can be of different ranks). The same issue also exists for
case (vii).

Is the POINTER associated with the TARGET if the POINTER is pointer
assigned to the TARGET with bounds-remapping?

ANSWER:

No, it is not intended that ASSOCIATED(POINTER, TARGET) return true
after pointer assignment using a bounds-remapping that changes the
shape or rank. This was a conscious decision made in response to a
Fortran 90 interpretation request concerning dummy arguments that are
different shaped versions of the same array in the calling procedure.

EDITS to 04-007:

none

SUBMITTED BY: Jim Xia

```
HISTORY: 07-259      m181  F03/0095 Submitted
         07-259r2    m181  Passed by J3 meeting
         07-279/321  m182  Passed by J3 letter ballot #14
         08-155      m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0097
TITLE: Blanks as separators in NAMELIST input
KEYWORDS: Namelist input, blanks, separators
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

1) Was it intended that blanks be allowed as separators in Namelist
Input?

Consider a namelist input fragment:

```
          I = 3          J = 4
```

Page 243:12 says that the name-value subsequences are separated by value separators.

Page 243:5 says that namelist value separators are the same as list directed value separators.

Page 239:7 says those value separators are "...blanks between values" and then defines what the values are.

The "J" above isn't a value, so the blanks aren't separators and the fragment is illegal for namelist input

2) Is there a similar problem with namelist comments as in this fragment?

```
          I = 3         !  this is a namelist comment
```

Page 245:29-30 says that a name-value subsequence is separated from the ! in a comment by a value separator.

ANSWER:

1) Yes, it was intended to allow blanks as separators for namelist input. Edits are supplied to correct the wording in the standard.

2) Yes, there is a similar problem with comments. The fragment is intended to be legal. The edits correct the error.

EDITS:

[243:5] Replace the paragraph by
"A value separator for namelist formatting is a value separator for list-directed formatting (10.9), or one or more contiguous blanks between a nonblank value and the following object designator or "!" comment initiator."

SUBMITTED BY: Dick Hendrickson

HISTORY: 07-267 m181 F03/0097 Submitted
07-267r2 m181 Passed by J3 meeting
07-279/321 m182 Passed as changed by J3 letter ballot #14
08-155 m184 Passed by WG5 ballot #4 N1711-N1721

NUMBER: F03/0101
TITLE: Is UDDTIO output suitable for namelist and list-directed input
KEYWORDS: UDDTIO, list-directed I/O, namelist I/O
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

The first paragraph of 10.9.2 says that the form of the values produced by list-directed output is the same as that required for input. It also says values are separated blanks or commas, etc.

The first paragraph of 10.10.2 has similar words for namelist output. It also requires that the variable name be produced in upper case and that the output consist of name-value pairs.

Is it intended that output produced by user-defined derived-type output routines conform to these rules?

ANSWER:

No, it was not intended to constrain the user-defined derived-type output values. There should be an exception similar to the one for adjacent undelimited character values. User-defined derived-type output fields do not need to be readable by either namelist or list-directed input.

EDITS:

[241:5] Add at the end of the paragraph
"The form of the values produced by a user-defined derived-type output routine invoked during list-directed output is specified by the invoked routine. This form need not be compatible with list-directed input."

[246:4] After "and logical values" add ", and output produced by user-defined derived-type output"

[246:7] Add at the end of the paragraph
"The form of the output produced by a user-defined derived-type output routine invoked during namelist output is specified by the invoked routine. This form need not be compatible with namelist input."

SUBMITTED BY: Dick Hendrickson

```
HISTORY: 07-275      m181  F03/0101 Submitted
         07-275r2    m181  Passed by J3 meeting
         07-279/321  m182  Passed as changed by J3 letter ballot #14
         08-155      m184  Passed by WG5 ballot #4 N1711-N1721
```

NUMBER: F03/0104
TITLE: Deallocation and finalization of bounds-remapped pointers
KEYWORDS: deallocate, finalization, bounds-remapping, pointer
DEFECT TYPE: Interpretation
STATUS: Passed by WG5 ballot

INTRODUCTION:

Consider the following example assuming a derived type of X is declared previously and made accessible to the current scoping unit,

