Annex A

Language Bindings Summary

In this section we summarize the specific bindings for C and Fortran. First we present the constants, type definitions, info values and keys. Then we present the routine prototypes separately for each binding. Listings are alphabetical within chapter.

A.1 Defined Values and Handles

A.1.1 Defined Constants

The C and Fortran names are listed below. Constants with the type `const int` may also be implemented as literal integer constants substituted by the preprocessor.

Error classes
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_SUCCESS
MPI_ERR_BUFFER
MPI_ERR_COUNT
MPI_ERR_TYPE
MPI_ERR_TAG
MPI_ERR_COMM
MPI_ERR_RANK
MPI_ERR_REQUEST
MPI_ERR_ROOT
MPI_ERR_GROUP
MPI_ERR_OP
MPI_ERR_TOPOLOGY
MPI_ERR_DIMS
MPI_ERR_ARG
MPI_ERR_UNKNOWN
MPI_ERR_TRUNCATE
MPI_ERR_OTHER
MPI_ERR_INTERN
MPI_ERR_PENDING

(Continued on next page)

Error classes (continued)

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_ERR_IN_STATUS
MPI_ERR_ACCESS
MPI_ERR_AMODE
MPI_ERR_ASSERT
MPI_ERR_BAD_FILE
MPI_ERR_BASE
MPI_ERR_CONVERSION
MPI_ERR_DISP
MPI_ERR_DUP_DATAREP
MPI_ERR_FILE_EXISTS
MPI_ERR_FILE_IN_USE
MPI_ERR_FILE
MPI_ERR_INFO_KEY
MPI_ERR_INFO_NOKEY
MPI_ERR_INFO_VALUE
MPI_ERR_INFO
MPI_ERR_IO
MPI_ERR_KEYVAL
MPI_ERR_LOCKTYPE
MPI_ERR_NAME
MPI_ERR_NO_MEM
MPI_ERR_NOT_SAME
MPI_ERR_NO_SPACE
MPI_ERR_NO_SUCH_FILE
MPI_ERR_PORT
MPI_ERR_QUOTA
MPI_ERR_READ_ONLY
MPI_ERR_RMA_ATTACH
MPI_ERR_RMA_CONFLICT
MPI_ERR_RMA_RANGE
MPI_ERR_RMA_SHARED
MPI_ERR_RMA_SYNC
MPI_ERR_RMA_FLAVOR
MPI_ERR_SERVICE
MPI_ERR_SIZE
MPI_ERR_SPAWN
MPI_ERR_UNSUPPORTED_DATAREP
MPI_ERR_UNSUPPORTED_OPERATION
MPI_ERR_WIN

(Continued on next page)

Error classes (continued)
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_T_ERR_CANNOT_INIT
MPI_T_ERR_NOT_INITIALIZED
MPI_T_ERR_MEMORY
MPI_T_ERR_INVALID
MPI_T_ERR_INVALID_INDEX
MPI_T_ERR_INVALID_ITEM
MPI_T_ERR_INVALID_SESSION
MPI_T_ERR_INVALID_HANDLE
MPI_T_ERR_INVALID_NAME
MPI_T_ERR_OUT_OF_HANDLES
MPI_T_ERR_OUT_OF_SESSIONS
MPI_T_ERR_CVAR_SET_NOT_NOW
MPI_T_ERR_CVAR_SET_NEVER
MPI_T_ERR_PVAR_NO_WRITE
MPI_T_ERR_PVAR_NO_STARTSTOP
MPI_T_ERR_PVAR_NO_ATOMIC
MPI_ERR_LASTCODE

Buffer Address Constants
C type: `void * const`
Fortran type: (predefined memory location)[1]
MPI_BOTTOM
MPI_IN_PLACE

[1] Note that in Fortran these constants are not usable for initialization expressions or assignment. See Section 2.5.4.

Assorted Constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_PROC_NULL
MPI_ANY_SOURCE
MPI_ANY_TAG
MPI_UNDEFINED
MPI_BSEND_OVERHEAD
MPI_KEYVAL_INVALID
MPI_LOCK_EXCLUSIVE
MPI_LOCK_SHARED
MPI_ROOT

No Process Message Handle
C type: `MPI_Message`
Fortran type: `INTEGER` or `TYPE(MPI_Message)`
MPI_MESSAGE_NO_PROC

Fortran Support Method Specific Constants
Fortran type: `LOGICAL`
MPI_SUBARRAYS_SUPPORTED (Fortran only)
MPI_ASYNC_PROTECTS_NONBLOCKING (Fortran only)

Status size and reserved index values (Fortran only)
Fortran type: `INTEGER`
MPI_STATUS_SIZE
MPI_SOURCE
MPI_TAG
MPI_ERROR

Variable Address Size (Fortran only)
Fortran type: `INTEGER`
MPI_ADDRESS_KIND
MPI_COUNT_KIND
MPI_INTEGER_KIND
MPI_OFFSET_KIND

Error-handling specifiers
C type: `MPI_Errhandler`
Fortran type: `INTEGER` or `TYPE(MPI_Errhandler)`
MPI_ERRORS_ARE_FATAL
MPI_ERRORS_RETURN

Maximum Sizes for Strings
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_MAX_DATAREP_STRING
MPI_MAX_ERROR_STRING
MPI_MAX_INFO_KEY
MPI_MAX_INFO_VAL
MPI_MAX_LIBRARY_VERSION_STRING
MPI_MAX_OBJECT_NAME
MPI_MAX_PORT_NAME
MPI_MAX_PROCESSOR_NAME

Named Predefined Datatypes	C types
C type: `MPI_Datatype`	
Fortran type: `INTEGER`	
or `TYPE(MPI_Datatype)`	
MPI_CHAR	`char`
	(treated as printable character)
MPI_SHORT	`signed short int`
MPI_INT	`signed int`
MPI_LONG	`signed long`
MPI_LONG_LONG_INT	`signed long long`
MPI_LONG_LONG (as a synonym)	`signed long long`
MPI_SIGNED_CHAR	`signed char`
	(treated as integral value)
MPI_UNSIGNED_CHAR	`unsigned char`
	(treated as integral value)
MPI_UNSIGNED_SHORT	`unsigned short`
MPI_UNSIGNED	`unsigned int`
MPI_UNSIGNED_LONG	`unsigned long`
MPI_UNSIGNED_LONG_LONG	`unsigned long long`
MPI_FLOAT	`float`
MPI_DOUBLE	`double`
MPI_LONG_DOUBLE	`long double`
MPI_WCHAR	`wchar_t`
	(defined in `<stddef.h>`)
	(treated as printable character)
MPI_C_BOOL	`_Bool`
MPI_INT8_T	`int8_t`
MPI_INT16_T	`int16_t`
MPI_INT32_T	`int32_t`
MPI_INT64_T	`int64_t`
MPI_UINT8_T	`uint8_t`
MPI_UINT16_T	`uint16_t`
MPI_UINT32_T	`uint32_t`
MPI_UINT64_T	`uint64_t`
MPI_AINT	`MPI_Aint`
MPI_COUNT	`MPI_Count`
MPI_OFFSET	`MPI_Offset`
MPI_C_COMPLEX	`float _Complex`
MPI_C_FLOAT_COMPLEX	`float _Complex`
MPI_C_DOUBLE_COMPLEX	`double _Complex`
MPI_C_LONG_DOUBLE_COMPLEX	`long double _Complex`
MPI_BYTE	(any C type)
MPI_PACKED	(any C type)

Named Predefined Datatypes	Fortran types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_INTEGER	INTEGER
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER(1)
MPI_AINT	INTEGER (KIND=MPI_ADDRESS_KIND)
MPI_COUNT	INTEGER (KIND=MPI_COUNT_KIND)
MPI_OFFSET	INTEGER (KIND=MPI_OFFSET_KIND)
MPI_BYTE	(any Fortran type)
MPI_PACKED	(any Fortran type)

Named Predefined Datatypes[1]	C++ types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_CXX_BOOL	bool
MPI_CXX_FLOAT_COMPLEX	std::complex<float>
MPI_CXX_DOUBLE_COMPLEX	std::complex<double>
MPI_CXX_LONG_DOUBLE_COMPLEX	std::complex<long double>

[1] If an accompanying C++ compiler is missing, then the MPI datatypes in this table are not defined.

Optional datatypes (Fortran)	Fortran types
C type: MPI_Datatype Fortran type: INTEGER or TYPE(MPI_Datatype)	
MPI_DOUBLE_COMPLEX	DOUBLE COMPLEX
MPI_INTEGER1	INTEGER*1
MPI_INTEGER2	INTEGER*2
MPI_INTEGER4	INTEGER*4
MPI_INTEGER8	INTEGER*8
MPI_INTEGER16	INTEGER*16
MPI_REAL2	REAL*2
MPI_REAL4	REAL*4
MPI_REAL8	REAL*8
MPI_REAL16	REAL*16
MPI_COMPLEX4	COMPLEX*4
MPI_COMPLEX8	COMPLEX*8
MPI_COMPLEX16	COMPLEX*16
MPI_COMPLEX32	COMPLEX*32

Datatypes for reduction functions (C)
C type: `MPI_Datatype`
Fortran type: `INTEGER` or `TYPE(MPI_Datatype)`
MPI_FLOAT_INT
MPI_DOUBLE_INT
MPI_LONG_INT
MPI_2INT
MPI_SHORT_INT
MPI_LONG_DOUBLE_INT

Datatypes for reduction functions (Fortran)
C type: `MPI_Datatype`
Fortran type: `INTEGER` or `TYPE(MPI_Datatype)`
MPI_2REAL
MPI_2DOUBLE_PRECISION
MPI_2INTEGER

Reserved communicators
C type: `MPI_Comm`
Fortran type: `INTEGER` or `TYPE(MPI_Comm)`
MPI_COMM_WORLD
MPI_COMM_SELF

Communicator split type constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_COMM_TYPE_SHARED

Results of communicator and group comparisons
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_IDENT
MPI_CONGRUENT
MPI_SIMILAR
MPI_UNEQUAL

Environmental inquiry info key
C type: `MPI_Info`
Fortran type: `INTEGER` or `TYPE(MPI_Info)`
MPI_INFO_ENV

Environmental inquiry keys
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_TAG_UB
MPI_IO
MPI_HOST
MPI_WTIME_IS_GLOBAL

Collective Operations

C type: `MPI_Op`
Fortran type: `INTEGER` or `TYPE(MPI_Op)`
MPI_MAX
MPI_MIN
MPI_SUM
MPI_PROD
MPI_MAXLOC
MPI_MINLOC
MPI_BAND
MPI_BOR
MPI_BXOR
MPI_LAND
MPI_LOR
MPI_LXOR
MPI_REPLACE
MPI_NO_OP

Null Handles

C/Fortran name C type / Fortran type
MPI_GROUP_NULL `MPI_Group` / `INTEGER` or `TYPE(MPI_Group)`
MPI_COMM_NULL `MPI_Comm` / `INTEGER` or `TYPE(MPI_Comm)`
MPI_DATATYPE_NULL `MPI_Datatype` / `INTEGER` or `TYPE(MPI_Datatype)`
MPI_REQUEST_NULL `MPI_Request` / `INTEGER` or `TYPE(MPI_Request)`
MPI_OP_NULL `MPI_Op` / `INTEGER` or `TYPE(MPI_Op)`
MPI_ERRHANDLER_NULL `MPI_Errhandler` / `INTEGER` or `TYPE(MPI_Errhandler)`
MPI_FILE_NULL `MPI_File` / `INTEGER` or `TYPE(MPI_File)`
MPI_INFO_NULL `MPI_Info` / `INTEGER` or `TYPE(MPI_Info)`
MPI_WIN_NULL `MPI_Win` / `INTEGER` or `TYPE(MPI_Win)`
MPI_MESSAGE_NULL `MPI_Message` / `INTEGER` or `TYPE(MPI_Message)`

Empty group

C type: `MPI_Group`
Fortran type: `INTEGER` or `TYPE(MPI_Group)`
MPI_GROUP_EMPTY

Topologies
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_GRAPH
MPI_CART
MPI_DIST_GRAPH

Predefined functions

C/Fortran name C type / Fortran type with `mpi` module / Fortran type with `mpi_f08` module
MPI_COMM_NULL_COPY_FN `MPI_Comm_copy_attr_function` / `COMM_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Comm_copy_attr_function)` [1])
MPI_COMM_DUP_FN `MPI_Comm_copy_attr_function` / `COMM_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Comm_copy_attr_function)` [1])
MPI_COMM_NULL_DELETE_FN `MPI_Comm_delete_attr_function` / `COMM_DELETE_ATTR_FUNCTION` / `PROCEDURE(MPI_Comm_delete_attr_function)` [1])
MPI_WIN_NULL_COPY_FN `MPI_Win_copy_attr_function` / `WIN_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Win_copy_attr_function)` [1])
MPI_WIN_DUP_FN `MPI_Win_copy_attr_function` / `WIN_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Win_copy_attr_function)` [1])
MPI_WIN_NULL_DELETE_FN `MPI_Win_delete_attr_function` / `WIN_DELETE_ATTR_FUNCTION` / `PROCEDURE(MPI_Win_delete_attr_function)` [1])
MPI_TYPE_NULL_COPY_FN `MPI_Type_copy_attr_function` / `TYPE_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Type_copy_attr_function)` [1])
MPI_TYPE_DUP_FN `MPI_Type_copy_attr_function` / `TYPE_COPY_ATTR_FUNCTION` / `PROCEDURE(MPI_Type_copy_attr_function)` [1])
MPI_TYPE_NULL_DELETE_FN `MPI_Type_delete_attr_function` / `TYPE_DELETE_ATTR_FUNCTION` / `PROCEDURE(MPI_Type_delete_attr_function)` [1])
MPI_CONVERSION_FN_NULL `MPI_Datarep_conversion_function` / `DATAREP_CONVERSION_FUNCTION` / `PROCEDURE(MPI_Datarep_conversion_function)` [1])

[1] See the advice to implementors (on page 270) and advice to users (on page 270) on the predefined Fortran functions MPI_COMM_NULL_COPY_FN, ... in Section 6.7.2.

Deprecated predefined functions

C/Fortran name C type / Fortran type with `mpi` module
MPI_NULL_COPY_FN `MPI_Copy_function` / `COPY_FUNCTION`
MPI_DUP_FN `MPI_Copy_function` / `COPY_FUNCTION`
MPI_NULL_DELETE_FN `MPI_Delete_function` / `DELETE_FUNCTION`

Predefined Attribute Keys

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_APPNUM
MPI_LASTUSEDCODE
MPI_UNIVERSE_SIZE
MPI_WIN_BASE
MPI_WIN_DISP_UNIT
MPI_WIN_SIZE
MPI_WIN_CREATE_FLAVOR
MPI_WIN_MODEL

MPI Window Create Flavors

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_WIN_FLAVOR_CREATE
MPI_WIN_FLAVOR_ALLOCATE
MPI_WIN_FLAVOR_DYNAMIC
MPI_WIN_FLAVOR_SHARED

MPI Window Models

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_WIN_SEPARATE
MPI_WIN_UNIFIED

Mode Constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_MODE_APPEND
MPI_MODE_CREATE
MPI_MODE_DELETE_ON_CLOSE
MPI_MODE_EXCL
MPI_MODE_NOCHECK
MPI_MODE_NOPRECEDE
MPI_MODE_NOPUT
MPI_MODE_NOSTORE
MPI_MODE_NOSUCCEED
MPI_MODE_RDONLY
MPI_MODE_RDWR
MPI_MODE_SEQUENTIAL
MPI_MODE_UNIQUE_OPEN
MPI_MODE_WRONLY

Datatype Decoding Constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_COMBINER_CONTIGUOUS
MPI_COMBINER_DARRAY
MPI_COMBINER_DUP
MPI_COMBINER_F90_COMPLEX
MPI_COMBINER_F90_INTEGER
MPI_COMBINER_F90_REAL
MPI_COMBINER_HINDEXED
MPI_COMBINER_HVECTOR
MPI_COMBINER_INDEXED_BLOCK
MPI_COMBINER_HINDEXED_BLOCK
MPI_COMBINER_INDEXED
MPI_COMBINER_NAMED
MPI_COMBINER_RESIZED
MPI_COMBINER_STRUCT
MPI_COMBINER_SUBARRAY
MPI_COMBINER_VECTOR

Threads Constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_THREAD_FUNNELED
MPI_THREAD_MULTIPLE
MPI_THREAD_SERIALIZED
MPI_THREAD_SINGLE

File Operation Constants, Part 1

C type: `const MPI_Offset` (or unnamed `enum`) Fortran type: `INTEGER (KIND=MPI_OFFSET_KIND)`
MPI_DISPLACEMENT_CURRENT

File Operation Constants, Part 2

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_DISTRIBUTE_BLOCK
MPI_DISTRIBUTE_CYCLIC
MPI_DISTRIBUTE_DFLT_DARG
MPI_DISTRIBUTE_NONE
MPI_ORDER_C
MPI_ORDER_FORTRAN
MPI_SEEK_CUR
MPI_SEEK_END
MPI_SEEK_SET

F90 Datatype Matching Constants

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_TYPECLASS_COMPLEX
MPI_TYPECLASS_INTEGER
MPI_TYPECLASS_REAL

Constants Specifying Empty or Ignored Input

C/Fortran name C type / Fortran type[1]
MPI_ARGVS_NULL `char***` / 2-dim. array of `CHARACTER*(*)`
MPI_ARGV_NULL `char**` / array of `CHARACTER*(*)`
MPI_ERRCODES_IGNORE `int*` / `INTEGER` array
MPI_STATUSES_IGNORE `MPI_Status*` / `INTEGER, DIMENSION(MPI_STATUS_SIZE,*)` or `TYPE(MPI_Status), DIMENSION(*)`
MPI_STATUS_IGNORE `MPI_Status*` / `INTEGER, DIMENSION(MPI_STATUS_SIZE)` or `TYPE(MPI_Status)`
MPI_UNWEIGHTED `int*` / `INTEGER` array
MPI_WEIGHTS_EMPTY `int*` / `INTEGER` array

[1] Note that in Fortran these constants are not usable for initialization expressions or assignment. See Section 2.5.4.

C Constants Specifying Ignored Input (no Fortran)

C type: `MPI_Fint*`	equivalent to Fortran
MPI_F_STATUSES_IGNORE	MPI_STATUSES_IGNORE in `mpi` / `mpif.h`
MPI_F_STATUS_IGNORE	MPI_STATUS_IGNORE in `mpi` / `mpif.h`
C type: `MPI_F08_status*`	equivalent to Fortran
MPI_F08_STATUSES_IGNORE	MPI_STATUSES_IGNORE in `mpi_f08`
MPI_F08_STATUS_IGNORE	MPI_STATUS_IGNORE in `mpi_f08`

C preprocessor Constants and Fortran Parameters

C type: C-preprocessor macro that expands to an `int` value Fortran type: `INTEGER`
MPI_SUBVERSION
MPI_VERSION

Null handles used in the MPI tool information interface

MPI_T_ENUM_NULL
`MPI_T_enum`
MPI_T_CVAR_HANDLE_NULL
`MPI_T_cvar_handle`
MPI_T_PVAR_HANDLE_NULL
`MPI_T_pvar_handle`
MPI_T_PVAR_SESSION_NULL
`MPI_T_pvar_session`

Verbosity Levels in the MPI tool information interface

C type: `const int` (or unnamed `enum`)
MPI_T_VERBOSITY_USER_BASIC
MPI_T_VERBOSITY_USER_DETAIL
MPI_T_VERBOSITY_USER_ALL
MPI_T_VERBOSITY_TUNER_BASIC
MPI_T_VERBOSITY_TUNER_DETAIL
MPI_T_VERBOSITY_TUNER_ALL
MPI_T_VERBOSITY_MPIDEV_BASIC
MPI_T_VERBOSITY_MPIDEV_DETAIL
MPI_T_VERBOSITY_MPIDEV_ALL

Constants to identify associations of variables in the MPI tool information interface

C type: `const int` (or unnamed `enum`)
MPI_T_BIND_NO_OBJECT
MPI_T_BIND_MPI_COMM
MPI_T_BIND_MPI_DATATYPE
MPI_T_BIND_MPI_ERRHANDLER
MPI_T_BIND_MPI_FILE
MPI_T_BIND_MPI_GROUP
MPI_T_BIND_MPI_OP
MPI_T_BIND_MPI_REQUEST
MPI_T_BIND_MPI_WIN
MPI_T_BIND_MPI_MESSAGE
MPI_T_BIND_MPI_INFO

Constants describing the scope of a control variable in the MPI tool information interface

C type: `const int` (or unnamed `enum`)
MPI_T_SCOPE_CONSTANT
MPI_T_SCOPE_READONLY
MPI_T_SCOPE_LOCAL
MPI_T_SCOPE_GROUP
MPI_T_SCOPE_GROUP_EQ
MPI_T_SCOPE_ALL
MPI_T_SCOPE_ALL_EQ

Additional constants used by the MPI tool information interface

C type: `MPI_T_pvar_handle`
MPI_T_PVAR_ALL_HANDLES

Performance variables classes used by the MPI tool information interface

C type: `const int` (or unnamed `enum`)
MPI_T_PVAR_CLASS_STATE
MPI_T_PVAR_CLASS_LEVEL
MPI_T_PVAR_CLASS_SIZE
MPI_T_PVAR_CLASS_PERCENTAGE
MPI_T_PVAR_CLASS_HIGHWATERMARK
MPI_T_PVAR_CLASS_LOWWATERMARK
MPI_T_PVAR_CLASS_COUNTER
MPI_T_PVAR_CLASS_AGGREGATE
MPI_T_PVAR_CLASS_TIMER
MPI_T_PVAR_CLASS_GENERIC

A.1.2 Types

The following are defined C type definitions, included in the file `mpi.h`.

```
/* C opaque types */
MPI_Aint
MPI_Count
MPI_Fint
MPI_Offset
MPI_Status
MPI_F08_status

/* C handles to assorted structures */
MPI_Comm
MPI_Datatype
MPI_Errhandler
MPI_File
MPI_Group
MPI_Info
MPI_Message
MPI_Op
MPI_Request
MPI_Win

/* Types for the MPI_T interface */
MPI_T_enum
MPI_T_cvar_handle
MPI_T_pvar_handle
MPI_T_pvar_session
```

