OAK RIDGE National Laboratory

Fortran Standardization Process and The Road to F202Y

FortranCon 2025, November 4-5, 2025

Reuben D. Budiardja
Chair, INCITS/Fortran (J3) Technical Committee
Group Leader, Advanced Computing for Nuclear, Particle, and Astrophysics
National Center for Computational Sciences
Oak Ridge National Laboratory

ORNL is managed by UT-Battelle LLC for the US Department of Energy

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

U.S. DEPARTMENT OF ENERGY

Outline

- Fortran standardization process
- Development work for “F202Y”
- Some feature preview of Fortran 202Y

Fortran Standardization Efforts

- Standards are published by International Organization for Standardization (ISO); ISO JTC1/SC22/**WG5** is the ISO working group for Fortran
  - consists of multiple "national bodies" (NB) **https://wg5-fortran.org/**
  - active NBs: Japan, U.K., Germany, U.S.

- INCITS is the U.S. NB, serving as Technical Advisory Group to the JTC1
  - INCITS/Fortran Technical Committee is responsible for the technical development of the standards https://www.incits.org/committees/pl22.3
  - Formerly known as ANSI X3J3, a.k.a **"J3"**

https://j3-fortran.org/

- ***In practice***, WG5 provides general directions & advice, while INCITS/Fortran (J3) does the technical work

Fortran Standardization Efforts

- INCITS/Fortran (J3) members are
  - application developers, compiler writers, libraries & tools developers, academics, anyone with passions for & interests in Fortran
  - US National labs, federal agencies, vendors / companies, Universities, other organizations, individuals
  - Significant overlap between WG5 and J3 members (including non-US members)
- Development are done openly (meetings are members-only)
  - Three meetings per year (1-2 virtual); at least one face-to-face, held adjacently with WG5 meeting
  - j3-fortran.org, wg5-fortran.org (credit to Steve Lionel aka "Dr. Fortran" for website donation & maintenance)
  - https://github.com/j3-fortran/
- Contribute!
  - Join your NB
  - Join INCITS/Fortran
  - Join the J3 mailing list

Fortran Standardization Efforts

- INCITS/Fortran (J3) and WG5 works by **consensus** building
  - Contrary views are seriously considered. Time were taken to resolve objections.
  - "What is needed to change a 'no' vote to 'yes'?" Results are typically better for this.

- j3-fortran.org
  - Papers are organized by meeting and by year
  - Papers document the developments of a feature (use case, rationale / discussion, requirements, specifications, edits)
  - Papers are discussed and "passed" (by voting) at J3 meetings
  - "Standing document" are long-lived documents spanning multiple meetings "007" is the committee working draft; "010" is Workplan (currently F202Y)
- wg5-fortran.org
  - the working home of ISO WG5
  - document WG-5 yearly meeting minutes & decisions

Fortran Standardization Efforts

- Performance and numerical centric
  - work hard such that feature specifications does not hamper optimization
- Continues evolution to accommodate development in scientific computing
  - while trying to avoid chasing passing fads
- Backward compatibility
  - previous standards is proper subset of current standards
  - with some features deemed obsolescent

Fortran 202Y (Goal is Y==8)

- The committee is working on the next standard, informally called F202Y
- An initial list of features was approved by NBs at 2023 WG5 ISO Fortran Meeting (N2222), including
  - Generic programming with Template
  - Auto-Generic subprograms
  - Standardize Fortran preprocessor
  - Improved rank-independent functionality
  - Feature list updated in N2249
- J3 is actively developing these features

Fortran 2028 WG5 Strategic Plan (WG5 N2248)

ISO/IEC JTC1/SC22/WG5 N2248

Revised strategic plans for WG5

Steve Lionel, June 27, 2025

1. Interpretations

Processing of interpretations against ISO/IEC 1539-1:2023
will continue, and Corrigenda constructed as appropriate.

