# 1 MPI Reference Card

| | |
|---|---|
| `int MPI_Abort( comm, errorcode )`<br>`MPI_Comm comm;`<br>`int errorcode;` | Terminates MPI execution environment |
| `int MPI_Address( location, address)`<br>`void *location;`<br>`MPI_Aint *address;` | Gets the address of a location in memory |
| `int MPI_Allgatherv ( sendbuf, sendcount,`<br>`sendtype, recvbuf, recvcounts,`<br>`displs, recvtype, comm`<br>`)`<br>`void *sendbuf;`<br>`int sendcount;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int *recvcounts;`<br>`int *displs;`<br>`MPI_Datatype recvtype;`<br>`MPI_Comm comm;` | Gathers data from all tasks and deliver it to all |
| `int MPI_Allgather ( sendbuf, sendcount,`<br>`sendtype, recvbuf, recvcount,`<br>`recvtype, comm`<br>`)`<br>`void *sendbuf;`<br>`int sendcount;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int recvcount;`<br>`MPI_Datatype recvtype;`<br>`MPI_Comm comm;` | Gathers data from all tasks and distribute it to all |
| `int MPI_Allreduce ( sendbuf, recvbuf,`<br>`count, datatype, op, comm )`<br>`void *sendbuf;`<br>`void *recvbuf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`MPI_Op op;`<br>`MPI_Comm comm;` | Combines values from all processes and distribute the result back to all processes |
| `int MPI_Alltoallv ( sendbuf, sendcnts,`<br>`sdispls, sendtype, recvbuf,`<br>`recvcnts, rdispls, recvtype,`<br>`comm`<br>`)`<br>`void *sendbuf;`<br>`int *sendcnts;`<br>`int *sdispls;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int *recvcnts;`<br>`int *rdispls;`<br>`MPI_Datatype recvtype;`<br>`MPI_Comm comm;` | Sends data from all to all processes, with a displacement |

| | |
|---|---|
| `int MPI_Alltoall( sendbuf, sendcount, sendtype, recvbuf, recvcnt, recvtype, comm )`<br>`void *sendbuf;`<br>`int sendcount;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int recvcnt;`<br>`MPI_Datatype recvtype;`<br>`MPI_Comm comm;` | Sends data from all to all processes |
| `int MPI_Attr_delete ( comm, keyval )`<br>`MPI_Comm comm;`<br>`int keyval;` | Deletes attribute value associated with a key |
| `int MPI_Attr_get ( comm, keyval, attr_value, flag )`<br>`MPI_Comm comm;`<br>`int keyval;`<br>`void *attr_value;`<br>`int *flag;` | Retrieves attribute value by key |
| `int MPI_Attr_put ( comm, keyval, attr_value )`<br>`MPI_Comm comm;`<br>`int keyval;`<br>`void *attr_value;` | Stores attribute value associated with a key |
| `int MPI_Barrier ( comm )`<br>`MPI_Comm comm;` | Blocks until all process have reached this routine. |
| `int MPI_Bcast ( buffer, count, datatype, root, comm )`<br>`void *buffer;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int root;`<br>`MPI_Comm comm;` | Broadcasts a message from the process with rank "root" to all other processes of the group. |
| `int MPI_Bsend_init( buf, count, datatype, dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Builds a handle for a buffered send |
| `int MPI_Bsend( buf, count, datatype, dest, tag, comm )`<br>`void *buf;`<br>`int count, dest, tag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;` | Basic send with user-specified buffering |
| `int MPI_Buffer_attach( buffer, size )`<br>`void *buffer;`<br>`int size;` | Attaches a user-defined buffer for sending |