The following are defined Fortran type definitions, included in the `mpi_f08` and `mpi` modules.

```
!  Fortran opaque types in the mpi_f08 and mpi modules
TYPE(MPI_Status)

!  Fortran handles in the mpi_f08 and mpi modules
TYPE(MPI_Comm)
TYPE(MPI_Datatype)
TYPE(MPI_Errhandler)
TYPE(MPI_File)
TYPE(MPI_Group)
TYPE(MPI_Info)
TYPE(MPI_Message)
TYPE(MPI_Op)
TYPE(MPI_Request)
TYPE(MPI_Win)
```

A.1.3 Prototype Definitions

C Bindings

The following are defined C typedefs for user-defined functions, also included in the file `mpi.h`.

```
/* prototypes for user-defined functions */
typedef void MPI_User_function(void *invec, void *inoutvec, int *len,
              MPI_Datatype *datatype);

typedef int MPI_Comm_copy_attr_function(MPI_Comm oldcomm,
              int comm_keyval, void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag);
typedef int MPI_Comm_delete_attr_function(MPI_Comm comm,
              int comm_keyval, void *attribute_val, void *extra_state);

typedef int MPI_Win_copy_attr_function(MPI_Win oldwin, int win_keyval,
              void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag);
typedef int MPI_Win_delete_attr_function(MPI_Win win, int win_keyval,
              void *attribute_val, void *extra_state);

typedef int MPI_Type_copy_attr_function(MPI_Datatype oldtype,
              int type_keyval, void *extra_state,
              void *attribute_val_in, void *attribute_val_out, int *flag);
typedef int MPI_Type_delete_attr_function(MPI_Datatype datatype,
              int type_keyval, void *attribute_val, void *extra_state);

typedef void MPI_Comm_errhandler_function(MPI_Comm *, int *, ...);
typedef void MPI_Win_errhandler_function(MPI_Win *, int *, ...);
typedef void MPI_File_errhandler_function(MPI_File *, int *, ...);

typedef int MPI_Grequest_query_function(void *extra_state,
            MPI_Status *status);
typedef int MPI_Grequest_free_function(void *extra_state);
typedef int MPI_Grequest_cancel_function(void *extra_state, int complete);

typedef int MPI_Datarep_extent_function(MPI_Datatype datatype,
            MPI_Aint *file_extent, void *extra_state);
typedef int MPI_Datarep_conversion_function(void *userbuf,
            MPI_Datatype datatype, int count, void *filebuf,
            MPI_Offset position, void *extra_state);
```

Fortran 2008 Bindings with the mpi_f08 Module

The callback prototypes when using the Fortran `mpi_f08` module are shown below:

The user-function argument to MPI_Op_create should be declared according to:

```
ABSTRACT INTERFACE
```

```
  SUBROUTINE MPI_User_function(invec, inoutvec, len, datatype)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      TYPE(C_PTR), VALUE ::  invec, inoutvec
      INTEGER ::  len
      TYPE(MPI_Datatype) ::  datatype
```

The copy and delete function arguments to MPI_Comm_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_copy_attr_function(oldcomm, comm_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Comm) ::  oldcomm
      INTEGER ::  comm_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in,
      attribute_val_out
      LOGICAL ::  flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_delete_attr_function(comm, comm_keyval,
  attribute_val, extra_state, ierror)
      TYPE(MPI_Comm) ::  comm
      INTEGER ::  comm_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
```

The copy and delete function arguments to MPI_Win_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_copy_attr_function(oldwin, win_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Win) ::  oldwin
      INTEGER ::  win_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in,
      attribute_val_out
      LOGICAL ::  flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_delete_attr_function(win, win_keyval, attribute_val,
  extra_state, ierror)
      TYPE(MPI_Win) ::  win
      INTEGER ::  win_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
```

The copy and delete function arguments to MPI_Type_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Type_copy_attr_function(oldtype, type_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Datatype) ::  oldtype
      INTEGER ::  type_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in,
```

```
      attribute_val_out
      LOGICAL ::  flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Type_delete_attr_function(datatype, type_keyval,
  attribute_val, extra_state, ierror)
      TYPE(MPI_Datatype) ::  datatype
      INTEGER ::  type_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
```

The handler-function argument to MPI_Comm_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_errhandler_function(comm, error_code)
      TYPE(MPI_Comm) ::  comm
      INTEGER ::  error_code
```

The handler-function argument to MPI_Win_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_errhandler_function(win, error_code)
      TYPE(MPI_Win) ::  win
      INTEGER ::  error_code
```

The handler-function argument to MPI_File_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_File_errhandler_function(file, error_code)
      TYPE(MPI_File) ::  file
      INTEGER ::  error_code
```

The query, free, and cancel function arguments to MPI_Grequest_start should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_query_function(extra_state, status, ierror)
      TYPE(MPI_Status) ::  status
      INTEGER ::  ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state

ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_free_function(extra_state, ierror)
      INTEGER ::  ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state

ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_cancel_function(extra_state, complete, ierror)
      INTEGER ::  ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state
      LOGICAL ::  complete
```

The extent and conversion function arguments to MPI_Register_datarep should be de-

clared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Datarep_extent_function(datatype, extent, extra_state,
  ierror)
      TYPE(MPI_Datatype) ::  datatype
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extent, extra_state
      INTEGER ::  ierror

ABSTRACT INTERFACE
  SUBROUTINE MPI_Datarep_conversion_function(userbuf, datatype, count,
  filebuf, position, extra_state, ierror)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      TYPE(C_PTR), VALUE ::  userbuf, filebuf
      TYPE(MPI_Datatype) ::  datatype
      INTEGER ::  count, ierror
      INTEGER(KIND=MPI_OFFSET_KIND) ::  position
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state
```

Fortran Bindings with mpif.h or the mpi Module

With the Fortran `mpi` module or `mpif.h`, here are examples of how each of the user-defined subroutines should be declared.

The user-function argument to MPI_OP_CREATE should be declared like this:

```
SUBROUTINE USER_FUNCTION(INVEC, INOUTVEC, LEN, DATATYPE)
   <type> INVEC(LEN), INOUTVEC(LEN)
   INTEGER LEN, DATATYPE
```

The copy and delete function arguments to MPI_COMM_CREATE_KEYVAL should be declared like these:

```
SUBROUTINE COMM_COPY_ATTR_FUNCTION(OLDCOMM, COMM_KEYVAL, EXTRA_STATE,
              ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
   INTEGER OLDCOMM, COMM_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT
   LOGICAL FLAG

SUBROUTINE COMM_DELETE_ATTR_FUNCTION(COMM, COMM_KEYVAL, ATTRIBUTE_VAL,
              EXTRA_STATE, IERROR)
   INTEGER COMM, COMM_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
```

The copy and delete function arguments to MPI_WIN_CREATE_KEYVAL should be declared like these:

```
SUBROUTINE WIN_COPY_ATTR_FUNCTION(OLDWIN, WIN_KEYVAL, EXTRA_STATE,
              ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
   INTEGER OLDWIN, WIN_KEYVAL, IERROR
```

```
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
             ATTRIBUTE_VAL_OUT
   LOGICAL FLAG

SUBROUTINE WIN_DELETE_ATTR_FUNCTION(WIN, WIN_KEYVAL, ATTRIBUTE_VAL,
             EXTRA_STATE, IERROR)
   INTEGER WIN, WIN_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
```

The copy and delete function arguments to MPI_TYPE_CREATE_KEYVAL should be declared like these:

```
SUBROUTINE TYPE_COPY_ATTR_FUNCTION(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE,
              ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
   INTEGER OLDTYPE, TYPE_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE,
              ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT
   LOGICAL FLAG

SUBROUTINE TYPE_DELETE_ATTR_FUNCTION(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL,
              EXTRA_STATE, IERROR)
   INTEGER DATATYPE, TYPE_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
```

The handler-function argument to MPI_COMM_CREATE_ERRHANDLER should be declared like this:

```
SUBROUTINE COMM_ERRHANDLER_FUNCTION(COMM, ERROR_CODE)
   INTEGER COMM, ERROR_CODE
```

The handler-function argument to MPI_WIN_CREATE_ERRHANDLER should be declared like this:

```
SUBROUTINE WIN_ERRHANDLER_FUNCTION(WIN, ERROR_CODE)
   INTEGER WIN, ERROR_CODE
```

The handler-function argument to MPI_FILE_CREATE_ERRHANDLER should be declared like this:

```
SUBROUTINE FILE_ERRHANDLER_FUNCTION(FILE, ERROR_CODE)
   INTEGER FILE, ERROR_CODE
```

The query, free, and cancel function arguments to MPI_GREQUEST_START should be declared like these:

```
SUBROUTINE GREQUEST_QUERY_FUNCTION(EXTRA_STATE, STATUS, IERROR)
   INTEGER STATUS(MPI_STATUS_SIZE), IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
```

```
SUBROUTINE GREQUEST_FREE_FUNCTION(EXTRA_STATE, IERROR)
   INTEGER IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

SUBROUTINE GREQUEST_CANCEL_FUNCTION(EXTRA_STATE, COMPLETE, IERROR)
   INTEGER IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
   LOGICAL COMPLETE
```

The extent and conversion function arguments to MPI_REGISTER_DATAREP should be declared like these:

```
SUBROUTINE DATAREP_EXTENT_FUNCTION(DATATYPE, EXTENT, EXTRA_STATE, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTENT, EXTRA_STATE

SUBROUTINE DATAREP_CONVERSION_FUNCTION(USERBUF, DATATYPE, COUNT, FILEBUF,
             POSITION, EXTRA_STATE, IERROR)
    <TYPE> USERBUF(*), FILEBUF(*)
    INTEGER COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) POSITION
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
```

A.1.4 Deprecated Prototype Definitions

The following are defined C typedefs for deprecated user-defined functions, also included in the file `mpi.h`.

```
/* prototypes for user-defined functions */
typedef int MPI_Copy_function(MPI_Comm oldcomm, int keyval,
              void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag);
typedef int MPI_Delete_function(MPI_Comm comm, int keyval,
              void *attribute_val, void *extra_state);
```

The following are deprecated Fortran user-defined callback subroutine prototypes. The deprecated copy and delete function arguments to MPI_KEYVAL_CREATE should be declared like these:

```
SUBROUTINE COPY_FUNCTION(OLDCOMM, KEYVAL, EXTRA_STATE,
               ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERR)
   INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
         ATTRIBUTE_VAL_OUT, IERR
   LOGICAL FLAG

SUBROUTINE DELETE_FUNCTION(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR
```

A.1.5 Info Keys

The following info keys are reserved. They are strings.

access_style
accumulate_ops
accumulate_ordering
alloc_shared_noncontig
appnum
arch
cb_block_size
cb_buffer_size
cb_nodes
chunked_item
chunked_size
chunked
collective_buffering
file_perm
filename
file
host
io_node_list
ip_address
ip_port
nb_proc
no_locks
num_io_nodes
path
same_disp_unit
same_size
soft
striping_factor
striping_unit
wdir

A.1.6 Info Values

The following info values are reserved. They are strings.

false
random
rar
raw
read_mostly
read_once
reverse_sequential
same_op
same_op_no_op
sequential

true
war
waw
write_mostly
write_once

A.2 C Bindings

A.2.1 Point-to-Point Communication C Bindings

```
int MPI_Bsend(const void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm)

int MPI_Bsend_init(const void* buf, int count, MPI_Datatype datatype,
              int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Buffer_attach(void* buffer, int size)

int MPI_Buffer_detach(void* buffer_addr, int* size)

int MPI_Cancel(MPI_Request *request)

int MPI_Get_count(const MPI_Status *status, MPI_Datatype datatype,
              int *count)

int MPI_Ibsend(const void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Improbe(int source, int tag, MPI_Comm comm, int *flag,
              MPI_Message *message, MPI_Status *status)

int MPI_Imrecv(void* buf, int count, MPI_Datatype datatype,
              MPI_Message *message, MPI_Request *request)

int MPI_Iprobe(int source, int tag, MPI_Comm comm, int *flag,
              MPI_Status *status)

int MPI_Irecv(void* buf, int count, MPI_Datatype datatype, int source,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Irsend(const void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Isend(const void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Issend(const void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Mprobe(int source, int tag, MPI_Comm comm, MPI_Message *message,
              MPI_Status *status)

int MPI_Mrecv(void* buf, int count, MPI_Datatype datatype,
              MPI_Message *message, MPI_Status *status)

int MPI_Probe(int source, int tag, MPI_Comm comm, MPI_Status *status)

int MPI_Recv_init(void* buf, int count, MPI_Datatype datatype, int source,
              int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source,
              int tag, MPI_Comm comm, MPI_Status *status)
```

```
int MPI_Request_free(MPI_Request *request)

int MPI_Request_get_status(MPI_Request request, int *flag,
             MPI_Status *status)

int MPI_Rsend(const void* buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)

int MPI_Rsend_init(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Send(const void* buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)

int MPI_Send_init(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
             int dest, int sendtag, void *recvbuf, int recvcount,
             MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm,
             MPI_Status *status)

int MPI_Sendrecv_replace(void* buf, int count, MPI_Datatype datatype,
             int dest, int sendtag, int source, int recvtag, MPI_Comm comm,
             MPI_Status *status)

int MPI_Ssend(const void* buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)

int MPI_Ssend_init(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Startall(int count, MPI_Request array_of_requests[])

int MPI_Start(MPI_Request *request)

int MPI_Testall(int count, MPI_Request array_of_requests[], int *flag,
             MPI_Status array_of_statuses[])

int MPI_Testany(int count, MPI_Request array_of_requests[], int *index,
             int *flag, MPI_Status *status)

int MPI_Test_cancelled(const MPI_Status *status, int *flag)

int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status)

int MPI_Testsome(int incount, MPI_Request array_of_requests[],
             int *outcount, int array_of_indices[],
             MPI_Status array_of_statuses[])

int MPI_Waitall(int count, MPI_Request array_of_requests[],
             MPI_Status array_of_statuses[])

int MPI_Waitany(int count, MPI_Request array_of_requests[], int *index,
             MPI_Status *status)
```

```
int MPI_Wait(MPI_Request *request, MPI_Status *status)

int MPI_Waitsome(int incount, MPI_Request array_of_requests[],
             int *outcount, int array_of_indices[],
             MPI_Status array_of_statuses[])
```

A.2.2 Datatypes C Bindings

```
int MPI_Get_address(const void *location, MPI_Aint *address)

int MPI_Get_elements(const MPI_Status *status, MPI_Datatype datatype,
             int *count)

int MPI_Get_elements_x(const MPI_Status *status, MPI_Datatype datatype,
             MPI_Count *count)

int MPI_Pack(const void* inbuf, int incount, MPI_Datatype datatype,
             void *outbuf, int outsize, int *position, MPI_Comm comm)

int MPI_Pack_external(const char datarep[], const void *inbuf, int incount,
             MPI_Datatype datatype, void *outbuf, MPI_Aint outsize,
             MPI_Aint *position)

int MPI_Pack_external_size(const char datarep[], int incount,
             MPI_Datatype datatype, MPI_Aint *size)

int MPI_Pack_size(int incount, MPI_Datatype datatype, MPI_Comm comm,
             int *size)

int MPI_Type_commit(MPI_Datatype *datatype)

int MPI_Type_contiguous(int count, MPI_Datatype oldtype,
             MPI_Datatype *newtype)

int MPI_Type_create_darray(int size, int rank, int ndims, const
             int array_of_gsizes[], const int array_of_distribs[], const
             int array_of_dargs[], const int array_of_psizes[], int order,
             MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_create_hindexed_block(int count, int blocklength, const
             MPI_Aint array_of_displacements[], MPI_Datatype oldtype,
             MPI_Datatype *newtype)

int MPI_Type_create_hindexed(int count, const int array_of_blocklengths[],
             const MPI_Aint array_of_displacements[], MPI_Datatype oldtype,
             MPI_Datatype *newtype)

int MPI_Type_create_hvector(int count, int blocklength, MPI_Aint stride,
             MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_create_indexed_block(int count, int blocklength, const
             int array_of_displacements[], MPI_Datatype oldtype,
             MPI_Datatype *newtype)
```

```
int MPI_Type_create_resized(MPI_Datatype oldtype, MPI_Aint lb, MPI_Aint
              extent, MPI_Datatype *newtype)

int MPI_Type_create_struct(int count, const int array_of_blocklengths[],
              const MPI_Aint array_of_displacements[], const
              MPI_Datatype array_of_types[], MPI_Datatype *newtype)

int MPI_Type_create_subarray(int ndims, const int array_of_sizes[], const
              int array_of_subsizes[], const int array_of_starts[], int
              order, MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_dup(MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_free(MPI_Datatype *datatype)

int MPI_Type_get_contents(MPI_Datatype datatype, int max_integers,
              int max_addresses, int max_datatypes, int array_of_integers[],
              MPI_Aint array_of_addresses[],
              MPI_Datatype array_of_datatypes[])

int MPI_Type_get_envelope(MPI_Datatype datatype, int *num_integers,
              int *num_addresses, int *num_datatypes, int *combiner)

int MPI_Type_get_extent(MPI_Datatype datatype, MPI_Aint *lb,
              MPI_Aint *extent)

int MPI_Type_get_extent_x(MPI_Datatype datatype, MPI_Count *lb,
              MPI_Count *extent)

int MPI_Type_get_true_extent(MPI_Datatype datatype, MPI_Aint *true_lb,
              MPI_Aint *true_extent)

int MPI_Type_get_true_extent_x(MPI_Datatype datatype, MPI_Count *true_lb,
              MPI_Count *true_extent)

int MPI_Type_indexed(int count, const int array_of_blocklengths[], const
              int array_of_displacements[], MPI_Datatype oldtype,
              MPI_Datatype *newtype)

int MPI_Type_size(MPI_Datatype datatype, int *size)

int MPI_Type_size_x(MPI_Datatype datatype, MPI_Count *size)

int MPI_Type_vector(int count, int blocklength, int stride,
              MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Unpack(const void* inbuf, int insize, int *position, void *outbuf,
              int outcount, MPI_Datatype datatype, MPI_Comm comm)

int MPI_Unpack_external(const char datarep[], const void *inbuf,
              MPI_Aint insize, MPI_Aint *position, void *outbuf,
              int outcount, MPI_Datatype datatype)

MPI_Aint MPI_Aint_add(MPI_Aint base, MPI_Aint disp)

MPI_Aint MPI_Aint_diff(MPI_Aint addr1, MPI_Aint addr2)
```

A.2.3 Collective Communication C Bindings

```
int MPI_Allgather(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, int recvcount,
             MPI_Datatype recvtype, MPI_Comm comm)

int MPI_Allgatherv(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
             const int displs[], MPI_Datatype recvtype, MPI_Comm comm)

int MPI_Allreduce(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, int recvcount, MPI_Datatype recvtype,
             MPI_Comm comm)

int MPI_Alltoallv(const void* sendbuf, const int sendcounts[], const
             int sdispls[], MPI_Datatype sendtype, void* recvbuf, const
             int recvcounts[], const int rdispls[], MPI_Datatype recvtype,
             MPI_Comm comm)

int MPI_Alltoallw(const void* sendbuf, const int sendcounts[], const
             int sdispls[], const MPI_Datatype sendtypes[], void* recvbuf,
             const int recvcounts[], const int rdispls[], const
             MPI_Datatype recvtypes[], MPI_Comm comm)

int MPI_Barrier(MPI_Comm comm)

int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype, int root,
             MPI_Comm comm)

int MPI_Exscan(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

int MPI_Gather(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
             MPI_Comm comm)

int MPI_Gatherv(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, const int recvcounts[], const int displs[],
             MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Iallgather(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, int recvcount,
             MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)

int MPI_Iallgatherv(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
             const int displs[], MPI_Datatype recvtype, MPI_Comm comm,
             MPI_Request* request)

int MPI_Iallreduce(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
```

```
            MPI_Request *request)

int MPI_Ialltoall(const void* sendbuf, int sendcount,
            MPI_Datatype sendtype, void* recvbuf, int recvcount,
            MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)

int MPI_Ialltoallv(const void* sendbuf, const int sendcounts[], const
            int sdispls[], MPI_Datatype sendtype, void* recvbuf, const
            int recvcounts[], const int rdispls[], MPI_Datatype recvtype,
            MPI_Comm comm, MPI_Request *request)

int MPI_Ialltoallw(const void* sendbuf, const int sendcounts[], const
            int sdispls[], const MPI_Datatype sendtypes[], void* recvbuf,
            const int recvcounts[], const int rdispls[], const
            MPI_Datatype recvtypes[], MPI_Comm comm, MPI_Request *request)

int MPI_Ibarrier(MPI_Comm comm, MPI_Request *request)

int MPI_Ibcast(void* buffer, int count, MPI_Datatype datatype, int root,
            MPI_Comm comm, MPI_Request *request)

int MPI_Iexscan(const void* sendbuf, void* recvbuf, int count,
            MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
            MPI_Request *request)

int MPI_Igather(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
            void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
            MPI_Comm comm, MPI_Request *request)

int MPI_Igatherv(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
            void* recvbuf, const int recvcounts[], const int displs[],
            MPI_Datatype recvtype, int root, MPI_Comm comm,
            MPI_Request *request)

int MPI_Ireduce(const void* sendbuf, void* recvbuf, int count,
            MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm,
            MPI_Request *request)

int MPI_Ireduce_scatter_block(const void* sendbuf, void* recvbuf,
            int recvcount, MPI_Datatype datatype, MPI_Op op,
            MPI_Comm comm, MPI_Request *request)

int MPI_Ireduce_scatter(const void* sendbuf, void* recvbuf, const
            int recvcounts[], MPI_Datatype datatype, MPI_Op op,
            MPI_Comm comm, MPI_Request *request)

int MPI_Iscan(const void* sendbuf, void* recvbuf, int count,
            MPI_Datatype datatype, MPI_Op op, MPI_Comm comm,
            MPI_Request *request)

int MPI_Iscatter(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
            void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
            MPI_Comm comm, MPI_Request *request)
```

```
int MPI_Iscatterv(const void* sendbuf, const int sendcounts[], const
             int displs[], MPI_Datatype sendtype, void* recvbuf,
             int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm,
             MPI_Request *request)

int MPI_Op_commutative(MPI_Op op, int *commute)

int MPI_Op_create(MPI_User_function* user_fn, int commute, MPI_Op* op)

int MPI_Op_free(MPI_Op *op)

int MPI_Reduce(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

int MPI_Reduce_local(const void* inbuf, void* inoutbuf, int count,
             MPI_Datatype datatype, MPI_Op op)

int MPI_Reduce_scatter_block(const void* sendbuf, void* recvbuf,
             int recvcount, MPI_Datatype datatype, MPI_Op op,
             MPI_Comm comm)

int MPI_Reduce_scatter(const void* sendbuf, void* recvbuf, const
             int recvcounts[], MPI_Datatype datatype, MPI_Op op,
             MPI_Comm comm)

int MPI_Scan(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

int MPI_Scatter(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
             void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,
             MPI_Comm comm)

int MPI_Scatterv(const void* sendbuf, const int sendcounts[], const
             int displs[], MPI_Datatype sendtype, void* recvbuf,
             int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

A.2.4 Groups, Contexts, Communicators, and Caching C Bindings

