Basic MPI Datatypes

1. **elementary datatypes in C**

MPI datatype	**C datatype**
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	
MPI_PACKED	

2. **optional datatypes in C**

MPI datatype	**C(99) datatype**
MPI_LONG_LONG_INT	long long int

3. **elementary datatypes in Fortran**

MPI datatype	**Fortan datatype**
MPI_INTEGER	INTEGER
MPI_REAL	REAL
MPI_DOUBLE_PRECISION	DOUBLE PRECISION
MPI_COMPLEX	COMPLEX
MPI_LOGICAL	LOGICAL
MPI_CHARACTER	CHARACTER(1)
MPI_BYTE	
MPI_PACKED	

4. **optional datatypes (Fortran)**

MPI datatype	**Fortan datatype**
MPI_INTEGER1	
MPI_INTEGER2	
MPI_INTEGER4	
MPI_REAL2	
MPI_REAL4	
MPI_REAL8	

5. **reserved communicators in C and Fortran**
   MPI_COMM_WORLD - all processes
   MPI_COMM_SELF - contains only the process itself

6. **tags and sources**
   MPI_ANY_TAG - to e.g. receive messages with any tag
   MPI_ANY_SOURCE - to receive from any source

7. **MPI_Status**
   In C, status is a structure that contains three fields named MPI_SOURCE, MPI_TAG, and MPI_ERROR; the structure may contain additional fields. Thus, status.MPI_SOURCE, status.MPI_TAG and status.MPI_ERROR contain the source, tag, and error code, respectively, of the received message.

   In Fortran, status is an array of INTEGERs of size MPI_STATUS_SIZE. The constants MPI_SOURCE, MPI_TAG and MPI_ERROR are the indices of the entries that store the source, tag and error fields. Thus, status(MPI_SOURCE), status(MPI_TAG) and status(MPI_ERROR) contain, respectively, the source, tag and error code of the received message.

Man Page extracts

- **MPI_Init - Initializes the MPI execution environment**
  C Syntax:
  int MPI_Init(int *argc, char ***argv)

  Fortran Syntax:
  MPI_INIT(IERROR)

  Description:
  All MPI programs must contain a call to MPI_INIT; this routine must be called before any other MPI routine (apart from MPI_INITIALIZED) is called. The version for ANSI C accepts the argc and argv that are provided by the arguments to main.

Parameter	**Type**	**Description**
argc	IN	C/C++ only: Pointer to the number of arguments.
argv	IN	C/C++ only: Argument vector.
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Finalize - Terminates MPI execution environment**
  C Syntax:
  int MPI_Finalize(void)

  Fortran Syntax:
  MPI_FINALIZE(IERROR)

  Description:
  This routines cleans up all MPI state. Once this routine is called, no MPI routine (even MPI_INIT) may be called.

Parameter	**Type**	**Description**
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Comm_rank - Determines the rank of the calling process in the communicator.**
  C Syntax:
  int MPI_Comm_rank(MPI_Comm comm, int *rank)

  Fortran Syntax:
  MPI_COMM_RANK(COMM, RANK, IERROR)

  Description:
  This function gives the rank of the process in the particular communicator's group.

Parameter	Type	Description
comm	IN	Communicator (handle)
rank	OUT	Rank of the calling process in group of comm (integer).
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Comm_size - Returns the size of the group associated with a communicator.**
  C Syntax:
  int MPI_Comm_size(MPI_Comm comm, int *size)

  Fortran Syntax:
  MPI_COMM_SIZE(COMM, SIZE, IERROR)

  Description:
  This function indicates the number of processes involved in a communicator. For MPI_COMM_WORLD, it indicates the total number of processes available.

Parameter	Type	Description
comm	IN	Communicator (handle)
size	OUT	Number of processes in the group of comm (integer).

- **MPI_Get_processor_name - Gets the name of the processor.**
  C Syntax:
  int MPI_Get_processor_name(char *name, int *resultlen)

  Fortran Syntax:
  MPI_GET_PROCESSOR_NAME(NAME, RESULTLEN, IERROR)

  Description:
  This routine returns the name of the processor on which it was called at the moment of the call. The name is a character string for maximum flexibility. From this value it must be possible to identify a specific piece of hardware. The argument name must represent storage that is at least MPI_MAX_PROCESSOR_NAME characters long.

Parameter	Type	Description
name	OUT	A unique specifier for the actual (as opposed to virtual) node.
resultlen	OUT	Length (in characters) of result returned in name.
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Send - Performs a standard send.**
  C Syntax:
  int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

  Fortran Syntax:
  MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)

  Description:
  MPI_Send performs a standard-mode, blocking send.

