To: J3 J3/22-208
From: R. Bleikamp & JOR
Subject: List and status of JOR's (potential) work items for F202y
Date: 2022-October-25
Reference: https://j3-fortran.org/forum/viewtopic.php?f=9&t=106&p=525#p525

Items recommended by JoR for inclusion in F202y

Work item name	Status / recent activity	References
Pre-processor	Started survey of existing implementations Fortran friendly support. See next table line below for current plans:	See https://j3-fortran.org/doc/year/22/22-186.pdf (or newest revision thereof)
Actively attempting to decide how Fortran friendly the preprocessor should be, and pondering other over-arching decisions. Possible steps (given an hours thought or so): 1. Understand what is being used in real codes. We know lots of packages we could examine for cpp-like directives, and we can find out what kinds of features they appear to be relying on. J3 could offer some especially important codes here. 2. Flesh out the potential requirement set, based both on what we learn above, and the features that have been proposed. 3. Flesh out the potential phases of the preprocessor, in a kind of functional programming way. That is, have some view of a shell-like pipeline: collecting-input | process-directives | expand-macros | reformat-output. Again, it doesn't have to be implemented this way, but it helps us describe the activities somewhat independently. 4. Map the requirement set to the phases. 5. Outline the consequences to the requirements to the implementation of the phases. There will be both plusses and minuses in terms of what the users get, and what the implementers have to do. JOR will be soliciting input from J3 along the way (discussion board, or a dedicated email list, or ?). Volunteers may be needed too.		
change F.P. model to be IEEE 754	Not started. Easy to do, low priority.	https://github.com/j3-fortran/fortran_proposals/issues/268
Remove some processor dependencies from Annex A	Not started.	
Immutable values	Not started.	https://github.com/j3-fortran/fortran_proposals/issues/221
Program specified default kinds for constants and intrinsic types.	Adopted by **DATA**	

Items JoR is undecided about for inclusion in F202y

Work item name	Status / recent activity	References
ASSERT	Not started. Need to evaluate Magne's comments.	https://github.com/j3-fortran/fortran_p... /issues/70; New details - viewtopic.php?f=9&t=113
scan/prefix sum	Not started. Need use cases, and possibly a volunteer to drive this item.	https://github.com/j3-fortran/fortran_proposals/issues/273
scan clause for do concurrent reduce	Not started. Need use cases, and possibly a volunteer to drive this item.	https://github.com/j3-fortran/fortran_proposals/issues/224
Disallow use of specific new F202y features in a program unit that uses any deprecated/deleted features	JoR is undecided if this is a desirable feature.	https://github.com/j3-fortran/fortran_proposals/issues/280

Work items that JoR is NOT planning on recommending for inclusion in F202y. Interested parties should contact JoR (rich@bleikamp.net) to arrange a time to present their views to the subgroup.

Work item name	Status / recent activity	References
Surprising results for UBOUND and LBOUND when arg has zero extent	JoR is leaning towards dropping this feature. A compelling use case would change our mind.	https://github.com/j3-fortran/fortran_proposals/issues/254
log2: just log2, or survey math.h and see what other base 2 intrinsics are missing from fortran.	We would like a compelling use case (HPC related) before we would tackle this feature. How many other vendors provide this now? Is support easily available thru C interop?	https://github.com/j3-fortran/fortran_proposals/issues/222
intrinsic to return the name of your caller, current procedure name, ...	JoR decided not to pursue this. Again, a compelling use case might change our mind. Overhead and possibly requiring debugging info is a concern. Seems like a companion processor (debugger) can do some of this.	https://github.com/j3-fortran/fortran_proposals/issues/180
Deprecate D format edit descriptor	the D edit descriptor serves no useful purpose anymore. But removing it from the standard may not be trivial.	https://github.com/j3-fortran/fortran_proposals/issues/226
constexpr	JoR needs to research this more. We want to know when C++ initializes constexprs. JoR would like to see a compelling use case. Seems expensive to implement if initialization happens at compile time. **Until JoR determines this is easier than we think to implement, this feature will remain in the Not Recommended catagory.**	https://github.com/j3-fortran/fortran_p ... issues/214https://fortran-lang.discourse.group/t/ ... fortranfan and from a DATA subgroup item https://github.com/j3-fortran/fortran_p ... issues/253
comments in list directed input	similar to namelist, but undelimited character input data may be a problem / incompatibility)	**Van's email description:** see the next table row.
	We allow comments in namelist input. In list-directed input, one can put comments after the last item desired by putting them after the slash that terminates the input. If one is reading several arrays, say one array per line, with one list-directed input statement, one cannot put a slash and comment on each line because that terminates the input. Would there be a problem to allow comments in list-directed input, beginning with "!" as in namelist input? JOR may reconsider if no backwards compatibility issues exist.	