Standard intrinsic module ISO_FORTRAN_ENV

March 8, 2024

1 Name

iso_fortran_env — standard intrinsic module

2 Usage

`USE,INTRINSIC :: ISO_FORTRAN_ENV`

(The '`,INTRINSIC ::`' part is optional, unless there is a user-defined module with the same name.)

3 Synopsis

`ISO_FORTRAN_ENV` provides named constants describing the Fortran environment, along with some derived types for coarray programming, and some utility procedures.

Kind parameter values for specific types are provided by `INT8`, `INT16`, `INT32`, `INT64`, `REAL16`, `REAL32`, `REAL64` and `REAL128`. Arrays listing all valid kind parameter values for each type are provided by `CHARACTER_KINDS`, `INTEGER_KINDS`, `LOGICAL_KINDS` and `REAL_KINDS`.

Storage units are described by `CHARACTER_STORAGE_SIZE` and `NUMERIC_STORAGE_SIZE`.

Input/output units are described by `ERROR_UNIT`, `INPUT_UNIT` and `OUTPUT_UNIT`.

The `RECL=` specifier for the `INQUIRE` and `OPEN` statements is described by `FILE_STORAGE_SIZE`.

`IOSTAT=` return values are described by `IOSTAT_END`, `IOSTAT_EOR` and `IOSTAT_INQUIRE_INTERNAL_UNIT`. These and many other `IOSTAT=` return values are provided by the intrinsic module `F90_IOSTAT` (except that it provides `IOSTAT_EOF` instead of `IOSTAT_END`).

`STAT=` (and `STAT` argument) return values for coarray programming are described by named constants beginning with `STAT_`. `KIND` parameter values for use with atomic subroutines are described by `ATOMIC_INT_KIND` and `ATOMIC_LOGICAL_KIND`. Argument values for the intrinsic function `GET_TEAM` are provided by `CURRENT_TEAM`, `INITIAL_TEAM` and `PARENT_TEAM`. Types for coarray programming are provided by `EVENT_TYPE`, `LOCK_TYPE` and `TEAM_TYPE`.

4 Parameter Descriptions

```
  INTEGER,PARAMETER :: atomic_int_kind = ...
```

The `KIND` value required for the `ATOM` argument of an atomic subroutine that is of type Integer.

```
  INTEGER,PARAMETER :: atomic_logical_kind = ...
```

The `KIND` value required for the `ATOM` argument of an atomic subroutine that is of type Logical.

```
INTEGER,PARAMETER :: character_kinds(4) = [ KIND('A'), &
                                             SELECTED_CHAR_KIND('JIS_0213'), &
                                             SELECTED_CHAR_KIND('UCS_2'), &
                                             SELECTED_CHAR_KIND('ISO_10646') ]
```

Array listing all valid kind type parameter values for `CHARACTER` type.

```
INTEGER,PARAMETER :: character_storage_size = 8
```

The size of a character storage unit in bits.

```
INTEGER,PARAMETER :: current_team = -3
```

Argument value for the intrinsic function `GET_TEAM`, specifying that it should return a team value identifying the current team.

```
INTEGER,PARAMETER :: error_unit = 0
```

The standard error reporting unit number.

```
INTEGER,PARAMETER :: file_storage_size = 8
```

The size of a file storage unit (used by `RECL=` in `OPEN` and `INQUIRE`) in bits.

```
INTEGER,PARAMETER :: initial_team = -1
```

Argument value for the intrinsic function `GET_TEAM`, specifying that it should return a team value identifying the initial team.

```
INTEGER,PARAMETER :: input_unit = 5
```

The standard input unit number. This is the one used by `READ` with an asterisk ('`*`') unit.

```
INTEGER,PARAMETER :: int8 = SELECTED_INT_KIND(2)
```

The kind parameter value for an 8-bit integer.

```
INTEGER,PARAMETER :: int16 = SELECTED_INT_KIND(4)
```

The kind parameter value for a 16-bit integer.

```
INTEGER,PARAMETER :: int32 = SELECTED_INT_KIND(9)
```

The kind parameter value for a 32-bit integer.

```
INTEGER,PARAMETER :: int64 = SELECTED_INT_KIND(18)
```

The kind parameter value for a 64-bit integer.

```
INTEGER,PARAMETER :: integer_kinds(4) = [ int8,int16,int32,int64 ]
```

Array listing all valid kind type parameter values for `INTEGER` type.

```
INTEGER,PARAMETER :: iostat_end = -1
```