```
int MPI_Comm_compare(MPI_Comm comm1, MPI_Comm comm2, int *result)

int MPI_Comm_create_group(MPI_Comm comm, MPI_Group group, int tag,
             MPI_Comm *newcomm)

int MPI_Comm_create_keyval(MPI_Comm_copy_attr_function *comm_copy_attr_fn,
             MPI_Comm_delete_attr_function *comm_delete_attr_fn,
             int *comm_keyval, void *extra_state)

int MPI_Comm_create(MPI_Comm comm, MPI_Group group, MPI_Comm *newcomm)

int MPI_Comm_delete_attr(MPI_Comm comm, int comm_keyval)

int MPI_COMM_DUP_FN(MPI_Comm oldcomm, int comm_keyval, void *extra_state,
             void *attribute_val_in, void *attribute_val_out, int *flag)

int MPI_Comm_dup(MPI_Comm comm, MPI_Comm *newcomm)
```

```
int MPI_Comm_dup_with_info(MPI_Comm comm, MPI_Info info, MPI_Comm *newcomm)

int MPI_Comm_free_keyval(int *comm_keyval)

int MPI_Comm_free(MPI_Comm *comm)

int MPI_Comm_get_attr(MPI_Comm comm, int comm_keyval, void *attribute_val,
             int *flag)

int MPI_Comm_get_info(MPI_Comm comm, MPI_Info *info_used)

int MPI_Comm_get_name(MPI_Comm comm, char *comm_name, int *resultlen)

int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)

int MPI_Comm_idup(MPI_Comm comm, MPI_Comm *newcomm, MPI_Request *request)

int MPI_COMM_NULL_COPY_FN(MPI_Comm oldcomm, int comm_keyval,
             void *extra_state, void *attribute_val_in,
             void *attribute_val_out, int *flag)

int MPI_COMM_NULL_DELETE_FN(MPI_Comm comm, int comm_keyval, void
             *attribute_val, void *extra_state)

int MPI_Comm_rank(MPI_Comm comm, int *rank)

int MPI_Comm_remote_group(MPI_Comm comm, MPI_Group *group)

int MPI_Comm_remote_size(MPI_Comm comm, int *size)

int MPI_Comm_set_attr(MPI_Comm comm, int comm_keyval, void *attribute_val)

int MPI_Comm_set_info(MPI_Comm comm, MPI_Info info)

int MPI_Comm_set_name(MPI_Comm comm, const char *comm_name)

int MPI_Comm_size(MPI_Comm comm, int *size)

int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)

int MPI_Comm_split_type(MPI_Comm comm, int split_type, int key,
             MPI_Info info, MPI_Comm *newcomm)

int MPI_Comm_test_inter(MPI_Comm comm, int *flag)

int MPI_Group_compare(MPI_Group group1,MPI_Group group2, int *result)

int MPI_Group_difference(MPI_Group group1, MPI_Group group2,
             MPI_Group *newgroup)

int MPI_Group_excl(MPI_Group group, int n, const int ranks[],
             MPI_Group *newgroup)

int MPI_Group_free(MPI_Group *group)

int MPI_Group_incl(MPI_Group group, int n, const int ranks[],
             MPI_Group *newgroup)
```

```
int MPI_Group_intersection(MPI_Group group1, MPI_Group group2,
              MPI_Group *newgroup)

int MPI_Group_range_excl(MPI_Group group, int n, int ranges[][3],
              MPI_Group *newgroup)

int MPI_Group_range_incl(MPI_Group group, int n, int ranges[][3],
              MPI_Group *newgroup)

int MPI_Group_rank(MPI_Group group, int *rank)

int MPI_Group_size(MPI_Group group, int *size)

int MPI_Group_translate_ranks(MPI_Group group1, int n, const int ranks1[],
              MPI_Group group2, int ranks2[])

int MPI_Group_union(MPI_Group group1, MPI_Group group2,
              MPI_Group *newgroup)

int MPI_Intercomm_create(MPI_Comm local_comm, int local_leader,
              MPI_Comm peer_comm, int remote_leader, int tag,
              MPI_Comm *newintercomm)

int MPI_Intercomm_merge(MPI_Comm intercomm, int high,
              MPI_Comm *newintracomm)

int MPI_Type_create_keyval(MPI_Type_copy_attr_function *type_copy_attr_fn,
              MPI_Type_delete_attr_function *type_delete_attr_fn,
              int *type_keyval, void *extra_state)

int MPI_Type_delete_attr(MPI_Datatype datatype, int type_keyval)

int MPI_TYPE_DUP_FN(MPI_Datatype oldtype, int type_keyval,
              void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag)

int MPI_Type_free_keyval(int *type_keyval)

int MPI_Type_get_attr(MPI_Datatype datatype, int type_keyval, void
              *attribute_val, int *flag)

int MPI_Type_get_name(MPI_Datatype datatype, char *type_name, int
              *resultlen)

int MPI_TYPE_NULL_COPY_FN(MPI_Datatype oldtype, int type_keyval,
              void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag)

int MPI_TYPE_NULL_DELETE_FN(MPI_Datatype datatype, int type_keyval, void
              *attribute_val, void *extra_state)

int MPI_Type_set_attr(MPI_Datatype datatype, int type_keyval,
              void *attribute_val)

int MPI_Type_set_name(MPI_Datatype datatype, const char *type_name)
```

```
int MPI_Win_create_keyval(MPI_Win_copy_attr_function *win_copy_attr_fn,
              MPI_Win_delete_attr_function *win_delete_attr_fn,
              int *win_keyval, void *extra_state)

int MPI_Win_delete_attr(MPI_Win win, int win_keyval)

int MPI_WIN_DUP_FN(MPI_Win oldwin, int win_keyval, void *extra_state,
              void *attribute_val_in, void *attribute_val_out, int *flag)

int MPI_Win_free_keyval(int *win_keyval)

int MPI_Win_get_attr(MPI_Win win, int win_keyval, void *attribute_val,
              int *flag)

int MPI_Win_get_name(MPI_Win win, char *win_name, int *resultlen)

int MPI_WIN_NULL_COPY_FN(MPI_Win oldwin, int win_keyval, void *extra_state,
              void *attribute_val_in, void *attribute_val_out, int *flag)

int MPI_WIN_NULL_DELETE_FN(MPI_Win win, int win_keyval, void
              *attribute_val, void *extra_state)

int MPI_Win_set_attr(MPI_Win win, int win_keyval, void *attribute_val)

int MPI_Win_set_name(MPI_Win win, const char *win_name)
```

A.2.5 Process Topologies C Bindings

```
int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int coords[])

int MPI_Cart_create(MPI_Comm comm_old, int ndims, const int dims[],
              const int periods[], int reorder, MPI_Comm *comm_cart)

int MPI_Cartdim_get(MPI_Comm comm, int *ndims)

int MPI_Cart_get(MPI_Comm comm, int maxdims, int dims[], int periods[],
              int coords[])

int MPI_Cart_map(MPI_Comm comm, int ndims, const int dims[],
              const int periods[], int *newrank)

int MPI_Cart_rank(MPI_Comm comm, const int coords[], int *rank)

int MPI_Cart_shift(MPI_Comm comm, int direction, int disp,
              int *rank_source, int *rank_dest)

int MPI_Cart_sub(MPI_Comm comm, const int remain_dims[], MPI_Comm *newcomm)

int MPI_Dims_create(int nnodes, int ndims, int dims[])

int MPI_Dist_graph_create_adjacent(MPI_Comm comm_old, int indegree,
              const int sources[], const int sourceweights[], int outdegree,
              const int destinations[], const int destweights[],
              MPI_Info info, int reorder, MPI_Comm *comm_dist_graph)

int MPI_Dist_graph_create(MPI_Comm comm_old, int n, const int sources[],
```

```
1               const int degrees[], const int destinations[],
2               const int weights[], MPI_Info info, int reorder,
3               MPI_Comm *comm_dist_graph)
4
5   int MPI_Dist_graph_neighbors_count(MPI_Comm comm, int *indegree,
6               int *outdegree, int *weighted)
7   int MPI_Dist_graph_neighbors(MPI_Comm comm, int maxindegree, int sources[],
8               int sourceweights[], int maxoutdegree, int destinations[],
9               int destweights[])
10
11  int MPI_Graph_create(MPI_Comm comm_old, int nnodes, const int index[],
12              const int edges[], int reorder, MPI_Comm *comm_graph)
13  int MPI_Graphdims_get(MPI_Comm comm, int *nnodes, int *nedges)
14
15  int MPI_Graph_get(MPI_Comm comm, int maxindex, int maxedges, int index[],
16              int edges[])
17  int MPI_Graph_map(MPI_Comm comm, int nnodes, const int index[],
18              const int edges[], int *newrank)
19
20  int MPI_Graph_neighbors_count(MPI_Comm comm, int rank, int *nneighbors)
21
22  int MPI_Graph_neighbors(MPI_Comm comm, int rank, int maxneighbors,
23              int neighbors[])
24  int MPI_Ineighbor_allgather(const void* sendbuf, int sendcount,
25              MPI_Datatype sendtype, void* recvbuf, int recvcount,
26              MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
27
28  int MPI_Ineighbor_allgatherv(const void* sendbuf, int sendcount,
29              MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
30              const int displs[], MPI_Datatype recvtype, MPI_Comm comm,
31              MPI_Request *request)
32  int MPI_Ineighbor_alltoall(const void* sendbuf, int sendcount,
33              MPI_Datatype sendtype, void* recvbuf, int recvcount,
34              MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
35
36  int MPI_Ineighbor_alltoallv(const void* sendbuf, const int sendcounts[],
37              const int sdispls[], MPI_Datatype sendtype, void* recvbuf,
38              const int recvcounts[], const int rdispls[],
39              MPI_Datatype recvtype, MPI_Comm comm, MPI_Request *request)
40  int MPI_Ineighbor_alltoallw(const void* sendbuf, const int sendcounts[],
41              const MPI_Aint sdispls[], const MPI_Datatype sendtypes[],
42              void* recvbuf, const int recvcounts[],
43              const MPI_Aint rdispls[], const MPI_Datatype recvtypes[],
44              MPI_Comm comm, MPI_Request *request)
45
46  int MPI_Neighbor_allgather(const void* sendbuf, int sendcount,
47              MPI_Datatype sendtype, void* recvbuf, int recvcount,
48              MPI_Datatype recvtype, MPI_Comm comm)
```

```
int MPI_Neighbor_allgatherv(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, const int recvcounts[],
             const int displs[], MPI_Datatype recvtype, MPI_Comm comm)

int MPI_Neighbor_alltoall(const void* sendbuf, int sendcount,
             MPI_Datatype sendtype, void* recvbuf, int recvcount,
             MPI_Datatype recvtype, MPI_Comm comm)

int MPI_Neighbor_alltoallv(const void* sendbuf, const int sendcounts[],
             const int sdispls[], MPI_Datatype sendtype, void* recvbuf,
             const int recvcounts[], const int rdispls[],
             MPI_Datatype recvtype, MPI_Comm comm)

int MPI_Neighbor_alltoallw(const void* sendbuf, const int sendcounts[],
             const MPI_Aint sdispls[], const MPI_Datatype sendtypes[],
             void* recvbuf, const int recvcounts[],
             const MPI_Aint rdispls[], const MPI_Datatype recvtypes[],
             MPI_Comm comm)

int MPI_Topo_test(MPI_Comm comm, int *status)
```

A.2.6 MPI Environmental Management C Bindings

```
int MPI_Abort(MPI_Comm comm, int errorcode)

int MPI_Add_error_class(int *errorclass)

int MPI_Add_error_code(int errorclass, int *errorcode)

int MPI_Add_error_string(int errorcode, const char *string)

int MPI_Alloc_mem(MPI_Aint size, MPI_Info info, void *baseptr)

int MPI_Comm_call_errhandler(MPI_Comm comm, int errorcode)

int MPI_Comm_create_errhandler(MPI_Comm_errhandler_function
             *comm_errhandler_fn, MPI_Errhandler *errhandler)

int MPI_Comm_get_errhandler(MPI_Comm comm, MPI_Errhandler *errhandler)

int MPI_Comm_set_errhandler(MPI_Comm comm, MPI_Errhandler errhandler)

int MPI_Errhandler_free(MPI_Errhandler *errhandler)

int MPI_Error_class(int errorcode, int *errorclass)

int MPI_Error_string(int errorcode, char *string, int *resultlen)

int MPI_File_call_errhandler(MPI_File fh, int errorcode)

int MPI_File_create_errhandler(MPI_File_errhandler_function
             *file_errhandler_fn, MPI_Errhandler *errhandler)

int MPI_File_get_errhandler(MPI_File file, MPI_Errhandler *errhandler)

int MPI_File_set_errhandler(MPI_File file, MPI_Errhandler errhandler)
```

```
int MPI_Finalized(int *flag)

int MPI_Finalize(void)

int MPI_Free_mem(void *base)

int MPI_Get_library_version(char *version, int *resultlen)

int MPI_Get_processor_name(char *name, int *resultlen)

int MPI_Get_version(int *version, int *subversion)

int MPI_Initialized(int *flag)

int MPI_Init(int *argc, char ***argv)

int MPI_Win_call_errhandler(MPI_Win win, int errorcode)

int MPI_Win_create_errhandler(MPI_Win_errhandler_function
             *win_errhandler_fn, MPI_Errhandler *errhandler)

int MPI_Win_get_errhandler(MPI_Win win, MPI_Errhandler *errhandler)

int MPI_Win_set_errhandler(MPI_Win win, MPI_Errhandler errhandler)

double MPI_Wtick(void)

double MPI_Wtime(void)
```

A.2.7 The Info Object C Bindings

```
int MPI_Info_create(MPI_Info *info)

int MPI_Info_delete(MPI_Info info, const char *key)

int MPI_Info_dup(MPI_Info info, MPI_Info *newinfo)

int MPI_Info_free(MPI_Info *info)

int MPI_Info_get(MPI_Info info, const char *key, int valuelen, char *value,
             int *flag)

int MPI_Info_get_nkeys(MPI_Info info, int *nkeys)

int MPI_Info_get_nthkey(MPI_Info info, int n, char *key)

int MPI_Info_get_valuelen(MPI_Info info, const char *key, int *valuelen,
             int *flag)

int MPI_Info_set(MPI_Info info, const char *key, const char *value)
```

A.2.8 Process Creation and Management C Bindings

```
int MPI_Close_port(const char *port_name)

int MPI_Comm_accept(const char *port_name, MPI_Info info, int root,
             MPI_Comm comm, MPI_Comm *newcomm)
```

```
int MPI_Comm_connect(const char *port_name, MPI_Info info, int root,
             MPI_Comm comm, MPI_Comm *newcomm)

int MPI_Comm_disconnect(MPI_Comm *comm)

int MPI_Comm_get_parent(MPI_Comm *parent)

int MPI_Comm_join(int fd, MPI_Comm *intercomm)

int MPI_Comm_spawn(const char *command, char *argv[], int maxprocs,
             MPI_Info info, int root, MPI_Comm comm, MPI_Comm *intercomm,
             int array_of_errcodes[])

int MPI_Comm_spawn_multiple(int count, char *array_of_commands[],
             char **array_of_argv[], const int array_of_maxprocs[], const
             MPI_Info array_of_info[], int root, MPI_Comm comm,
             MPI_Comm *intercomm, int array_of_errcodes[])

int MPI_Lookup_name(const char *service_name, MPI_Info info,
             char *port_name)

int MPI_Open_port(MPI_Info info, char *port_name)

int MPI_Publish_name(const char *service_name, MPI_Info info, const
             char *port_name)

int MPI_Unpublish_name(const char *service_name, MPI_Info info, const
             char *port_name)
```

A.2.9 One-Sided Communications C Bindings

```
int MPI_Accumulate(const void *origin_addr, int origin_count,
             MPI_Datatype origin_datatype, int target_rank,
             MPI_Aint target_disp, int target_count,
             MPI_Datatype target_datatype, MPI_Op op, MPI_Win win)

int MPI_Compare_and_swap(const void *origin_addr, const void *compare_addr,
             void *result_addr, MPI_Datatype datatype, int target_rank,
             MPI_Aint target_disp, MPI_Win win)

int MPI_Fetch_and_op(const void *origin_addr, void *result_addr,
             MPI_Datatype datatype, int target_rank, MPI_Aint target_disp,
             MPI_Op op, MPI_Win win)

int MPI_Get_accumulate(const void *origin_addr, int origin_count,
             MPI_Datatype origin_datatype, void *result_addr,
             int result_count, MPI_Datatype result_datatype,
             int target_rank, MPI_Aint target_disp, int target_count,
             MPI_Datatype target_datatype, MPI_Op op, MPI_Win win)

int MPI_Get(void *origin_addr, int origin_count,
             MPI_Datatype origin_datatype, int target_rank,
             MPI_Aint target_disp, int target_count,
```

```
              MPI_Datatype target_datatype, MPI_Win win)

int MPI_Put(const void *origin_addr, int origin_count, MPI_Datatype
              origin_datatype, int target_rank, MPI_Aint target_disp, int
              target_count, MPI_Datatype target_datatype, MPI_Win win)

int MPI_Raccumulate(const void *origin_addr, int origin_count,
              MPI_Datatype origin_datatype, int target_rank,
              MPI_Aint target_disp, int target_count,
              MPI_Datatype target_datatype, MPI_Op op, MPI_Win win,
              MPI_Request *request)

int MPI_Rget_accumulate(const void *origin_addr, int origin_count,
              MPI_Datatype origin_datatype, void *result_addr,
              int result_count, MPI_Datatype result_datatype,
              int target_rank, MPI_Aint target_disp, int target_count,
              MPI_Datatype target_datatype, MPI_Op op, MPI_Win win,
              MPI_Request *request)

int MPI_Rget(void *origin_addr, int origin_count,
              MPI_Datatype origin_datatype, int target_rank,
              MPI_Aint target_disp, int target_count,
              MPI_Datatype target_datatype, MPI_Win win,
              MPI_Request *request)

int MPI_Rput(const void *origin_addr, int origin_count,
              MPI_Datatype origin_datatype, int target_rank,
              MPI_Aint target_disp, int target_count,
              MPI_Datatype target_datatype, MPI_Win win,
              MPI_Request *request)

int MPI_Win_allocate(MPI_Aint size, int disp_unit, MPI_Info info,
              MPI_Comm comm, void *baseptr, MPI_Win *win)

int MPI_Win_allocate_shared(MPI_Aint size, int disp_unit, MPI_Info info,
              MPI_Comm comm, void *baseptr, MPI_Win *win)

int MPI_Win_attach(MPI_Win win, void *base, MPI_Aint size)

int MPI_Win_complete(MPI_Win win)

int MPI_Win_create_dynamic(MPI_Info info, MPI_Comm comm, MPI_Win *win)

int MPI_Win_create(void *base, MPI_Aint size, int disp_unit, MPI_Info info,
              MPI_Comm comm, MPI_Win *win)

int MPI_Win_detach(MPI_Win win, const void *base)

int MPI_Win_fence(int assert, MPI_Win win)

int MPI_Win_flush_all(MPI_Win win)

int MPI_Win_flush(int rank, MPI_Win win)

int MPI_Win_flush_local_all(MPI_Win win)
```

```
int MPI_Win_flush_local(int rank, MPI_Win win)

int MPI_Win_free(MPI_Win *win)

int MPI_Win_get_group(MPI_Win win, MPI_Group *group)

int MPI_Win_get_info(MPI_Win win, MPI_Info *info_used)

int MPI_Win_lock_all(int assert, MPI_Win win)

int MPI_Win_lock(int lock_type, int rank, int assert, MPI_Win win)

int MPI_Win_post(MPI_Group group, int assert, MPI_Win win)

int MPI_Win_set_info(MPI_Win win, MPI_Info info)

int MPI_Win_shared_query(MPI_Win win, int rank, MPI_Aint *size,
             int *disp_unit, void *baseptr)

int MPI_Win_start(MPI_Group group, int assert, MPI_Win win)

int MPI_Win_sync(MPI_Win win)

int MPI_Win_test(MPI_Win win, int *flag)

int MPI_Win_unlock_all(MPI_Win win)

int MPI_Win_unlock(int rank, MPI_Win win)

int MPI_Win_wait(MPI_Win win)
```

A.2.10 External Interfaces C Bindings

```
int MPI_Grequest_complete(MPI_Request request)

int MPI_Grequest_start(MPI_Grequest_query_function *query_fn,
             MPI_Grequest_free_function *free_fn,
             MPI_Grequest_cancel_function *cancel_fn, void *extra_state,
             MPI_Request *request)

int MPI_Init_thread(int *argc, char ***argv, int required, int *provided)

int MPI_Is_thread_main(int *flag)

int MPI_Query_thread(int *provided)

int MPI_Status_set_cancelled(MPI_Status *status, int flag)

int MPI_Status_set_elements(MPI_Status *status, MPI_Datatype datatype,
             int count)

int MPI_Status_set_elements_x(MPI_Status *status, MPI_Datatype datatype,
             MPI_Count count)
```

A.2.11 I/O C Bindings

```
int MPI_CONVERSION_FN_NULL(void *userbuf, MPI_Datatype datatype, int count,
             void *filebuf, MPI_Offset position, void *extra_state)

int MPI_File_close(MPI_File *fh)

int MPI_File_delete(const char *filename, MPI_Info info)

int MPI_File_get_amode(MPI_File fh, int *amode)

int MPI_File_get_atomicity(MPI_File fh, int *flag)

int MPI_File_get_byte_offset(MPI_File fh, MPI_Offset offset,
             MPI_Offset *disp)

int MPI_File_get_group(MPI_File fh, MPI_Group *group)

int MPI_File_get_info(MPI_File fh, MPI_Info *info_used)

int MPI_File_get_position(MPI_File fh, MPI_Offset *offset)

int MPI_File_get_position_shared(MPI_File fh, MPI_Offset *offset)

int MPI_File_get_size(MPI_File fh, MPI_Offset *size)

int MPI_File_get_type_extent(MPI_File fh, MPI_Datatype datatype,
             MPI_Aint *extent)

int MPI_File_get_view(MPI_File fh, MPI_Offset *disp, MPI_Datatype *etype,
             MPI_Datatype *filetype, char *datarep)

int MPI_File_iread_all(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iread_at_all(MPI_File fh, MPI_Offset offset, void *buf,
             int count, MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iread_at(MPI_File fh, MPI_Offset offset, void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iread(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iread_shared(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iwrite_all(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iwrite_at_all(MPI_File fh, MPI_Offset offset, const void *buf,
             int count, MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iwrite_at(MPI_File fh, MPI_Offset offset, const void *buf,
             int count, MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iwrite(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)
```