```
   type(X), pointer :: a(:), b(:,:)

   allocate (a(100))
   b(1:10, 1:10) => a

   DEALLOCATE (b)
```

QUESTION:

(a) Is DEALLOCATE (b) in the example intended to be standard conforming?

(b) If the answer to (a) is yes, and also assume type X has finalizers of both rank-one and rank-two, then which finalizer should be invoked by the DEALLOCATE statement.

ANSWER:

(a) Yes, the example is intended to be standard conforming. The deallocation of pointer b should be executed successfully.

(b) The Standard is clear about how the finalizations are processed in this case. In 4.5.5.1, the first step in invoking the appropriate final subroutine requires a finalizer matching the rank of the entity being finalized. In this case, object b is being finalized and therefore the rank-two final subroutine of type X will be invoked with object b as the actual argument.

EDITS:

None.

SUBMITTED BY: Jim Xia

```
HISTORY: 07-299    m182  F03/0104 Submitted; Passed by J3 meeting
        08-133r2  m183  Passed by letter ballot #15 08-101
        08-164    m184  Passed by WG5 ballot N1722-N1726
```

NUMBER: F03/0106
TITLE: Inquire by unit inconsistencies
KEYWORDS: inquire, unit, not connected
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

There are many things that can be inquired about, such as ACTION or READ, that are purely file or connection properties. In some cases, such as ACTION, the specifier description includes "If there is no connection [the result is] the value UNDEFINED" or similar words. In other cases, such as READ, there seems to be a tacit assumption that there is a file connected to the unit. The descriptions refer to "the file" and don't specify a result if there is no connection. In most cases, there is a phrase like "if the processor is unable to determine if the

file ... [the result is] {UNDEFINED, UNKNOWN, -1, etc.}".

Question 1) Are the inquire specifiers DIRECT, ENCODING, FORMATTED, NAMED, NEXTREC, NUMBER, POS, READ, READWRITE, SEQUENTIAL, SIZE, STREAM, UNFORMATTED, and WRITE allowed in an INQUIRE by unit when there is no file connected to the unit?

Question 2) If so, should the descriptions for the above specifiers be clarified by adding phrases such as "if there is no file specified or connected" to the "UNKNOWN" result descriptions?

ANSWER:

Question 1) Yes. In an inquiry by unit, the specifiers have little meaning when there is no file connected to the unit. However, the standard should specify the results.

Question 2) Yes, edits are supplied below.

Note: 9.9.1.15 NAMED= [213:10] needs no edit; the value will be false if the unit specified by UNIT= is not connected to a file

EDITS:

9.9.1.8 DIRECT= At [212:15], add to the end of the last sentence "or if the unit specified by UNIT= is not connected to a file"

9.9.1.9 ENCODING= At [212:21], after "file" insert "or if the unit specified by UNIT= is not connected to a file"

9.9.1.12 FORMATTED= At [212:36], add to the end of the last sentence "or if the unit specified by UNIT= is not connected to a file"

9.9.1.16 NEXTREC= At [213:15], change "or if" to ", if" and At [213:16], after "condition" insert ", or if the unit specified by UNIT= is not connected to a file"

9.9.1.17 NUMBER= Replace [213:20-21] with "Execution of an INQUIRE by file statement causes the <scalar-int-variable> in the NUMBER= specifier to be assigned the

value of the external unit number of the unit that is connected
to the file. If there is no unit connected to the file, the
value -1 is assigned. Execution of an INQUIRE by unit statement
causes the <scalar-int-variable> to be assigned the value specified
by UNIT=."

9.9.1.21 POS= At [214:19], change "or if" to ", if" and
At [214:20], after "conditions" insert ", or if the
unit specified by UNIT= is not connected to a file"

9.9.1.23 READ= At [215:2], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

9.9.1.24 READWRITE= At [215:7], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

9.9.1.27 SEQUENTIAL= At [215:26], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

9.9.1.29 SIZE= At [215:34], after "determined" insert "or if the unit
specified by UNIT= is not connected to a file"

9.9.1.30 STREAM= At [216:5], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

9.9.1.31 UNFORMATTED= At [216:10], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

9.9.1.32 WRITE= At [216:15], add to the end of the last sentence
"or if the unit specified by UNIT= is not connected to a file"

SUBMITTED BY: Dick Hendrickson

HISTORY: 07-309   m182 F03/0106 Submitted
        07-309r1 m182 Answer based on 07-310; Passed by J3 meeting
        08-133r2 m183 Passed letter ballot #15 08-101
        08-164   m184 Passed WG5 ballot #5 N1722-N1726
        N1727   m184 Note edit changes in F2003 Corrigendum 3

In the edit to 9.9.1.17, N1727 puts "scalar-int-variable" in italics,
ie, <scalar-int-variable>.

NUMBER: F03/0107
TITLE: Are the IEEE_* elemental routines required
KEYWORDS: IEEE, elemental routines
DEFECT TYPE: Erratum
STATUS: Passed by WG5 ballot

QUESTION:

The descriptions for all of the IEEE elemental intrinsics listed in 14.9 say something like "shall not be invoked if IEEE_SUPPORT_DATATYPE(X) is false".

I believe this was to allow a careful programmer to do something like