2. Revision of ISO/IEC 1539-1:2023 (Fortran 2023). This schedule
has not changed since 2024.

Final choice of technical content	2024-06
CD constructed	2026-06
CD ballot initiated	2026-07
CD ballot results available	2026-09
DIS constructed	2027-02
DIS ballot initiated	2027-03
DIS ballot results available	2027-09
DIS revised	2027-11
FDIS constructed	2028-02
FDIS ballot initiated	2028-05
FDIS ballot results available	2028-08
Standard published	2028-10

F202Y Current Status (j3-fortran.org standing doc 010)

F202Y Workplan

Live doc link

Label	Short Desc	Level (see 24-167)	Background / Use Cases	Subgroup	Rationale / Discussion Papers	Requirements	Specifications	Syntax	Edits
JP01	Include generic subprograms described in N2217	5	N2217 (WG5) 25-102 25-150	Data	24-140r2 (info, straw votes) 24-184r1 24-188r1 24-180 24-181 25-115	23-223r2 24-147r1 25-163r2	23-244r1 24-148r1 25-103 25-128 25-129 25-164r2	24-139r2 25-103 25-128 25-168r2	
US01	Obsolete default implicit typing	2	23-177	JOR					25-192r2
US02	Obsolete D format edit descriptor	2	23-178	JOR					25-113
US03	Add note that the real model is not IEEE 754	1	23-180	JOR					25-193r1
US04	Asynchronous Collective Subroutines		N2245 (WG5)	HPC		25-162r2	25-165r1	25-166r2	
US05	Add extended floating-point types defined in ISO C23 to ISO_C_BINDING	3	23-176	HPC		25-189r1	25-189r1		
US06	Provide a mechanism to specify global binding name for non C-interoperable	3	23-201	HPC					
US07	Improve rank-independent functionality	4	23-184r1	Data / JOR					
US08	Polymorphic PURE function results	4	23-186	Data		25-116r1	25-116r1 25-178	25-116r1 25-178	25-153r1
US09	Allow I/O of enumeration type names	4	23-151r1	JOR					25-126r1
US10	Define a standard Fortran preprocessor	6	23-192r1, 24-108, 24-109	FPP		25-114r2	25-142r2		
US11	Provide intrinsics for source location	3	23-193r1	JOR					
US12	Add maximum rank/corank constants to ISO_FORTRAN_ENV	3	23-194	Data (JP01)					
US13	Remote access to module entities	3	23-196r1	Data	25-119r1				
US14	Provide scoped access to enumeration enumerators	4	23-197	Data		25-180	25-180	25-180	
US15	Readonly pointers: a new attribute be application to pointers that determined whether the target of the pointer cannot be changed trhough the pointer	4	23-198	Data		25-179r1	25-179r1	25-182r1	25-197r2
US16	Provide a facility to specify the default kinds within the program unit	3	23-199	Data					
US17	Generic programming using templates	8	23-148, 23-103	Generic	23-211r1 23-212r2 23-213r2 23-214r3 23-215r1 25-158r1	24-105r3	24-105r3	24-125r5 24-126r4 24-127r4 24-159r4	25-109r2 25-135r2 25-136r2 25-172r1 25-173r1 25-174r2
US18	Allow Polymorphism in Coarrays	4	23-217	HPC (?)					
US19	Add more math functions from IEEE-754	4	23-234r2	JOR		25-143	25-143		
US20	Add Intrinsic and collective subroutines for prefix operations (formerly "SCAN" and "CO_SCAN")	4	23-235r2, 24-157	HPC + JOR	25-195r1	25-144r1 (co_*) 25-145r1 (local)	25-177r1 (co_*) 25-186r2 (sum) 25-196r1 (reduce)	25-177r1 (co_*) 25-186r2 (sum) 25-196r1 (reduce)	
US23/DIN4	Add generic processing of assumed-rank object	5	24-136r1	Data	24-169	24-171 24-172	24-183r1	24-183r1	24-183r1
US24	Add rank-independent looping	4	24-143	Data + JOR					
US25	Add ability to interpret complex as real and vice versa	5	24-129	Data		24-173	25-188r1	25-188r1	
DIN1	Add execution of collective procedures on a specified team	4	N2230 (WG5) 25-125r1	HPC		25-127r1	25-127r1	25-127r1	25-127r1
DIN3a	Add support for atomic operations in local memory	5	N2230 (WG5)	HPC					

OAK RIDGE National Laboratory

Fortran 202Y Preview

F202Y **Preview**: Generic Programming

Details are subject to change

A motivating _AXPY example:

```
subroutine axpy(a, x, y)
  real, intent(in) :: a
  real, intent(in) :: x(:)
  real, intent(inout) :: y(:)
  y = a*x + y
end subroutine axpy
```

In Fortran 2023, supporting AXPY for different data types require user to write similar subroutines with suitable interface for each supported data types.