The `IOSTAT=` return value for end of file.

```
INTEGER,PARAMETER :: iostat_eor = -2
```

The `IOSTAT=` return value for end of record.

```
INTEGER,PARAMETER :: iostat_inquire_internal_unit = 242
```

The `IOSTAT=` return value for an `INQUIRE` statement within a child i/o procedure that references a unit number that is associated with an internal file.

```
INTEGER,PARAMETER :: logical_kinds(4) = integer_kinds
```

Array listing all valid kind type parameter values for `LOGICAL` type.

```
INTEGER,PARAMETER :: numeric_storage_size = BIT_SIZE(0)
```

The size of a numeric storage unit in bits.

```
INTEGER,PARAMETER :: output_unit = 6
```

The standard output unit number. This is the one used by `PRINT`, and by `WRITE` with an asterisk ('`*`') unit.

```
INTEGER,PARAMETER :: parent_team = -2
```

Argument value for the intrinsic function `GET_TEAM`, specifying that it should return a team value identifying the parent team.

```
INTEGER,PARAMETER :: real_kinds(4) = [ real16,real32,real64,real128 ]
```

Array listing all valid kind type parameter values for `REAL` type.

```
INTEGER,PARAMETER :: real16 = SELECTED_REAL_KIND(3)
```

The kind parameter value for a 16-bit real.

```
INTEGER,PARAMETER :: real32 = SELECTED_REAL_KIND(6)
```

The kind parameter value for a 32-bit real.

```
INTEGER,PARAMETER :: real64 = SELECTED_REAL_KIND(15)
```

The kind parameter value for a 64-bit real.

```
INTEGER,PARAMETER :: real128 = SELECTED_REAL_KIND(30)
```

The kind parameter value for a 128-bit real.

```
INTEGER,PARAMETER :: stat_failed_image = 314
```

The `STAT=` value returned from an image control statement, coindexed object access, or atomic or collective subroutine reference when an image involved has failed.

```
INTEGER,PARAMETER :: stat_locked = 310
```

The `STAT=` value returned from the `LOCK` statement when the lock was already locked by the executing image.

```
INTEGER,PARAMETER :: stat_locked_other_image = 312
```

The `STAT=` value returned from the `UNLOCK` statement when the lock was locked by another image.

```
INTEGER,PARAMETER :: stat_stopped_image = 315
```

The `STAT=` value returned from an image control statement or reference to a collective subroutine if an image being synchronised with has stopped.

```
INTEGER,PARAMETER :: stat_unlocked = 311
```

The `STAT=` value returned from the `UNLOCK` statement when the lock was already unlocked.

```
INTEGER,PARAMETER :: stat_unlocked_failed_image = 313
```

The `STAT=` value returned from the `LOCK` statement when the lock became unlocked due to failure of the image previously holding the lock.

5 Derived Type Descriptions

```
TYPE event_type
  PRIVATE
  ...
END TYPE
```

Type for use by the `EVENT POST` and `EVENT WAIT` statements, and the `EVENT_QUERY` intrinsic function. Named entities of this type must be coarrays. Named entities that have a potential subobject component of this type must also be coarrays. Variables of this type can only be operated on by the special statements provided.

```
TYPE lock_type
  PRIVATE
  ...
END TYPE
```

Type for use by the `LOCK` and `UNLOCK` statements. Named entities of this type must be coarrays. Named entities that have a potential subobject component of this type must also be coarrays. Variables of this type can only be operated on by the special statements provided.

```
TYPE team_type
  PRIVATE
  ...
END TYPE
```

Type for use by the `FORM TEAM` statement and the `CHANGE TEAM` construct, and as a TEAM argument for several related intrinsic functions. Named entities of this type must not be coarrays.

6 Procedure Descriptions

```
PURE CHARACTER(...) FUNCTION compiler_options()
```

Function that returns a character string containing the compiler options used to compile the program unit. This function may be used in constant expressions.

```
PURE CHARACTER(...) FUNCTION compiler_version()
```

Function that returns a character string describing the version of the compiler that was used to compile the program unit. This function may be used in constant expressions.

7 Files

The source code for this module may be found in the NAG Fortran runtime library directory (usually `/usr/local/lib/NAG_Fortran`).

8 See Also

f90_iostat(3), **f90_kind**(3), **nag_modules**(3).

9 Bugs

Please report any bugs found to ‘support@nag.co.uk’ or ‘support@nag.com’, along with any suggestions for improvements.

10 Author

Malcolm Cohen, Nihon Numerical Algorithms Group KK, Tokyo, Japan.