

```

1
2
3
4
5
6 Chapter 16
7
8
9
10
11
12
13
14 16.1 Existing Fortran Bindings
15
16 Arbitrarily picked MPI_SEND, MPI_STARTALL, MPI_TYPE_CREATE_STRUCT to show as
17 samples.
18
19
20 MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
21 <type> BUF(*)
22 INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
23
24
25 MPI_STARTALL(COUNT, ARRAY_OF_REQUESTS, IERROR)
26 INTEGER COUNT, ARRAY_OF_REQUESTS(*), IERROR
27
28
29 MPI_TYPE_CREATE_STRUCT(COUNT, ARRAY_OF_BLOCKLENGTHS,
30   ARRAY_OF_DISPLACEMENTS, ARRAY_OF_TYPES, NEWTYPE, IERROR)
31 INTEGER COUNT, ARRAY_OF_BLOCKLENGTHS(*), ARRAY_OF_TYPES(*), NEWTYPE,
32   IERROR
33 INTEGER(KIND=MPI_ADDRESS_KIND) ARRAY_OF_DISPLACEMENTS(*)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

```

16.2 Verbatim Fortran Bindings Samples

3 This is using simple “verbatim” latex style. Note that the INTEGER KIND stuff has not
 4 been finalized yet; we put in MPI_INT_KIND everywhere just to show what it would look
 5 like.

6 It is *possible* (likely?) that the MPI handles will be some other type in the new MPI-3
 7 Fortran interface, not INTEGER.

```

8
9

10 subroutine MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
11 import
12 <type>, dimension(*), intent(IN) :: BUF
13 integer(MPI_INT_KIND), intent(IN) :: COUNT
14 integer(MPI_INT_KIND), intent(IN) :: DATATYPE
15 integer(MPI_INT_KIND), intent(IN) :: DEST
16 integer(MPI_INT_KIND), intent(IN) :: TAG
17 integer(MPI_INT_KIND), intent(IN) :: COMM
18 integer(MPI_INT_KIND), intent(OUT) :: IERROR
19 end subroutine MPI_SEND
20

21 subroutine MPI_STARTALL(COUNT, ARRAY_OF_REQUESTS, IERROR)
22 import
23 integer(MPI_INT_KIND), intent(IN) :: COUNT
24 integer(MPI_INT_KIND), dimension(*), intent(INOUT) :: ARRAY_OF_REQUESTS
25 integer(MPI_INT_KIND), intent(OUT) :: IERROR
26 end subroutine MPI_STARTALL
27

28 subroutine MPI_TYPE_CREATE_STRUCT(COUNT, ARRAY_OF_BLOCKLENGTHS, &
29     ARRAY_OF_DISPLACEMENTS, ARRAY_OF_TYPES, NEWTYPE, IERROR)
30 import
31 integer(MPI_INT_KIND), intent(IN) :: COUNT
32 integer(MPI_INT_KIND), dimension(*), intent(IN) :: ARRAY_OF_BLOCKLENGTHS
33 integer(MPI_ADDRESS_KIND), dimension(*), intent(IN) :: ARRAY_OF_DISPLACEMENTS
34 integer(MPI_INT_KIND), dimension(*), intent(IN) :: ARRAY_OF_TYPES
35 integer(MPI_INT_KIND), intent(OUT) :: NEWTYPE
36 integer(MPI_INT_KIND), intent(OUT) :: IERROR
37 end subroutine MPI_TYPE_CREATE_STRUCT
38

39
40
41
42
43
44
45
46
47
48
```

16.3 Sample 1 Fortran Bindings

This uses similar L^AT_EX mechanisms as the existing “mpifbind” macros, but slightly modified.

Same disclaimers as above – I used MPI_INT_KIND everywhere just to see what it would look like.

```

1 subroutine MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
2   import
3   <type>, dimension(*), intent(IN) ::  BUF
4   integer(MPI_INT_KIND), intent(IN) ::  COUNT
5   integer(MPI_INT_KIND), intent(IN) ::  DATATYPE
6   integer(MPI_INT_KIND), intent(IN) ::  DEST
7   integer(MPI_INT_KIND), intent(IN) ::  TAG
8   integer(MPI_INT_KIND), intent(IN) ::  COMM
9   integer(MPI_INT_KIND), intent(OUT) ::  IERROR
10  end subroutine MPI_SEND
11
12 subroutine MPI_STARTALL(COUNT, ARRAY_OF_REQUESTS, IERROR)
13   import
14   integer(MPI_INT_KIND), intent(IN) ::  COUNT
15   integer(MPI_INT_KIND), dimension(*), intent(INOUT) :: :
16   ARRAY_OF_REQUESTS
17   integer(MPI_INT_KIND), intent(OUT) ::  IERROR
18  end subroutine MPI_STARTALL
19
20 subroutine MPI_TYPE_CREATE_STRUCT(COUNT, ARRAY_OF_BLOCKLENGTHS,
21   ARRAY_OF_DISPLACEMENTS, ARRAY_OF_TYPES, NEWTYPE, IERROR)
22   import
23   integer(MPI_INT_KIND), intent(IN) ::  COUNT
24   integer(MPI_INT_KIND), dimension(*), intent(IN) :: :
25   ARRAY_OF_BLOCKLENGTHS
26   integer(MPI_ADDRESS_KIND), dimension(*), intent(IN) :: :
27   ARRAY_OF_DISPLACEMENTS
28   integer(MPI_INT_KIND), dimension(*), intent(IN) ::  ARRAY_OF_TYPES
29   integer(MPI_INT_KIND), intent(OUT) ::  NEWTYPE
30   integer(MPI_INT_KIND), intent(OUT) ::  IERROR
31  end subroutine MPI_TYPE_CREATE_STRUCT
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

```