```
int MPI_File_iwrite_shared(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Request *request)

int MPI_File_open(MPI_Comm comm, const char *filename, int amode,
             MPI_Info info, MPI_File *fh)

int MPI_File_preallocate(MPI_File fh, MPI_Offset size)

int MPI_File_read_all_begin(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype)

int MPI_File_read_all_end(MPI_File fh, void *buf, MPI_Status *status)

int MPI_File_read_all(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_read_at_all_begin(MPI_File fh, MPI_Offset offset, void *buf,
             int count, MPI_Datatype datatype)

int MPI_File_read_at_all_end(MPI_File fh, void *buf, MPI_Status *status)

int MPI_File_read_at_all(MPI_File fh, MPI_Offset offset, void *buf,
             int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_read_at(MPI_File fh, MPI_Offset offset, void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_read(MPI_File fh, void *buf, int count, MPI_Datatype datatype,
             MPI_Status *status)

int MPI_File_read_ordered_begin(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype)

int MPI_File_read_ordered_end(MPI_File fh, void *buf, MPI_Status *status)

int MPI_File_read_ordered(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_read_shared(MPI_File fh, void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_seek(MPI_File fh, MPI_Offset offset, int whence)

int MPI_File_seek_shared(MPI_File fh, MPI_Offset offset, int whence)

int MPI_File_set_atomicity(MPI_File fh, int flag)

int MPI_File_set_info(MPI_File fh, MPI_Info info)

int MPI_File_set_size(MPI_File fh, MPI_Offset size)

int MPI_File_set_view(MPI_File fh, MPI_Offset disp, MPI_Datatype etype,
             MPI_Datatype filetype, const char *datarep, MPI_Info info)

int MPI_File_sync(MPI_File fh)

int MPI_File_write_all_begin(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype)
```

```
int MPI_File_write_all_end(MPI_File fh, const void *buf,
             MPI_Status *status)

int MPI_File_write_all(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_at_all_begin(MPI_File fh, MPI_Offset offset, const
             void *buf, int count, MPI_Datatype datatype)

int MPI_File_write_at_all_end(MPI_File fh, const void *buf,
             MPI_Status *status)

int MPI_File_write_at_all(MPI_File fh, MPI_Offset offset, const void *buf,
             int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_at(MPI_File fh, MPI_Offset offset, const void *buf,
             int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_ordered_begin(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype)

int MPI_File_write_ordered_end(MPI_File fh, const void *buf,
             MPI_Status *status)

int MPI_File_write_ordered(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_shared(MPI_File fh, const void *buf, int count,
             MPI_Datatype datatype, MPI_Status *status)

int MPI_Register_datarep(const char *datarep,
             MPI_Datarep_conversion_function *read_conversion_fn,
             MPI_Datarep_conversion_function *write_conversion_fn,
             MPI_Datarep_extent_function *dtype_file_extent_fn,
             void *extra_state)
```

A.2.12 Language Bindings C Bindings

```
int MPI_Status_f082f(MPI_F08_status *f08_status, MPI_Fint *f_status)

int MPI_Status_f2f08(MPI_Fint *f_status, MPI_F08_status *f08_status)

int MPI_Type_create_f90_complex(int p, int r, MPI_Datatype *newtype)

int MPI_Type_create_f90_integer(int r, MPI_Datatype *newtype)

int MPI_Type_create_f90_real(int p, int r, MPI_Datatype *newtype)

int MPI_Type_match_size(int typeclass, int size, MPI_Datatype *datatype)

MPI_Fint MPI_Comm_c2f(MPI_Comm comm)

MPI_Comm MPI_Comm_f2c(MPI_Fint comm)
```

```
MPI_Fint MPI_Errhandler_c2f(MPI_Errhandler errhandler)

MPI_Errhandler MPI_Errhandler_f2c(MPI_Fint errhandler)

MPI_Fint MPI_File_c2f(MPI_File file)

MPI_File MPI_File_f2c(MPI_Fint file)

MPI_Fint MPI_Group_c2f(MPI_Group group)

MPI_Group MPI_Group_f2c(MPI_Fint group)

MPI_Fint MPI_Info_c2f(MPI_Info info)

MPI_Info MPI_Info_f2c(MPI_Fint info)

MPI_Fint MPI_Message_c2f(MPI_Message message)

MPI_Message MPI_Message_f2c(MPI_Fint message)

MPI_Fint MPI_Op_c2f(MPI_Op op)

MPI_Op MPI_Op_f2c(MPI_Fint op)

MPI_Fint MPI_Request_c2f(MPI_Request request)

MPI_Request MPI_Request_f2c(MPI_Fint request)

int MPI_Status_c2f08(const MPI_Status *c_status, MPI_F08_status
            *f08_status)

int MPI_Status_c2f(const MPI_Status *c_status, MPI_Fint *f_status)

int MPI_Status_f082c(const MPI_F08_status *f08_status, MPI_Status
            *c_status)

int MPI_Status_f2c(const MPI_Fint *f_status, MPI_Status *c_status)

MPI_Fint MPI_Type_c2f(MPI_Datatype datatype)

MPI_Datatype MPI_Type_f2c(MPI_Fint datatype)

MPI_Fint MPI_Win_c2f(MPI_Win win)

MPI_Win MPI_Win_f2c(MPI_Fint win)
```

A.2.13 Tools / Profiling Interface C Bindings

```
int MPI_Pcontrol(const int level, ...)
```

A.2.14 Tools / MPI Tool Information Interface C Bindings

```
int MPI_T_category_changed(int *stamp)

int MPI_T_category_get_categories(int cat_index, int len, int indices[])

int MPI_T_category_get_cvars(int cat_index, int len, int indices[])
```

```
int MPI_T_category_get_index(const char *name, int *cat_index)

int MPI_T_category_get_info(int cat_index, char *name, int *name_len,
              char *desc, int *desc_len, int *num_cvars, int *num_pvars,
              int *num_categories)

int MPI_T_category_get_num(int *num_cat)

int MPI_T_category_get_pvars(int cat_index, int len, int indices[])

int MPI_T_cvar_get_index(const char *name, int *cvar_index)

int MPI_T_cvar_get_info(int cvar_index, char *name, int *name_len, int
              *verbosity, MPI_Datatype *datatype, MPI_T_enum *enumtype, char
              *desc, int *desc_len, int *bind, int *scope)

int MPI_T_cvar_get_num(int *num_cvar)

int MPI_T_cvar_handle_alloc(int cvar_index, void *obj_handle,
              MPI_T_cvar_handle *handle, int *count)

int MPI_T_cvar_handle_free(MPI_T_cvar_handle *handle)

int MPI_T_cvar_read(MPI_T_cvar_handle handle, void* buf)

int MPI_T_cvar_write(MPI_T_cvar_handle handle, const void* buf)

int MPI_T_enum_get_info(MPI_T_enum enumtype, int *num, char *name, int
              *name_len)

int MPI_T_enum_get_item(MPI_T_enum enumtype, int index, int *value, char
              *name, int *name_len)

int MPI_T_finalize(void)

int MPI_T_init_thread(int required, int *provided)

int MPI_T_pvar_get_index(const char *name, int var_class, int *pvar_index)

int MPI_T_pvar_get_info(int pvar_index, char *name, int *name_len,
              int *verbosity, int *var_class, MPI_Datatype *datatype,
              MPI_T_enum *enumtype, char *desc, int *desc_len, int *bind,
              int *readonly, int *continuous, int *atomic)

int MPI_T_pvar_get_num(int *num_pvar)

int MPI_T_pvar_handle_alloc(MPI_T_pvar_session session, int pvar_index,
              void *obj_handle, MPI_T_pvar_handle *handle, int *count)

int MPI_T_pvar_handle_free(MPI_T_pvar_session session, MPI_T_pvar_handle
              *handle)

int MPI_T_pvar_read(MPI_T_pvar_session session, MPI_T_pvar_handle handle,
              void* buf)

int MPI_T_pvar_readreset(MPI_T_pvar_session session, MPI_T_pvar_handle
              handle, void* buf)
```

```
int MPI_T_pvar_reset(MPI_T_pvar_session session, MPI_T_pvar_handle handle)

int MPI_T_pvar_session_create(MPI_T_pvar_session *session)

int MPI_T_pvar_session_free(MPI_T_pvar_session *session)

int MPI_T_pvar_start(MPI_T_pvar_session session, MPI_T_pvar_handle handle)

int MPI_T_pvar_stop(MPI_T_pvar_session session, MPI_T_pvar_handle handle)

int MPI_T_pvar_write(MPI_T_pvar_session session, MPI_T_pvar_handle handle,
             const void* buf)
```

A.2.15 Deprecated C Bindings

```
int MPI_Attr_delete(MPI_Comm comm, int keyval)

int MPI_Attr_get(MPI_Comm comm, int keyval, void *attribute_val, int *flag)

int MPI_Attr_put(MPI_Comm comm, int keyval, void* attribute_val)

int MPI_DUP_FN(MPI_Comm oldcomm, int keyval, void *extra_state,
             void *attribute_val_in, void *attribute_val_out, int *flag)

int MPI_Keyval_create(MPI_Copy_function *copy_fn, MPI_Delete_function
             *delete_fn, int *keyval, void* extra_state)

int MPI_Keyval_free(int *keyval)

int MPI_NULL_COPY_FN(MPI_Comm oldcomm, int keyval, void *extra_state,
             void *attribute_val_in, void *attribute_val_out, int *flag)

int MPI_NULL_DELETE_FN(MPI_Comm comm, int keyval, void *attribute_val,
             void *extra_state)
```

A.3 Fortran 2008 Bindings with the mpi_f08 Module

A.3.1 Point-to-Point Communication Fortran 2008 Bindings

```
MPI_Bsend(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Bsend_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Buffer_attach(buffer, size, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buffer
    INTEGER, INTENT(IN) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Buffer_detach(buffer_addr, size, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    TYPE(C_PTR), INTENT(OUT) ::  buffer_addr
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cancel(request, ierror)
    TYPE(MPI_Request), INTENT(IN) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_count(status, datatype, count, ierror)
    TYPE(MPI_Status), INTENT(IN) ::  status
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(OUT) ::  count
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ibsend(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Improbe(source, tag, comm, flag, message, status, ierror)
    INTEGER, INTENT(IN) ::  source, tag
    TYPE(MPI_Comm), INTENT(IN) ::  comm
```

```
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Message), INTENT(OUT) ::  message
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Imrecv(buf, count, datatype, message, request, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Message), INTENT(INOUT) ::  message
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iprobe(source, tag, comm, flag, status, ierror)
    INTEGER, INTENT(IN) ::  source, tag
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Irecv(buf, count, datatype, source, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, source, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Irsend(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Isend(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Issend(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
```

```
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Mprobe(source, tag, comm, message, status, ierror)
    INTEGER, INTENT(IN) ::  source, tag
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Message), INTENT(OUT) ::  message
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Mrecv(buf, count, datatype, message, status, ierror)
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Message), INTENT(INOUT) ::  message
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Probe(source, tag, comm, status, ierror)
    INTEGER, INTENT(IN) ::  source, tag
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Recv(buf, count, datatype, source, tag, comm, status, ierror)
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count, source, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Recv_init(buf, count, datatype, source, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, source, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Request_free(request, ierror)
    TYPE(MPI_Request), INTENT(INOUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Request_get_status(request, flag, status, ierror)
    TYPE(MPI_Request), INTENT(IN) ::  request
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Rsend(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
```

```
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Rsend_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Send(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Send_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Sendrecv_replace(buf, count, datatype, dest, sendtag, source, recvtag,
            comm, status, ierror)
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count, dest, sendtag, source, recvtag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Sendrecv(sendbuf, sendcount, sendtype, dest, sendtag, recvbuf,
            recvcount, recvtype, source, recvtag, comm, status, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, dest, sendtag, recvcount, source,
    recvtag
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ssend(buf, count, datatype, dest, tag, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
```

```
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ssend_init(buf, count, datatype, dest, tag, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count, dest, tag
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Startall(count, array_of_requests, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(count)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Start(request, ierror)
    TYPE(MPI_Request), INTENT(INOUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Testall(count, array_of_requests, flag, array_of_statuses, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(count)
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Status) ::  array_of_statuses(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Testany(count, array_of_requests, index, flag, status, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(count)
    INTEGER, INTENT(OUT) ::  index
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Test_cancelled(status, flag, ierror)
    TYPE(MPI_Status), INTENT(IN) ::  status
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Test(request, flag, status, ierror)
    TYPE(MPI_Request), INTENT(INOUT) ::  request
    LOGICAL, INTENT(OUT) ::  flag
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Testsome(incount, array_of_requests, outcount, array_of_indices,
             array_of_statuses, ierror)
    INTEGER, INTENT(IN) ::  incount
```

```
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(incount)
    INTEGER, INTENT(OUT) ::  outcount, array_of_indices(*)
    TYPE(MPI_Status) ::  array_of_statuses(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Waitall(count, array_of_requests, array_of_statuses, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(count)
    TYPE(MPI_Status) ::  array_of_statuses(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Waitany(count, array_of_requests, index, status, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(count)
    INTEGER, INTENT(OUT) ::  index
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Wait(request, status, ierror)
    TYPE(MPI_Request), INTENT(INOUT) ::  request
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Waitsome(incount, array_of_requests, outcount, array_of_indices,
             array_of_statuses, ierror)
    INTEGER, INTENT(IN) ::  incount
    TYPE(MPI_Request), INTENT(INOUT) ::  array_of_requests(incount)
    INTEGER, INTENT(OUT) ::  outcount, array_of_indices(*)
    TYPE(MPI_Status) ::  array_of_statuses(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.2 Datatypes Fortran 2008 Bindings

```
INTEGER(KIND=MPI_ADDRESS_KIND) MPI_Aint_add(base, disp)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  base, disp

INTEGER(KIND=MPI_ADDRESS_KIND) MPI_Aint_diff(addr1, addr2)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  addr1, addr2

MPI_Get_address(location, address, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  location
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  address
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_elements(status, datatype, count, ierror)
    TYPE(MPI_Status), INTENT(IN) ::  status
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(OUT) ::  count
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_elements_x(status, datatype, count, ierror)
```

```
    TYPE(MPI_Status), INTENT(IN) ::  status
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) ::  count
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Pack_external(datarep, inbuf, incount, datatype, outbuf, outsize,
             position, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  datarep
    TYPE(*), DIMENSION(..), INTENT(IN) ::  inbuf
    TYPE(*), DIMENSION(..)  ::  outbuf
    INTEGER, INTENT(IN) ::  incount
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  outsize
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(INOUT) ::  position
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Pack_external_size(datarep, incount, datatype, size, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  incount
    CHARACTER(LEN=*), INTENT(IN) ::  datarep
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Pack(inbuf, incount, datatype, outbuf, outsize, position, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  inbuf
    TYPE(*), DIMENSION(..)  ::  outbuf
    INTEGER, INTENT(IN) ::  incount, outsize
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(INOUT) ::  position
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Pack_size(incount, datatype, comm, size, ierror)
    INTEGER, INTENT(IN) ::  incount
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_commit(datatype, ierror)
    TYPE(MPI_Datatype), INTENT(INOUT) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_contiguous(count, oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_darray(size, rank, ndims, array_of_gsizes,
             array_of_distribs, array_of_dargs, array_of_psizes, order,
```

```
              oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  size, rank, ndims, array_of_gsizes(ndims),
    array_of_distribs(ndims), array_of_dargs(ndims),
    array_of_psizes(ndims), order
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_hindexed_block(count, blocklength, array_of_displacements,
              oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count, blocklength
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
    array_of_displacements(count)
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_hindexed(count, array_of_blocklengths,
              array_of_displacements, oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count, array_of_blocklengths(count)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
    array_of_displacements(count)
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_hvector(count, blocklength, stride, oldtype, newtype,
              ierror)
    INTEGER, INTENT(IN) ::  count, blocklength
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  stride
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_indexed_block(count, blocklength, array_of_displacements,
              oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count, blocklength,
    array_of_displacements(count)
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_resized(oldtype, lb, extent, newtype, ierror)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  lb, extent
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_struct(count, array_of_blocklengths,
              array_of_displacements, array_of_types, newtype, ierror)
```

```
    INTEGER, INTENT(IN) ::  count, array_of_blocklengths(count)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::
    array_of_displacements(count)
    TYPE(MPI_Datatype), INTENT(IN) ::  array_of_types(count)
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_subarray(ndims, array_of_sizes, array_of_subsizes,
             array_of_starts, order, oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  ndims, array_of_sizes(ndims),
    array_of_subsizes(ndims), array_of_starts(ndims), order
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_dup(oldtype, newtype, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_free(datatype, ierror)
    TYPE(MPI_Datatype), INTENT(INOUT) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_contents(datatype, max_integers, max_addresses, max_datatypes,
             array_of_integers, array_of_addresses, array_of_datatypes,
             ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  max_integers, max_addresses, max_datatypes
    INTEGER, INTENT(OUT) ::  array_of_integers(max_integers)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::
    array_of_addresses(max_addresses)
    TYPE(MPI_Datatype), INTENT(OUT) ::  array_of_datatypes(max_datatypes)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_envelope(datatype, num_integers, num_addresses, num_datatypes,
             combiner, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(OUT) ::  num_integers, num_addresses, num_datatypes,
    combiner
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_extent(datatype, lb, extent, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  lb, extent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_extent_x(datatype, lb, extent, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) ::  lb, extent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Type_get_true_extent(datatype, true_lb, true_extent, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  true_lb, true_extent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_true_extent_x(datatype, true_lb, true_extent, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND = MPI_COUNT_KIND), INTENT(OUT) ::  true_lb, true_extent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_indexed(count, array_of_blocklengths, array_of_displacements,
            oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count, array_of_blocklengths(count),
    array_of_displacements(count)
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_size(datatype, size, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_size_x(datatype, size, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND=MPI_COUNT_KIND), INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_vector(count, blocklength, stride, oldtype, newtype, ierror)
    INTEGER, INTENT(IN) ::  count, blocklength, stride
    TYPE(MPI_Datatype), INTENT(IN) ::  oldtype
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Unpack_external(datarep, inbuf, insize, position, outbuf, outcount,
            datatype, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  datarep
    TYPE(*), DIMENSION(..), INTENT(IN) ::  inbuf
    TYPE(*), DIMENSION(..)  ::  outbuf
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  insize
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(INOUT) ::  position
    INTEGER, INTENT(IN) ::  outcount
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Unpack(inbuf, insize, position, outbuf, outcount, datatype, comm,
            ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  inbuf
    TYPE(*), DIMENSION(..)  ::  outbuf
    INTEGER, INTENT(IN) ::  insize, outcount
    INTEGER, INTENT(INOUT) ::  position
```

```
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.3 Collective Communication Fortran 2008 Bindings

```
MPI_Allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
             comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
             recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcounts(*), displs(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Allreduce(sendbuf, recvbuf, count, datatype, op, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
             comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts,
             rdispls, recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcounts(*), sdispls(*), recvcounts(*),
    rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
```

```
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf, recvcounts,
            rdispls, recvtypes, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcounts(*), sdispls(*), recvcounts(*),
    rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtypes(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  recvtypes(*)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Barrier(comm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Bcast(buffer, count, datatype, root, comm, ierror)
    TYPE(*), DIMENSION(..)  ::  buffer
    INTEGER, INTENT(IN) ::  count, root
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Exscan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Gather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
            root, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount, root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Gatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
            recvtype, root, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcounts(*), displs(*), root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
```

```
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iallgather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
              comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iallgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
              recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  recvcounts(*), displs(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iallreduce(sendbuf, recvbuf, count, datatype, op, comm, request,
              ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ialltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
              comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ialltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf, recvcounts,
              rdispls, recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  sendcounts(*), sdispls(*),
    recvcounts(*), rdispls(*)
```

```
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ialltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf,
             recvcounts, rdispls, recvtypes, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  sendcounts(*), sdispls(*),
    recvcounts(*), rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN), ASYNCHRONOUS ::  sendtypes(*),
    recvtypes(*)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ibarrier(comm, request, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ibcast(buffer, count, datatype, root, comm, request, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buffer
    INTEGER, INTENT(IN) ::  count, root
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iexscan(sendbuf, recvbuf, count, datatype, op, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Igather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
             root, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount, root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Igatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
             recvtype, root, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, root
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  recvcounts(*), displs(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ireduce_scatter_block(sendbuf, recvbuf, recvcount, datatype, op, comm,
             request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ireduce_scatter(sendbuf, recvbuf, recvcounts, datatype, op, comm,
             request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  recvcounts(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ireduce(sendbuf, recvbuf, count, datatype, op, root, comm, request,
             ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  count, root
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iscan(sendbuf, recvbuf, count, datatype, op, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
```

```
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iscatter(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
             root, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount, root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Iscatterv(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcount,
             recvtype, root, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  sendcounts(*), displs(*)
    INTEGER, INTENT(IN) ::  recvcount, root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Op_commutative(op, commute, ierror)
    TYPE(MPI_Op), INTENT(IN) ::  op
    LOGICAL, INTENT(OUT) ::  commute
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Op_create(user_fn, commute, op, ierror)
    PROCEDURE(MPI_User_function) ::  user_fn
    LOGICAL, INTENT(IN) ::  commute
    TYPE(MPI_Op), INTENT(OUT) ::  op
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Op_free(op, ierror)
    TYPE(MPI_Op), INTENT(INOUT) ::  op
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Reduce_local(inbuf, inoutbuf, count, datatype, op, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  inbuf
    TYPE(*), DIMENSION(..)  ::  inoutbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Reduce_scatter_block(sendbuf, recvbuf, recvcount, datatype, op, comm,
             ierror)
```

```
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Reduce_scatter(sendbuf, recvbuf, recvcounts, datatype, op, comm,
            ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  recvcounts(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  count, root
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Scan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Scatter(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
            root, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount, root
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Scatterv(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcount,
            recvtype, root, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcounts(*), displs(*), recvcount, root
```