| | |
|---|---|
| `int MPI_Buffer_detach( bufferptr, size )`<br>`void *bufferptr;`<br>`int *size;` | Removes an existing buffer (for use in MPI_Bsend etc) |
| `int MPI_Cancel( request )`<br>`MPI_Request *request;` | Cancels a communication request |
| `int MPI_Cart_coords ( comm, rank,`<br>`maxdims, coords )`<br>`MPI_Comm comm;`<br>`int rank;`<br>`int maxdims;`<br>`int *coords;` | Determines process coords in cartesian topology given rank in group |
| `int MPI_Cart_create ( comm_old, ndims,`<br>`dims, periods, reorder,`<br>`comm_cart )`<br>`MPI_Comm comm_old;`<br>`int ndims;`<br>`int *dims;`<br>`int *periods;`<br>`int reorder;`<br>`MPI_Comm *comm_cart;` | Makes a new communicator to which topology information has been attached |
| `int MPI_Cart_get ( comm, maxdims, dims,`<br>`periods, coords )`<br>`MPI_Comm comm;`<br>`int maxdims;`<br>`int *dims, *periods, *coords;` | Retrieves Cartesian topology information associated with a communicator |
| `int MPI_Cart_map ( comm_old, ndims, dims,`<br>`periods, newrank )`<br>`MPI_Comm comm_old;`<br>`int ndims;`<br>`int *dims;`<br>`int *periods;`<br>`int *newrank;` | Maps process to Cartesian topology information |
| `int MPI_Cart_rank ( comm, coords, rank )`<br>`MPI_Comm comm;`<br>`int *coords;`<br>`int *rank;` | Determines process rank in communicator given Cartesian location |
| `int MPI_Cart_shift ( comm, direction,`<br>`displ, source, dest )`<br>`MPI_Comm comm;`<br>`int direction;`<br>`int displ;`<br>`int *source;`<br>`int *dest;` | Returns the shifted source and destination ranks, given a shift direction and amount |
| `int MPI_Cart_sub ( comm, remain_dims,`<br>`comm_new )`<br>`MPI_Comm comm;`<br>`int *remain_dims;`<br>`MPI_Comm *comm_new;` | Partitions a communicator into subgroups which form lower-dimensional cartesian subgrids |
| `int MPI_Cartdim_get ( comm, ndims )`<br>`MPI_Comm comm;`<br>`int *ndims;` | Retrieves Cartesian topology information associated with a communicator |

| | |
|---|---|
| `int MPI_Comm_compare ( comm1, comm2, result )`<br>`MPI_Comm comm1;`<br>`MPI_Comm comm2;`<br>`int *result;` | Compares two communicators |
| `int MPI_Comm_create ( comm, group, comm_out )`<br>`MPI_Comm comm;`<br>`MPI_Group group;`<br>`MPI_Comm *comm_out;` | Creates a new communicator |
| `int MPI_Comm_dup ( comm, comm_out )`<br>`MPI_Comm comm, *comm_out;` | Duplicates an existing communicator with all its cached information |
| `int MPI_Comm_free ( commp )`<br>`MPI_Comm *commp;` | Marks the communicator object for deallocation |
| `int MPI_Comm_group ( comm, group )`<br>`MPI_Comm comm;`<br>`MPI_Group *group;` | Accesses the group associated with given communicator |
| `int MPI_Comm_rank ( comm, rank )`<br>`MPI_Comm comm;`<br>`int *rank;` | Determines the rank of the calling process in the communicator |
| `int MPI_Comm_remote_group ( comm, group )`<br>`MPI_Comm comm;`<br>`MPI_Group *group;` | Accesses the remote group associated with the given inter-communicator |
| `int MPI_Comm_remote_size ( comm, size )`<br>`MPI_Comm comm;`<br>`int *size;` | Determines the size of the remote group associated with an inter-communictor |
| `int MPI_Comm_size ( comm, size )`<br>`MPI_Comm comm;`<br>`int *size;` | Determines the size of the group associated with a communictor |
| `int MPI_Comm_split ( comm, color, key, comm_out )`<br>`MPI_Comm comm;`<br>`int color, key;`<br>`MPI_Comm *comm_out;` | Creates new communicators based on colors and keys |
| `int MPI_Comm_test_inter ( comm, flag )`<br>`MPI_Comm comm;`<br>`int *flag;` | Tests to see if a comm is an inter-communicator |
| `int MPIR_dup_fn ( comm, keyval, extra_state, attr_in, attr_out, flag )`<br>`MPI_Comm *comm;`<br>`int *keyval;`<br>`void *extra_state;`<br>`void *attr_in;`<br>`void *attr_out;`<br>`int *flag;` | A function to simple-mindedly copy attributes |
| `int MPI_Dims_create(nnodes, ndims, dims)`<br>`int nnodes;`<br>`int ndims;`<br>`int *dims;` | Creates a division of processors in a cartesian grid |

