```
        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:

#390

```
  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)
```

```
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:

#390

```
  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:

#390
```
  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:

#390

```
  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)
```