F202Y **Preview**: Auto-Generic Procedures

Details are subject to change

Original proposed feature with use-cases: WG5 N2217

There shall be a mechanism for defining an auto-generic procedure, with a single procedure body rather than a set of specific procedures.

- have *generic* name but no *specific* name
- may have zero, one, or more "generic" dummy arguments
- an anonymous set of specific procedures is implicitly defined, one for every combinations of **type**, **kind-type** parameter, and **rank** → "generic combinations"
- rest of requirements: 24-147r1

Provide a relatively simple way to leverage existing compile-time polymorphism ("genericity") already existed in the language.

F202Y **Preview**: Auto-Generic Procedures - AXPY Example

Details are subject to change

```
generic subroutine axpy(a, x, y)
  type(real([real32, real64, real128]), &
     integer([integer_kinds]), intent(in) :: a
  typeof(a), dimension(:), intent(in) :: x
  typeof(a), rank(rank(x)), intent(inout) :: y
  …
end subroutine axpy
```

generic dummy argument

generic-dependent

credit: Malcolm Cohen (NAG), Hidetoshi Iwashita, John Reid

F202Y **Preview**: Auto-Generic - Kind & Rank Genericity Examples

Details are subject to change

```
generic subroutine lift(x,y)
  type(integer([int32,int64]), real), rank(1:2), allocatable :: x
  type(integer([int32,int64]), real), rank(1:2), allocatable :: y
  typeof(x), rank(rank(y)), allocatable :: z
  !-- Defines 36 specific procedures with generic name “lift”
end subroutine
```

```
generic subroutine lift(x,y)
  type(integer([int32,int64]), real), rank(1:2), allocatable :: x
  typeof(x), rank(rank(x)), allocatable :: y
  typeof(x),rank(rank(y)),allocatable :: z
  !-- Defines 6 specific procedures with generic name “lift”
end subroutine
```

F202Y **Preview**: Auto-Generic - Rank Genericity Examples

Details are subject to change

```
generic function fun(x) result(y)
  type(type1), rank(0:7) :: x
  typeof(x), rank(rank(x)) :: y

  select generic rank (x)
  rank (0)
    !! code if x is a scalar
  rank (1:3)
    !! code if x is an array of 1 to 3 dimensions
  rank default
    !! code if x is an array of 4 to 7 dimensions
  end select
end function fun
```

F202Y **Preview**: Auto-Generic - Type Genericity Examples

Details are subject to change

```
module example
  interface operator(.myop.)
    procedure s ! all of the specific procedures of s.
  end interface
contains
  generic function s(a,b) result(c)
    use iso_fortran_env
    type(real,complex), intent(in), rank(1:max_rank()) :: a
    typeof(a),rank(rank(a)), intent(in) :: b
    typeof(a), rank(rank(a)) :: c, temp
    …
    select generic type (a)
      declared type is (real)
         temp = temp * (1-b)
      declared type is (complex)
        ! just this once, we want the conjugate.
        temp = temp * (1-conjg(b))
    end select
    …
    c = temp
  end function
end module
```

F202Y **Preview**: Generic Programming with Template

Details are subject to change

A motivating _AXPY example:

```
subroutine axpy(a, x, y)
  real, intent(in) :: a
  real, intent(in) :: x(:)
  real, intent(inout) :: y(:)
  y = a*x + y
end subroutine axpy
```

In Fortran 2023, supporting AXPY for different data types require user to write similar subroutines with suitable interface for each supported data types.

F202Y **Preview**: Generic Programming with Templates

Details are subject to change

```
requirement bin_op { T, Op }
  deferred type :: T
  deferred interface
    elemental function Op ( a, b )
      type ( T ), intent ( in ) :: a, b
      type ( T ) :: op
    end function
  end interface
end requirement
```

```
integer, parameter :: SP = kind ( 1.0 ), DP = kind ( 1.d0 )

real(sp) :: f_A, f_X ( 10 ), f_Y ( 10 )
real(dp) :: d_A, d_X ( 10 ), d_Y ( 10 )
integer  :: i_A, i_X ( 10 ), i_Y ( 10 )

instantiate axpy_tmpl { real(sp), operator(+), operator(*) }
instantiate axpy_tmpl { real(dp), operator(+), operator(*) }
instantiate axpy_tmpl { integer, operator(+), operator(*) }

...

call axpy ( f_A, f_X, f_Y )
call axpy ( d_A, d_X, d_Y )
call axpy ( i_A, i_X, i_Y )
```

```
template axpy_tmpl { T, plus, times }

  private
  public :: axpy
  require bin_op { T, plus }
  require bin_op { T, times }
  generic :: axpy => axpy_

contains

  subroutine axpy_ ( A, X, Y )
    type ( T ), intent ( in ) :: A
    type ( T ), dimension ( : ), intent ( in ) :: X
    type ( T ), dimension ( : ), intent ( inout ) :: Y

    Y = plus ( times ( A, X ), Y )
  end subroutine

end template
```

credit: Tom Clune (NASA), Brad Richardson (NASA), & INCITS/Fortran Generic Subgroup