```
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.4 Groups, Contexts, Communicators, and Caching Fortran 2008 Bindings

```
MPI_Comm_compare(comm1, comm2, result, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm1, comm2
    INTEGER, INTENT(OUT) ::  result
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_create(comm, group, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Group), INTENT(IN) ::  group
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_create_group(comm, group, tag, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  tag
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_create_keyval(comm_copy_attr_fn, comm_delete_attr_fn, comm_keyval,
            extra_state, ierror)
    PROCEDURE(MPI_Comm_copy_attr_function) ::  comm_copy_attr_fn
    PROCEDURE(MPI_Comm_delete_attr_function) ::  comm_delete_attr_fn
    INTEGER, INTENT(OUT) ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  extra_state
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_delete_attr(comm, comm_keyval, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  comm_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_dup(comm, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_COMM_DUP_FN(oldcomm, comm_keyval, extra_state, attribute_val_in,
            attribute_val_out, flag, ierror)
    TYPE(MPI_Comm) ::  oldcomm
    INTEGER ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror
```

```
MPI_Comm_dup_with_info(comm, info, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_free(comm, ierror)
    TYPE(MPI_Comm), INTENT(INOUT) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_free_keyval(comm_keyval, ierror)
    INTEGER, INTENT(INOUT) ::  comm_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_get_attr(comm, comm_keyval, attribute_val, flag, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  attribute_val
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_get_info(comm, info_used, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Info), INTENT(OUT) ::  info_used
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_get_name(comm, comm_name, resultlen, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) ::  comm_name
    INTEGER, INTENT(OUT) ::  resultlen
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_group(comm, group, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Group), INTENT(OUT) ::  group
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_idup(comm, newcomm, request, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Comm), INTENT(OUT), ASYNCHRONOUS ::  newcomm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_COMM_NULL_COPY_FN(oldcomm, comm_keyval, extra_state, attribute_val_in,
            attribute_val_out, flag, ierror)
    TYPE(MPI_Comm) ::  oldcomm
    INTEGER ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror

```

```
MPI_COMM_NULL_DELETE_FN(comm, comm_keyval, attribute_val, extra_state,
            ierror)
    TYPE(MPI_Comm) ::  comm
    INTEGER ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
    INTEGER ::  ierror

MPI_Comm_rank(comm, rank, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  rank
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_remote_group(comm, group, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Group), INTENT(OUT) ::  group
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_remote_size(comm, size, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_set_attr(comm, comm_keyval, attribute_val, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  comm_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  attribute_val
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_set_info(comm, info, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_set_name(comm, comm_name, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    CHARACTER(LEN=*), INTENT(IN) ::  comm_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_size(comm, size, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_split(comm, color, key, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  color, key
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_split_type(comm, split_type, key, info, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  split_type, key
```

```
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_test_inter(comm, flag, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_compare(group1, group2, result, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group1, group2
    INTEGER, INTENT(OUT) ::  result
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_difference(group1, group2, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group1, group2
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_excl(group, n, ranks, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  n, ranks(n)
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_free(group, ierror)
    TYPE(MPI_Group), INTENT(INOUT) ::  group
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_incl(group, n, ranks, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  n, ranks(n)
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_intersection(group1, group2, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group1, group2
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_range_excl(group, n, ranges, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  n, ranges(3,n)
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_range_incl(group, n, ranges, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  n, ranges(3,n)
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Group_rank(group, rank, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(OUT) ::  rank
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_size(group, size, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_translate_ranks(group1, n, ranks1, group2, ranks2, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group1, group2
    INTEGER, INTENT(IN) ::  n, ranks1(n)
    INTEGER, INTENT(OUT) ::  ranks2(n)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Group_union(group1, group2, newgroup, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group1, group2
    TYPE(MPI_Group), INTENT(OUT) ::  newgroup
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Intercomm_create(local_comm, local_leader, peer_comm, remote_leader,
            tag, newintercomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  local_comm, peer_comm
    INTEGER, INTENT(IN) ::  local_leader, remote_leader, tag
    TYPE(MPI_Comm), INTENT(OUT) ::  newintercomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Intercomm_merge(intercomm, high, newintracomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  intercomm
    LOGICAL, INTENT(IN) ::  high
    TYPE(MPI_Comm), INTENT(OUT) ::  newintracomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_keyval(type_copy_attr_fn, type_delete_attr_fn, type_keyval,
            extra_state, ierror)
    PROCEDURE(MPI_Type_copy_attr_function) ::  type_copy_attr_fn
    PROCEDURE(MPI_Type_delete_attr_function) ::  type_delete_attr_fn
    INTEGER, INTENT(OUT) ::  type_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  extra_state
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_delete_attr(datatype, type_keyval, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  type_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_TYPE_DUP_FN(oldtype, type_keyval, extra_state, attribute_val_in,
            attribute_val_out, flag, ierror)
    TYPE(MPI_Datatype) ::  oldtype
    INTEGER ::  type_keyval
```

```
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror

MPI_Type_free_keyval(type_keyval, ierror)
    INTEGER, INTENT(INOUT) ::  type_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_attr(datatype, type_keyval, attribute_val, flag, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  type_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  attribute_val
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_get_name(datatype, type_name, resultlen, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) ::  type_name
    INTEGER, INTENT(OUT) ::  resultlen
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_TYPE_NULL_COPY_FN(oldtype, type_keyval, extra_state, attribute_val_in,
             attribute_val_out, flag, ierror)
    TYPE(MPI_Datatype) ::  oldtype
    INTEGER ::  type_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror

MPI_TYPE_NULL_DELETE_FN(datatype, type_keyval, attribute_val, extra_state,
             ierror)
    TYPE(MPI_Datatype) ::  datatype
    INTEGER ::  type_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
    INTEGER, INTENT(OUT) ::  ierror

MPI_Type_set_attr(datatype, type_keyval, attribute_val, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  type_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  attribute_val
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_set_name(datatype, type_name, ierror)
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    CHARACTER(LEN=*), INTENT(IN) ::  type_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_create_keyval(win_copy_attr_fn, win_delete_attr_fn, win_keyval,
             extra_state, ierror)
```

```
    PROCEDURE(MPI_Win_copy_attr_function) ::  win_copy_attr_fn
    PROCEDURE(MPI_Win_delete_attr_function) ::  win_delete_attr_fn
    INTEGER, INTENT(OUT) ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  extra_state
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_delete_attr(win, win_keyval, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, INTENT(IN) ::  win_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_WIN_DUP_FN(oldwin, win_keyval, extra_state, attribute_val_in,
            attribute_val_out, flag, ierror)
    TYPE(MPI_Win) ::  oldwin
    INTEGER ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror

MPI_Win_free_keyval(win_keyval, ierror)
    INTEGER, INTENT(INOUT) ::  win_keyval
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_get_attr(win, win_keyval, attribute_val, flag, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, INTENT(IN) ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  attribute_val
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_get_name(win, win_name, resultlen, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    CHARACTER(LEN=MPI_MAX_OBJECT_NAME), INTENT(OUT) ::  win_name
    INTEGER, INTENT(OUT) ::  resultlen
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_WIN_NULL_COPY_FN(oldwin, win_keyval, extra_state, attribute_val_in,
            attribute_val_out, flag, ierror)
    TYPE(MPI_Win) ::  oldwin
    INTEGER ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state, attribute_val_in
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val_out
    LOGICAL ::  flag
    INTEGER ::  ierror

MPI_WIN_NULL_DELETE_FN(win, win_keyval, attribute_val, extra_state, ierror)
    TYPE(MPI_Win) ::  win
    INTEGER ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  attribute_val, extra_state
    INTEGER ::  ierror
```

```
MPI_Win_set_attr(win, win_keyval, attribute_val, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, INTENT(IN) ::  win_keyval
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  attribute_val
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_set_name(win, win_name, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    CHARACTER(LEN=*), INTENT(IN) ::  win_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.5 Process Topologies Fortran 2008 Bindings

```
MPI_Cart_coords(comm, rank, maxdims, coords, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  rank, maxdims
    INTEGER, INTENT(OUT) ::  coords(maxdims)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cart_create(comm_old, ndims, dims, periods, reorder, comm_cart, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm_old
    INTEGER, INTENT(IN) ::  ndims, dims(ndims)
    LOGICAL, INTENT(IN) ::  periods(ndims), reorder
    TYPE(MPI_Comm), INTENT(OUT) ::  comm_cart
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cartdim_get(comm, ndims, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  ndims
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cart_get(comm, maxdims, dims, periods, coords, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  maxdims
    INTEGER, INTENT(OUT) ::  dims(maxdims), coords(maxdims)
    LOGICAL, INTENT(OUT) ::  periods(maxdims)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cart_map(comm, ndims, dims, periods, newrank, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  ndims, dims(ndims)
    LOGICAL, INTENT(IN) ::  periods(ndims)
    INTEGER, INTENT(OUT) ::  newrank
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cart_rank(comm, coords, rank, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  coords(*)
    INTEGER, INTENT(OUT) ::  rank
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Cart_shift(comm, direction, disp, rank_source, rank_dest, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  direction, disp
    INTEGER, INTENT(OUT) ::  rank_source, rank_dest
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Cart_sub(comm, remain_dims, newcomm, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    LOGICAL, INTENT(IN) ::  remain_dims(*)
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Dims_create(nnodes, ndims, dims, ierror)
    INTEGER, INTENT(IN) ::  nnodes, ndims
    INTEGER, INTENT(INOUT) ::  dims(ndims)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Dist_graph_create_adjacent(comm_old, indegree, sources, sourceweights,
             outdegree, destinations, destweights, info, reorder,
             comm_dist_graph, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm_old
    INTEGER, INTENT(IN) ::  indegree, sources(indegree), outdegree,
        destinations(outdegree)
    INTEGER, INTENT(IN) ::  sourceweights(*), destweights(*)
    TYPE(MPI_Info), INTENT(IN) ::  info
    LOGICAL, INTENT(IN) ::  reorder
    TYPE(MPI_Comm), INTENT(OUT) ::  comm_dist_graph
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Dist_graph_create(comm_old, n, sources, degrees, destinations, weights,
             info, reorder, comm_dist_graph, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm_old
    INTEGER, INTENT(IN) ::  n, sources(n), degrees(n), destinations(*)
    INTEGER, INTENT(IN) ::  weights(*)
    TYPE(MPI_Info), INTENT(IN) ::  info
    LOGICAL, INTENT(IN) ::  reorder
    TYPE(MPI_Comm), INTENT(OUT) ::  comm_dist_graph
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Dist_graph_neighbors(comm, maxindegree, sources, sourceweights,
             maxoutdegree, destinations, destweights, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  maxindegree, maxoutdegree
    INTEGER, INTENT(OUT) ::  sources(maxindegree),
        destinations(maxoutdegree)
    INTEGER ::  sourceweights(*), destweights(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Dist_graph_neighbors_count(comm, indegree, outdegree, weighted, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
```

```
    INTEGER, INTENT(OUT) ::  indegree, outdegree
    LOGICAL, INTENT(OUT) ::  weighted
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graph_create(comm_old, nnodes, index, edges, reorder, comm_graph,
             ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm_old
    INTEGER, INTENT(IN) ::  nnodes, index(nnodes), edges(*)
    LOGICAL, INTENT(IN) ::  reorder
    TYPE(MPI_Comm), INTENT(OUT) ::  comm_graph
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graphdims_get(comm, nnodes, nedges, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  nnodes, nedges
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graph_get(comm, maxindex, maxedges, index, edges, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  maxindex, maxedges
    INTEGER, INTENT(OUT) ::  index(maxindex), edges(maxedges)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graph_map(comm, nnodes, index, edges, newrank, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  nnodes, index(nnodes), edges(*)
    INTEGER, INTENT(OUT) ::  newrank
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graph_neighbors(comm, rank, maxneighbors, neighbors, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  rank, maxneighbors
    INTEGER, INTENT(OUT) ::  neighbors(maxneighbors)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Graph_neighbors_count(comm, rank, nneighbors, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  rank
    INTEGER, INTENT(OUT) ::  nneighbors
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ineighbor_allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
             recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Ineighbor_allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts,
            displs, recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  recvcounts(*), displs(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ineighbor_alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount,
            recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ineighbor_alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf,
            recvcounts, rdispls, recvtype, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  sendcounts(*), sdispls(*),
        recvcounts(*), rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Ineighbor_alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf,
            recvcounts, rdispls, recvtypes, comm, request, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  sendbuf
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  recvbuf
    INTEGER, INTENT(IN), ASYNCHRONOUS ::  sendcounts(*), recvcounts(*)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN), ASYNCHRONOUS ::
        sdispls(*), rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN), ASYNCHRONOUS ::  sendtypes(*),
        recvtypes(*)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Neighbor_allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
            recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
```

```
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Neighbor_allgatherv(sendbuf, sendcount, sendtype, recvbuf, recvcounts,
             displs, recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcounts(*), displs(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Neighbor_alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount,
             recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcount, recvcount
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Neighbor_alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf,
             recvcounts, rdispls, recvtype, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcounts(*), sdispls(*), recvcounts(*),
        rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtype, recvtype
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Neighbor_alltoallw(sendbuf, sendcounts, sdispls, sendtypes, recvbuf,
             recvcounts, rdispls, recvtypes, comm, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN) ::  sendbuf
    TYPE(*), DIMENSION(..)  ::  recvbuf
    INTEGER, INTENT(IN) ::  sendcounts(*), recvcounts(*)
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  sdispls(*), rdispls(*)
    TYPE(MPI_Datatype), INTENT(IN) ::  sendtypes(*), recvtypes(*)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Topo_test(comm, status, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(OUT) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.6 MPI Environmental Management Fortran 2008 Bindings

```
DOUBLE PRECISION MPI_Wtick()

DOUBLE PRECISION MPI_Wtime()

MPI_Abort(comm, errorcode, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  errorcode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Add_error_class(errorclass, ierror)
    INTEGER, INTENT(OUT) ::  errorclass
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Add_error_code(errorclass, errorcode, ierror)
    INTEGER, INTENT(IN) ::  errorclass
    INTEGER, INTENT(OUT) ::  errorcode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Add_error_string(errorcode, string, ierror)
    INTEGER, INTENT(IN) ::  errorcode
    CHARACTER(LEN=*), INTENT(IN) ::  string
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Alloc_mem(size, info, baseptr, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  size
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(C_PTR), INTENT(OUT) ::  baseptr
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_call_errhandler(comm, errorcode, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    INTEGER, INTENT(IN) ::  errorcode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_create_errhandler(comm_errhandler_fn, errhandler, ierror)
    PROCEDURE(MPI_Comm_errhandler_function) ::  comm_errhandler_fn
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_get_errhandler(comm, errhandler, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_set_errhandler(comm, errhandler, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Errhandler), INTENT(IN) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Errhandler_free(errhandler, ierror)
```

```
    TYPE(MPI_Errhandler), INTENT(INOUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Error_class(errorcode, errorclass, ierror)
    INTEGER, INTENT(IN) ::  errorcode
    INTEGER, INTENT(OUT) ::  errorclass
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Error_string(errorcode, string, resultlen, ierror)
    INTEGER, INTENT(IN) ::  errorcode
    CHARACTER(LEN=MPI_MAX_ERROR_STRING), INTENT(OUT) ::  string
    INTEGER, INTENT(OUT) ::  resultlen
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_call_errhandler(fh, errorcode, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER, INTENT(IN) ::  errorcode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_create_errhandler(file_errhandler_fn, errhandler, ierror)
    PROCEDURE(MPI_File_errhandler_function) ::  file_errhandler_fn
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_errhandler(file, errhandler, ierror)
    TYPE(MPI_File), INTENT(IN) ::  file
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_set_errhandler(file, errhandler, ierror)
    TYPE(MPI_File), INTENT(IN) ::  file
    TYPE(MPI_Errhandler), INTENT(IN) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Finalized(flag, ierror)
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Finalize(ierror)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Free_mem(base, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  base
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_library_version(version, resultlen, ierror)
    CHARACTER(LEN=MPI_MAX_LIBRARY_VERSION_STRING), INTENT(OUT) ::  version
    INTEGER, INTENT(OUT) ::  resultlen
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_processor_name(name, resultlen, ierror)
    CHARACTER(LEN=MPI_MAX_PROCESSOR_NAME), INTENT(OUT) ::  name
    INTEGER, INTENT(OUT) ::  resultlen
```

```
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_version(version, subversion, ierror)
    INTEGER, INTENT(OUT) ::  version, subversion
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Initialized(flag, ierror)
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Init(ierror)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_call_errhandler(win, errorcode, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, INTENT(IN) ::  errorcode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_create_errhandler(win_errhandler_fn, errhandler, ierror)
    PROCEDURE(MPI_Win_errhandler_function) ::  win_errhandler_fn
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_get_errhandler(win, errhandler, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Errhandler), INTENT(OUT) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_set_errhandler(win, errhandler, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Errhandler), INTENT(IN) ::  errhandler
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.7 The Info Object Fortran 2008 Bindings

```
MPI_Info_create(info, ierror)
    TYPE(MPI_Info), INTENT(OUT) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_delete(info, key, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=*), INTENT(IN) ::  key
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_dup(info, newinfo, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Info), INTENT(OUT) ::  newinfo
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_free(info, ierror)
    TYPE(MPI_Info), INTENT(INOUT) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Info_get(info, key, valuelen, value, flag, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=*), INTENT(IN) ::  key
    INTEGER, INTENT(IN) ::  valuelen
    CHARACTER(LEN=valuelen), INTENT(OUT) ::  value
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_get_nkeys(info, nkeys, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, INTENT(OUT) ::  nkeys
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_get_nthkey(info, n, key, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, INTENT(IN) ::  n
    CHARACTER(LEN=*), INTENT(OUT) ::  key
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_get_valuelen(info, key, valuelen, flag, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=*), INTENT(IN) ::  key
    INTEGER, INTENT(OUT) ::  valuelen
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Info_set(info, key, value, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=*), INTENT(IN) ::  key, value
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.8 Process Creation and Management Fortran 2008 Bindings

```
MPI_Close_port(port_name, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  port_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_accept(port_name, info, root, comm, newcomm, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  port_name
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, INTENT(IN) ::  root
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_connect(port_name, info, root, comm, newcomm, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  port_name
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, INTENT(IN) ::  root
    TYPE(MPI_Comm), INTENT(IN) ::  comm
```

```
    TYPE(MPI_Comm), INTENT(OUT) ::  newcomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_disconnect(comm, ierror)
    TYPE(MPI_Comm), INTENT(INOUT) ::  comm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_get_parent(parent, ierror)
    TYPE(MPI_Comm), INTENT(OUT) ::  parent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_join(fd, intercomm, ierror)
    INTEGER, INTENT(IN) ::  fd
    TYPE(MPI_Comm), INTENT(OUT) ::  intercomm
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_spawn(command, argv, maxprocs, info, root, comm, intercomm,
            array_of_errcodes, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  command, argv(*)
    INTEGER, INTENT(IN) ::  maxprocs, root
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Comm), INTENT(OUT) ::  intercomm
    INTEGER ::  array_of_errcodes(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Comm_spawn_multiple(count, array_of_commands, array_of_argv,
            array_of_maxprocs, array_of_info, root, comm, intercomm,
            array_of_errcodes, ierror)
    INTEGER, INTENT(IN) ::  count, array_of_maxprocs(*), root
    CHARACTER(LEN=*), INTENT(IN) ::  array_of_commands(*)
    CHARACTER(LEN=*), INTENT(IN) ::  array_of_argv(count, *)
    TYPE(MPI_Info), INTENT(IN) ::  array_of_info(*)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Comm), INTENT(OUT) ::  intercomm
    INTEGER ::  array_of_errcodes(*)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Lookup_name(service_name, info, port_name, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  service_name
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=MPI_MAX_PORT_NAME), INTENT(OUT) ::  port_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Open_port(info, port_name, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    CHARACTER(LEN=MPI_MAX_PORT_NAME), INTENT(OUT) ::  port_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Publish_name(service_name, info, port_name, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
```

```
    CHARACTER(LEN=*), INTENT(IN) ::  service_name, port_name
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Unpublish_name(service_name, info, port_name, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  service_name, port_name
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.9 One-Sided Communications Fortran 2008 Bindings

```
MPI_Accumulate(origin_addr, origin_count, origin_datatype, target_rank,
            target_disp, target_count, target_datatype, op, win, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Compare_and_swap(origin_addr, compare_addr, result_addr, datatype,
            target_rank, target_disp, win, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  compare_addr
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  result_addr
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  target_rank
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Fetch_and_op(origin_addr, result_addr, datatype, target_rank,
            target_disp, op, win, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  result_addr
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, INTENT(IN) ::  target_rank
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get_accumulate(origin_addr, origin_count, origin_datatype, result_addr,
            result_count, result_datatype, target_rank, target_disp,
            target_count, target_datatype, op, win, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  result_addr
    INTEGER, INTENT(IN) ::  origin_count, result_count, target_rank,
    target_count
```

```
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype,
    result_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Get(origin_addr, origin_count, origin_datatype, target_rank,
            target_disp, target_count, target_datatype, win, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Put(origin_addr, origin_count, origin_datatype, target_rank,
            target_disp, target_count, target_datatype, win, ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Raccumulate(origin_addr, origin_count, origin_datatype, target_rank,
            target_disp, target_count, target_datatype, op, win, request,
            ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Op), INTENT(IN) ::  op
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Rget_accumulate(origin_addr, origin_count, origin_datatype,
            result_addr, result_count, result_datatype, target_rank,
            target_disp, target_count, target_datatype, op, win, request,
            ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  result_addr
    INTEGER, INTENT(IN) ::  origin_count, result_count, target_rank,
    target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype,
    result_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Op), INTENT(IN) ::  op
```

```
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Rget(origin_addr, origin_count, origin_datatype, target_rank,
             target_disp, target_count, target_datatype, win, request,
             ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Rput(origin_addr, origin_count, origin_datatype, target_rank,
             target_disp, target_count, target_datatype, win, request,
             ierror)
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  origin_addr
    INTEGER, INTENT(IN) ::  origin_count, target_rank, target_count
    TYPE(MPI_Datatype), INTENT(IN) ::  origin_datatype, target_datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  target_disp
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_allocate_shared(size, disp_unit, info, comm, baseptr, win, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  size
    INTEGER, INTENT(IN) ::  disp_unit
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(C_PTR), INTENT(OUT) ::  baseptr
    TYPE(MPI_Win), INTENT(OUT) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_allocate(size, disp_unit, info, comm, baseptr, win, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  size
    INTEGER, INTENT(IN) ::  disp_unit
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(C_PTR), INTENT(OUT) ::  baseptr
    TYPE(MPI_Win), INTENT(OUT) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_attach(win, base, size, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  base
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  size
```

