ISO/IEC JTC1/SC22/WG5 N1335

To: WG5
From: DIN Working Group for Fortran

Some Technical Comments as Input to WG5 Meeting at Cadarache

Among many other technical issues, DIN discussed I18n as well as J3's answer to our previous remarks on DTIO. Concerning these issues, we have the following questions/remarks:

Questions concerning Internationalization:

According to N1320 and J3/98-199r1, there is some support (inquiry intrinsic for the character KIND value) planned for the ISO 10646 character set. Why is there no similar support for the 15 variants of ISO 8859?

In fact, ISO 8859-15 (8-Bit character set) will be the most commonly used character set because it includes the EURO symbol, so some support would be helpful. Or is it a general policy to support ISO 10646 only?

Maybe this question should be answered by a character code expert rather than within our Fortran community. Can we get any help on this?

Remarks concerning Derived-Type I/O:

The following remarks refer to the X3J3 answer to the DIN comments (J3/99-111):

– "New requirements at a late date"?

Our most essential requirement - support of full nesting of encapsulated derived types - is neither new nor controversial: we just stated it again clearly to stress the reasons for our dissatisfaction.

We think that this requirement is now adequately supported by the current specifications for the I/O handler routines (which now allow fully recursive calls of derived-type handlers), but not yet adequately supported by the format syntax which should also reflect the recursive nature of derived types - although J3/98-189 was a considerable improvement over previous versions (see details below).

The other requirements - esp. concerning the support of the format handling which needs to be programmed within I/O handlers - have indeed never been stated before, but were born (as reasoning behind our dissatisfaction) when we discussed the current proposal and imagined how derived type I/O handlers must be programmed in Fortran2000: we found it not very user-friendly that

* with intrinsic I/O, all standard format editing features are automatically available for all derived-type components,
* whenever a single component of a derived type needs to be treated differently, the whole formatting has to be done completely by the I/O handler and none of the format editing features are available any more. The derived-type programmer then has to re-invent some kind of format syntax for his derived type components (as contents of the DT"..." string) and implement the whole format analysis himself without any Fortran support, although every Fortran processor does have the format analysis features - they are merely not available to the user.

So we believe that the language should give some support in order to facilitate the task of writing an I/O handler (see details below).

– Syntax of the derived-type format edit descriptor:

The enhancements described in J3/98-189 are certainly a large improvement, since the DT"..." syntax now allows some nesting of derived type edit descriptors.

The present syntax DT".." however creates the problem that - when derived type edit descriptors are nested in a straightforward manner - the "-delimiters have to be doubled at each level:

Example:

DT"2I5,A10,A,2F10,DT""ROUND_TO_LOW,DT""""X"""",'abcde',I5""2A6"

So we still would prefer some syntax which allows nesting of DT-specifiers in an obvious and easy way (as e.g. parentheses) at least as alternate syntax to the current draft (we cannot see any contradiction with N1322 or earlier WG5 directions in this).

– Help for parsing standard format strings:

It does not seem very complicated to add a standard module with e.g. the following 3 subroutines/interfaces:

* OPEN_FORMAT_PARSING (Input: Format-String, Output: Handle)
* GET_NEXT_FORMAT_ITEM (Input: Handle, Output: (1) String describing exactly one edit descriptor - without repeat specifier - to be used in internal I/O ; (2) EOString indicator
* CLOSE_FORMAT_PARSING (Input: Handle).

Such functions must exist in every runtime library (they are just not accessible) and would satisfy the requirement described above.

– Recursive I/O:

Formally, it may be true that "Reentrant I/O" is a different issue than derived-type I/O. So far, however, the disallowance of reentrant I/O could only be felt in function calls within I/O lists, and there was an easy workaround by placing function results first into intermediate variables before starting the I/O statement.

With the introduction of derived type I/O, however, the problem becomes much more urgent since user-defined routines will intercept I/O much more often and with much more complicated activities than before; such routines may require initialization, user dialogs in

case of errors, etc. We think that there really *is* an intimate relationship between reentrant I/O and derived-type I/O from the requirement side.

– INQUIRE by IOLENGTH:

In principle, we agree with not providing INQUIRE by IOLENGTH for derived-type I/O.

We could not see any edits for this, though (maybe we just missed them). What happens if a Fortran program contains INQUIRE by IOLENGTH with some derived type (without POINTER component) in the output list and where some I/O handler has been defined? (Hopefully, this will be forbidden. With the current wording, some IOLENGTH value should be returned - which value?)