F202Y **Preview**: Generic Programming with Templates

Details are subject to change

```
requirement bin_op { T, Op }
  deferred type :: T
  deferred interface
    elemental function Op ( a, b )
      type ( T ), intent ( in ) :: a, b
      type ( T ) :: op
    end function
  end interface
end requirement
```

Library writer - avail. to user

```
integer, parameter :: SP = kind ( 1.0 ), DP = kind ( 1.d0 )

real(sp) :: f_A, f_X ( 10 ), f_Y ( 10 )
real(dp) :: d_A, d_X ( 10 ), d_Y ( 10 )
integer  :: i_A, i_X ( 10 ), i_Y ( 10 )

instantiate axpy_tmpl { real(sp), operator(+), operator(*) }
instantiate axpy_tmpl { real(dp), operator(+), operator(*) }
instantiate axpy_tmpl { integer, operator(+), operator(*) }

...

call axpy ( f_A, f_X, f_Y )
call axpy ( d_A, d_X, d_Y )
call axpy ( i_A, i_X, i_Y )
```

User

```
template axpy_tmpl { T, plus, times }

  private
  public :: axpy
  require bin_op { T, plus }
  require bin_op { T, times }
  generic :: axpy => axpy_

contains

  subroutine axpy_ ( A, X, Y )
    type ( T ), intent ( in ) :: A
    type ( T ), dimension ( : ), intent ( in ) :: X
    type ( T ), dimension ( : ), intent ( inout ) :: Y

    Y = plus ( times ( A, X ), Y )
  end subroutine

end template
```

Library writer

credit: Tom Clune (NASA), Brad Richardson (NASA), & INCITS/Fortran Generic Subgroup

Auto-generic Procedures and/or Template?

- Some overlap in functionalities, but can also be complementary
- Auto-generic is conceptually simpler
  - extension of current genericity in Fortran with which users are already familiar
- Template is more extensible and powerful
  - can be written for future-/user-defined types as long as `requirements` are satisfied
- Both are compile-time resolvable → no/minimal runtime impact
- Edge-cases, integrations, and compositing these features will still need to be ironed out in the standardization process

F202Y **Preview**: Asynchronous Collectives

F202Y introduces asynchrony for collective subroutines

```
CO_BROADCAST (A, SOURCE_IMAGE  [, STAT, ERRMSG])
CO_BROADCAST (A, SOURCE_IMAGE, COMPLETION [, STAT, ERRMSG])
CO_BROADCAST (A, SOURCE_IMAGE, TEAM [, STAT, ERRMSG])
CO_BROADCAST (A, SOURCE_IMAGE, TEAM, COMPLETION [, STAT, ERRMSG])

CO_MAX/MIN/SUM (A [, RESULT_IMAGE, STAT, ERRMSG] )
CO_MAX/MIN/SUM (A [, RESULT_IMAGE], COMPLETION [,STAT, ERRMSG] )
CO_MAX/MIN/SUM (A [, RESULT_IMAGE], TEAM [,STAT, ERRMSG] )
CO_MAX/MIN/SUM (A [, RESULT_IMAGE], TEAM, COMPLETION [,STAT, ERRMSG] )

CO_REDUCE (A, OPERATION  [, STAT, ERRMSG])
CO_REDUCE (A, OPERATION, COMPLETION [, STAT, ERRMSG])
CO_REDUCE (A, OPERATION, TEAM [, STAT, ERRMSG])
CO_REDUCE (A, OPERATION, TEAM, COMPLETION [, STAT, ERRMSG])

COMPLETE (COMPLETION [, FINISHED])
```

TEAM: if appear, collective operation over the specified TEAM

COMPLETION: if appear, initiate an asynchronous collective operation.