| | |
|---|---|
| `int MPI_Errhandler_create( function, errhandler )`<br>`MPI_Handler_function *function;`<br>`MPI_Errhandler *errhandler;` | Creates an MPI-style errorhandler |
| `int MPI_Errhandler_free( errhandler )`<br>`MPI_Errhandler *errhandler;` | Frees an MPI-style errorhandler |
| `int MPI_Errhandler_get( comm, errhandler )`<br>`MPI_Comm comm;`<br>`MPI_Errhandler *errhandler;` | Gets the error handler for a communicator |
| `int MPI_Errhandler_set( comm, errhandler )`<br>`MPI_Comm comm;`<br>`MPI_Errhandler errhandler;` | Sets the error handler for a communicator |
| `int MPI_Error_class( errorcode, errorclass )`<br>`int errorcode, *errorclass;` | Converts an error code into an error class |
| `int MPI_Error_string( errorcode, string, resultlen )`<br>`int errorcode, *resultlen;`<br>`char *string;` | Return a string for a given error code |
| `int MPI_Finalize()` | Terminates MPI execution environment |
| `int MPI_Gatherv ( sendbuf, sendcnt, sendtype, recvbuf, recvcnts, displs, recvtype, root, comm )`<br>`void *sendbuf;`<br>`int sendcnt;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int *recvcnts;`<br>`int *displs;`<br>`MPI_Datatype recvtype;`<br>`int root;`<br>`MPI_Comm comm;` | Gathers into specified locations from all processes in a group |
| `int MPI_Gather ( sendbuf, sendcnt, sendtype, recvbuf, recvcount, recvtype, root, comm )`<br>`void *sendbuf;`<br>`int sendcnt;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int recvcount;`<br>`MPI_Datatype recvtype;`<br>`int root;`<br>`MPI_Comm comm;` | Gathers together values from a group of processes |
| `int MPI_Get_count( status, datatype, count )`<br>`MPI_Status *status;`<br>`MPI_Datatype datatype;`<br>`int *count;` | Gets the number of "top level" elements |

| | |
|---|---|
| `int MPI_Get_elements ( status, datatype, elements )`<br>`MPI_Status *status;`<br>`MPI_Datatype datatype;`<br>`int *elements;` | Returns the number of basic elements in a datatype |
| `int MPI_Get_processor_name( name, resultlen )`<br>`char *name;`<br>`int *resultlen;` | Gets the name of the processor |
| `int MPI_Graph_create ( comm_old, nnodes, index, edges, reorder, comm_graph )`<br>`MPI_Comm comm_old;`<br>`int nnodes;`<br>`int *index;`<br>`int *edges;`<br>`int reorder;`<br>`MPI_Comm *comm_graph;` | Makes a new communicator to which topology information has been attached |
| `int MPI_Graph_get ( comm, maxindex, maxedges, index, edges )`<br>`MPI_Comm comm;`<br>`int maxindex, maxedges;`<br>`int *index, *edges;` | Retrieves graph topology information associated with a communicator |
| `int MPI_Graph_map ( comm_old, nnodes, index, edges, newrank )`<br>`MPI_Comm comm_old;`<br>`int nnodes;`<br>`int *index;`<br>`int *edges;`<br>`int *newrank;` | Maps process to graph topology information |
| `int MPI_Graph_neighbors_count ( comm, rank, nneighbors )`<br>`MPI_Comm comm;`<br>`int rank;`<br>`int *nneighbors;` | Returns the number of neighbors of a node associated with a graph topology |
| `int MPI_Graph_neighbors ( comm, rank, maxneighbors, neighbors )`<br>`MPI_Comm comm;`<br>`int rank;`<br>`int maxneighbors;`<br>`int *neighbors;` | Returns the neighbors of a node associated with a graph topology |
| `int MPI_Graphdims_get ( comm, nnodes, nedges )`<br>`MPI_Comm comm;`<br>`int *nnodes;`<br>`int *nedges;` | Retrieves graph topology information associated with a communicator |
| `int MPI_Group_compare ( group1, group2, result )`<br>`MPI_Group group1;`<br>`MPI_Group group2;`<br>`int *result;` | Compares two groups |