```
    if (IEEE_SUPPORT_DATATYPE(x)) then
        x = IEEE_SCALB(x,2)
    else
        x = x*4
    endif
```

and program around partial IEEE support.

But 14.9.2 says that "IEEE_ARITHMETIC contains the following [routines] for which IEEE_SUPPORT_DATATYPE(X) [is] true"

I'd read that as saying the functions aren't there for cases where IEEE_SUPPORT_DATATYPE is false. But, then, there is no way to program around their absence. The example above will fail at load time because IEEE_SCALB is absent.

If a processor provides the IEEE_ARITHMETIC module must it provide versions of all of the intrinsics for all of the available datatypes, including those for which IEEE_SUPPORT_DATATYPE() is false?

ANSWER:

Yes, edits are provided to make this clear.

DISCUSSION: It was intended that the above coding snippet could be used by a careful programmer to program portably for processors which have varying degrees of IEEE support. This might require processors

to provide some stub function for each routine and for each non-IEEE datatype they support. If a program invokes one of the stub routines, it is a run-time programming error. Nevertheless, a program which has references to the routines, but doesn't invoke them, must load and execute.

EDITS:

In the first paragraph of subclause 14.9.2 [370:8-9] Replace

"for reals X and Y for which IEEE_SUPPORT_DATATYPE(X) and IEEE_SUPPORT_DATATYPE(Y) are true"

with

"for all reals X and Y"

NOTE:

The following note should be inserted at the end of the section on IEEE arithmetic in a future standard:

"The standard requires that code such as

```
if (IEEE_SUPPORT_DATATYPE(x)) then
      x = IEEE_SCALB(x,2)
else
      x = x*4
endif
```

be executable. The elemental functions in the IEEE_ARITHMETIC module (14.9.2) must exist for all real kinds supported by the processor, even if IEEE_SUPPORT_DATATYPE returns false for some kinds. However, if IEEE_SUPPORT_DATATYPE returns false for a particular kind, these functions must not be invoked with arguments of that kind. This allows a careful programmer to write programs that work on processors that do not support IEEE arithmetic for all real kinds.

The processor might provide stub routines which allow the program to link and execute, but which will abort if they are invoked."

SUBMITTED BY: Dick Hendrickson

HISTORY: 07-312   m182 F03/0107 Submitted
        07-312r2 m182 Passed by J3 meeting
        08-133r2 m183 Passed letter ballot #15 08-101
        08-164   m184 Passed WG5 ballot #5 N1722-N1726

NUMBER: F03/0108
TITLE: Is IEEE_SUPPORT_NAN consistent with the other IEEE_SUPPORT
      functions
KEYWORDS: IEEE_SUPPORT_NAN, IEEE support functions
DEFECT TYPE: Clarification
STATUS: Passed by WG5 ballot

QUESTION:

The restriction of IEEE_IS_NAN requires that IEEE_SUPPORT_NAN returns the value true. The restrictions for the similar functions IEEE_IS_{FINITE, NEGATIVE, and NORMAL} all require that IEEE_SUPPORT_DATATYPE be true. This is a much stronger restriction.

Should IEEE_SUPPORT_NAN also require that IEEE_SUPPORT_DATATYPE return true?

ANSWER:

No. The IEEE_SUPPORT_NAN restriction is weaker than requiring IEEE_SUPPORT_DATATYPE but IEEE_SUPPORT_NAN is sufficient. IEEE_SUPPORT_DATATYPE is used in IEEE_IS_FINITE, IEEE_IS_NEGATIVE, and IEEE_IS_NORMAL because there are no IEEE_SUPPORT_* inquiry functions to query support for finite, negative, or normal. IEEE_SUPPORT_INF asks about infinities not finites and IEEE_SUPPORT_DENORMAL only covers denormals and not the other non-finites (NaNs and Infinities).

EDITS:

None.

SUBMITTED BY: Dick Hendrickson

HISTORY: 07-328   m182 F03/0108 Submitted

ISO/IEC JTC 1/SC 22 N 4342

2008-05-19

07-328r2 m182 Passed by J3 meeting
08-133r2 m183 Passed letter ballot #15 08-101
08-164   m184 Passed WG5 ballot #5 N1722-N1726
