
**Information technology —
Programming languages — Fortran —
Part 1:
Base language**

*Technologies de l'information — Langages de programmation —
Fortran —*

Partie 1: Langage de base

ITeH Standards
(<https://standards.iteh.ai>)
Document Preview

ISO/IEC 1539-1:2018

<https://standards.iteh.ai/catalog/standards/iso/5cf376c5-5a69-4be4-a143-84361b77d375/iso-iec-1539-1-2018>



iTeh Standards
(<https://standards.itih.ai>)
Document Preview

ISO/IEC 1539-1:2018

<https://standards.itih.ai/catalog/standards/iso/5cf376c5-5a69-4be4-a143-84361b77d375/iso-iec-1539-1-2018>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2018

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword	xii
Introduction	xiii
1 Scope	1
2 Normative references	2
3 Terms and definitions	3
4 Notation, conformance, and compatibility	24
4.1 Notation, symbols and abbreviated terms	24
4.1.1 Syntax rules	24
4.1.2 Constraints	25
4.1.3 Assumed syntax rules	25
4.1.4 Syntax conventions and characteristics	25
4.1.5 Text conventions	26
4.2 Conformance	26
4.3 Compatibility	27
4.3.1 Previous Fortran standards	27
4.3.2 New intrinsic procedures	27
4.3.3 Fortran 2008 compatibility	27
4.3.4 Fortran 2003 compatibility	28
4.3.5 Fortran 95 compatibility	29
4.3.6 Fortran 90 compatibility	29
4.3.7 FORTRAN 77 compatibility	29
4.4 Deleted and obsolescent features	30
4.4.1 General	30
4.4.2 Nature of deleted features	30
4.4.3 Nature of obsolescent features	30
5 Fortran concepts	31
5.1 High level syntax	31
5.2 Program unit concepts	34
5.2.1 Program units and scoping units	34
5.2.2 Program	34
5.2.3 Procedure	34
5.2.4 Module	35
5.2.5 Submodule	35
5.3 Execution concepts	35
5.3.1 Statement classification	35
5.3.2 Statement order	35
5.3.3 The END statement	36
5.3.4 Program execution	36
5.3.5 Execution sequence	37
5.3.6 Image execution states	37

5.3.7	Termination of execution	38
5.4	Data concepts	38
5.4.1	Type	38
5.4.2	Data value	39
5.4.3	Data entity	39
5.4.4	Definition of objects and pointers	40
5.4.5	Reference	41
5.4.6	Array	41
5.4.7	Coarray	41
5.4.8	Established coarrays	42
5.4.9	Pointer	42
5.4.10	Allocatable variables	42
5.4.11	Storage	42
5.5	Fundamental concepts	43
5.5.1	Names and designators	43
5.5.2	Statement keyword	43
5.5.3	Other keywords	43
5.5.4	Association	43
5.5.5	Intrinsic	43
5.5.6	Operator	43
5.5.7	Companion processors	44
6	Lexical tokens and source form	45
6.1	Processor character set	45
6.1.1	Characters	45
6.1.2	Letters	45
6.1.3	Digits	45
6.1.4	Underscore	45
6.1.5	Special characters	45
6.1.6	Other characters	46
6.2	Low-level syntax	46
6.2.1	Tokens	46
6.2.2	Names	46
6.2.3	Constants	47
6.2.4	Operators	47
6.2.5	Statement labels	48
6.2.6	Delimiters	48
6.3	Source form	49
6.3.1	Program units, statements, and lines	49
6.3.2	Free source form	49
6.3.3	Fixed source form	50
6.4	Including source text	51
7	Types	53
7.1	Characteristics of types	53
7.1.1	The concept of type	53
7.1.2	Type classification	53
7.1.3	Set of values	53
7.1.4	Constants	53
7.1.5	Operations	53
7.2	Type parameters	54
7.3	Types, type specifiers, and values	54
7.3.1	Relationship of types and values to objects	54
7.3.2	Type specifiers and type compatibility	55
7.4	Intrinsic types	57
7.4.1	Classification and specification	57