```
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_complete(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_create(base, size, disp_unit, info, comm, win, ierror)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  base
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  size
    INTEGER, INTENT(IN) ::  disp_unit
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Win), INTENT(OUT) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_create_dynamic(info, comm, win, ierror)
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    TYPE(MPI_Win), INTENT(OUT) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_detach(win, base, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  base
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_fence(assert, win, ierror)
    INTEGER, INTENT(IN) ::  assert
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_flush_all(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_flush_local_all(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_flush_local(rank, win, ierror)
    INTEGER, INTENT(IN) ::  rank
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_flush(rank, win, ierror)
    INTEGER, INTENT(IN) ::  rank
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_free(win, ierror)
    TYPE(MPI_Win), INTENT(INOUT) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_Win_get_group(win, group, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Group), INTENT(OUT) ::  group
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_get_info(win, info_used, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Info), INTENT(OUT) ::  info_used
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_lock_all(assert, win, ierror)
    INTEGER, INTENT(IN) ::  assert
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_lock(lock_type, rank, assert, win, ierror)
    INTEGER, INTENT(IN) ::  lock_type, rank, assert
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_post(group, assert, win, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  assert
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_set_info(win, info, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_shared_query(win, rank, size, disp_unit, baseptr, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, INTENT(IN) ::  rank
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  size
    INTEGER, INTENT(OUT) ::  disp_unit
    TYPE(C_PTR), INTENT(OUT) ::  baseptr
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_start(group, assert, win, ierror)
    TYPE(MPI_Group), INTENT(IN) ::  group
    INTEGER, INTENT(IN) ::  assert
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_sync(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_test(win, flag, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
```

```
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_unlock_all(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_unlock(rank, win, ierror)
    INTEGER, INTENT(IN) ::  rank
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Win_wait(win, ierror)
    TYPE(MPI_Win), INTENT(IN) ::  win
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.10 External Interfaces Fortran 2008 Bindings

```
MPI_Grequest_complete(request, ierror)
    TYPE(MPI_Request), INTENT(IN) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Grequest_start(query_fn, free_fn, cancel_fn, extra_state, request,
            ierror)
    PROCEDURE(MPI_Grequest_query_function) ::  query_fn
    PROCEDURE(MPI_Grequest_free_function) ::  free_fn
    PROCEDURE(MPI_Grequest_cancel_function) ::  cancel_fn
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  extra_state
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Init_thread(required, provided, ierror)
    INTEGER, INTENT(IN) ::  required
    INTEGER, INTENT(OUT) ::  provided
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Is_thread_main(flag, ierror)
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Query_thread(provided, ierror)
    INTEGER, INTENT(OUT) ::  provided
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Status_set_cancelled(status, flag, ierror)
    TYPE(MPI_Status), INTENT(INOUT) ::  status
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Status_set_elements(status, datatype, count, ierror)
    TYPE(MPI_Status), INTENT(INOUT) ::  status
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
```

```
    INTEGER, INTENT(IN) ::  count
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Status_set_elements_x(status, datatype, count, ierror)
    TYPE(MPI_Status), INTENT(INOUT) ::  status
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND = MPI_COUNT_KIND), INTENT(IN) ::  count
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.11 I/O Fortran 2008 Bindings

```
MPI_CONVERSION_FN_NULL(userbuf, datatype, count, filebuf, position,
             extra_state, ierror)
    USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
    TYPE(C_PTR), VALUE ::  userbuf, filebuf
    TYPE(MPI_Datatype) ::  datatype
    INTEGER ::  count, ierror
    INTEGER(KIND=MPI_OFFSET_KIND) ::  position
    INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state

MPI_File_close(fh, ierror)
    TYPE(MPI_File), INTENT(INOUT) ::  fh
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_delete(filename, info, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  filename
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_amode(fh, amode, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER, INTENT(OUT) ::  amode
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_atomicity(fh, flag, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    LOGICAL, INTENT(OUT) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_byte_offset(fh, offset, disp, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) ::  disp
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_group(fh, group, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(MPI_Group), INTENT(OUT) ::  group
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_info(fh, info_used, ierror)
```

```
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(MPI_Info), INTENT(OUT) ::  info_used
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_position(fh, offset, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) ::  offset
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_position_shared(fh, offset, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) ::  offset
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_size(fh, size, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_type_extent(fh, datatype, extent, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(OUT) ::  extent
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_get_view(fh, disp, etype, filetype, datarep, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(OUT) ::  disp
    TYPE(MPI_Datatype), INTENT(OUT) ::  etype, filetype
    CHARACTER(LEN=*), INTENT(OUT) ::  datarep
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iread_all(fh, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iread_at_all(fh, offset, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iread_at(fh, offset, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
```

```
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iread(fh, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iread_shared(fh, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iwrite_all(fh, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iwrite_at_all(fh, offset, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iwrite_at(fh, offset, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iwrite(fh, buf, count, datatype, request, ierror)
```

```
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_iwrite_shared(fh, buf, count, datatype, request, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Request), INTENT(OUT) ::  request
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_open(comm, filename, amode, info, fh, ierror)
    TYPE(MPI_Comm), INTENT(IN) ::  comm
    CHARACTER(LEN=*), INTENT(IN) ::  filename
    INTEGER, INTENT(IN) ::  amode
    TYPE(MPI_Info), INTENT(IN) ::  info
    TYPE(MPI_File), INTENT(OUT) ::  fh
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_preallocate(fh, size, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_all_begin(fh, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_all_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_all(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_at_all_begin(fh, offset, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
```

```
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_at_all_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_at_all(fh, offset, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_at(fh, offset, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_ordered_begin(fh, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_ordered_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_File_read_ordered(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_read_shared(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..)  ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_seek(fh, offset, whence, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    INTEGER, INTENT(IN) ::  whence
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_seek_shared(fh, offset, whence, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    INTEGER, INTENT(IN) ::  whence
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_set_atomicity(fh, flag, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    LOGICAL, INTENT(IN) ::  flag
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_set_info(fh, info, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_set_size(fh, size, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_set_view(fh, disp, etype, filetype, datarep, info, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  disp
    TYPE(MPI_Datatype), INTENT(IN) ::  etype, filetype
    CHARACTER(LEN=*), INTENT(IN) ::  datarep
    TYPE(MPI_Info), INTENT(IN) ::  info
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

```
MPI_File_sync(fh, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_all_begin(fh, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_all_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_all(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_at_all_begin(fh, offset, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_at_all_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_at_all(fh, offset, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_at(fh, offset, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    INTEGER(KIND=MPI_OFFSET_KIND), INTENT(IN) ::  offset
```

```
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_ordered_begin(fh, buf, count, datatype, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_ordered_end(fh, buf, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN), ASYNCHRONOUS ::  buf
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_ordered(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_File_write_shared(fh, buf, count, datatype, status, ierror)
    TYPE(MPI_File), INTENT(IN) ::  fh
    TYPE(*), DIMENSION(..), INTENT(IN) ::  buf
    INTEGER, INTENT(IN) ::  count
    TYPE(MPI_Datatype), INTENT(IN) ::  datatype
    TYPE(MPI_Status) ::  status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Register_datarep(datarep, read_conversion_fn, write_conversion_fn,
            dtype_file_extent_fn, extra_state, ierror)
    CHARACTER(LEN=*), INTENT(IN) ::  datarep
    PROCEDURE(MPI_Datarep_conversion_function) ::  read_conversion_fn
    PROCEDURE(MPI_Datarep_conversion_function) ::  write_conversion_fn
    PROCEDURE(MPI_Datarep_extent_function) ::  dtype_file_extent_fn
    INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) ::  extra_state
```

```
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.12 Language Bindings Fortran 2008 Bindings

```
MPI_F_sync_reg(buf)
    TYPE(*), DIMENSION(..), ASYNCHRONOUS ::  buf

MPI_Sizeof(x, size, ierror)
    TYPE(*), DIMENSION(..)  ::  x
    INTEGER, INTENT(OUT) ::  size
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Status_f082f(f08_status, f_status, ierror)
    TYPE(MPI_Status), INTENT(IN) ::  f08_status
    INTEGER, INTENT(OUT) ::  f_status(MPI_STATUS_SIZE)
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Status_f2f08(f_status, f08_status, ierror)
    INTEGER, INTENT(IN) ::  f_status(MPI_STATUS_SIZE)
    TYPE(MPI_Status), INTENT(OUT) ::  f08_status
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_f90_complex(p, r, newtype, ierror)
    INTEGER, INTENT(IN) ::  p, r
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_f90_integer(r, newtype, ierror)
    INTEGER, INTENT(IN) ::  r
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_create_f90_real(p, r, newtype, ierror)
    INTEGER, INTENT(IN) ::  p, r
    TYPE(MPI_Datatype), INTENT(OUT) ::  newtype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror

MPI_Type_match_size(typeclass, size, datatype, ierror)
    INTEGER, INTENT(IN) ::  typeclass, size
    TYPE(MPI_Datatype), INTENT(OUT) ::  datatype
    INTEGER, OPTIONAL, INTENT(OUT) ::  ierror
```

A.3.13 Tools / Profiling Interface Fortran 2008 Bindings

```
MPI_Pcontrol(level)
    INTEGER, INTENT(IN) ::  level
```

A.4 Fortran Bindings with mpif.h or the mpi Module

A.4.1 Point-to-Point Communication Fortran Bindings

```
MPI_BSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR

MPI_BSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_BUFFER_ATTACH(BUFFER, SIZE, IERROR)
    <type> BUFFER(*)
    INTEGER SIZE, IERROR

MPI_BUFFER_DETACH(BUFFER_ADDR, SIZE, IERROR)
    <type> BUFFER_ADDR(*)
    INTEGER SIZE, IERROR

MPI_CANCEL(REQUEST, IERROR)
    INTEGER REQUEST, IERROR

MPI_GET_COUNT(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR

MPI_IBSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_IMPROBE(SOURCE, TAG, COMM, FLAG, MESSAGE, STATUS, IERROR)
    INTEGER SOURCE, TAG, COMM, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR
    LOGICAL FLAG

MPI_IMRECV(BUF, COUNT, DATATYPE, MESSAGE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, MESSAGE, REQUEST, IERROR

MPI_IPROBE(SOURCE, TAG, COMM, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR

MPI_IRECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR

MPI_IRSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_ISEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
```

```
MPI_ISSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_MPROBE(SOURCE, TAG, COMM, MESSAGE, STATUS, IERROR)
    INTEGER SOURCE, TAG, COMM, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_MRECV(BUF, COUNT, DATATYPE, MESSAGE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, MESSAGE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_PROBE(SOURCE, TAG, COMM, STATUS, IERROR)
    INTEGER SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR

MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE),
    IERROR

MPI_RECV_INIT(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR

MPI_REQUEST_FREE(REQUEST, IERROR)
    INTEGER REQUEST, IERROR

MPI_REQUEST_GET_STATUS( REQUEST, FLAG, STATUS, IERROR)
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR
    LOGICAL FLAG

MPI_RSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR

MPI_RSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR

MPI_SEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_SENDRECV_REPLACE(BUF, COUNT, DATATYPE, DEST, SENDTAG, SOURCE, RECVTAG,
             COMM, STATUS, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, SENDTAG, SOURCE, RECVTAG, COMM,
    STATUS(MPI_STATUS_SIZE), IERROR

MPI_SENDRECV(SENDBUF, SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVBUF,
             RECVCOUNT, RECVTYPE, SOURCE, RECVTAG, COMM, STATUS, IERROR)
```

```
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVCOUNT, RECVTYPE,
    SOURCE, RECVTAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR

MPI_SSEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR

MPI_SSEND_INIT(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR

MPI_STARTALL(COUNT, ARRAY_OF_REQUESTS, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), IERROR

MPI_START(REQUEST, IERROR)
    INTEGER REQUEST, IERROR

MPI_TESTALL(COUNT, ARRAY_OF_REQUESTS, FLAG, ARRAY_OF_STATUSES, IERROR)
    LOGICAL FLAG
    INTEGER COUNT, ARRAY_OF_REQUESTS(*),
    ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR

MPI_TESTANY(COUNT, ARRAY_OF_REQUESTS, INDEX, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), INDEX, STATUS(MPI_STATUS_SIZE),
    IERROR

MPI_TEST_CANCELLED(STATUS, FLAG, IERROR)
    LOGICAL FLAG
    INTEGER STATUS(MPI_STATUS_SIZE), IERROR

MPI_TEST(REQUEST, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR

MPI_TESTSOME(INCOUNT, ARRAY_OF_REQUESTS, OUTCOUNT, ARRAY_OF_INDICES,
             ARRAY_OF_STATUSES, IERROR)
    INTEGER INCOUNT, ARRAY_OF_REQUESTS(*), OUTCOUNT, ARRAY_OF_INDICES(*),
    ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR

MPI_WAITALL(COUNT, ARRAY_OF_REQUESTS, ARRAY_OF_STATUSES, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*)
    INTEGER ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR

MPI_WAITANY(COUNT, ARRAY_OF_REQUESTS, INDEX, STATUS, IERROR)
    INTEGER COUNT, ARRAY_OF_REQUESTS(*), INDEX, STATUS(MPI_STATUS_SIZE),
    IERROR

MPI_WAIT(REQUEST, STATUS, IERROR)
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR

MPI_WAITSOME(INCOUNT, ARRAY_OF_REQUESTS, OUTCOUNT, ARRAY_OF_INDICES,
             ARRAY_OF_STATUSES, IERROR)
```

```
    INTEGER INCOUNT, ARRAY_OF_REQUESTS(*), OUTCOUNT, ARRAY_OF_INDICES(*),
    ARRAY_OF_STATUSES(MPI_STATUS_SIZE,*), IERROR
```

A.4.2 Datatypes Fortran Bindings

```
INTEGER(KIND=MPI_ADDRESS_KIND) MPI_AINT_ADD(BASE, DISP)
    INTEGER(KIND=MPI_ADDRESS_KIND) BASE, DISP

INTEGER(KIND=MPI_ADDRESS_KIND) MPI_AINT_DIFF(ADDR1, ADDR2)
    INTEGER(KIND=MPI_ADDRESS_KIND) ADDR1, ADDR2

MPI_GET_ADDRESS(LOCATION, ADDRESS, IERROR)
    <type> LOCATION(*)
    INTEGER IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ADDRESS

MPI_GET_ELEMENTS(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR

MPI_GET_ELEMENTS_X(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, IERROR
    INTEGER(KIND=MPI_COUNT_KIND) COUNT

MPI_PACK_EXTERNAL(DATAREP, INBUF, INCOUNT, DATATYPE, OUTBUF, OUTSIZE,
             POSITION, IERROR)
    INTEGER INCOUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) OUTSIZE, POSITION
    CHARACTER*(*) DATAREP
    <type> INBUF(*), OUTBUF(*)

MPI_PACK_EXTERNAL_SIZE(DATAREP, INCOUNT, DATATYPE, SIZE, IERROR)
    INTEGER INCOUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) SIZE
    CHARACTER*(*) DATAREP

MPI_PACK(INBUF, INCOUNT, DATATYPE, OUTBUF, OUTSIZE, POSITION, COMM, IERROR)
    <type> INBUF(*), OUTBUF(*)
    INTEGER INCOUNT, DATATYPE, OUTSIZE, POSITION, COMM, IERROR

MPI_PACK_SIZE(INCOUNT, DATATYPE, COMM, SIZE, IERROR)
    INTEGER INCOUNT, DATATYPE, COMM, SIZE, IERROR

MPI_TYPE_COMMIT(DATATYPE, IERROR)
    INTEGER DATATYPE, IERROR

MPI_TYPE_CONTIGUOUS(COUNT, OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_CREATE_DARRAY(SIZE, RANK, NDIMS, ARRAY_OF_GSIZES,
             ARRAY_OF_DISTRIBS, ARRAY_OF_DARGS, ARRAY_OF_PSIZES, ORDER,
             OLDTYPE, NEWTYPE, IERROR)
    INTEGER SIZE, RANK, NDIMS, ARRAY_OF_GSIZES(*), ARRAY_OF_DISTRIBS(*),
    ARRAY_OF_DARGS(*), ARRAY_OF_PSIZES(*), ORDER, OLDTYPE, NEWTYPE, IERROR
```

```
MPI_TYPE_CREATE_HINDEXED_BLOCK(COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS,
              OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_HINDEXED(COUNT, ARRAY_OF_BLOCKLENGTHS,
              ARRAY_OF_DISPLACEMENTS, OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE,
              IERROR)
    INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE

MPI_TYPE_CREATE_INDEXED_BLOCK(COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS,
              OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, BLOCKLENGTH, ARRAY_OF_DISPLACEMENTS(*), OLDTYPE,
    NEWTYPE, IERROR

MPI_TYPE_CREATE_RESIZED(OLDTYPE, LB, EXTENT, NEWTYPE, IERROR)
    INTEGER OLDTYPE, NEWTYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) LB, EXTENT

MPI_TYPE_CREATE_STRUCT(COUNT, ARRAY_OF_BLOCKLENGTHS,
              ARRAY_OF_DISPLACEMENTS, ARRAY_OF_TYPES, NEWTYPE, IERROR)
    INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), ARRAY_OF_TYPES(*), NEWTYPE,
    IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)

MPI_TYPE_CREATE_SUBARRAY(NDIMS, ARRAY_OF_SIZES, ARRAY_OF_SUBSIZES,
              ARRAY_OF_STARTS, ORDER, OLDTYPE, NEWTYPE, IERROR)
    INTEGER NDIMS, ARRAY_OF_SIZES(*), ARRAY_OF_SUBSIZES(*),
    ARRAY_OF_STARTS(*), ORDER, OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_DUP(OLDTYPE, NEWTYPE, IERROR)
    INTEGER OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_FREE(DATATYPE, IERROR)
    INTEGER DATATYPE, IERROR

MPI_TYPE_GET_CONTENTS(DATATYPE, MAX_INTEGERS, MAX_ADDRESSES, MAX_DATATYPES,
              ARRAY_OF_INTEGERS, ARRAY_OF_ADDRESSES, ARRAY_OF_DATATYPES,
              IERROR)
    INTEGER DATATYPE, MAX_INTEGERS, MAX_ADDRESSES, MAX_DATATYPES,
    ARRAY_OF_INTEGERS(*), ARRAY_OF_DATATYPES(*), IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_ADDRESSES(*)

MPI_TYPE_GET_ENVELOPE(DATATYPE, NUM_INTEGERS, NUM_ADDRESSES, NUM_DATATYPES,
              COMBINER, IERROR)
    INTEGER DATATYPE, NUM_INTEGERS, NUM_ADDRESSES, NUM_DATATYPES, COMBINER,
    IERROR
```

```
MPI_TYPE_GET_EXTENT(DATATYPE, LB, EXTENT, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND = MPI_ADDRESS_KIND) LB, EXTENT

MPI_TYPE_GET_EXTENT_X(DATATYPE, LB, EXTENT, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND = MPI_COUNT_KIND) LB, EXTENT

MPI_TYPE_GET_TRUE_EXTENT(DATATYPE, TRUE_LB, TRUE_EXTENT, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND = MPI_ADDRESS_KIND) TRUE_LB, TRUE_EXTENT

MPI_TYPE_GET_TRUE_EXTENT_X(DATATYPE, TRUE_LB, TRUE_EXTENT, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND = MPI_COUNT_KIND) TRUE_LB, TRUE_EXTENT

MPI_TYPE_INDEXED(COUNT, ARRAY_OF_BLOCKLENGTHS, ARRAY_OF_DISPLACEMENTS,
             OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), ARRAY_OF_DISPLACEMENTS(*),
    OLDTYPE, NEWTYPE, IERROR

MPI_TYPE_SIZE(DATATYPE, SIZE, IERROR)
    INTEGER DATATYPE, SIZE, IERROR

MPI_TYPE_SIZE_X(DATATYPE, SIZE, IERROR)
    INTEGER DATATYPE, IERROR
    INTEGER(KIND = MPI_COUNT_KIND) SIZE

MPI_TYPE_VECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE, IERROR)
    INTEGER COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE, IERROR

MPI_UNPACK_EXTERNAL(DATAREP, INBUF, INSIZE, POSITION, OUTBUF, OUTCOUNT,
             DATATYPE, IERROR)
    INTEGER OUTCOUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) INSIZE, POSITION
    CHARACTER*(*) DATAREP
    <type> INBUF(*), OUTBUF(*)

MPI_UNPACK(INBUF, INSIZE, POSITION, OUTBUF, OUTCOUNT, DATATYPE, COMM,
             IERROR)
    <type> INBUF(*), OUTBUF(*)
    INTEGER INSIZE, POSITION, OUTCOUNT, DATATYPE, COMM, IERROR
```

A.4.3 Collective Communication Fortran Bindings

```
MPI_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR

MPI_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
             RECVTYPE, COMM, IERROR)
```

```
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
    IERROR

MPI_ALLREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR

MPI_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR

MPI_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF, RECVCOUNTS,
             RDISPLS, RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
    RECVTYPE, COMM, IERROR

MPI_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF, RECVCOUNTS,
             RDISPLS, RECVTYPES, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPES(*), RECVCOUNTS(*),
    RDISPLS(*), RECVTYPES(*), COMM, IERROR

MPI_BARRIER(COMM, IERROR)
    INTEGER COMM, IERROR

MPI_BCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, IERROR)
    <type> BUFFER(*)
    INTEGER COUNT, DATATYPE, ROOT, COMM, IERROR

MPI_EXSCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR

MPI_GATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, IERROR

MPI_GATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
             RECVTYPE, ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, ROOT,
    COMM, IERROR

MPI_IALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR

MPI_IALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
```

```
              RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
    REQUEST, IERROR

MPI_IALLREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST,
              IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR

MPI_IALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
              COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR

MPI_IALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF, RECVCOUNTS,
              RDISPLS, RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
    RECVTYPE, COMM, REQUEST, IERROR

MPI_IALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
              RECVCOUNTS, RDISPLS, RECVTYPES, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPES(*), RECVCOUNTS(*),
    RDISPLS(*), RECVTYPES(*), COMM, REQUEST, IERROR

MPI_IBARRIER(COMM, REQUEST, IERROR)
    INTEGER COMM, REQUEST, IERROR

MPI_IBCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, REQUEST, IERROR)
    <type> BUFFER(*)
    INTEGER COUNT, DATATYPE, ROOT, COMM, REQUEST, IERROR

MPI_IEXSCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR

MPI_IGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
              ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, REQUEST,
    IERROR

