Compiler support for the

Fortran 2008

and

Fortran 2018

standards

Revision 4

22/12/2025

Contents

Table of Contents

1 Compiler support for the Fortran 2008 and 2018 standards

1.1 Introduction

This document looks at compiler support for the 2008 and 2018 Fortran standards. It is a successor to previous documents that looked at support for earlier standards. As most actively developed compilers now fully support the Fortran 2003 standard (with some minor exceptions) our starting point is the Fortran 2008 standard.

The last set of tables with details of Fortran 2003 conformance appeared in the August 2019 edition of Fortran Forum

1.2 Fortran 2008 support

Here is the Fortran 2008 compliance table. The original table was based on John Reid's paper

- N1828 - The new features of Fortran 2008 (Reid)

which was published in 2010. This was superseded by

- N1891 The new features of Fortran 2008 (Reid)

which was published in 2011.

Additional features were taken from Annex C of the Fortran 2018 standard.

Van Snyder raised the issue of

- Interpretation F18/012 which was passed as paper 19-179.

He suggested that the feature could be listed as

- Internal specific for generic

We have added this to the Fortran 2008 table.

The references section has more details about these documents.

We provide three tables. The first table has a description of the feature. We have added an artificial feature number to make linking the tables easier. The column labelled 2008 number refers to the numbering scheme in John Reid's papers. The column labelled 2018 number refers to the numbering in Annex C of the Fortran 2018 standard.

The second table summarises implementation status.

We have created a third table that summarises the vendor support.

1.2.1 Fortran 2008 feature list table

Feature number	Fortran 2008 feature	2018 number	2008 number
1	Submodules		2
2	Coarrays		3
	Performance enhancements		4
3	do concurrent		4.1

Feature number	Fortran 2008 feature	2018 number	2008 number
4	Contiguous attribute		4.2
	Data Declaration		5
5	$Maximum rank + corank <=15$		5.1
6	Long integers (18 digit or 64 bit)		5.2
7	Allocatable components of recursive type		5.3
8	Implied-shape arrays		5.4
9	Pointer initialization		5.5
10	Data statement restrictions lifted		5.6
11	Kind of a forall index		5.7
12	Type statement for intrinsic types TYPE (intrinsic type) specifier		5.8
13	Declaring type-bound procedures		5.9
14	Value attribute is permitted for any nonallocatable nonpointer noncoarray		5.10.1
15	In a pure procedure the intent of an argument need not be specified if it has the value attribute		5.10.2
	Data Usage		6
16	Simply contiguous arrays rank remapping to rank>1 target		4.3
17	Omitting an allocatable component in a structure constructor		6.1
18	Multiple allocations with source=		6.2
19	Copying the properties of an object in an allocate statement		6.3
20	MOLD= specifier for ALLOCATE		6.3
21	Copying the bounds of a source array in an allocate statement		6,3
22	Polymorphic assignment		6.4
23	Accessing real and imaginary parts		6.5
24	Pointer function reference is a variable		6.6

Feature number	Fortran 2008 feature	2018 number	2008 number
25	Elemental dummy argument restrictions lifted		6.7
	Input/Output		7
26	Finding a unit when opening a file		7.1
27	g0 edit descriptor		7.2
28	Unlimited format item		7.3
29	Recursive i/o		7.4
	Execution control		8
30	The block construct		8.1
31	Exit statement		8.2
32	Stop code		8.3
33	ERROR STOP		8.4
	Intrinsic procedures and modules		9
	Bit processing		9.1
34	Bit sequence comparison		9.1-1
35	Combined shifting		9.1-2
36	Counting bits		9.1-3
37	Masking bits		9.1-4
38	Shifting bits		9.1-5
39	Merging bits		9.1-6
40	Bit transformational functions		9.1-7
41	Storage size		9.2
42	Optional argument radix added to selected_real_kind		9.3
43	Extensions to trigonometric and hyperbolic intrinsic functions		9.4

Feature number	Fortran 2008 feature	2018 number	2008 number
44	Bessel functions		9.5
45	Error and gamma functions		9.6
46	Euclidean vector norms		9.7
47	Parity		9.8
48	Execute command line		9.9
49	Optional back argument added to maxloc and minloc		9.1
50	Find location in an array		9.1.1
51	String comparison		9.1.2
52	Constants		9.1.3
53	COMPILER_VERSION		9.1.4
54	COMPILER_OPTIONS		9.1.4
55	Function for C sizeof		9.1.5
56	Added optional argument for ieee_selected_real_kind		9.1.6
	Programs and procedures		10
57	Save attribute for module and submodule data		
58	Empty contains section		10.2
59	Form of the end statement for an internal or module procedure		10.3
60	Internal procedure as an actual argument or pointer target		10.4
61	Null pointer or unallocated allocatable as an absent dummy argument		10.5
62	Non pointer actual for pointer dummy argument		10.6
63	Generic resolution by procedureness		10.7.1
64	Generic resolution by pointer versus allocatable		10.7.2
65	Impure elemental procedures		10.8
66	Entry statement becomes obsolescent		10.9
	Source form		11
67	Semicolon at line start		11.1