7.4.2	Intrinsic operations on intrinsic types	57
7.4.3	Numeric intrinsic types	57
7.4.4	Character type	61
7.4.5	Logical type	64
7.5	Derived types	64
7.5.1	Derived type concepts	64
7.5.2	Derived-type definition	65
7.5.3	Derived-type parameters	69
7.5.4	Components	70
7.5.5	Type-bound procedures	77
7.5.6	Final subroutines	79
7.5.7	Type extension	81
7.5.8	Derived-type values	83
7.5.9	Derived-type specifier	83
7.5.10	Construction of derived-type values	84
7.5.11	Derived-type operations and assignment	86
7.6	Enumerations and enumerators	86
7.7	Binary, octal, and hexadecimal literal constants	87
7.8	Construction of array values	88
8	Attribute declarations and specifications	91
8.1	Attributes of procedures and data objects	91
8.2	Type declaration statement	91
8.3	Automatic data objects	93
8.4	Initialization	93
8.5	Attributes	93
8.5.1	Attribute specification	93
8.5.2	Accessibility attribute	94
8.5.3	ALLOCATABLE attribute	94
8.5.4	ASYNCHRONOUS attribute	94
8.5.5	BIND attribute for data entities	95
8.5.6	CODIMENSION attribute	95
8.5.7	CONTIGUOUS attribute	97
8.5.8	DIMENSION attribute	98
8.5.9	EXTERNAL attribute	101
8.5.10	INTENT attribute	101
8.5.11	INTRINSIC attribute	103
8.5.12	OPTIONAL attribute	104
8.5.13	PARAMETER attribute	104
8.5.14	POINTER attribute	104
8.5.15	PROTECTED attribute	105
8.5.16	SAVE attribute	105
8.5.17	TARGET attribute	106
8.5.18	VALUE attribute	106
8.5.19	VOLATILE attribute	106
8.6	Attribute specification statements	107
8.6.1	Accessibility statement	107
8.6.2	ALLOCATABLE statement	108
8.6.3	ASYNCHRONOUS statement	108
8.6.4	BIND statement	108
8.6.5	CODIMENSION statement	108
8.6.6	CONTIGUOUS statement	109
8.6.7	DATA statement	109
8.6.8	DIMENSION statement	111
8.6.9	INTENT statement	111
8.6.10	OPTIONAL statement	112

8.6.11	PARAMETER statement	112
8.6.12	POINTER statement	112
8.6.13	PROTECTED statement	113
8.6.14	SAVE statement	113
8.6.15	TARGET statement	113
8.6.16	VALUE statement	113
8.6.17	VOLATILE statement	114
8.7	IMPLICIT statement	114
8.8	IMPORT statement	116
8.9	NAMelist statement	118
8.10	Storage association of data objects	119
8.10.1	EQUIVALENCE statement	119
8.10.2	COMMON statement	121
8.10.3	Restrictions on common and equivalence	122
9	Use of data objects	123
9.1	Designator	123
9.2	Variable	123
9.3	Constants	124
9.4	Scalars	124
9.4.1	Substrings	124
9.4.2	Structure components	124
9.4.3	Coindexed named objects	126
9.4.4	Complex parts	126
9.4.5	Type parameter inquiry	126
9.5	Arrays	127
9.5.1	Order of reference	127
9.5.2	Whole arrays	127
9.5.3	Array elements and array sections	127
9.5.4	Simply contiguous array designators	130
9.6	Image selectors	131
9.7	Dynamic association	132
9.7.1	ALLOCATE statement	132
9.7.2	NULLIFY statement	136
9.7.3	DEALLOCATE statement	136
9.7.4	STAT= specifier	138
9.7.5	ERRMSG= specifier	139
10	Expressions and assignment	140
10.1	Expressions	140
10.1.1	Expression semantics	140
10.1.2	Form of an expression	140
10.1.3	Precedence of operators	144
10.1.4	Evaluation of operations	145
10.1.5	Intrinsic operations	146
10.1.6	Defined operations	153
10.1.7	Evaluation of operands	154
10.1.8	Integrity of parentheses	154
10.1.9	Type, type parameters, and shape of an expression	155
10.1.10	Conformability rules for elemental operations	156
10.1.11	Specification expression	156
10.1.12	Constant expression	158
10.2	Assignment	159
10.2.1	Assignment statement	159
10.2.2	Pointer assignment	164
10.2.3	Masked array assignment – WHERE	168