| | |
|---|---|
| `int MPI_Group_difference ( group1, group2, group_out )`<br>`MPI_Group group1, group2, *group_out;` | Makes a group from the difference of two groups |
| `int MPI_Group_excl ( group, n, ranks, newgroup )`<br>`MPI_Group group, *newgroup;`<br>`int n, *ranks;` | Produces a group by reordering an existing group and taking only unlisted members |
| `int MPI_Group_free ( group )`<br>`MPI_Group *group;` | Frees a group |
| `int MPI_Group_incl ( group, n, ranks, group_out )`<br>`MPI_Group group, *group_out;`<br>`int n, *ranks;` | Produces a group by reordering an existing group and taking only listed members |
| `int MPI_Group_intersection ( group1, group2, group_out )`<br>`MPI_Group group1, group2, *group_out;` | Produces a group as the intersection of two existing groups |
| `int MPI_Group_range_excl ( group, n, ranges, newgroup )`<br>`MPI_Group group, *newgroup;`<br>`int n, ranges[][3];` | Produces a group by excluding ranges of processes from an existing group |
| `int MPI_Group_range_incl ( group, n, ranges, newgroup )`<br>`MPI_Group group, *newgroup;`<br>`int n, ranges[][3];` | Creates a new group from ranges of ranks in an existing group |
| `int MPI_Group_rank ( group, rank )`<br>`MPI_Group group;`<br>`int *rank;` | Returns the rank of this process in the given group |
| `int MPI_Group_size ( group, size )`<br>`MPI_Group group;`<br>`int *size;` | Returns the size of a group |
| `int MPI_Group_translate_ranks ( group_a, n, ranks_a, group_b, ranks_b )`<br>`MPI_Group group_a;`<br>`int n;`<br>`int *ranks_a;`<br>`MPI_Group group_b;`<br>`int *ranks_b;` | Translates the ranks of processes in one group to those in another group |
| `int MPI_Group_union ( group1, group2, group_out )`<br>`MPI_Group group1, group2, *group_out;` | Produces a group by combining two groups |
| `int MPI_Ibsend( buf, count, datatype, dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Starts a nonblocking buffered send |
| `int MPI_Initialized( flag )`<br>`int *flag;` | Indicates whether MPI_Init has been called. |

| | |
|---|---|
| `int MPI_Init(argc,argv)`<br>`int *argc;`<br>`char ***argv;` | Initialize the MPI execution environment |
| `int MPI_Intercomm_create ( local_comm,`<br>`  local_leader, peer_comm,`<br>`  remote_leader, tag, comm_out`<br>`)`<br>`MPI_Comm local_comm;`<br>`int local_leader;`<br>`MPI_Comm peer_comm;`<br>`int remote_leader;`<br>`int tag;`<br>`MPI_Comm *comm_out;` | Creates an intercommuncator from two intracommunicators |
| `int MPI_Intercomm_merge ( comm, high,`<br>`  comm_out )`<br>`MPI_Comm comm;`<br>`int high;`<br>`MPI_Comm *comm_out;` | Creates an intracommuncator from an intercommunicator |
| `int MPI_Iprobe( source, tag, comm, flag,`<br>`  status )`<br>`int source;`<br>`int tag;`<br>`int *flag;`<br>`MPI_Comm comm;`<br>`MPI_Status *status;` | Nonblocking test for a message |
| `int MPI_Irecv( buf, count, datatype,`<br>`  source, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int source;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Begins a nonblocking receive |
| `int MPI_Irsend( buf, count, datatype,`<br>`  dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Starts a nonblocking ready send |
| `int MPI_Isend( buf, count, datatype,`<br>`  dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Begins a nonblocking send |