Feature number	Fortran 2008 feature	2018 number	2008 number
	Annex C Fortran 2018 standard		
	The following features were new in Fortran 2008 but not originally listed in its introduction as being new features		
68	An array or object with a nonconstant length type parameter can have the VALUE attribute.	1	
69	Multiple allocations are permitted in a single ALLOCATE statement with the SOURCE= specifier.	2	
70	A PROCEDURE statement can have a double colon before the first procedure name	3	
71	An argument to a pure procedure can have default INTENT if it has the VALUE attribute.	4	
72	The PROTECTED attribute can be specified by the procedure declaration statement.	5	
73	A defined-operator can be used in a specification expression	6	
74	All transformational functions from the intrinsic module ISO_C_BINDING can be used in specification expressions	7	
75	A contiguous array variable that is not interoperable but which has interoperable type and kind type parameter (if any); and a scalar character variable with length greater than 1 and kind C_CHAR in the intrinsic module ISO_C_BINDING; can be used as the argument of the function C_LOC in the intrinsic module ISO_C_BINDING; provided the variable has the POINTER or TARGET attribute.	8	
76	The name of an external procedure that has a binding label is a local identifier and not a global identifier	9	
77	A procedure that is not a procedure pointer can be an actual argument that corresponds to a procedure pointer dummy argument with the INTENT (IN) attribute.	10	

Feature number	Fortran 2008 feature	2018 number	2008 number
78	An interface body for an external procedure that does not exist in a program can be used to specify an explicit specific interface	11	
79	Internal specific for generic; See note 1 below.		

1.2.1.1 Notes

1. Interpretation F18/012 which was passed as paper 19-179.

1.2.2 Fortran 2008 implementation table

Here is the implementation summary.

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	NEC	Nvidia
						ifort	ifx			nvfortran
Version	20.1	4.0.2	9.x	8.4.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
1	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
2	N	Y	Y:2	Y	N	Y	N	Y	Y	N
3	Y	Y	P	Y	Y	Y	Y	Y	Y	Y
4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
5	N	Y	N	Y	Y	Y	Y	Y	Y	N
6	Y	Y:3	Y	Y	Y	Y	Y	Y	Y	Y
7	N	Y	N	Y	N	Y	Y	Y	Y	N
8	N	Y	Y	Y	Y	Y	Y	Y	Y	N
9	N	Y	Y	Y	N	Y	Y	Y	Y	P:14
10	N	Y	N	Y	N	Y	Y	Y	Y	N
11	N	Y	N	Y	Y	Y	Y	Y	Y	Y
12	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
13	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
14	N	Y	Y	Y	P:5	Y	Y	Y	Y	Y
15	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
16	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
17	N	Y	N	Y	N	Y	Y	Y	Y	Y
18	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
19	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
20	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	NEC	Nvidia
						ifort	ifx			nvfortran
Version	20.1	4.0.2	9.x	8.4.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
21	Y	Y	N:6	Y	Y	Y	Y	Y	Y	Y
22	N	Y	P:6	Y	Y	Y	Y	Y	Y	Y
23	P:7	Y	N	Y	Y	Y	Y	Y	Y	Y
24	N	Y	P	Y	N	Y	Y	Y	Y	N
25	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
26	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
27	N	Y	Y	Y	N	Y	Y	Y	Y	Y
28	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
29	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
30	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
31	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
32	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
33	N	Y	Y		Y	Y	Y	Y	Y	Y
34	N	Y	Y	Y	N	Y	Y	Y	Y	N
35	N	Y	Y	Y	Y	Y	Y	Y	Y	N
36	P:8	Y	Y	Y	Y	Y	Y	Y	Y	P:8
37	N	Y	Y	Y	Y	Y	Y	Y	Y	N
38	N	Y	Y	Y	Y	Y	Y	Y	Y	N
39	N	Y	Y	Y	Y	Y	Y	Y	Y	N
40	N	Y	Y	Y	N	Y	Y	Y	Y	N
41	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
42	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
43	P:9	Y	Y	Y	Y	Y	Y	Y	Y	Y
44	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
45	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
46	N	Y	Y	Y	N	Y	Y	Y	Y	Y
47	N	Y	Y	Y	N	Y	Y	Y	Y	N
48	N	Y	Y	Y	Y	Y	Y	Y	Y	Y