10.2.4	FORALL	170
11	Execution control	173
11.1	Executable constructs containing blocks	173
11.1.1	Blocks	173
11.1.2	Rules governing blocks	173
11.1.3	ASSOCIATE construct	174
11.1.4	BLOCK construct	175
11.1.5	CHANGE TEAM construct	177
11.1.6	CRITICAL construct	179
11.1.7	DO construct	180
11.1.8	IF construct and statement	187
11.1.9	SELECT CASE construct	188
11.1.10	SELECT RANK construct	191
11.1.11	SELECT TYPE construct	193
11.1.12	EXIT statement	196
11.2	Branching	196
11.2.1	Branch concepts	196
11.2.2	GO TO statement	196
11.2.3	Computed GO TO statement	197
11.3	CONTINUE statement	197
11.4	STOP and ERROR STOP statements	197
11.5	FAIL IMAGE statement	198
11.6	Image execution control	198
11.6.1	Image control statements	198
11.6.2	Segments	199
11.6.3	SYNC ALL statement	200
11.6.4	SYNC IMAGES statement	201
11.6.5	SYNC MEMORY statement	202
11.6.6	SYNC TEAM statement	203
11.6.7	EVENT POST statement	204
11.6.8	EVENT WAIT statement	204
11.6.9	FORM TEAM statement	204
11.6.10	LOCK and UNLOCK statements	205
11.6.11	STAT= and ERRMSG= specifiers in image control statements	207
12	Input/output statements	210
12.1	Input/output concepts	210
12.2	Records	210
12.2.1	Definition of a record	210
12.2.2	Formatted record	210
12.2.3	Unformatted record	210
12.2.4	Endfile record	211
12.3	External files	211
12.3.1	External file concepts	211
12.3.2	File existence	211
12.3.3	File access	212
12.3.4	File position	214
12.3.5	File storage units	215
12.4	Internal files	216
12.5	File connection	216
12.5.1	Referring to a file	216
12.5.2	Connection modes	217
12.5.3	Unit existence	218
12.5.4	Connection of a file to a unit	218
12.5.5	Preconnection	219

12.5.6	OPEN statement	219
12.5.7	CLOSE statement	223
12.6	Data transfer statements	224
12.6.1	Form of input and output statements	224
12.6.2	Control information list	225
12.6.3	Data transfer input/output list	229
12.6.4	Execution of a data transfer input/output statement	231
12.6.5	Termination of data transfer statements	242
12.7	Waiting on pending data transfer	242
12.7.1	Wait operation	242
12.7.2	WAIT statement	243
12.8	File positioning statements	243
12.8.1	Syntax	243
12.8.2	BACKSPACE statement	244
12.8.3	ENDFILE statement	244
12.8.4	REWIND statement	245
12.9	FLUSH statement	245
12.10	File inquiry statement	246
12.10.1	Forms of the INQUIRE statement	246
12.10.2	Inquiry specifiers	246
12.10.3	Inquire by output list	252
12.11	Error, end-of-record, and end-of-file conditions	253
12.11.1	Occurrence of input/output conditions	253
12.11.2	Error conditions and the ERR= specifier	253
12.11.3	End-of-file condition and the END= specifier	253
12.11.4	End-of-record condition and the EOR= specifier	254
12.11.5	IOSTAT= specifier	254
12.11.6	IOMSG= specifier	255
12.12	Restrictions on input/output statements	255
13	Input/output editing	256
13.1	Format specifications	256
13.2	Explicit format specification methods	256
13.2.1	FORMAT statement	256
13.2.2	Character format specification	256
13.3	Form of a format item list	257
13.3.1	Syntax	257
13.3.2	Edit descriptors	257
13.3.3	Fields	259
13.4	Interaction between input/output list and format	259
13.5	Positioning by format control	260
13.6	Decimal symbol	261
13.7	Data edit descriptors	261
13.7.1	Purpose of data edit descriptors	261
13.7.2	Numeric editing	261
13.7.3	Logical editing	269
13.7.4	Character editing	269
13.7.5	Generalized editing	270
13.7.6	User-defined derived-type editing	271
13.8	Control edit descriptors	271
13.8.1	Position edit descriptors	271
13.8.2	Slash editing	272
13.8.3	Colon editing	272
13.8.4	SS, SP, and S editing	273
13.8.5	P editing	273
13.8.6	BN and BZ editing	273