Parameter	**Type**	**Description**
buf	IN	Initial address of send buffer (choice).
count	IN	Number of elements send (nonnegative integer).
datatype	IN	Datatype of each send buffer element (handle).
dest	IN	Rank of destination (integer).
tag	IN	Message tag (integer).
comm	IN	Communicator (handle).
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Recv - Basic receive.**
  C Syntax:
  int MPI_Recv(void *buf, int count, MPI_Datatype datatype,int source, int tag, MPI_Comm comm,MPI_Status *status)

  Fortran Syntax:
  MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS, IERROR)

  Description:
  This basic receive operation, MPI_Recv, is blocking: it returns only after the receive buffer contains the newly received message.

Parameter	**Type**	**Description**
count	IN	Maximum number of elements to receive (integer).
datatype	IN	Datatype of each receive buffer entry (handle).
source	IN	Rank of source (integer).
tag	IN	Message tag (integer).
comm	IN	Communicator (handle).
buf	OUT	Initial address of receive buffer (choice).
status	OUT	Status object (status).
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Wtime - Returns an elapsed time on the calling processor.**
  C Syntax:
  double MPI_Wtime(void)

  Fortran Syntax:
  DOUBLE PRECISION MPI_WTIME()

  Description:
  MPI_WTIME returns a floating-point number of seconds, representing elapsed wall-clock time since some time in the past. The ”time in the past” is guaranteed not to change during the life of the process. The user is responsible for converting large numbers of seconds to other units if they are preferred. This function is portable (it returns seconds, not ”ticks”), it allows high-resolution, and carries no unnecessary baggage.

- **MPI_Bcast - Broadcasts a message from the process with rank root to all other processes of the group.**
  C Syntax:
  int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

  Fortran Syntax:
  MPI_BCAST(BUFFER, COUNT, DATATYPE, ROOT, COMM, IERROR)

Description:
MPI_Bcast broadcasts a message from the process with rank root to all processes of the group, itself included. It is called by all members of group using the same arguments for comm, root. On return, the contents of root's communication buffer has been copied to all processes. General, derived datatypes are allowed for datatype. The type signature of count, datatype on any process must be equal to the type signature of count, datatype at the root.

Parameter	**Type**	**Description**
buffer	IN/OUT	Starting address of buffer (choice)
count	IN/OUT	Number of entries in buffer (integer)
datatype	IN/OUT	Data type of buffer (handle)
root	IN/OUT	Rank of broadcast root (integer)
comm	IN/OUT	Communicator (handle)
IERROR	OUT	Fortran only: Error status (integer)

- **MPI_Reduce - Reduces values on all processes within a group.**
C Syntax:
int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)

Fortran Syntax:
MPI_REDUCE(SENDBUF, RECVBUF, COUNT, DATATYPE, OP, ROOT, COMM, IERROR)

Description:
The global reduce functions (MPI_Reduce, MPI_Op_create, MPI_Op_free, MPI_Allreduce, MPI_Reduce_scatter, MPI_Scan) perform a global reduce operation (such as sum, max, logical AND, etc.) across all the members of a group.

Parameter	**Type**	**Description**
sendbuf	IN	Address of send buffer (choice).
count	IN	Number of elements in send buffer (integer).
datatype	IN	Data type of elements of send buffer (handle).
op	IN	Reduce operation (handle).
root	IN	Rank of root process (integer).
comm	IN	Communicator (handle).
recvbuf	OUT	Address of receive buffer (choice, significant only at root).
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Isend - Starts a standard-mode, nonblocking send.**
C Syntax:
int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request)

Fortran Syntax:
MPI_ISEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)

Description:
MPI_Isend starts a standard-mode, nonblocking send. Nonblocking calls allocate a communication request object and associate it with the request handle (the argument request). The request can be used later to query the status of the communication or wait for its completion.

Parameter	**Type**	**Description**
buf	IN	Initial address of send buffer (choice).
count	IN	Number of elements in send buffer (integer).
datatype	IN	Datatype of each send buffer element (handle).
dest	IN	Rank of destination (integer).
tag	IN	Message tag (integer).
comm	IN	Communicator (handle).
request	OUT	Communication request (handle).
IERROR	OUT	Fortran only: Error status (integer).

- **MPI_Irecv - Begins a nonblocking receive.**

  C Syntax:

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

  Fortran Syntax:

  MPI_IRECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)

  Description:

  Nonblocking calls allocate a communication request object and associate it with the request handle (the argument request). The request can be used later to query the status of the communication or wait for its completion.

  A nonblocking receive call indicates that the system may start writing data into the receive buffer. The receiver should not access any part of the receive buffer after a nonblocking receive operation is called, until the receive completes.

Parameter	**Type**	**Description**
buf	IN	Initial address of receive buffer (choice).
count	IN	Number of elements in receive buffer (integer).
datatype	IN	Datatype of each receive buffer entry (handle).
source	IN	Rank of source (integer).
tag	IN	Message tag (integer).
comm	IN	Communicator (handle).
request	OUT	Communication request (handle).
IERROR	OUT	Fortran only: Error status (integer).