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	NEC	Nvidia
						ifort	ifx			nvfortran
Version	20.1	4.0.2	9.x	8.4.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
49	N	Y	N	Y	Y	Y	Y	Y	Y	Y
50	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
51	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
52	Y	Y	P:10	Y	Y	Y	Y	Y	Y	Y
53	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
54	N	Y	N	Y	Y	Y	Y	Y	Y	Y
55	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
56	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
57	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
58	P:11	Y	Y	Y	Y	Y	Y	Y	Y	Y
59	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
60	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
61	P:12	Y	Y	Y	Y	Y	Y	Y	Y	Y
62	Y	Y	N	Y		Y	Y	Y	Y	Y
63	N	Y	N	Y	Y	Y	Y	Y	Y	Y
64	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
65	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
66	P:13	Y	Y	Y	Y	Y	Y	Y	Y	Y
67	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
68	N	Y			Y	Y	Y	Y	Y	N
69	N	Y			N	Y	Y	Y	Y	Y
70	N	Y			Y	Y	Y	Y	Y	Y
71	N	Y			N	Y	Y	Y	Y	Y
72	N	Y			N	Y	Y	Y	Y	Y
73	N	Y			N	Y	Y	Y	Y	Y
74		Y			N	Y	Y	Y	Y	N
75		Y			Y	Y	Y	Y	Y	Y
76		Y			N	Y	Y	Y	Y	Y

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	NEC	Nvidia
						ifort	ifx			nvfortran
Version	20.1	4.0.2	9.x	8.4.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
77		Y			Y	Y	Y	Y	Y	N
78		Y				Y	Y	Y	Y	Y
79		Y						Y	Y	Y

1.2.2.1 Notes

Y Yes

N No

P Partial

Not known

2 Single image support since 4.6. Multi-image support using OpenCoarrays (including the Fortran 2018 collective subroutines) since 5.1, except allocatable or pointer components of derived type coarrays.

3 INTEGER (KIND=8)

4 but only for NULL as initialiser

5 Missing VALUE on dummy with non-constant type parameters

6 gfortran via allocate but not via intrinsic assignment

7 Not supported for complex arrays.

8 leadz, popcnt, and poppar supported. No trailz

9 Complex types are not accepted for acosh, asinh, and atanh, Additionally, atan2 cannot be accessed via atan.

10 int, real, and coarray

11 Not supported for procedures.

12 Not supported for null pointer.

13 Only shows a warning with the -Mstandard flag.

14 Pointer initialisation with a non-null() target is available for procedure pointers only.

1.2.3 Fortran 2008 vendor summary

Here is the summary table with counts.

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	NEC	Nvidia
						ifort	ifx			nvfortran
Version	19.1	4.0.2	9.x	8.4.0	15.1.5	2021.6	2022.1	7.1	2.3.1	22.7
Yes	28	78	48	66	53	78	78	79	79	60
Yes, Notes		1	1							0
No	39	0	13	0	22	0	1	0	0	17
No, Notes			1							
P	0	0	2	0	0	0	0	0	0	0
P, Notes	6	0	2		1					2
No Info	6	0	12	13	3	1	1	0	0	0
Total	79	79	79	79	79	79	79	79	79	79

1.3 Fortran 2018 support

We provide three tables. The first is a feature list table, the second is an implementation table and the third is a vendor summary table.

1.3.1 Fortran 2018 feature list table

Here is the feature list table for Fortran 2018. It is based on the Introduction of the Fortran 2018 standard, pages xiii-xv. John Reid has produced two papers that provide a more descriptive coverage.

- N2161 - The New Features of Fortran 2018 (Reid - Replaces N2145)
- N2145 - Summary of Fortran 2018 (Reid - Replaced by N2161)

These can be found at the WG5 site.

Here is the feature table.

Section	Sub section	Description
1		Data declaration:
	1.1	Constant properties of an object declared in its entity-decl can be used in its initialization.
	1.2	The EQUIVALENCE and COMMON statements and the block data program unit have been redundant since Fortran 90 and are now specified to be obsolescent.
	1.3	Diagnosis of the appearance of a PROTECTED TARGET variable accessed by use association as a data-target in a structure constructor is required.
2		Data usage and computation:
	2.1	The declared type of the value supplied for a polymorphic allocatable component in a structure constructor is no longer required to be the same as the declared type of the component.

Section	Sub section	Description
	2.2	FORALL is now specified to be obsolescent.
	2.3	The type and kind of an implied DO variable in an array constructor or DATA statement can be specified within the constructor or statement.
	2.4	The SELECT RANK construct provides structured access to the elements of an assumed-rank array.
	2.5	The standard intrinsic operations <, <=, >, and >= (also known as .LT., .LE., .GT., and .GE.) on IEEE numbers provide compareSignaling (relation) operations;
	2.6	The = and /= operations (also known as .EQ. and .NE.) provide compareQuiet (relation) operations.
	2.7	Finalization of an allocatable subobject during intrinsic assignment has been clarified.
	2.8	The char-length in an executable statement is no longer required to be a specification expression.
3		Input/output:
	3.1	The SIZE= specifier can be used with advancing input.
	3.2	It is no longer prohibited to open a file on more than one unit.
	3.3	The value assigned by the RECL= specifier in an INQUIRE statement has been standardized.
	3.4	The values assigned by the POS= and SIZE= specifiers in an INQUIRE statement for a unit that has pending asynchronous operations have been standardized.
	3.5	The G0.d edit descriptor can be used for list items of type Integer, Logical, and Character.
	3.6	The D, E, EN, and ES edit descriptors can have a field width of zero, analogous to the F edit descriptor.
	3.7	The exponent width e in a data edit descriptor can be zero, analogous to a field width of zero.
	3.8	Floating-point formatted input accepts hexadecimal-significand numbers that conform to ISO/IEC/IEEE 60559:2011.
	3.9	The EX edit descriptor provides hexadecimal-significand formatted output conforming to
	3.10	An error condition occurs if unacceptable characters are presented for logical or numeric editing during execution of a formatted input statement.
4		Execution control:
	4.1	The arithmetic IF statement has been deleted.