Asynchronous collective operation needs to be wait-ed on, or its status can be queried by call to `COMPLETE ()`

What other features are planned for F202Y?

See WG-5 N2249 document for full list, or j3-fortran.org standing doc 010.

Some features to mention:

- US07. Improve rank-independent functionality.
- US10. Define a standard "Fortran-friendly" preprocessor.
- US16. Define default KIND values to use throughout a program unit.
- US19. Add more math functions from IEEE-754
- US20. Add Intrinsic and collective subroutines for prefix operations
- US23/DIN4 Add generic processing of assumed-rank object
- … and more

What other features are planned for F202Y?

F202Y Workplan

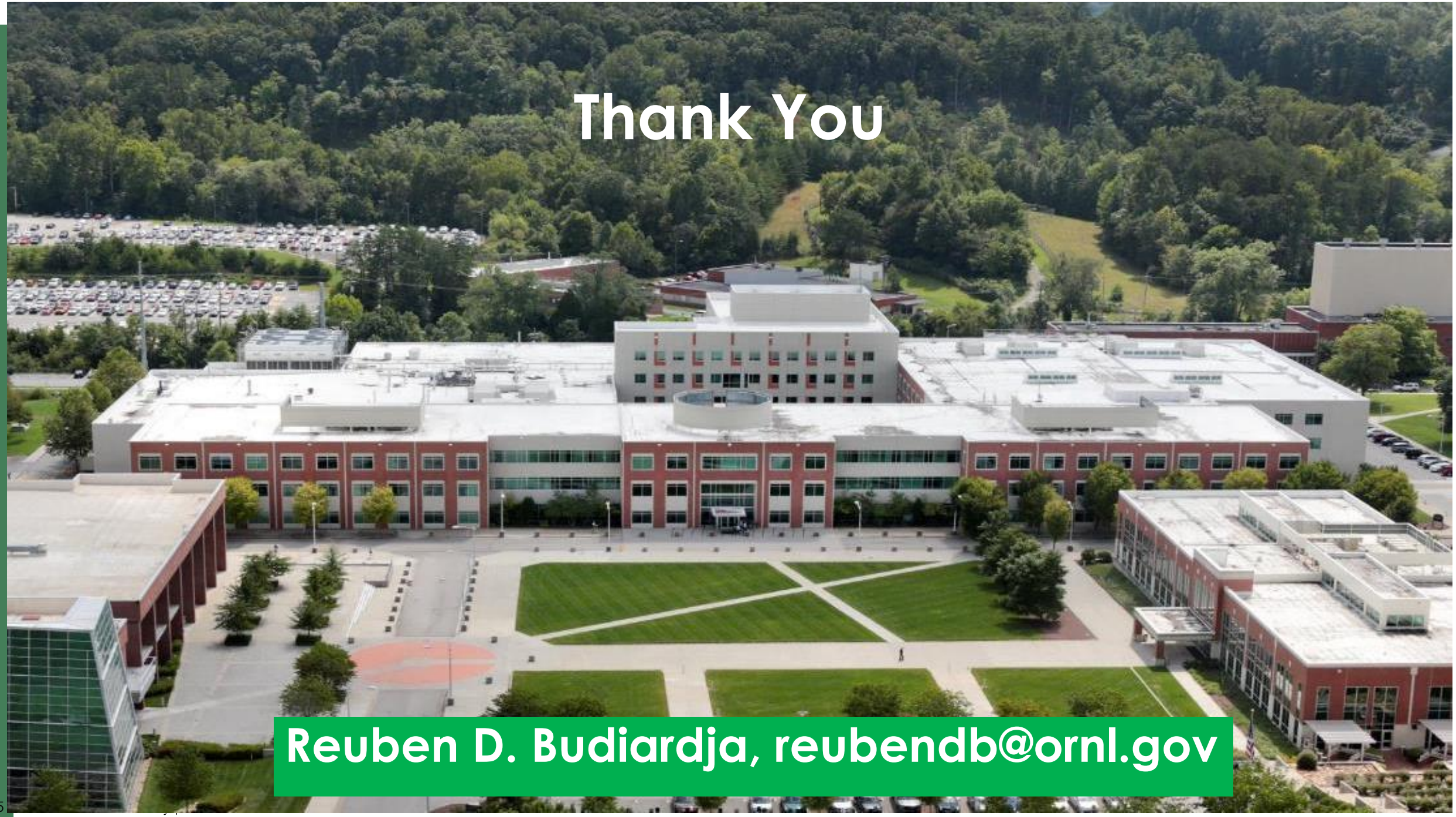
Live doc link

Label	Short Desc	Level (see 24-167)	Background / Use Cases	Subgroup	Rationale / Discussion Papers	Requirements	Specifications	Syntax	Edits
JP01	Include generic subprograms described in N2217	5	N2217 (WG5) 25-102 25-150	Data	24-140r2 (info, straw votes) 24-184r1 24-188r1 24-180 24-181 25-115	23-223r2 24-147r1 25-163r2	23-244r1 24-148r1 25-103 25-128 25-129 25-164r2	24-139r2 25-103 25-128 25-168r2	
US01	Obsolete default implicit typing	2	23-177	JOR					25-192r2
US02	Obsolete D format edit descriptor	2	23-178	JOR					25-113
US03	Add note that the real model is not IEEE 754	1	23-180	JOR					25-193r1
US04	Asynchronous Collective Subroutines		N2245 (WG5)	HPC		25-162r2	25-165r1	25-166r2	
US05	Add extended floating-point types defined in ISO C23 to ISO_C_BINDING	3	23-176	HPC		25-189r1	25-189r1		
US06	Provide a mechanism to specify global binding name for non C-interoperable	3	23-201	HPC					
US07	Improve rank-independent functionality	4	23-184r1	Data / JOR					
US08	Polymorphic PURE function results	4	23-186	Data		25-116r1	25-116r1 25-178	25-116r1 25-178	25-153r1
US09	Allow I/O of enumeration type names	4	23-151r1	JOR					25-126r1
US10	Define a standard Fortran preprocessor	6	23-192r1, 24-108, 24-109	FPP		25-114r2	25-142r2		
US11	Provide intrinsics for source location	3	23-193r1	JOR					
US12	Add maximum rank/corank constants to ISO_FORTRAN_ENV	3	23-194	Data (JP01)					