13.8.7	RU, RD, RZ, RN, RC, and RP editing	273
13.8.8	DC and DP editing	274
13.9	Character string edit descriptors	274
13.10	List-directed formatting	274
13.10.1	Purpose of list-directed formatting	274
13.10.2	Values and value separators	274
13.10.3	List-directed input	275
13.10.4	List-directed output	277
13.11	Namelist formatting	278
13.11.1	Purpose of namelist formatting	278
13.11.2	Name-value subsequences	278
13.11.3	Namelist input	279
13.11.4	Namelist output	282
14	Program units	283
14.1	Main program	283
14.2	Modules	283
14.2.1	Module syntax and semantics	283
14.2.2	The USE statement and use association	284
14.2.3	Submodules	287
14.3	Block data program units	287
15	Procedures	289
15.1	Concepts	289
15.2	Procedure classifications	289
15.2.1	Procedure classification by reference	289
15.2.2	Procedure classification by means of definition	289
15.3	Characteristics	290
15.3.1	Characteristics of procedures	290
15.3.2	Characteristics of dummy arguments	290
15.3.3	Characteristics of function results	290
15.4	Procedure interface	291
15.4.1	Interface and abstract interface	291
15.4.2	Implicit and explicit interfaces	291
15.4.3	Specification of the procedure interface	292
15.5	Procedure reference	300
15.5.1	Syntax of a procedure reference	300
15.5.2	Actual arguments, dummy arguments, and argument association	303
15.5.3	Function reference	314
15.5.4	Subroutine reference	314
15.5.5	Resolving named procedure references	314
15.5.6	Resolving type-bound procedure references	316
15.6	Procedure definition	317
15.6.1	Intrinsic procedure definition	317
15.6.2	Procedures defined by subprograms	317
15.6.3	Definition and invocation of procedures by means other than Fortran	322
15.6.4	Statement function	323
15.7	Pure procedures	323
15.8	Elemental procedures	325
15.8.1	Elemental procedure declaration and interface	325
15.8.2	Elemental function actual arguments and results	326
15.8.3	Elemental subroutine actual arguments	326
16	Intrinsic procedures and modules	327
16.1	Classes of intrinsic procedures	327
16.2	Arguments to intrinsic procedures	327