Section	Sub section	Description
	4.2	Labelled DO loops have been redundant since Fortran 90 and are now specified to be obsolescent.
	4.3	The nonblock DO construct has been deleted.
	4.4	The locality of a variable used in a DO CONCURRENT construct can be explicitly specified.
	4.5	The stop code in a STOP or ERROR STOP statement can be nonconstant.
	4.6	Output of the stop code and exception summary from the STOP and ERROR STOP statements can be controlled.
5		Intrinsic procedures and modules:
	5.1	In a reference to the intrinsic function CMPLX with an actual argument of type complex, no keyword is needed for a KIND argument.
	5.2	In references to the intrinsic functions ALL, ANY, FINDLOC, IALL, IANY, IPARITY, MAXLOC, MAXVAL, MINLOC, MINVAL, NORM2, PARITY, PRODUCT, SUM, and THIS_IMAGE, the actual argument for DIM can be a present optional dummy argument.
	5.3	The new intrinsic function COSHAPE returns the coshape of a coarray.
	5.4	The new intrinsic function OUT_OF_RANGE tests whether a numeric value can be safely converted to a different type or kind.
	5.5	The new intrinsic subroutine RANDOM_INIT establishes the initial state of the pseudorandom number generator used by RANDOM_NUMBER.
	5.6	The new intrinsic function REDUCE performs user-specified array reductions.
	5.7	A processor is required to report use of a nonstandard intrinsic procedure, use of a nonstandard intrinsic module, and use of a nonstandard procedure from a standard intrinsic module.
	5.8	Integer and logical arguments to intrinsic procedures and intrinsic module procedures that were previously required to be of default kind no longer have that requirement, except for RANDOM_SEED.
	5.9	Specific names for intrinsic functions are now deemed obsolescent.
	5.10	All standard procedures in the intrinsic module ISO_C_BINDING, other than C_F_POINTER, are now pure.
	5.11	The arguments to the intrinsic function SIGN can be of different kind.
	5.12	Nonpolymorphic pointer arguments to the intrinsic functions EXTENDS_TYPE_OF and SAME_TYPE_AS need not have defined pointer association status.

Section	Sub section	Description
	5.13	The effects of invoking the intrinsic procedures COMMAND_ARGUMENT_COUNT, GET_COMMAND, and GET_COMMAND_ARGUMENT, on images other than image one, are no longer processor dependent.
	5.14	Access to error messages from the intrinsic subroutines GET_COMMAND, GET_COMMAND_ARGUMENT and GET_ENVIRONMENT_VARIABLE is provided by an optional ERRMSG argument.
	5.15	The result of NORM2 for a zero-sized array argument has been clarified.
6		Program units and procedures:
	6.1	The IMPORT statement can appear in a contained subprogram or BLOCK construct, and can restrict access via host association;
	6.2	Diagnosis of violation of the IMPORT restrictions is required.
	6.3	The GENERIC statement can be used to declare generic interfaces.
	6.4	The number of procedure arguments is used in generic resolution.
	6.5	In a module, the default accessibility of entities accessed from another module can be controlled separately from the default accessibility of entities declared in the using module.
	6.6	An IMPLICIT NONE statement can require explicit declaration of the EXTERNAL attribute throughout a scoping unit and its contained scoping units.
	6.7	A defined operation need not specify INTENT (IN) for a dummy argument with the VALUE attribute.
	6.8	A defined assignment need not specify INTENT (IN) for the second dummy argument if it has the VALUE attribute.
	6.9	Procedures that are not declared with an asterisk type-param-value, including ELEMENTAL procedures, can be invoked recursively by default;
	6.10	The RECURSIVE keyword is advisory (most procedures are recursive by default) only.
	6.11	The NON_RECURSIVE keyword specifies that a procedure is not recursive.
	6.12	The ERROR STOP statement can appear in a pure subprogram.
	6.13	A dummy argument of a pure function is permitted in a variable definition context, if it has the VALUE attribute.
	6.14	A coarray dummy argument can be referenced or defined by another image.
7		Features previously described by ISO/IEC TS 29113:2012:
	7.1	A dummy data object can assume its rank from its effective argument.
	7.2	A dummy data object can assume the type from its effective argument, without having the ability to perform type selection.

