INTERNATIONAL STANDARD

ISO/IEC 1539-1

Fourth edition
2018-11

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

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

*Partie 1: Langage de base*

Reference number
ISO/IEC 1539-1:2018(E)

# Contents

# 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_PROCPOINTER 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_GE, 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_-

REAL performs rounded conversions to real type. The function IEEE_REM now requires its arguments to have the same radix. The function IEEE_RINT now has a ROUND argument to perform specific rounding. The new function IEEE_SIGNBIT tests the sign bit of an IEEE number.

- Features previously described by ISO/IEC TS 18508:2015:
The CRITICAL statement has optional ERRMSG= and STAT= specifiers. The intrinsic subroutines ATOMIC_DEFINE and ATOMIC_REF have an optional STAT argument. 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. The new intrinsic functions FAILED_IMAGES and STOPPED_IMAGES return indices of images known to have failed or stopped respectively. The new intrinsic function IMAGE_STATUS returns the image execution status of an image. The intrinsic subroutine MOVE_ALLOC has optional ERRMSG and STAT arguments. The intrinsic functions IMAGE_INDEX and NUM_IMAGES have additional forms with a TEAM or TEAM_NUMBER argument. The intrinsic function THIS_IMAGE has an optional TEAM argument. 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. 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 program's 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. 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. 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.
- Changes to features previously described by ISO/IEC TS 18508:2015:
The CHANGE TEAM and SYNC TEAM statements, and the TEAM= specifier on image selectors, permit the team to be specified by an expression. The intrinsic functions FAILED_IMAGES and STOPPED_-IMAGES have no restriction on the kind of their result. 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. 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. The team number for the initial team can be used in image selectors, and in the intrinsic functions NUM_IMAGES and IMAGE_INDEX. 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. All images of the current team are no longer required to execute the same CHANGE TEAM statement. A variable of type TEAM_-TYPE from the intrinsic module ISO_FORTRAN_ENV is not permitted to be a coarray. 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. The intrinsic function UCOBOUND produces a value for the final upper cobound that is always relative to the current team. An EXIT statement can be used to complete execution of a CHANGE TEAM or CRITICAL construct.

3 This document is organized in 19 clauses, dealing with 8 conceptual areas. These 8 areas, and the clauses in which they are treated, are:

| | |
|---|---|
| High/low level concepts | Clauses 4, 5, 6 |
| Data concepts | Clauses 7, 8, 9 |
| Computations | Clauses 10, 16, 17 |
| Execution control | Clause 11 |
| Input/output | Clauses 12, 13 |
| Program units | Clauses 14, 15 |
| Interoperability with C | Clause 18 |
| Scoping and association rules | Clause 19 |

4 It also contains the following nonnormative material:

# Information technology — Programming languages — Fortran —

# Part 1: Base language

## 1 Scope

1 This document specifies the form and establishes the interpretation of programs expressed in the base Fortran language. The purpose of this document is to promote portability, reliability, maintainability, and efficient execution of Fortran programs for use on a variety of computing systems.

2 This document specifies

- the forms that a program written in the Fortran language can take,
- the rules for interpreting the meaning of a program and its data,
- the form of the input data to be processed by such a program, and
- the form of the output data resulting from the use of such a program.

3 Except where stated otherwise, requirements and prohibitions specified by this document apply to programs rather than processors.

4 This document does not specify

- the mechanism by which programs are transformed for use on computing systems,
- the operations required for setup and control of the use of programs on computing systems,
- the method of transcription of programs or their input or output data to or from a storage medium,
- the program and processor behavior when this document fails to establish an interpretation except for the processor detection and reporting requirements in items (2) to (10) of 4.2,
- the maximum number of images, or the size or complexity of a program and its data that will exceed the capacity of any particular computing system or the capability of a particular processor,
- the mechanism for determining the number of images of a program,
- the physical properties of an image or the relationship between images and the computational elements of a computing system,
- the physical properties of the representation of quantities and the method of rounding, approximating, or computing numeric values on a particular processor, except by reference to ISO/IEC/IEEE 60559:2011 under conditions specified in Clause 17,
- the physical properties of input/output records, files, and units, or
- the physical properties and implementation of storage.

# 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 646:1991 (International Reference Version), *Information technology—ISO 7-bit coded character set for information interchange*

ISO/IEC 9899:2011, *Programming languages—C*

ISO/IEC 10646, *Information technology—Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC/IEEE 60559:2011, *Information technology — Microprocessor Systems — Floating-Point arithmetic*