Subject: Unresolved issue 362
From: John Reid, as revised by Van Snyder

1 Introduction

Subgroup suggests one change to 02-178, shown below on page 2. For the convenience of members, we also show how the TEXisms in 02-178 will be rendered.

2 Original text of 02-178, with modification indicated

```
I would like to acknowledge the help of Malcolm Cohen, Richard Maine,
Dick Hendrickson, and Dan Nagle in constructing this paper.

Unresolved issue 362 is

   The comments apply to the IEEE SUPPORT * functions, and possibly
   others.

   There are several problems here

   A) IEEE_SUPPORT_DENORMAL can't be a generic function since the rules
   do not allow an optional argument to be the generic decider. Do we
   need to have 2 cases in the header IEEE_SUPPORT_DENORMAL() and
   IEEE_SUPPORT_DENORMAL(X)? Does that match the intent?

   B) How should the restriction be read? It's clear that
   IEEE_SUPPORT_DENORMAL(3.14) can't be invoked unless
   IEEE_SUPPORT_DATATYPE(3.14) is true; but can IEEE_SUPPORT_DENORMAL()
   be invoked if IEEE_SUPPORT_DATATYPE() is false? The restriction
   could be read either way. What was the intent?

   C) How should this be treated if X is an optional argument in the
   invoker?
        Call z ()
        ...
        Subroutine z(X)
        Real, optional :: X
        Print *, IEEE_SUPPORT_DENORMAL(X)

   Is this illegal because it's a reference to a not-present optional
   argument?  Is it the same as Print *, IEEE SUPPORT DENORMAL(3.14)
   because only the properties of X apply?  Is it the same as Print *,
   IEEE SUPPORT DENORMAL() because the argument isn't present and
   unpresentness flows down the call chain?

I think the answer to A) should be yes. The sensible interpretation of
the code C) is that the inquiry is about a compile-time property of X
```

that is independent of its presence. I do not think anyone foresaw this difficulty when we were writing the TR. Furthermore (see 02-161r1), similar considerations apply to several existing inquiry functions, such as KIND. Malcolm Cohen has suggested edits to pp. 270-271 to cover the general case (see below). To make this cover the IEEE inquiries, we need to to allow an IEEE inquiry function to be an initialization expression, a derirable change anyway. A simple way to do this is to extend the definition of specification inquiry so that 7.1.7 (8) applies to IEEE inquiry functions.

B) highlights an ambiguity that needs resolution anyway. It would be more friendly to the user to allow the case where IEEE_SUPPORT_DATATYPE is false and return false in this case. This is just a minor extension from what is in the TR.

Edits:

```
127:41. Set 'specification inquiry' in bold and add an index reference
      to it.

128:5.  Delete 'or'
128:6.  Change '.' to ', or'
128:6+. Add
    (7) an IEEE inquiry function (14.8.1).
```

Begin Revised Edit

```
270:32. After 'PRESENT intrinsic function' add 'or as an argument of a
      function reference that meets the requirements of (7) and (8) in
      7.1.7'.
```