Section	Sub section	Description
	7.3	An interoperable procedure can have dummy arguments that are assumed-type and/or assumed-rank.
	7.4	An interoperable procedure can have dummy data objects that are allocatable, assumed-shape, optional, or pointers.
	7.5	The character length of a dummy data object of an interoperable procedure can be assumed.
	7.6	The argument to C_LOC can be a noninteroperable array.
	7.7	The FPTR argument to C_F_POINTER can be a noninteroperable array pointer.
	7.8	The argument to C_FUNLOC can be a noninteroperable procedure.
	7.9	The FPTR argument to C_F_PROCPOINTER can be a noninteroperable procedure pointer.
	7.10	There is a new named constant C_PTRDIFF_T to provide interoperability with the C type ptrdiff_t.
	7.11	Additionally to ISO/IEC TS 29113:2012, a scalar actual argument can be associated with an assumed-type assumed-size dummy argument, an assumed-rank dummy data object that is not associated with an assumed-size array can be used as the argument to the function C_SIZEOF from the intrinsic module ISO_C_BINDING, and the type argument to CFI_establish can have a positive value corresponding to an interoperable C type. “8"
8		Changes to the intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS, and IEEE_FEATURES for conformance with ISO/IEC/IEEE 60559:2011:
	8.1	There is a new, optional, rounding mode IEEE_AWAY.
	8.2	The new type IEEE_MODES_TYPE encapsulates all floating-point modes.
	8.3	Features associated with subnormal numbers can be accessed with functions and types named ...SUBNORMALquestion (the old ...DENORMALquestion names remain).
	8.4	The new function IEEE_FMA performs fused multiply-add operations.
	8.5	The function IEEE_INT performs rounded conversions to integer type.
	8.6	The new functions IEEE_MAX_NUM, IEEE_MAX_NUM_MAG, IEEE_MIN_NUM, and IEEE_MIN_NUM_MAG calculate maximum and minimum numeric values.
	8.7	The new functions IEEE_NEXT_DOWN and IEEE_NEXT_UP return the adjacent machine numbers.
	8.8	The new functions IEEE_QUIET_EQ, IEEE_QUIET_GE, IEEE_QUIET_GT, IEEE_QUIET_LE, IEEE_QUIET_LT, and IEEE_QUIET_NE perform quiet comparisons.

Section	Sub section	Description
	8.9	The new functions IEEE_SIGNALING_EQ, IEEE_SIGNALING_GE, IEEE_SIGNALING_GT, IEEE_SIGNALING_GE, IEEE_SIGNALING_LE, IEEE_SIGNALING_LT, and IEEE_SIGNALING_NE perform signalling comparisons.
	8.10	The decimal rounding mode can be inquired and set independently of the binary rounding mode, using the RADIX argument to IEEE_GET_ROUNDING_MODE, and IEEE_SET_ROUNDING_MODE.
	8.11	The new function IEEE_REAL performs rounded conversions to real type.
	8.12	The function IEEE_REM now requires its arguments to have the same radix.
	8.13	The function IEEE_RINT now has a ROUND argument to perform specific rounding.
	8.14	The new function IEEE_SIGNBIT tests the sign bit of an IEEE number.
9		Features previously described by ISO/IEC TS 18508:2015:
	9.1	The CRITICAL statement has optional ERRMSG= and STAT= specifiers.
	9.2	The intrinsic subroutines ATOMIC_DEFINE and ATOMIC_REF have an optional STAT argument.
	9.3	The new intrinsic subroutines ATOMIC_ADD, ATOMIC_AND, ATOMIC_CAS, ATOMIC_FETCH_ADD, ATOMIC_FETCH_AND ATOMIC_FETCH_OR, ATOMIC_FETCH_XOR, ATOMIC_OR, and ATOMIC_XOR perform atomic operations.
	9.4	The new intrinsic functions FAILED_IMAGES and STOPPED_IMAGES return indices of images known to have failed or stopped respectively.
	9.5	The new intrinsic function IMAGE_STATUS returns the image execution status of an image.
	9.6	The intrinsic subroutine MOVE_ALLOC has optional ERRMSG and STAT arguments.
	9.7	The intrinsic functions IMAGE_INDEX and NUM_IMAGES have additional forms with a TEAM or TEAM_NUMBER argument.
	9.8	The intrinsic function THIS_IMAGE has an optional TEAM argument.
	9.9	The EVENT POST and EVENT WAIT statements, the intrinsic subroutine EVENT_QUERY, and the type EVENT_TYPE provide an event facility for one-sided segment ordering.