US13	Remote access to module entities	3	23-196r1	Data	25-119r1				
US14	Provide scoped access to enumeration enumerators	4	23-197	Data		25-180	25-180	25-180	
US15	Readonly pointers: a new attribute be application to pointers that determined whether the target of the pointer cannot be changed trhough the pointer	4	23-198	Data		25-179r1	25-179r1	25-182r1	25-197r2
US16	Provide a facility to specify the default kinds within the program unit	3	23-199	Data					
US17	Generic programming using templates	8	23-148, 23-103	Generic	23-211r1 23-212r2 23-213r2 23-214r3 23-215r1 25-158r1	24-105r3	24-105r3	24-125r5 24-126r4 24-127r4 24-159r4	25-109r2 25-135r2 25-136r2 25-172r1 25-173r1 25-174r2
US18	Allow Polymorphism in Coarrays	4	23-217	HPC (?)					
US19	Add more math functions from IEEE-754	4	23-234r2	JOR		25-143	25-143		
US20	Add Intrinsic and collective subroutines for prefix operations (formerly "SCAN" and "CO_SCAN")	4	23-235r2, 24-157	HPC + JOR	25-195r1	25-144r1 (co_*) 25-145r1 (local)	25-177r1 (co_*) 25-186r2 (sum) 25-196r1 (reduce)	25-177r1 (co_*) 25-186r2 (sum) 25-196r1 (reduce)	
US23/DIN4	Add generic processing of assumed-rank object	5	24-136r1	Data	24-169	24-171 24-172	24-183r1	24-183r1	24-183r1
US24	Add rank-independent looping	4	24-143	Data + JOR					
US25	Add ability to interpret complex as real and vice versa	5	24-129	Data		24-173	25-188r1	25-188r1	
DIN1	Add execution of collective procedures on a specified team	4	N2230 (WG5) 25-125r1	HPC		25-127r1	25-127r1	25-127r1	25-127r1
DIN3a	Add support for atomic operations in local memory	5	N2230 (WG5)	HPC					

Conclusions & Acknowledgements

- We are well on our way to Fortran 2028.
  - INCITS/Fortran (J3) and WG5 are working very hard to publish the standard in 2028
- More contributions from community are welcomed
  - join the standardization process
  - share your wishlists / pain points
  - contribute / write “papers” to J3

Thank you to all the INCITS/Fortran (J3) and WG5 members for their contributions to the development of the Fortran standard.


Thank You
Reuben D. Budiardja, reubendb@ornl.gov