| | |
|---|---|
| `int MPI_Issend( buf, count, datatype, dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Starts a nonblocking synchronous send |
| `int MPI_Keyval_create ( copy_fn, delete_fn, keyval, extra_state )`<br>`MPI_Copy_function *copy_fn;`<br>`MPI_Delete_function *delete_fn;`<br>`int *keyval;`<br>`void *extra_state;` | Generates a new attribute key |
| `int MPI_Keyval_free ( keyval )`<br>`int *keyval;` | Frees attribute key for communicator cache attribute |
| `int MPI_Op_create( function, commute, op )`<br>`MPI_User_function *function;`<br>`int commute;`<br>`MPI_Op *op;` | Creates a user-defined combination function handle |
| `int MPI_Op_free( op )`<br>`MPI_Op *op;` | Frees a user-defined combination function handle |
| `int MPI_Pack_size ( incount, datatype, comm, size )`<br>`int incount;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;`<br>`int *size;` | Returns the upper bound on the amount of space needed to pack a message |
| `int MPI_Pack ( inbuf, incount, type, outbuf, outcount, position, comm )`<br>`void *inbuf;`<br>`int incount;`<br>`MPI_Datatype type;`<br>`void *outbuf;`<br>`int outcount;`<br>`int *position;`<br>`MPI_Comm comm;` | Packs a datatype into contiguous memory |
| `int MPI_Pcontrol( level )`<br>`int level;` | Controls profiling |
| `int MPI_Probe( source, tag, comm, status )`<br>`int source;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Status *status;` | Blocking test for a message |

| | |
|---|---|
| `int MPI_Recv_init( buf, count, datatype,`<br>`  source, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Request *request;`<br>`MPI_Datatype datatype;`<br>`int source;`<br>`int tag;`<br>`MPI_Comm comm;` | Builds a handle for a receive |
| `int MPI_Recv( buf, count, datatype,`<br>`  source, tag, comm, status )`<br>`void *buf;`<br>`int count, source, tag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;`<br>`MPI_Status *status;` | Basic receive |
| `int MPI_Reduce_scatter ( sendbuf,`<br>`  recvbuf, recvcnts, datatype,`<br>`  op, comm )`<br>`void *sendbuf;`<br>`void *recvbuf;`<br>`int *recvcnts;`<br>`MPI_Datatype datatype;`<br>`MPI_Op op;`<br>`MPI_Comm comm;` | Combines values and scatters the results |
| `int MPI_Reduce ( sendbuf, recvbuf, count,`<br>`  datatype, op, root, comm )`<br>`void *sendbuf;`<br>`void *recvbuf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`MPI_Op op;`<br>`int root;`<br>`MPI_Comm comm;` | Reduces values on all processes to a single value |
| `int MPI_Request_free( request )`<br>`MPI_Request *request;` | Frees a communication request object |
| `int MPI_Rsend_init( buf, count, datatype,`<br>`  dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Builds a handle for a ready send |
| `int MPI_Rsend( buf, count, datatype,`<br>`  dest, tag, comm )`<br>`void *buf;`<br>`int count, dest, tag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;` | Basic ready send |

| | |
|---|---|
| `int MPI_Scan ( sendbuf, recvbuf, count, datatype, op, comm )`<br>`void *sendbuf;`<br>`void *recvbuf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`MPI_Op op;`<br>`MPI_Comm comm;` | Computes the scan (partial reductions) of data on a collection of processes |
| `int MPI_Scatterv ( sendbuf, sendcnts, displs, sendtype, recvbuf, recvcnt, recvtype, root, comm )`<br>`void *sendbuf;`<br>`int *sendcnts;`<br>`int *displs;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int recvcnt;`<br>`MPI_Datatype recvtype;`<br>`int root;`<br>`MPI_Comm comm;` | Scatters a buffer in parts to all tasks in a group |
| `int MPI_Scatter ( sendbuf, sendcnt, sendtype, recvbuf, recvcnt, recvtype, root, comm )`<br>`void *sendbuf;`<br>`int sendcnt;`<br>`MPI_Datatype sendtype;`<br>`void *recvbuf;`<br>`int recvcnt;`<br>`MPI_Datatype recvtype;`<br>`int root;`<br>`MPI_Comm comm;` | Sends data from one task to all other tasks in a group |
| `int MPI_Send_init( buf, count, datatype, dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Builds a handle for a standard send |
| `int MPI_Sendrecv_replace( buf, count, datatype, dest, sendtag, source, recvtag, comm, status )`<br>`void *buf;`<br>`int count, dest, sendtag, source, recvtag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;`<br>`MPI_Status *status;` | Sends and receives using a single buffer |