Section	Sub section	Description
	9.10	The CHANGE TEAM construct, derived type TEAM_TYPE, FORM TEAM and SYNC TEAM statements, intrinsic functions GET_TEAM and TEAM_NUMBER, and the TEAM= and TEAM_NUMBER= specifiers on image selectors, provide a team facility for a subset of the programs images to act in concert as if it were the set of all images. This team facility allows an allocatable coarray to be allocated or deallocated on a subset of images.
	9.11	The new intrinsic subroutines CO_BROADCAST, CO_MAX, CO_MIN, CO_REDUCE, and CO_SUM perform collective reduction operations on the images of the current team.
	9.12	The concept of failed images, the FAIL IMAGE statement, the STAT= specifier on image selectors, and the named constant STAT_FAILED_IMAGE from the intrinsic module ISO_FORTRAN_ENV provide support for fault-tolerant parallel execution.
10		Changes to features previously described by ISO/IEC TS 18508:2015:
	10.1	The CHANGE TEAM and SYNC TEAM statements, and the TEAM= specifier on image selectors, permit the team to be specified by an expression.
	10.2	The intrinsic functions FAILED_IMAGES and STOPPED_IMAGES have no restriction on the kind of their result.
	10.3	The name of the function argument to the intrinsic function CO_REDUCE is OPERATION instead of OPERATOR; this argument is not required to be commutative.
	10.4	The named constant STAT_UNLOCKED_FAILED_IMAGE from the intrinsic module ISO_FORTRAN_ENV indicates that a lock variable was locked by an image that failed.
	10.5	The team number for the initial team can be used in image selectors, and in the intrinsic functions NUM_IMAGES and IMAGE_INDEX.
	10.6	A team variable that appears in a CHANGE TEAM statement can no longer be defined or become undefined during execution of the CHANGE TEAM construct.
	10.7	All images of the current team are no longer required to execute the same CHANGE TEAM statement.
	10.8	A variable of type TEAM_TYPE from the intrinsic module ISO_FORTRAN_ENV is not permitted to be a coarray.
	10.9	A variable of type TEAM_TYPE from the intrinsic module ISO_FORTRAN_ENV can have a pointer component, and a team variable becomes undefined if assigned a value from another image.
	10.10	The intrinsic function UCOBOUND produces a value for the final upper cobound that is always relative to the current team.
	10.11	An EXIT statement can be used to complete execution of a CHANGE TEAM or CRITICAL construct.

1.3.2 Fortran 2018 implementation table

Here is the implementation table.

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	Nec	Nvidia
						ifort	ifx			
Version	18.1	4.0.2	13.2	9.1.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
1.1		Y	Y	Y		Y	Y	Y	Y	Y
1.2		N	Y:14	Y		Y	Y	Y:10	Y	N
1.3		Y	Y	Y		Y	Y	Y	Y	Y
2.1		Y	N:15	Y		Y	Y	Y	Y	N
2.2		N	Y	Y		Y	Y	N	N	N
2.3		Y	N	Y		y	Y	N	N	N
2.4		N	Y	Y		Y	Y	Y	N	N
2.5		N	Y	Y		Y	Y	Y	Y	N
2.6		N	Y	Y		Y	Y	Y	Y	N
2.7		N	Y	Y		Y	Y	Y	Y	Y
2.8		Y	Y	Y		Y	Y	Y	Y	Y
3.1		N	N	Y		Y	Y	Y	Y	Y
3.2		N	N	Y		Y	Y	Y	Y	N
3.3		N	P	Y		Y	Y	Y	Y	N
3.4		N	N	Y		Y	Y	Y	Y	Y
3.5		N	N	Y		Y	Y	N	N	Y
3.6		N	N	Y		Y	Y	N	N	N
3.7		N	N	Y		Y	Y	N	N	N
3.8		N	N	Y		Y	Y	N	N	N
3.9		N	N	Y		Y	Y	N	N	N
3.1		N	Y	Y		Y	Y	Y	Y	Y
4.1		N	Y	Y		Y	Y	N	N	N
4.2		N	Y	Y		Y	Y	N	N	N
4.3		N	Y	Y		Y	Y	N	N	N
4.4		N	N	N		Y	Y	N	N	Y
4.5		N	Y	Y		Y	Y	Y	Y	Y

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	Nec	Nvidia
						ifort	ifx			
Version	18.1	4.0.2	13.2	9.1.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
4.6		N	Y	Y		Y	Y	Y	Y	Y
5.1		N	N	Y		Y	Y	Y	Y	N
5.2		N	N	Y		Y	Y	Y	Y	Y
5.3		N	Y	Y		Y	N	Y	Y	N
5.4		N	N	Y		Y	Y	N	N	N
5.5		N	Y:16	Y		Y	Y	N	N	N
5.6		N	N	N		Y	N	Y	Y	N
5.7		N	N	Y		Y	Y	Y	Y	N
5.8		N	Y	Y		Y	Y	P:11	N	N
5.9		N	N	Y		Y	Y	Y	Y	N
5.10		N	Y	N		Y	Y	N	N	N
5.11		N	N	Y		Y	Y	Y	Y	Y
5.12		N	Y	Y		Y	Y	Y	Y	Y
5.13		N		Y		Y	Y	Y	Y	N
5.14		N	N	Y		Y	Y	N	N	N
5.15		N	Y	Y		Y	Y	Y	Y	Y
6.1		N	N	Y		Y	Y	N	N	N
6.2		N	Y	Y		Y	Y	Y	Y	N
6.3		N	Y	Y		Y	Y	N	N	N
6.4		N	N	Y		Y	Y	Y	Y	N
6.5		N	Y	Y		Y	Y	N	N	Y
6.6		N	N	Y		Y	Y	N	N	N
6.7		N	N	Y		Y	Y	Y	Y	Y
6.8		N	N	Y		Y	Y	Y	Y	Y
6.9		N	N:17	Y		Y:1	Y:1	Y:12	Y	N
6.10		N	N	Y		Y:1	Y:1	y:12	Y	N
6.11		N	N	Y		Y	Y	Y	Y	N
6.12		N	Y	Y		Y	N	Y	Y	Y