16.2.1	General rules	327
16.2.2	The shape of array arguments	328
16.2.3	Mask arguments	328
16.2.4	DIM arguments and reduction functions	328
16.3	Bit model	328
16.3.1	General	328
16.3.2	Bit sequence comparisons	329
16.3.3	Bit sequences as arguments to INT and REAL	329
16.4	Numeric models	329
16.5	Atomic subroutines	330
16.6	Collective subroutines	331
16.7	Standard generic intrinsic procedures	332
16.8	Specific names for standard intrinsic functions	338
16.9	Specifications of the standard intrinsic procedures	339
16.9.1	General	339
16.10	Standard intrinsic modules	426
16.10.1	General	426
16.10.2	The ISO_FORTRAN_ENV intrinsic module	426
17	Exceptions and IEEE arithmetic	433
17.1	Overview of IEEE arithmetic support	433
17.2	Derived types, constants, and operators defined in the modules	434
17.3	The exceptions	434
17.4	The rounding modes	436
17.5	Underflow mode	437
17.6	Halting	437
17.7	The floating-point modes and status	438
17.8	Exceptional values	438
17.9	IEEE arithmetic	438
17.10	Summary of the procedures	439
17.11	Specifications of the procedures	441
17.11.1	General	441
17.12	Examples	466
18	Interoperability with C	468
18.1	General	468
18.2	The ISO_C_BINDING intrinsic module	468
18.2.1	Summary of contents	468
18.2.2	Named constants and derived types in the module	468
18.2.3	Procedures in the module	469
18.3	Interoperability between Fortran and C entities	474
18.3.1	Interoperability of intrinsic types	474
18.3.2	Interoperability with C pointer types	475
18.3.3	Interoperability of derived types and C structure types	475
18.3.4	Interoperability of scalar variables	476
18.3.5	Interoperability of array variables	477
18.3.6	Interoperability of procedures and procedure interfaces	477
18.4	C descriptors	480
18.5	The source file ISO_Fortran_binding.h	480
18.5.1	Summary of contents	480
18.5.2	The CFI_dim_t structure type	480
18.5.3	The CFI_cdesc_t structure type	481
18.5.4	Macros and typedefs in ISO_Fortran_binding.h	482
18.5.5	Functions declared in ISO_Fortran_binding.h	484
18.6	Restrictions on C descriptors	492
18.7	Restrictions on formal parameters	492

18.8	Restrictions on lifetimes	492
18.9	Interoperation with C global variables	493
18.9.1	General	493
18.9.2	Binding labels for common blocks and variables	494
18.10	Interoperation with C functions	494
18.10.1	Definition and reference of interoperable procedures	494
18.10.2	Binding labels for procedures	495
18.10.3	Exceptions and IEEE arithmetic procedures	496
18.10.4	Asynchronous communication	496
19	Scope, association, and definition	497
19.1	Scopes, identifiers, and entities	497
19.2	Global identifiers	497
19.3	Local identifiers	498
19.3.1	Classes of local identifiers	498
19.3.2	Local identifiers that are the same as common block names	499
19.3.3	Function results	499
19.3.4	Components, type parameters, and bindings	499
19.3.5	Argument keywords	499
19.4	Statement and construct entities	500
19.5	Association	501
19.5.1	Name association	501
19.5.2	Pointer association	505
19.5.3	Storage association	508
19.5.4	Inheritance association	510
19.5.5	Establishing associations	510
19.6	Definition and undefinition of variables	511
19.6.1	Definition of objects and subobjects	511
19.6.2	Variables that are always defined	511
19.6.3	Variables that are initially defined	511
19.6.4	Variables that are initially undefined	512
19.6.5	Events that cause variables to become defined	512
19.6.6	Events that cause variables to become undefined	514
19.6.7	Variable definition context	516
19.6.8	Pointer association context	516
Annex A	(informative) Processor dependencies	518
Annex B	(informative) Deleted and obsolescent features	524
Annex C	(informative) Extended notes	528
Index		600

Foreword

- 1 ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.
- 2 The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).
- 3 Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).
- 4 Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.
- 5 For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.
- 6 This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.
- 7 This fourth edition cancels and replaces the third edition (ISO 1539-1:2010), which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 1539-1:2010/Cor. 1:2012, ISO/IEC 1539-1:2010/Cor. 2:2013, ISO/IEC 1539-1:2010/Cor. 3:2014, and ISO/IEC 1539-1:2010/Cor. 4:2016. The main changes compared to the previous edition are as follows:
 - The Technical Specifications ISO/IEC TS 29113:2012 and ISO/IEC TS 18508:2015 have been incorporated.
 - Support for IEEE floating-point arithmetic has been updated to ISO/IEC/IEEE 60559:2011.
- 8 A list of all parts in the ISO 1539 series can be found on the ISO website.
- 9 Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