MPI_IGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS, DISPLS,
              RECVTYPE, ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, ROOT,
    COMM, REQUEST, IERROR

MPI_IREDUCE_SCATTER_BLOCK(SENDBUF, RECVBUF, RECVCOUNT, DATATYPE, OP, COMM,
              REQUEST, IERROR)
```

```
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNT, DATATYPE, OP, COMM, REQUEST, IERROR

MPI_IREDUCE_SCATTER(SENDBUF, RECVBUF, RECVCOUNTS, DATATYPE, OP, COMM,
             REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNTS(*), DATATYPE, OP, COMM, REQUEST, IERROR

MPI_IREDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, REQUEST,
             IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, ROOT, COMM, REQUEST, IERROR

MPI_ISCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, REQUEST, IERROR

MPI_ISCATTER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, REQUEST,
    IERROR

MPI_ISCATTERV(SENDBUF, SENDCOUNTS, DISPLS, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, ROOT, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), DISPLS(*), SENDTYPE, RECVCOUNT, RECVTYPE, ROOT,
    COMM, REQUEST, IERROR

MPI_OP_COMMUTATIVE(OP, COMMUTE, IERROR)
    LOGICAL COMMUTE
    INTEGER OP, IERROR

MPI_OP_CREATE( USER_FN, COMMUTE, OP, IERROR)
    EXTERNAL USER_FN
    LOGICAL COMMUTE
    INTEGER OP, IERROR

MPI_OP_FREE(OP, IERROR)
    INTEGER OP, IERROR

MPI_REDUCE_LOCAL(INBUF, INOUTBUF, COUNT, DATATYPE, OP, IERROR)
    <type> INBUF(*), INOUTBUF(*)
    INTEGER COUNT, DATATYPE, OP, IERROR

MPI_REDUCE_SCATTER_BLOCK(SENDBUF, RECVBUF, RECVCOUNT, DATATYPE, OP, COMM,
             IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER RECVCOUNT, DATATYPE, OP, COMM, IERROR

MPI_REDUCE_SCATTER(SENDBUF, RECVBUF, RECVCOUNTS, DATATYPE, OP, COMM,
             IERROR)
    <type> SENDBUF(*), RECVBUF(*)
```

```
    INTEGER RECVCOUNTS(*), DATATYPE, OP, COMM, IERROR

MPI_REDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, ROOT, COMM, IERROR

MPI_SCAN(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER COUNT, DATATYPE, OP, COMM, IERROR

MPI_SCATTER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE,
             ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, IERROR

MPI_SCATTERV(SENDBUF, SENDCOUNTS, DISPLS, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, ROOT, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), DISPLS(*), SENDTYPE, RECVCOUNT, RECVTYPE, ROOT,
    COMM, IERROR
```

A.4.4 Groups, Contexts, Communicators, and Caching Fortran Bindings

```
MPI_COMM_COMPARE(COMM1, COMM2, RESULT, IERROR)
    INTEGER COMM1, COMM2, RESULT, IERROR

MPI_COMM_CREATE(COMM, GROUP, NEWCOMM, IERROR)
    INTEGER COMM, GROUP, NEWCOMM, IERROR

MPI_COMM_CREATE_GROUP(COMM, GROUP, TAG, NEWCOMM, IERROR)
    INTEGER COMM, GROUP, TAG, NEWCOMM, IERROR

MPI_COMM_CREATE_KEYVAL(COMM_COPY_ATTR_FN, COMM_DELETE_ATTR_FN, COMM_KEYVAL,
             EXTRA_STATE, IERROR)
    EXTERNAL COMM_COPY_ATTR_FN, COMM_DELETE_ATTR_FN
    INTEGER COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

MPI_COMM_DELETE_ATTR(COMM, COMM_KEYVAL, IERROR)
    INTEGER COMM, COMM_KEYVAL, IERROR

MPI_COMM_DUP(COMM, NEWCOMM, IERROR)
    INTEGER COMM, NEWCOMM, IERROR

MPI_COMM_DUP_FN(OLDCOMM, COMM_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
             ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDCOMM, COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_COMM_DUP_WITH_INFO(COMM, INFO, NEWCOMM, IERROR)
    INTEGER COMM, INFO, NEWCOMM, IERROR
```

```
MPI_COMM_FREE(COMM, IERROR)
    INTEGER COMM, IERROR

MPI_COMM_FREE_KEYVAL(COMM_KEYVAL, IERROR)
    INTEGER COMM_KEYVAL, IERROR

MPI_COMM_GET_ATTR(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
    INTEGER COMM, COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL
    LOGICAL FLAG

MPI_COMM_GET_INFO(COMM, INFO_USED, IERROR)
    INTEGER COMM, INFO_USED, IERROR

MPI_COMM_GET_NAME(COMM, COMM_NAME, RESULTLEN, IERROR)
    INTEGER COMM, RESULTLEN, IERROR
    CHARACTER*(*) COMM_NAME

MPI_COMM_GROUP(COMM, GROUP, IERROR)
    INTEGER COMM, GROUP, IERROR

MPI_COMM_IDUP(COMM, NEWCOMM, REQUEST, IERROR)
    INTEGER COMM, NEWCOMM, REQUEST, IERROR

MPI_COMM_NULL_COPY_FN(OLDCOMM, COMM_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
            ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDCOMM, COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_COMM_NULL_DELETE_FN(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE,
            IERROR)
    INTEGER COMM, COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE

MPI_COMM_RANK(COMM, RANK, IERROR)
    INTEGER COMM, RANK, IERROR

MPI_COMM_REMOTE_GROUP(COMM, GROUP, IERROR)
    INTEGER COMM, GROUP, IERROR

MPI_COMM_REMOTE_SIZE(COMM, SIZE, IERROR)
    INTEGER COMM, SIZE, IERROR

MPI_COMM_SET_ATTR(COMM, COMM_KEYVAL, ATTRIBUTE_VAL, IERROR)
    INTEGER COMM, COMM_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL

MPI_COMM_SET_INFO(COMM, INFO, IERROR)
    INTEGER COMM, INFO, IERROR

MPI_COMM_SET_NAME(COMM, COMM_NAME, IERROR)
    INTEGER COMM, IERROR
```

```
    CHARACTER*(*) COMM_NAME

MPI_COMM_SIZE(COMM, SIZE, IERROR)
    INTEGER COMM, SIZE, IERROR

MPI_COMM_SPLIT(COMM, COLOR, KEY, NEWCOMM, IERROR)
    INTEGER COMM, COLOR, KEY, NEWCOMM, IERROR

MPI_COMM_SPLIT_TYPE(COMM, SPLIT_TYPE, KEY, INFO, NEWCOMM, IERROR)
    INTEGER COMM, SPLIT_TYPE, KEY, INFO, NEWCOMM, IERROR

MPI_COMM_TEST_INTER(COMM, FLAG, IERROR)
    INTEGER COMM, IERROR
    LOGICAL FLAG

MPI_GROUP_COMPARE(GROUP1, GROUP2, RESULT, IERROR)
    INTEGER GROUP1, GROUP2, RESULT, IERROR

MPI_GROUP_DIFFERENCE(GROUP1, GROUP2, NEWGROUP, IERROR)
    INTEGER GROUP1, GROUP2, NEWGROUP, IERROR

MPI_GROUP_EXCL(GROUP, N, RANKS, NEWGROUP, IERROR)
    INTEGER GROUP, N, RANKS(*), NEWGROUP, IERROR

MPI_GROUP_FREE(GROUP, IERROR)
    INTEGER GROUP, IERROR

MPI_GROUP_INCL(GROUP, N, RANKS, NEWGROUP, IERROR)
    INTEGER GROUP, N, RANKS(*), NEWGROUP, IERROR

MPI_GROUP_INTERSECTION(GROUP1, GROUP2, NEWGROUP, IERROR)
    INTEGER GROUP1, GROUP2, NEWGROUP, IERROR

MPI_GROUP_RANGE_EXCL(GROUP, N, RANGES, NEWGROUP, IERROR)
    INTEGER GROUP, N, RANGES(3,*), NEWGROUP, IERROR

MPI_GROUP_RANGE_INCL(GROUP, N, RANGES, NEWGROUP, IERROR)
    INTEGER GROUP, N, RANGES(3,*), NEWGROUP, IERROR

MPI_GROUP_RANK(GROUP, RANK, IERROR)
    INTEGER GROUP, RANK, IERROR

MPI_GROUP_SIZE(GROUP, SIZE, IERROR)
    INTEGER GROUP, SIZE, IERROR

MPI_GROUP_TRANSLATE_RANKS(GROUP1, N, RANKS1, GROUP2, RANKS2, IERROR)
    INTEGER GROUP1, N, RANKS1(*), GROUP2, RANKS2(*), IERROR

MPI_GROUP_UNION(GROUP1, GROUP2, NEWGROUP, IERROR)
    INTEGER GROUP1, GROUP2, NEWGROUP, IERROR

MPI_INTERCOMM_CREATE(LOCAL_COMM, LOCAL_LEADER, PEER_COMM, REMOTE_LEADER,
              TAG, NEWINTERCOMM, IERROR)
    INTEGER LOCAL_COMM, LOCAL_LEADER, PEER_COMM, REMOTE_LEADER, TAG,
    NEWINTERCOMM, IERROR
```

```
MPI_INTERCOMM_MERGE(INTERCOMM, HIGH, NEWINTRACOMM, IERROR)
    INTEGER INTERCOMM, NEWINTRACOMM, IERROR
    LOGICAL HIGH

MPI_TYPE_CREATE_KEYVAL(TYPE_COPY_ATTR_FN, TYPE_DELETE_ATTR_FN, TYPE_KEYVAL,
            EXTRA_STATE, IERROR)
    EXTERNAL TYPE_COPY_ATTR_FN, TYPE_DELETE_ATTR_FN
    INTEGER TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

MPI_TYPE_DELETE_ATTR(DATATYPE, TYPE_KEYVAL, IERROR)
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR

MPI_TYPE_DUP_FN(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
            ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDTYPE, TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_TYPE_FREE_KEYVAL(TYPE_KEYVAL, IERROR)
    INTEGER TYPE_KEYVAL, IERROR

MPI_TYPE_GET_ATTR(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL
    LOGICAL FLAG

MPI_TYPE_GET_NAME(DATATYPE, TYPE_NAME, RESULTLEN, IERROR)
    INTEGER DATATYPE, RESULTLEN, IERROR
    CHARACTER*(*) TYPE_NAME

MPI_TYPE_NULL_COPY_FN(OLDTYPE, TYPE_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
            ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDTYPE, TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_TYPE_NULL_DELETE_FN(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE,
            IERROR)
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE

MPI_TYPE_SET_ATTR(DATATYPE, TYPE_KEYVAL, ATTRIBUTE_VAL, IERROR)
    INTEGER DATATYPE, TYPE_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL

MPI_TYPE_SET_NAME(DATATYPE, TYPE_NAME, IERROR)
    INTEGER DATATYPE, IERROR
    CHARACTER*(*) TYPE_NAME

MPI_WIN_CREATE_KEYVAL(WIN_COPY_ATTR_FN, WIN_DELETE_ATTR_FN, WIN_KEYVAL,
```

```
              EXTRA_STATE, IERROR)
    EXTERNAL WIN_COPY_ATTR_FN, WIN_DELETE_ATTR_FN
    INTEGER WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

MPI_WIN_DELETE_ATTR(WIN, WIN_KEYVAL, IERROR)
    INTEGER WIN, WIN_KEYVAL, IERROR

MPI_WIN_DUP_FN(OLDWIN, WIN_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDWIN, WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_WIN_FREE_KEYVAL(WIN_KEYVAL, IERROR)
    INTEGER WIN_KEYVAL, IERROR

MPI_WIN_GET_ATTR(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
    INTEGER WIN, WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL
    LOGICAL FLAG

MPI_WIN_GET_NAME(WIN, WIN_NAME, RESULTLEN, IERROR)
    INTEGER WIN, RESULTLEN, IERROR
    CHARACTER*(*) WIN_NAME

MPI_WIN_NULL_COPY_FN(OLDWIN, WIN_KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT, FLAG, IERROR)
    INTEGER OLDWIN, WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
        ATTRIBUTE_VAL_OUT
    LOGICAL FLAG

MPI_WIN_NULL_DELETE_FN(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR)
    INTEGER WIN, WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE

MPI_WIN_SET_ATTR(WIN, WIN_KEYVAL, ATTRIBUTE_VAL, IERROR)
    INTEGER WIN, WIN_KEYVAL, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL

MPI_WIN_SET_NAME(WIN, WIN_NAME, IERROR)
    INTEGER WIN, IERROR
    CHARACTER*(*) WIN_NAME
```

A.4.5 Process Topologies Fortran Bindings

```
MPI_CART_COORDS(COMM, RANK, MAXDIMS, COORDS, IERROR)
    INTEGER COMM, RANK, MAXDIMS, COORDS(*), IERROR

MPI_CART_CREATE(COMM_OLD, NDIMS, DIMS, PERIODS, REORDER, COMM_CART, IERROR)
```

```
    INTEGER COMM_OLD, NDIMS, DIMS(*), COMM_CART, IERROR
    LOGICAL PERIODS(*), REORDER

MPI_CARTDIM_GET(COMM, NDIMS, IERROR)
    INTEGER COMM, NDIMS, IERROR

MPI_CART_GET(COMM, MAXDIMS, DIMS, PERIODS, COORDS, IERROR)
    INTEGER COMM, MAXDIMS, DIMS(*), COORDS(*), IERROR
    LOGICAL PERIODS(*)

MPI_CART_MAP(COMM, NDIMS, DIMS, PERIODS, NEWRANK, IERROR)
    INTEGER COMM, NDIMS, DIMS(*), NEWRANK, IERROR
    LOGICAL PERIODS(*)

MPI_CART_RANK(COMM, COORDS, RANK, IERROR)
    INTEGER COMM, COORDS(*), RANK, IERROR

MPI_CART_SHIFT(COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR)
    INTEGER COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR

MPI_CART_SUB(COMM, REMAIN_DIMS, NEWCOMM, IERROR)
    INTEGER COMM, NEWCOMM, IERROR
    LOGICAL REMAIN_DIMS(*)

MPI_DIMS_CREATE(NNODES, NDIMS, DIMS, IERROR)
    INTEGER NNODES, NDIMS, DIMS(*), IERROR

MPI_DIST_GRAPH_CREATE_ADJACENT(COMM_OLD, INDEGREE, SOURCES, SOURCEWEIGHTS,
             OUTDEGREE, DESTINATIONS, DESTWEIGHTS, INFO, REORDER,
             COMM_DIST_GRAPH, IERROR)
    INTEGER COMM_OLD, INDEGREE, SOURCES(*), SOURCEWEIGHTS(*), OUTDEGREE,
        DESTINATIONS(*), DESTWEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
    LOGICAL REORDER

MPI_DIST_GRAPH_CREATE(COMM_OLD, N, SOURCES, DEGREES, DESTINATIONS, WEIGHTS,
             INFO, REORDER, COMM_DIST_GRAPH, IERROR)
    INTEGER COMM_OLD, N, SOURCES(*), DEGREES(*), DESTINATIONS(*),
        WEIGHTS(*), INFO, COMM_DIST_GRAPH, IERROR
    LOGICAL REORDER

MPI_DIST_GRAPH_NEIGHBORS(COMM, MAXINDEGREE, SOURCES, SOURCEWEIGHTS,
             MAXOUTDEGREE, DESTINATIONS, DESTWEIGHTS, IERROR)
    INTEGER COMM, MAXINDEGREE, SOURCES(*), SOURCEWEIGHTS(*), MAXOUTDEGREE,
        DESTINATIONS(*), DESTWEIGHTS(*), IERROR

MPI_DIST_GRAPH_NEIGHBORS_COUNT(COMM, INDEGREE, OUTDEGREE, WEIGHTED, IERROR)
    INTEGER COMM, INDEGREE, OUTDEGREE, IERROR
    LOGICAL WEIGHTED

MPI_GRAPH_CREATE(COMM_OLD, NNODES, INDEX, EDGES, REORDER, COMM_GRAPH,
             IERROR)
    INTEGER COMM_OLD, NNODES, INDEX(*), EDGES(*), COMM_GRAPH, IERROR
    LOGICAL REORDER
```

```
MPI_GRAPHDIMS_GET(COMM, NNODES, NEDGES, IERROR)
    INTEGER COMM, NNODES, NEDGES, IERROR

MPI_GRAPH_GET(COMM, MAXINDEX, MAXEDGES, INDEX, EDGES, IERROR)
    INTEGER COMM, MAXINDEX, MAXEDGES, INDEX(*), EDGES(*), IERROR

MPI_GRAPH_MAP(COMM, NNODES, INDEX, EDGES, NEWRANK, IERROR)
    INTEGER COMM, NNODES, INDEX(*), EDGES(*), NEWRANK, IERROR

MPI_GRAPH_NEIGHBORS(COMM, RANK, MAXNEIGHBORS, NEIGHBORS, IERROR)
    INTEGER COMM, RANK, MAXNEIGHBORS, NEIGHBORS(*), IERROR

MPI_GRAPH_NEIGHBORS_COUNT(COMM, RANK, NNEIGHBORS, IERROR)
    INTEGER COMM, RANK, NNEIGHBORS, IERROR

MPI_INEIGHBOR_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR

MPI_INEIGHBOR_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS,
             DISPLS, RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
        REQUEST, IERROR

MPI_INEIGHBOR_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, REQUEST, IERROR

MPI_INEIGHBOR_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF,
             RECVCOUNTS, RDISPLS, RECVTYPE, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
        RECVTYPE, COMM, REQUEST, IERROR

MPI_INEIGHBOR_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
             RECVCOUNTS, RDISPLS, RECVTYPES, COMM, REQUEST, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) SDISPLS(*), RDISPLS(*)
    INTEGER SENDCOUNTS(*), SENDTYPES(*), RECVCOUNTS(*), RECVTYPES(*), COMM,
        REQUEST, IERROR

MPI_NEIGHBOR_ALLGATHER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR

MPI_NEIGHBOR_ALLGATHERV(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNTS,
             DISPLS, RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNTS(*), DISPLS(*), RECVTYPE, COMM,
```

```
        IERROR

MPI_NEIGHBOR_ALLTOALL(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT,
             RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, COMM, IERROR

MPI_NEIGHBOR_ALLTOALLV(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPE, RECVBUF,
             RECVCOUNTS, RDISPLS,
    RECVTYPE, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER SENDCOUNTS(*), SDISPLS(*), SENDTYPE, RECVCOUNTS(*), RDISPLS(*),
    RECVTYPE, COMM, IERROR

MPI_NEIGHBOR_ALLTOALLW(SENDBUF, SENDCOUNTS, SDISPLS, SENDTYPES, RECVBUF,
             RECVCOUNTS, RDISPLS, RECVTYPES, COMM, IERROR)
    <type> SENDBUF(*), RECVBUF(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) SDISPLS(*), RDISPLS(*)
    INTEGER SENDCOUNTS(*), SENDTYPES(*), RECVCOUNTS(*), RECVTYPES(*), COMM,
        IERROR

MPI_TOPO_TEST(COMM, STATUS, IERROR)
    INTEGER COMM, STATUS, IERROR
```

A.4.6 MPI Environmental Management Fortran Bindings

```
DOUBLE PRECISION MPI_WTICK()

DOUBLE PRECISION MPI_WTIME()

MPI_ABORT(COMM, ERRORCODE, IERROR)
    INTEGER COMM, ERRORCODE, IERROR

MPI_ADD_ERROR_CLASS(ERRORCLASS, IERROR)
    INTEGER ERRORCLASS, IERROR

MPI_ADD_ERROR_CODE(ERRORCLASS, ERRORCODE, IERROR)
    INTEGER ERRORCLASS, ERRORCODE, IERROR

MPI_ADD_ERROR_STRING(ERRORCODE, STRING, IERROR)
    INTEGER ERRORCODE, IERROR
    CHARACTER*(*) STRING

MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
    INTEGER INFO, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

If the Fortran compiler provides `TYPE(C_PTR)`, then overloaded by:

```
  INTERFACE MPI_ALLOC_MEM
    SUBROUTINE MPI_ALLOC_MEM(SIZE, INFO, BASEPTR, IERROR)
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  INFO, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE, BASEPTR
```

```
    END SUBROUTINE
    SUBROUTINE MPI_ALLOC_MEM_CPTR(SIZE, INFO, BASEPTR, IERROR)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  INFO, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
      TYPE(C_PTR) ::  BASEPTR
    END SUBROUTINE
  END INTERFACE

MPI_COMM_CALL_ERRHANDLER(COMM, ERRORCODE, IERROR)
    INTEGER COMM, ERRORCODE, IERROR

MPI_COMM_CREATE_ERRHANDLER(COMM_ERRHANDLER_FN, ERRHANDLER, IERROR)
    EXTERNAL COMM_ERRHANDLER_FN
    INTEGER ERRHANDLER, IERROR

MPI_COMM_GET_ERRHANDLER(COMM, ERRHANDLER, IERROR)
    INTEGER COMM, ERRHANDLER, IERROR

MPI_COMM_SET_ERRHANDLER(COMM, ERRHANDLER, IERROR)
    INTEGER COMM, ERRHANDLER, IERROR

MPI_ERRHANDLER_FREE(ERRHANDLER, IERROR)
    INTEGER ERRHANDLER, IERROR

MPI_ERROR_CLASS(ERRORCODE, ERRORCLASS, IERROR)
    INTEGER ERRORCODE, ERRORCLASS, IERROR

MPI_ERROR_STRING(ERRORCODE, STRING, RESULTLEN, IERROR)
    INTEGER ERRORCODE, RESULTLEN, IERROR
    CHARACTER*(*) STRING

MPI_FILE_CALL_ERRHANDLER(FH, ERRORCODE, IERROR)
    INTEGER FH, ERRORCODE, IERROR

MPI_FILE_CREATE_ERRHANDLER(FILE_ERRHANDLER_FN, ERRHANDLER, IERROR)
    EXTERNAL FILE_ERRHANDLER_FN
    INTEGER ERRHANDLER, IERROR

MPI_FILE_GET_ERRHANDLER(FILE, ERRHANDLER, IERROR)
    INTEGER FILE, ERRHANDLER, IERROR

MPI_FILE_SET_ERRHANDLER(FILE, ERRHANDLER, IERROR)
    INTEGER FILE, ERRHANDLER, IERROR

MPI_FINALIZED(FLAG, IERROR)
    LOGICAL FLAG
    INTEGER IERROR

MPI_FINALIZE(IERROR)
    INTEGER IERROR

MPI_FREE_MEM(BASE, IERROR)
```

```
    <type> BASE(*)
    INTEGER IERROR

MPI_GET_LIBRARY_VERSION(VERSION, RESULTLEN, IERROR)
    CHARACTER*(*) VERSION
    INTEGER RESULTLEN,IERROR

MPI_GET_PROCESSOR_NAME( NAME, RESULTLEN, IERROR)
    CHARACTER*(*) NAME
    INTEGER RESULTLEN,IERROR

MPI_GET_VERSION(VERSION, SUBVERSION, IERROR)
    INTEGER VERSION, SUBVERSION, IERROR

MPI_INITIALIZED(FLAG, IERROR)
    LOGICAL FLAG
    INTEGER IERROR

MPI_INIT(IERROR)
    INTEGER IERROR

MPI_WIN_CALL_ERRHANDLER(WIN, ERRORCODE, IERROR)
    INTEGER WIN, ERRORCODE, IERROR

MPI_WIN_CREATE_ERRHANDLER(WIN_ERRHANDLER_FN, ERRHANDLER, IERROR)
    EXTERNAL WIN_ERRHANDLER_FN
    INTEGER ERRHANDLER, IERROR

MPI_WIN_GET_ERRHANDLER(WIN, ERRHANDLER, IERROR)
    INTEGER WIN, ERRHANDLER, IERROR

MPI_WIN_SET_ERRHANDLER(WIN, ERRHANDLER, IERROR)
    INTEGER WIN, ERRHANDLER, IERROR
```