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	Nec	Nvidia
						ifort	ifx			
Version	18.1	4.0.2	13.2	9.1.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
6.13		Y	Y	Y		Y	Y	Y	Y	Y
6.14		Y		Y		Y	Y	Y	Y	N
7.1		Y	Y	Y		Y	Y	Y	Y	Y
7.2		Y	Y	Y		Y	Y	Y	Y	N
7.3		Y	Y	Y		Y	Y	Y	Y	N
7.4		N	Y	Y		Y	Y	Y	Y	N
7.5		N	Y	Y		Y	Y	Y	Y	N
7.6		Y	Y	Y		Y	Y	Y	Y	Y
7.7		Y	Y	Y		Y	Y	Y	Y	Y
7.8		Y	Y	Y		Y	Y	Y	Y	Y
7.9		Y	Y	Y		Y	Y	Y	Y	Y
7.10		N	Y	Y		Y	Y	N	N	Y
7.11		N	Y	Y		Y	Y	N	N	N
8.1		N	Y	N		Y	Y	N	N	N
8.2		N	Y	N		Y	Y	N	N	N
8.3		N	Y	Y		Y	Y	Y	Y	N
8.4		N	Y	Y		Y	Y	N	N	N
8.5		N	N	Y		Y	Y	N	N	N
8.6		N	Y	Y		Y	Y	N	N	N
8.7		N	N	Y		Y	Y	Y	Y	N
8.8		N	Y	Y		Y	Y	N	N	N
8.9		N	Y	Y		Y	Y	N	N	N
8.1		N	Y	Y		Y	Y	N:13	N	N
8.11		N	N	Y		Y	Y	N	N	N
8.12		N	N	Y		Y	Y	Y:13	Y	N
8.13		N	N	Y		Y	Y	N	N	N
8.14		N	Y	Y		Y	Y	N	N	N
9.1		N	N	N		Y	Y	Y	Y	N

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	Nec	Nvidia
						ifort	ifx			
Version	18.1	4.0.2	13.2	9.1.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Feature number										
9.2		Y	Y	Y		Y	N	Y	Y	N
9.3		N	Y	Y		Y	N	Y	Y	N
9.4		N	P18	N		Y:2	N	Y	Y	N
9.5		N	Y	N		Y:2	N	Y	Y	N
9.6		N	N	Y		Y	N	Y	Y	N
9.7		N	N	N		Y	N	Y	Y	N
9.8		N	Y	N		Y	N	Y	Y	N
9.9		N	P	Y		Y	N	Y	Y	N
9.10		N	P18	N		Y	N	Y	Y	N
9.11		P	Y	Y		Y	N	Y	N	N
9.12		N	P	N		Y	N	Y	Y	N
10.1		N	N	N		Y	N	Y	Y	N
10.2		N	Y	N		Y	N	Y	Y	N
10.3		N	Y	N		Y	N	Y	Y	N
10.4		N	N	N		Y	N	Y	Y	N
10.5		N	P	N		Y	N	Y	Y	N
10.6		N		N		Y	N	N	N	N
10.7		N		N		Y	N	Y	Y	N
10.8		N		N		Y	N	Y	Y	N
10.9		N		N		Y	N	Y	Y	N
10.10		N		Y		Y	N	Y	Y	N
10.11		N		N		Y	Y	Y	Y	N

1.3.2.1 Notes

Y Yes

N No

P Partial

Not known

1 Must use the non-default -assume recursion compiler option

2 The TEAM= argument may not be specified for the intrinsics FAILED_IMAGES, IMAGE_STATUS, or STOPPED_IMAGES at this time.

10 The -f2018 option

11 THIS_IMAGE and the six IEEE_GET and IEEE_SET routines

12 The -recursive or -f2018 options

13 There are no decimal REAL kinds

14 gfortran

15 gfortran

16 gfortran

17 gfortran

18 gfortran

1.3.3 2018 vendor summary table

Here is the summary table.

