Non-blocking point-to-point routines

- faster
- help avoid deadlocks
- possibility of overlapping communication with computation

mpi_wait

`mpi_wait(request, status, ierror)`

mpi_test

`mpi_test(req, flag, status, ierror)`

The flag argument is a logical. If `flag == true`, then the operation identified by the request has completed, otherwise it has not. Here is an example.

```
...
...
call mpi_isend(A(1), n, dp, dest, tag, comm, request, ierror)
...
...
10 call mpi_test(request, flag, status, ierror)
if(.not. flag) then
  goto 10
else
  ...
  ...
endif
```

Other routines

- `mpi_waitany`
- `mpi_testany`
- `mpi_waitsome`
- `mpi_testsome`
- `mpi_testall`

mpi_waitall

`mpi_waitall(count, array_of_requests, array_of_statuses, ierror)`

Here is an example of how it can be used.

```
! Instead of this...
!do i=1, p-1
!  call mpi\_wait(req(i), status, ierror)
!enddo
! ...do this
!
integer, allocatable :: array\_of\_requests(:), array\_of\_requests(:, :)
...
...
call mpi\_comm\_size(comm, p, ierror)
allocate(array\_of\_requests(p-1), array\_ofstatuses(mpi\_status\_size, p-1))
...
...
```

```
call mpi\_waitall(p-1, array\_of\_requests, array\_of\_statuses, ierror)
```