| | |
|---|---|
| `int MPI_Sendrecv( sendbuf, sendcount,`<br>`  sendtype, dest, sendtag,`<br>`  recvbuf, recvcount, recvtype,`<br>`  source, recvtag, comm, status`<br>`)`<br>`void *sendbuf;`<br>`int sendcount;`<br>`MPI_Datatype sendtype;`<br>`int dest, sendtag;`<br>`void *recvbuf;`<br>`int recvcount;`<br>`MPI_Datatype recvtype;`<br>`int source, recvtag;`<br>`MPI_Comm comm;`<br>`MPI_Status *status;` | Sends and receives a message |
| `int MPI_Send( buf, count, datatype, dest,`<br>`  tag, comm )`<br>`void *buf;`<br>`int count, dest, tag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;` | Performs a basic send |
| `int MPI_Ssend_init( buf, count, datatype,`<br>`  dest, tag, comm, request )`<br>`void *buf;`<br>`int count;`<br>`MPI_Datatype datatype;`<br>`int dest;`<br>`int tag;`<br>`MPI_Comm comm;`<br>`MPI_Request *request;` | Builds a handle for a synchronous send |
| `int MPI_Ssend( buf, count, datatype,`<br>`  dest, tag, comm )`<br>`void *buf;`<br>`int count, dest, tag;`<br>`MPI_Datatype datatype;`<br>`MPI_Comm comm;` | Basic synchronous send |
| `int MPI_Startall( count,`<br>`  array_of_requests )`<br>`int count;`<br>`MPI_Request array_of_requests[];` | Starts a collection of requests |
| `int MPI_Start( request )`<br>`MPI_Request *request;` | Initiates a communication with a persistent request handle |
| `int MPI_Test_cancelled( status, flag )`<br>`MPI_Status *status;`<br>`int *flag;` | Tests to see if a request was canceled |
| `int MPI_Testall( count,`<br>`  array_of_requests, flag,`<br>`  array_of_statuses )`<br>`int count;`<br>`MPI_Request array_of_requests[];`<br>`int *flag;`<br>`MPI_Status *array_of_statuses;` | Tests for the completion of all previously initiated communications |

| | |
|---|---|
| `int MPI_Testany( count,`<br>`  array_of_requests, index, flag,`<br>`  status )`<br>`int count;`<br>`MPI_Request array_of_requests[];`<br>`int *index, *flag;`<br>`MPI_Status *status;` | Tests for completion of any previdously initiated communication |
| `int MPI_Testsome( incount,`<br>`  array_of_requests, outcount,`<br>`  array_of_indices,`<br>`  array_of_statuses`<br>`)`<br>`int incount, *outcount,`<br>`array_of_indices[];`<br>`MPI_Request array_of_requests[];`<br>`MPI_Status array_of_statuses[];` | Tests for some given communications to complete |
| `int MPI_Test ( request, flag, status )`<br>`MPI_Request *request;`<br>`int *flag;`<br>`MPI_Status *status;` | Tests for the completion of a send or receive |
| `int MPI_Topo_test ( comm, top_type )`<br>`MPI_Comm comm;`<br>`int *top_type;` | Determines the type of topology (if any) associated with a communicator |
| `int MPI_Type_commit ( datatype )`<br>`MPI_Datatype *datatype;` | Commits the datatype |
| `int MPI_Type_contiguous( count, old_type,`<br>`  newtype )`<br>`int count;`<br>`MPI_Datatype old_type;`<br>`MPI_Datatype *newtype;` | Creates a contiguous datatype |
| `int MPI_Type_extent( datatype, extent )`<br>`MPI_Datatype datatype;`<br>`MPI_Aint *extent;` | Returns the extent of a datatype |
| `int MPI_Type_free ( datatype )`<br>`MPI_Datatype *datatype;` | Frees the datatype |
| `int MPI_Type_hindexed( count, blocklens,`<br>`  indices, old_type, newtype )`<br>`int count;`<br>`int blocklens[];`<br>`MPI_Aint indices[];`<br>`MPI_Datatype old_type;`<br>`MPI_Datatype *newtype;` | Creates an indexed datatype with offsets in bytes |
| `int MPI_Type_hvector( count, blocklen,`<br>`  stride, old_type, newtype )`<br>`int count;`<br>`int blocklen;`<br>`MPI_Aint stride;`<br>`MPI_Datatype old_type;`<br>`MPI_Datatype *newtype;` | Creates a vector (strided) datatype with offset in bytes |