End of Revised Edit

```
270:38-40. Change
     If it is a pointer, it shall not be allocated, deallocated,
     nullified, pointer-assigned, or supplied as an actual argument
     corresponding to a nonpointer dummy argument other than as the
     argument of the PRESENT intrinsic function.
   to
     If it is a pointer, it shall not be allocated, deallocated,
     nullified, pointer-assigned, or supplied as an actual argument
     corresponding to an optional nonpointer dummy argument.
   This change simply factors OUT the cases already covered by the ordinary
   one (lines 31-32).

270:41-42. Change
     If it is allocatable, it shall not be allocated, deallocated, or
     supplied as an actual argument corresponding to a nonallocatable
     dummy argument other than as the argument of the PRESENT intrinsic
     function.
   to
     If it is allocatable, it shall not be allocated, deallocated, or
     supplied as an actual argument corresponding to an optional
     nonallocatable dummy argument.

271:1. Change
     If it has type parameters, they shall not be inquired about.
   to
     If it has nonkind type parameters, they shall not be the
     subject of an inquiry.

371:23. Change the title to
            IEEE_SUPPORT_DATATYPE() or IEEE_SUPPORT_DATATYPE(X)

Elsewhere. Make a similar change on each of the lines 372:1, 373:1,
    373:14, 374:11, 374:23, 374:37, 375:12,  375:27,  376:4.

371:26. Delete '(optional)'.

Elsewhere. Make the same change on each of the lines 372:4, 373:5,
    373:19, 374:14, 374:27, 375:3, 375:17,  375:31,  376:8.

372:5-6. Delete these lines and the J3 internal note.

Elsewhere. Delete the lines  373:6-7, 373:20-21, 374:15-16, 374:28-29,
    375:4-5, 375:18-19,  375:32-33.

372:8-11. Replace by

\resvalue{}

\begin{incase}
  \item IEEE\_SUPPORT\_DENORMAL(X) has the value true if
  IEEE\_SUPPORT\_DATATYPE(X) has the value true and the processor
  supports arithmetic operations and assignments with denormalized
  numbers (biased exponent $e = 0$ and fraction $f \neq 0$, see section
  3.2 of the IEEE standard) for real variables of the same kind type
  parameter as X; otherwise, it has the value false.

  \item IEEE\_SUPPORT\_DENORMAL() has the value true if and only if
  IEEE\_SUPPORT\_DENORMAL(X) has the value true for all real X.
\end{incase}
```

This is how the above TEXism compiles:

Result Value.

Case (i): IEEE_SUPPORT_DENORMAL(X) has the value true if IEEE_SUPPORT_DATATYPE(X) has the value true and the processor supports arithmetic operations and assignments with denormalized numbers (biased exponent $e = 0$ and fraction $f \neq 0$, see section 3.2 of the IEEE standard) for real variables of the same kind type parameter as X; otherwise, it has the value false.

Case (ii): IEEE_SUPPORT_DENORMAL() has the value true if and only if IEEE_SUPPORT_DENORMAL(X) has the value true for all real X.

```
Elsewhere. Make similar changes to lines 373:9-11, 373:23-25,
   374:18-20, 374:31-34, 375:7-9, 375:21-24, 375:35-376:1.

376:11-24. Change to

\begin{incase}

  \item IEEE\_SUPPORT\_STANDARD(X) has the value true if the results
  of all the functions IEEE\_SUPPORT\_DATATYPE(X),
  IEEE\_SUPPORT\_DENORMAL(X), IEEE\_SUPPORT\_DIVIDE(X),
  IEEE\_SUPPORT\_FLAG(FLAG,X) for valid FLAG,
  IEEE\_SUPPORT\_HALTING(FLAG) for valid FLAG, IEEE\_SUPPORT\_INF(X),
  IEEE\_SUPPORT\_NAN(X), IEEE\_SUPPORT\_ROUNDING(ROUND\_VALUE,X) for
  valid ROUND\_VALUE, and IEEE\_SUPPORT\_SQRT(X) are all true;
  otherwise, the result has the value false.

  \item IEEE\_SUPPORT\_STANDARD() has the value true if and only if
  IEEE\_SUPPORT\_STANDARD(X) has the value true for all real X.
\end{incase}
```

This is how the above TEXism compiles:

Result Value.

Case (i): IEEE_SUPPORT_STANDARD(X) has the value true if the results of all the functions IEEE_SUPPORT_DATATYPE(X), IEEE_SUPPORT_DENORMAL(X), IEEE_SUPPORT_DIVIDE(X), IEEE_SUPPORT_FLAG(FLAG,X) for valid FLAG, IEEE_SUPPORT_HALTING(FLAG) for valid FLAG, IEEE_SUPPORT_INF(X), IEEE_SUPPORT_NAN(X), IEEE_SUPPORT_ROUNDING(ROUND_VALUE,X) for valid ROUND_VALUE, and IEEE_SUPPORT_SQRT(X) are all true; otherwise, the result has the value false.

Case (ii): IEEE_SUPPORT_STANDARD() has the value true if and only if IEEE_SUPPORT_STANDARD(X) has the value true for all real X.