A.4.7 The Info Object Fortran Bindings

```
MPI_INFO_CREATE(INFO, IERROR)
    INTEGER INFO, IERROR

MPI_INFO_DELETE(INFO, KEY, IERROR)
    INTEGER INFO, IERROR
    CHARACTER*(*) KEY

MPI_INFO_DUP(INFO, NEWINFO, IERROR)
    INTEGER INFO, NEWINFO, IERROR

MPI_INFO_FREE(INFO, IERROR)
    INTEGER INFO, IERROR

MPI_INFO_GET(INFO, KEY, VALUELEN, VALUE, FLAG, IERROR)
    INTEGER INFO, VALUELEN, IERROR
    CHARACTER*(*) KEY, VALUE
    LOGICAL FLAG
```

```
MPI_INFO_GET_NKEYS(INFO, NKEYS, IERROR)
    INTEGER INFO, NKEYS, IERROR

MPI_INFO_GET_NTHKEY(INFO, N, KEY, IERROR)
    INTEGER INFO, N, IERROR
    CHARACTER*(*) KEY

MPI_INFO_GET_VALUELEN(INFO, KEY, VALUELEN, FLAG, IERROR)
    INTEGER INFO, VALUELEN, IERROR
    LOGICAL FLAG
    CHARACTER*(*) KEY

MPI_INFO_SET(INFO, KEY, VALUE, IERROR)
    INTEGER INFO, IERROR
    CHARACTER*(*) KEY, VALUE
```

A.4.8 Process Creation and Management Fortran Bindings

```
MPI_CLOSE_PORT(PORT_NAME, IERROR)
    CHARACTER*(*) PORT_NAME
    INTEGER IERROR

MPI_COMM_ACCEPT(PORT_NAME, INFO, ROOT, COMM, NEWCOMM, IERROR)
    CHARACTER*(*) PORT_NAME
    INTEGER INFO, ROOT, COMM, NEWCOMM, IERROR

MPI_COMM_CONNECT(PORT_NAME, INFO, ROOT, COMM, NEWCOMM, IERROR)
    CHARACTER*(*) PORT_NAME
    INTEGER INFO, ROOT, COMM, NEWCOMM, IERROR

MPI_COMM_DISCONNECT(COMM, IERROR)
    INTEGER COMM, IERROR

MPI_COMM_GET_PARENT(PARENT, IERROR)
    INTEGER PARENT, IERROR

MPI_COMM_JOIN(FD, INTERCOMM, IERROR)
    INTEGER FD, INTERCOMM, IERROR

MPI_COMM_SPAWN(COMMAND, ARGV, MAXPROCS, INFO, ROOT, COMM, INTERCOMM,
              ARRAY_OF_ERRCODES, IERROR)
    CHARACTER*(*) COMMAND, ARGV(*)
    INTEGER INFO, MAXPROCS, ROOT, COMM, INTERCOMM, ARRAY_OF_ERRCODES(*),
    IERROR

MPI_COMM_SPAWN_MULTIPLE(COUNT, ARRAY_OF_COMMANDS, ARRAY_OF_ARGV,
              ARRAY_OF_MAXPROCS, ARRAY_OF_INFO, ROOT, COMM, INTERCOMM,
              ARRAY_OF_ERRCODES, IERROR)
    INTEGER COUNT, ARRAY_OF_INFO(*), ARRAY_OF_MAXPROCS(*), ROOT, COMM,
    INTERCOMM, ARRAY_OF_ERRCODES(*), IERROR
    CHARACTER*(*) ARRAY_OF_COMMANDS(*), ARRAY_OF_ARGV(COUNT, *)

MPI_LOOKUP_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)
```

```
    CHARACTER*(*) SERVICE_NAME, PORT_NAME
    INTEGER INFO, IERROR

MPI_OPEN_PORT(INFO, PORT_NAME, IERROR)
    CHARACTER*(*) PORT_NAME
    INTEGER INFO, IERROR

MPI_PUBLISH_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)
    INTEGER INFO, IERROR
    CHARACTER*(*) SERVICE_NAME, PORT_NAME

MPI_UNPUBLISH_NAME(SERVICE_NAME, INFO, PORT_NAME, IERROR)
    INTEGER INFO, IERROR
    CHARACTER*(*) SERVICE_NAME, PORT_NAME
```

A.4.9 One-Sided Communications Fortran Bindings

```
MPI_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
            TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE,TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, OP, WIN, IERROR

MPI_COMPARE_AND_SWAP(ORIGIN_ADDR, COMPARE_ADDR, RESULT_ADDR, DATATYPE,
            TARGET_RANK, TARGET_DISP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), COMPARE_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER DATATYPE, TARGET_RANK, WIN, IERROR

MPI_FETCH_AND_OP(ORIGIN_ADDR, RESULT_ADDR, DATATYPE, TARGET_RANK,
            TARGET_DISP, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER DATATYPE, TARGET_RANK, OP, WIN, IERROR

MPI_GET_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_ADDR,
            RESULT_COUNT, RESULT_DATATYPE, TARGET_RANK, TARGET_DISP,
            TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR)
    <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_COUNT, RESULT_DATATYPE,
    TARGET_RANK, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, IERROR

MPI_GET(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
            TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, IERROR
```

```
MPI_PUT(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
             TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, IERROR

MPI_RACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
             TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST,
             IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, OP, WIN, REQUEST, IERROR

MPI_RGET_ACCUMULATE(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE,
             RESULT_ADDR, RESULT_COUNT, RESULT_DATATYPE, TARGET_RANK,
             TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST,
             IERROR)
    <type> ORIGIN_ADDR(*), RESULT_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, RESULT_COUNT, RESULT_DATATYPE,
    TARGET_RANK, TARGET_COUNT, TARGET_DATATYPE, OP, WIN, REQUEST, IERROR

MPI_RGET(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
             TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, REQUEST,
             IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, REQUEST, IERROR

MPI_RPUT(ORIGIN_ADDR, ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK,
             TARGET_DISP, TARGET_COUNT, TARGET_DATATYPE, WIN, REQUEST,
             IERROR)
    <type> ORIGIN_ADDR(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) TARGET_DISP
    INTEGER ORIGIN_COUNT, ORIGIN_DATATYPE, TARGET_RANK, TARGET_COUNT,
    TARGET_DATATYPE, WIN, REQUEST, IERROR

MPI_WIN_ALLOCATE_SHARED(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
    INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

If the Fortran compiler provides `TYPE(C_PTR)`, then overloaded by:

```
  INTERFACE MPI_WIN_ALLOCATE_SHARED
    SUBROUTINE MPI_WIN_ALLOCATE_SHARED(SIZE, DISP_UNIT, INFO, COMM, &
          BASEPTR, WIN, IERROR)
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE, BASEPTR
```

```
    END SUBROUTINE
    SUBROUTINE MPI_WIN_ALLOCATE_SHARED_CPTR(SIZE, DISP_UNIT, INFO, COMM, &
          BASEPTR, WIN, IERROR)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
      TYPE(C_PTR) ::  BASEPTR
    END SUBROUTINE
  END INTERFACE

MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, WIN, IERROR)
    INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

If the Fortran compiler provides `TYPE(C_PTR)`, then overloaded by:

```
  INTERFACE MPI_WIN_ALLOCATE
    SUBROUTINE MPI_WIN_ALLOCATE(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, &
          WIN, IERROR)
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE, BASEPTR
    END SUBROUTINE
    SUBROUTINE MPI_WIN_ALLOCATE_CPTR(SIZE, DISP_UNIT, INFO, COMM, BASEPTR, &
          WIN, IERROR)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  DISP_UNIT, INFO, COMM, WIN, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
      TYPE(C_PTR) ::  BASEPTR
    END SUBROUTINE
  END INTERFACE

MPI_WIN_ATTACH(WIN, BASE, SIZE, IERROR)
    INTEGER WIN, IERROR
    <type> BASE(*)
    INTEGER (KIND=MPI_ADDRESS_KIND) SIZE

MPI_WIN_COMPLETE(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_CREATE(BASE, SIZE, DISP_UNIT, INFO, COMM, WIN, IERROR)
    <type> BASE(*)
    INTEGER(KIND=MPI_ADDRESS_KIND) SIZE
    INTEGER DISP_UNIT, INFO, COMM, WIN, IERROR

MPI_WIN_CREATE_DYNAMIC(INFO, COMM, WIN, IERROR)
    INTEGER INFO, COMM, WIN, IERROR

MPI_WIN_DETACH(WIN, BASE, IERROR)
    INTEGER WIN, IERROR
```

```
    <type> BASE(*)

MPI_WIN_FENCE(ASSERT, WIN, IERROR)
    INTEGER ASSERT, WIN, IERROR

MPI_WIN_FLUSH_ALL(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_FLUSH_LOCAL_ALL(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_FLUSH_LOCAL(RANK, WIN, IERROR)
    INTEGER RANK, WIN, IERROR

MPI_WIN_FLUSH(RANK, WIN, IERROR)
    INTEGER RANK, WIN, IERROR

MPI_WIN_FREE(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_GET_GROUP(WIN, GROUP, IERROR)
    INTEGER WIN, GROUP, IERROR

MPI_WIN_GET_INFO(WIN, INFO_USED, IERROR)
    INTEGER WIN, INFO_USED, IERROR

MPI_WIN_LOCK_ALL(ASSERT, WIN, IERROR)
    INTEGER ASSERT, WIN, IERROR

MPI_WIN_LOCK(LOCK_TYPE, RANK, ASSERT, WIN, IERROR)
    INTEGER LOCK_TYPE, RANK, ASSERT, WIN, IERROR

MPI_WIN_POST(GROUP, ASSERT, WIN, IERROR)
    INTEGER GROUP, ASSERT, WIN, IERROR

MPI_WIN_SET_INFO(WIN, INFO, IERROR)
    INTEGER WIN, INFO, IERROR

MPI_WIN_SHARED_QUERY(WIN, RANK, SIZE, DISP_UNIT, BASEPTR, IERROR)
    INTEGER WIN, RANK, DISP_UNIT, IERROR
    INTEGER (KIND=MPI_ADDRESS_KIND) SIZE, BASEPTR
```

If the Fortran compiler provides TYPE(C_PTR), then overloaded by:

```
  INTERFACE MPI_WIN_SHARED_QUERY
    SUBROUTINE MPI_WIN_SHARED_QUERY(WIN, RANK, SIZE, DISP_UNIT, &
          BASEPTR, IERROR)
      IMPORT ::  MPI_ADDRESS_KIND
      INTEGER ::  WIN, RANK, DISP_UNIT, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE, BASEPTR
    END SUBROUTINE
    SUBROUTINE MPI_WIN_SHARED_QUERY_CPTR(WIN, RANK, SIZE, DISP_UNIT, &
          BASEPTR, IERROR)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      IMPORT ::  MPI_ADDRESS_KIND
```

```
      INTEGER ::  WIN, RANK, DISP_UNIT, IERROR
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  SIZE
      TYPE(C_PTR) ::  BASEPTR
    END SUBROUTINE
  END INTERFACE

MPI_WIN_START(GROUP, ASSERT, WIN, IERROR)
    INTEGER GROUP, ASSERT, WIN, IERROR

MPI_WIN_SYNC(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_TEST(WIN, FLAG, IERROR)
    INTEGER WIN, IERROR
    LOGICAL FLAG

MPI_WIN_UNLOCK_ALL(WIN, IERROR)
    INTEGER WIN, IERROR

MPI_WIN_UNLOCK(RANK, WIN, IERROR)
    INTEGER RANK, WIN, IERROR

MPI_WIN_WAIT(WIN, IERROR)
    INTEGER WIN, IERROR
```

A.4.10 External Interfaces Fortran Bindings

```
MPI_GREQUEST_COMPLETE(REQUEST, IERROR)
    INTEGER REQUEST, IERROR

MPI_GREQUEST_START(QUERY_FN, FREE_FN, CANCEL_FN, EXTRA_STATE, REQUEST,
            IERROR)
    INTEGER REQUEST, IERROR
    EXTERNAL QUERY_FN, FREE_FN, CANCEL_FN
    INTEGER (KIND=MPI_ADDRESS_KIND) EXTRA_STATE

MPI_INIT_THREAD(REQUIRED, PROVIDED, IERROR)
    INTEGER REQUIRED, PROVIDED, IERROR

MPI_IS_THREAD_MAIN(FLAG, IERROR)
    LOGICAL FLAG
    INTEGER IERROR

MPI_QUERY_THREAD(PROVIDED, IERROR)
    INTEGER PROVIDED, IERROR

MPI_STATUS_SET_CANCELLED(STATUS, FLAG, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), IERROR
    LOGICAL FLAG

MPI_STATUS_SET_ELEMENTS(STATUS, DATATYPE, COUNT, IERROR)
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, COUNT, IERROR

MPI_STATUS_SET_ELEMENTS_X(STATUS, DATATYPE, COUNT, IERROR)
```

```
    INTEGER STATUS(MPI_STATUS_SIZE), DATATYPE, IERROR
    INTEGER (KIND=MPI_COUNT_KIND) COUNT
```

A.4.11 I/O Fortran Bindings

```
MPI_CONVERSION_FN_NULL(USERBUF, DATATYPE, COUNT, FILEBUF, POSITION,
             EXTRA_STATE, IERROR)
    <TYPE> USERBUF(*), FILEBUF(*)
    INTEGER COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) POSITION
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE

MPI_FILE_CLOSE(FH, IERROR)
    INTEGER FH, IERROR

MPI_FILE_DELETE(FILENAME, INFO, IERROR)
    CHARACTER*(*) FILENAME
    INTEGER INFO, IERROR

MPI_FILE_GET_AMODE(FH, AMODE, IERROR)
    INTEGER FH, AMODE, IERROR

MPI_FILE_GET_ATOMICITY(FH, FLAG, IERROR)
    INTEGER FH, IERROR
    LOGICAL FLAG

MPI_FILE_GET_BYTE_OFFSET(FH, OFFSET, DISP, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET, DISP

MPI_FILE_GET_GROUP(FH, GROUP, IERROR)
    INTEGER FH, GROUP, IERROR

MPI_FILE_GET_INFO(FH, INFO_USED, IERROR)
    INTEGER FH, INFO_USED, IERROR

MPI_FILE_GET_POSITION(FH, OFFSET, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_GET_POSITION_SHARED(FH, OFFSET, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_GET_SIZE(FH, SIZE, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) SIZE

MPI_FILE_GET_TYPE_EXTENT(FH, DATATYPE, EXTENT, IERROR)
    INTEGER FH, DATATYPE, IERROR
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTENT

MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)
```

```
    INTEGER FH, ETYPE, FILETYPE, IERROR
    CHARACTER*(*) DATAREP
    INTEGER(KIND=MPI_OFFSET_KIND) DISP

MPI_FILE_IREAD_ALL(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_IREAD_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_IREAD_AT(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_IREAD(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_IREAD_SHARED(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_IWRITE_ALL(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_IWRITE_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_IWRITE_AT(FH, OFFSET, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_IWRITE(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_IWRITE_SHARED(FH, BUF, COUNT, DATATYPE, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, REQUEST, IERROR

MPI_FILE_OPEN(COMM, FILENAME, AMODE, INFO, FH, IERROR)
    CHARACTER*(*) FILENAME
    INTEGER COMM, AMODE, INFO, FH, IERROR

MPI_FILE_PREALLOCATE(FH, SIZE, IERROR)
```

```
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) SIZE

MPI_FILE_READ_ALL_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR

MPI_FILE_READ_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_AT_ALL_BEGIN(FH, OFFSET, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_READ_AT_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_READ_AT(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_READ(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_ORDERED_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR

MPI_FILE_READ_ORDERED_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_ORDERED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_READ_SHARED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
```

```
MPI_FILE_SEEK(FH, OFFSET, WHENCE, IERROR)
    INTEGER FH, WHENCE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_SEEK_SHARED(FH, OFFSET, WHENCE, IERROR)
    INTEGER FH, WHENCE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_SET_ATOMICITY(FH, FLAG, IERROR)
    INTEGER FH, IERROR
    LOGICAL FLAG

MPI_FILE_SET_INFO(FH, INFO, IERROR)
    INTEGER FH, INFO, IERROR

MPI_FILE_SET_SIZE(FH, SIZE, IERROR)
    INTEGER FH, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) SIZE

MPI_FILE_SET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, INFO, IERROR)
    INTEGER FH, ETYPE, FILETYPE, INFO, IERROR
    CHARACTER*(*) DATAREP
    INTEGER(KIND=MPI_OFFSET_KIND) DISP

MPI_FILE_SYNC(FH, IERROR)
    INTEGER FH, IERROR

MPI_FILE_WRITE_ALL_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR

MPI_FILE_WRITE_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_ALL(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_AT_ALL_BEGIN(FH, OFFSET, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_WRITE_AT_ALL_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_AT_ALL(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_WRITE_AT(FH, OFFSET, BUF, COUNT, DATATYPE, STATUS, IERROR)
```

```
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR
    INTEGER(KIND=MPI_OFFSET_KIND) OFFSET

MPI_FILE_WRITE(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_ORDERED_BEGIN(FH, BUF, COUNT, DATATYPE, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, IERROR

MPI_FILE_WRITE_ORDERED_END(FH, BUF, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_ORDERED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_FILE_WRITE_SHARED(FH, BUF, COUNT, DATATYPE, STATUS, IERROR)
    <type> BUF(*)
    INTEGER FH, COUNT, DATATYPE, STATUS(MPI_STATUS_SIZE), IERROR

MPI_REGISTER_DATAREP(DATAREP, READ_CONVERSION_FN, WRITE_CONVERSION_FN,
            DTYPE_FILE_EXTENT_FN, EXTRA_STATE, IERROR)
    CHARACTER*(*) DATAREP
    EXTERNAL READ_CONVERSION_FN, WRITE_CONVERSION_FN, DTYPE_FILE_EXTENT_FN
    INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE
    INTEGER IERROR
```

A.4.12 Language Bindings Fortran Bindings

```
MPI_F_SYNC_REG(buf)
    <type> buf(*)

MPI_SIZEOF(X, SIZE, IERROR)
    <type> X
    INTEGER SIZE, IERROR

MPI_STATUS_F082F(F08_STATUS, F_STATUS, IERROR)
    TYPE(MPI_Status) ::  F08_STATUS
    INTEGER ::  F_STATUS(MPI_STATUS_SIZE)
    INTEGER IERROR

MPI_STATUS_F2F08(F_STATUS, F08_STATUS, IERROR)
    INTEGER ::  F_STATUS(MPI_STATUS_SIZE)
    TYPE(MPI_Status) ::  F08_STATUS
    INTEGER IERROR

MPI_TYPE_CREATE_F90_COMPLEX(P, R, NEWTYPE, IERROR)
    INTEGER P, R, NEWTYPE, IERROR
```

```
MPI_TYPE_CREATE_F90_INTEGER(R, NEWTYPE, IERROR)
    INTEGER R, NEWTYPE, IERROR

MPI_TYPE_CREATE_F90_REAL(P, R, NEWTYPE, IERROR)
    INTEGER P, R, NEWTYPE, IERROR

MPI_TYPE_MATCH_SIZE(TYPECLASS, SIZE, DATATYPE, IERROR)
    INTEGER TYPECLASS, SIZE, DATATYPE, IERROR
```

A.4.13 Tools / Profiling Interface Fortran Bindings

```
MPI_PCONTROL(LEVEL)
    INTEGER LEVEL
```

A.4.14 Deprecated Fortran Bindings

```
MPI_ATTR_DELETE(COMM, KEYVAL, IERROR)
    INTEGER COMM, KEYVAL, IERROR

MPI_ATTR_GET(COMM, KEYVAL, ATTRIBUTE_VAL, FLAG, IERROR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, IERROR
    LOGICAL FLAG

MPI_ATTR_PUT(COMM, KEYVAL, ATTRIBUTE_VAL, IERROR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, IERROR

MPI_DUP_FN(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT, FLAG, IERR)
    INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, IERR
    LOGICAL FLAG

MPI_KEYVAL_CREATE(COPY_FN, DELETE_FN, KEYVAL, EXTRA_STATE, IERROR)
    EXTERNAL COPY_FN, DELETE_FN
    INTEGER KEYVAL, EXTRA_STATE, IERROR

MPI_KEYVAL_FREE(KEYVAL, IERROR)
    INTEGER KEYVAL, IERROR

MPI_NULL_COPY_FN(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT, FLAG, IERR)
    INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, IERR
    LOGICAL FLAG

MPI_NULL_DELETE_FN(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERROR

SUBROUTINE COPY_FUNCTION(OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
              ATTRIBUTE_VAL_OUT, FLAG, IERR)
    INTEGER OLDCOMM, KEYVAL, EXTRA_STATE, ATTRIBUTE_VAL_IN,
    ATTRIBUTE_VAL_OUT, IERR
```

```
    LOGICAL FLAG

SUBROUTINE DELETE_FUNCTION(COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR)
    INTEGER COMM, KEYVAL, ATTRIBUTE_VAL, EXTRA_STATE, IERR
```