- 1 This document comprises the specification of the base Fortran language, informally known as Fortran 2018. With the limitations noted in 4.3.3, the syntax and semantics of Fortran 2008 are contained entirely within Fortran 2018. Therefore, any standard-conforming Fortran 2008 program not affected by such limitations is a standard-conforming Fortran 2018 program. New features of Fortran 2018 can be compatibly incorporated into such Fortran 2008 programs, with any exceptions indicated in the text of this document.
- 2 Fortran 2018 contains several extensions to Fortran 2008; these are listed below.
 - Data declaration:
Constant properties of an object declared in its *entity-decl* can be used in its *initialization*. The **EQUIVALENCE** and **COMMON** statements and the **block data program unit** have been redundant since Fortran 90 and are now specified to be *obsolescent*. Diagnosis of the appearance of a **PROTECTED TARGET** variable accessed by *use association* as a *data-target* in a structure constructor is required.
 - Data usage and computation:
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*. **FORALL** is now specified to be *obsolescent*. The type and kind of an implied DO variable in an *array constructor* or **DATA** statement can be specified within the constructor or statement. The **SELECT RANK** construct provides structured access to the elements of an *assumed-rank array*. Completing execution of a **BLOCK** construct can cause the *association status of a pointer* with the **PROTECTED** attribute to become undefined. The standard intrinsic operations **<**, **<=**, **>**, and **>=** (also known as **.LT.**, **.LE.**, **.GT.**, and **.GE.**) on IEEE numbers provide *compareSignaling{relation}* operations; the **=** and **/=** operations (also known as **.EQ.** and **.NE.**) provide *compareQuiet{relation}* operations. Finalization of an allocatable subobject during intrinsic assignment has been clarified. The *char-length* in an executable statement is no longer required to be a specification expression.
 - Input/output:
The **SIZE=** specifier can be used with advancing input. It is no longer prohibited to open a file on more than one unit. The value assigned by the **RECL=** specifier in an **INQUIRE** statement has been standardized. The values assigned by the **POS=** and **SIZE=** specifiers in an **INQUIRE** statement for a unit that has pending asynchronous operations have been standardized. The **G0.d** edit descriptor can be used for list items of type **Integer**, **Logical**, and **Character**. The **D**, **E**, **EN**, and **ES** edit descriptors can have a field width of zero, analogous to the **F** edit descriptor. The exponent width *e* in a *data edit descriptor* can be zero, analogous to a field width of zero. Floating-point formatted input accepts hexadecimal-significand numbers that conform to ISO/IEC/IEEE 60559:2011. The **EX** edit descriptor provides hexadecimal-significand formatted output conforming to ISO/IEC/IEEE 60559:2011. An error condition occurs if unacceptable characters are presented for logical or numeric editing during execution of a formatted input statement.
 - Execution control:
The **arithmetic IF** statement has been deleted. Labeled **DO** loops have been redundant since Fortran 90 and are now specified to be *obsolescent*. The **nonblock DO** construct has been deleted. The locality of a variable used in a **DO CONCURRENT** construct can be explicitly specified. The stop code in a **STOP** or **ERROR STOP** statement can be nonconstant. Output of the stop code and exception summary from the **STOP** and **ERROR STOP** statements can be controlled.
 - Intrinsic procedures and modules:
In a reference to the intrinsic function **CMPLX** with an actual argument of type complex, no keyword is needed for a **KIND** argument. 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*. The new intrinsic function **COSHAPE** returns the coshape of a *coarray*. The new intrinsic function **OUT_OF_RANGE** tests whether a numeric value can be safely converted to a different type or kind. The new intrinsic subroutine **RANDOM_INIT** establishes the initial state of the pseudorandom number generator used by **RANDOM_NUMBER**. The new intrinsic function **REDUCE** performs user-specified array reductions. 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. Integer and logical arguments to intrinsic proced-