Vendor	Arm	Fujitsu	gfortran	HPE	IBM	Intel	Intel	Nag	Nec	Nvidia
						ifort	ifx			nvfortran
version	18.1	4.0.2	13.2	9.1.0	15.1.5	2021.6	2022.1	7.1	3.5.0	22.7
Yes	0	14	51	82	0	100	78	66	68	26
Yes, Notes			2			4	2	4		
No	0	88	35	22	0	0	24	32	36	78
No, Notes			2					1		
P	0	1	4	0	0	0	0	0	0	0
P, Notes			2					1		
No Info	104	1	8	0	104	0	0	0	0	0
Total	104	104	104	104	104	104	104	104	104	104

1.4 References and history

The WG5 site has copies of John Reid's papers. These are

- N2212 - The New Features of Fortran 2023 (Replaces N2194, Reid)
- N2161 - The New Features of Fortran 2018 (Reid - Replaces N2145)
- N2145 - Summary of Fortran 2018 (Reid - Replaced by N2161)
- N1891 The new features of Fortran 2008 (Reid)
- N1828 - The new features of Fortran 2008 (Reid - Replaced by N1891)

`https://wg5-fortran.org/`

The ISO has copies of the standards. The current standard is:

- ISO/IEC 1539-1:2023 Programming languages — Fortran Part 1: Base language. Published (Edition 5, 2023)

`https://www.iso.org/standards.html`

Here is a summary of recent standards.

- ISO/IEC 1539-1:2018. Withdrawn.
- ISO/IEC 1539-1:2018/Cor 1:2021. Withdrawn.
- ISO/IEC 1539-1:2018/Cor 2:2023. Withdrawn.
- TR 15580:1998 1998 December Floating-point exception handling Withdrawn
- TR 15580:2001 2001 June Withdrawn
- TR 15581:1998 1998 December Enhanced data type facilities Withdrawn
- TR 15581:2001 2001 June Withdrawn
- TR 19767:2005 2005 February Enhanced module facilities Withdrawn
- TS 29113:2012 2012 December Further interoperability of Fortran with C Current
- TS 18508:2015 2015 December Additional Parallel Features in Fortran Current

J3 also has a wide range of documents available. Details of interpretation request 19-179 are given below.

`https://j3-fortran.org/doc/year/19/19-179.txt`

1.4.1 Fortran Forum articles on standard conformance

Fortran Forum was an ACM newsletter about Fortran. Here is a link

`https://dl.acm.org/newsletter/sigplan-fortran`

It was published between 1982 and 2021.

The first article on Fortran 2003 conformance appeared in 2007. Here is the revision history

- Original April 2007
- Revision 1 August 2007
- Revision 2 August 2008
- Revision 3 April 2009
- Revision 4 August 2009
- Revision 5 August 2010
- Revision 6 December 2010
- Revision 7 August 2011
- Revision 8 April 2012
- Revision 9 April 2012
- Revision 10 August 2012
- Revision 11 December 2012
- Revision 12 April 2013
- Revision 13 August 2013
- Revision 14 December 2013
- Revision 15 August 2014

- Revision 16 August 2015
- Revision 17 December 2015
- Revision 18 December 2015
- Revision 19 August 2016
- Revision 20 December 2016
- Revision 21 April 2017
- Revision 22 December 2017
- Revision 23 April 2018
- Revision 24 August 2018
- Revision 25 December 2018
- Revision 26 August 2019

Here is a link with more detailed information

`https://www.rhymneyconsulting.co.uk/fortran/fortran_2003_compiler_conformance_tables/`

Copies of each of the above articles are available as Adobe Acrobat Portable Document format - pdf.

The first edition of the document upon which this document is based appeared in the April 2020 edition of Fortran Forum. Here is a link

`https://dl.acm.org/doi/10.1145/3432987.3432991`

The article appeared on-line in 2021.

1.5 Acknowledgements

The following people have contributed to the tables since the first version that appeared in Fortran Forum in April 2007.

Absoft:	Wood Lotz
Arm:	Richard Barton, John MacCallum, Ashok Bhat, Nathan Sircombe, Kiran Chandramohan, Caroline Concatto, Peter Waller
Cray:	Bill Long
Fujitsu:	Minoru Tanaka, Yuuji Tsujimori, Suzuk Toshihiro
gfortran:	fxcoudert (fx) , Paul Richard Thomas, Tobias Burnus
IBM:	Rafik Zurob, Daniel C Chen
HPE	Bill Long
Intel:	Lorri Menard, Jon L Steidel, Steve Lionel, Stan Whitlock
Nag:	Malcolm Cohen
NEC:	Yasuharu Hayashi, Shoichi Sakon
Nvidia	Graham Lopez, Jeff Larkin, Jeff Hammond, Mark LeAir.
Oracle:	Calvin Vu
PGI:	Dave Norton, Pat Brooks, Brent Leback, Gary Klimowicz, Mark Leair
Other:	Takata Masayuki, Richard Maine, Van Snyder

Thanks to everyone.