| | |
|---|---|
| `int MPI_Type_indexed( count, blocklens, indices, old_type, newtype )`<br>`int count;`<br>`int blocklens[];`<br>`int indices[];`<br>`MPI_Datatype old_type;`<br>`MPI_Datatype *newtype;` | Creates an indexed datatype |
| `int MPI_Type_lb ( datatype, displacement )`<br>`MPI_Datatype datatype;`<br>`MPI_Aint *displacement;` | Returns the lower-bound of a datatype |
| `int MPI_Type_size ( datatype, size )`<br>`MPI_Datatype datatype;`<br>`MPI_Aint *size;` | Return the number of bytes occupied by entries in the datatype |
| `int MPI_Type_struct( count, blocklens, indices, old_types, newtype )`<br>`int count;`<br>`int blocklens[];`<br>`MPI_Aint indices[];`<br>`MPI_Datatype old_types[];`<br>`MPI_Datatype *newtype;` | Creates a struct datatype |
| `int MPI_Type_ub ( datatype, displacement )`<br>`MPI_Datatype datatype;`<br>`MPI_Aint *displacement;` | Returns the upper bound of a datatype |
| `int MPI_Type_vector( count, blocklen, stride, old_type, newtype )`<br>`int count;`<br>`int blocklen;`<br>`int stride;`<br>`MPI_Datatype old_type;`<br>`MPI_Datatype *newtype;` | Creates a vector (strided) datatype |
| `int MPI_Unpack ( inbuf, insize, position, outbuf, outcount, type, comm )`<br>`void *inbuf;`<br>`int insize;`<br>`int *position;`<br>`void *outbuf;`<br>`int outcount;`<br>`MPI_Datatype type;`<br>`MPI_Comm comm;` | Unpack a datatype into contiguous memory |
| `int MPI_Waitall(count, array_of_requests, array_of_statuses )`<br>`int count;`<br>`MPI_Request array_of_requests[];`<br>`MPI_Status array_of_statuses[];` | Waits for all given communications to complete |
| `int MPI_Waitany(count, array_of_requests, index, status )`<br>`int count;`<br>`MPI_Request array_of_requests[];`<br>`int *index;`<br>`MPI_Status *status;` | Waits for any specified send or receive to complete |

| | |
|---|---|
| `int MPI_Waitsome( incount,`<br>`  array_of_requests, outcount,`<br>`  array_of_indices,`<br>`  array_of_statuses`<br>`)`<br>`int incount, *outcount,`<br>`array_of_indices[];`<br>`MPI_Request array_of_requests[];`<br>`MPI_Status array_of_statuses[];` | Waits for some given communications to complete |
| `int MPI_Wait ( request, status )`<br>`MPI_Request *request;`<br>`MPI_Status *status;` | Waits for an MPI send or receive to complete |
| `double MPI_Wtick()` | Returns the resolution of MPI_Wtime |
| `double MPI_Wtime()` | Returns an elapsed time on the calling processor |