ures and intrinsic module procedures that were previously required to be of default kind no longer have that requirement, except for `RANDOM_SEED`. Specific names for intrinsic functions are now deemed *obsolescent*. All standard procedures in the intrinsic module `ISO_C_BINDING`, other than `C_F_POINTER`, are now *pure*. The arguments to the intrinsic function `SIGN` can be of different kind. Nonpolymorphic pointer arguments to the intrinsic functions `EXTENDS_TYPE_OF` and `SAME_TYPE_AS` need not have defined pointer association status. 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. Access to error messages from the intrinsic subroutines `GET_COMMAND`, `GET_COMMAND_ARGUMENT`, and `GET_ENVIRONMENT_VARIABLE` is provided by an optional `ERRMSG` argument. The result of `NORM2` for a zero-sized array argument has been clarified.

- Program units and procedures:

The `IMPORT` statement can appear in a contained subprogram or `BLOCK` construct, and can restrict access via host association; diagnosis of violation of the `IMPORT` restrictions is required. The `GENERIC` statement can be used to declare generic interfaces. The number of procedure arguments is used in *generic resolution*. 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. An `IMPLICIT NONE` statement can require explicit declaration of the `EXTERNAL` attribute throughout a *scoping unit* and its contained *scoping units*. A defined operation need not specify `INTENT (IN)` for a dummy argument with the `VALUE` attribute. A defined assignment need not specify `INTENT (IN)` for the second dummy argument if it has the `VALUE` attribute. Procedures that are not declared with an asterisk *type-param-value*, including `ELEMENTAL` procedures, can be invoked recursively by default; the `RECURSIVE` keyword is advisory only. The `NON_RECURSIVE` keyword specifies that a procedure is not recursive. The `ERROR STOP` statement can appear in a *pure subprogram*. A dummy argument of a *pure function* is permitted in a variable definition context, if it has the `VALUE` attribute. A *coarray dummy argument* can be referenced or defined by another *image*.

- Features previously described by ISO/IEC TS 29113:2012:

A dummy data object can assume its rank from its effective argument. A dummy data object can assume the type from its effective argument, without having the ability to perform type selection. An interoperable procedure can have dummy arguments that are *assumed-type* and/or *assumed-rank*. An interoperable procedure can have dummy data objects that are *allocatable*, *assumed-shape*, *optional*, or *pointers*. The *character length* of a dummy data object of an interoperable procedure can be assumed. The argument to `C_LOC` can be a noninteroperable array. The `FPTR` argument to `C_F_POINTER` can be a noninteroperable array pointer. The argument to `C_FUNLOC` can be a noninteroperable procedure. The `FPTR` argument to `C_F_PROCPTR` can be a noninteroperable procedure pointer. There is a new named constant `C_PTRDIFF_T` to provide interoperability with the C type `ptrdiff_t`.

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.

- Changes to the intrinsic modules `IEEE_ARITHMETIC`, `IEEE_EXCEPTIONS`, and `IEEE_FEATURES` for conformance with ISO/IEC/IEEE 60559:2011:

There is a new, optional, rounding mode `IEEE_AWAY`. The new type `IEEE_MODES_TYPE` encapsulates all floating-point modes. Features associated with subnormal numbers can be accessed with functions and types named `...SUBNORMAL...` (the old `...DENORMAL...` names remain). The new function `IEEE_FMA` performs fused multiply-add operations. The function `IEEE_INT` performs rounded conversions to integer type. The new functions `IEEE_MAX_NUM`, `IEEE_MAX_NUM_MAG`, `IEEE_MIN_NUM`, and `IEEE_MIN_NUM_MAG` calculate maximum and minimum numeric values. The new functions `IEEE_NEXT_DOWN` and `IEEE_NEXT_UP` return the adjacent machine numbers. 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. The new functions `IEEE_SIGNALING_EQ`, `IEEE_SIGNALING_GE`, `IEEE_SIGNALING_GT`, `IEEE_SIGNALING_LE`, `IEEE_SIGNALING_LT`, and `IEEE_SIGNALING_NE` perform signaling comparisons. 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`. The new function `IEEE_`