Summary of the Semantics of all Operation-Related MPI Procedures

Message Passing Interface Forum

February 24, 2021

Version 1.0: February 24, 2021 This supplementary document provides the list of MPI procedures that are associated with an MPI operation, or inquiry procedures providing information about an operation.

Summary of the Semantics of all Operation-Related MPI Procedures

This is a supplementary document for the MPI Standard Version MPI-4.0 that provides the list of MPI procedures that are associated with an MPI operation, or inquiry procedures providing information about an operation.

Table Legend:

- **Stages:** i=initialization, s=starting, c=completion, f=freeing. The procedure does at least part of the indicated stage(s).
- **Cpl:** ic=incomplete procedure, c=completing procedure, f=freeing procedure
- **Loc:** l=local procedure, nl=non-local procedure
- $*$: exceptions, e.g., ic+nl = incomplete+non-local, and c+l = completing+local (both are defined as blocking)
- **Blk:** b=blocking procedure, nb=nonblocking procedure. Note that from a user's view point, this column is only a hint. Relevant is, whether a routine is local or not and which resources are blocked until when. See both previous and last columns.
- ‡: exceptions, e.g., nonblocking procedures without prefix I, or that prefix I only marks immediate return.
- **Op:** part of operation type: b-op = blocking operation, nb-op = nonblocking operation, p-op = persistent operation, pp-op = persistent partitioned operation
- **Collective procedures:**
  - C = all processes of the group must call the procedure
  - sq = in the same sequence
  - S1 = blocking synchronization, i.e., no process shall return from this procedure until all processes on the associated process group called this procedure
  - W1 = the implementation is permitted to do S1 but not required to do S1
  - S2 = start-complete-synchronization, i.e., no process shall complete the associated operation until all processes on the associated process group have called the associated starting procedure
  - W2 = the implementation is permitted to do S2 but not required to do S2
- **Blocked resources:** They are blocked after the call until the end of the subsequent stage where this resource is not mentioned further in the table.

Table Remarks:

1) Must not return before the corresponding MPI receive operation is started.

2) Not related to an MPI operation. Prior to MPI-4.0, MPI_PROBE and MPI_IPROBE were also described as blocking and nonblocking. From MPI-4.0 onwards, only non-local and local are used to describe these procedures.

3) Usually, MPI_WAIT is non-local, but in this case it is local.

4) In case of a MPI_(I)BARRIER on an intra-communicator, the S1/S2 synchronization is required (instead of W1/W2).

5) Collective: all processes must complete, but with the free choice of using MPI_WAIT or MPI_TEST returning flag = TRUE.

6) It also may not return until MPI_INIT was called in the children.

7) Addresses are cached on the request handle.

8) One of the rare cases that an incomplete call is non-local and therefore blocking.

9) One shall not free or deallocate the buffer before the operation is freed, that is MPI_REQUEST_FREE returned.

10) For MPI_WAIT and MPI_TEST, see corresponding lines for a) MPI_BSEND, or b) MPI_IBCAST.

11) The prefix I marks only that this procedure returns immediately.

12) One of the exceptions that a completing and therefore blocking operation-related procedure is local.

13) MPI_(I)MPROBE initializes the operation through generating the message handle whereas MPI_(I)MRECV initializes the receive buffer (i.e., two MPI procedures together implement the initialization stage).

14) Nonblocking procedure without an I prefix.

15) Initialization stage ("i") only if flag = TRUE is returned else no operation is progressed.

16) Collective: all processes must start, but with the free choice of using MPI_START or MPI_STARTALL for a given persistent request handle (i.e., if one process starts a persistent request handle then all processes of the associated process group must start their corresponding request handle, and if any process starts then all processes must complete their handles).

17) In a correct MPI program, a call to MPI_(I)RSEND requires that the receiver has already started the corresponding receive. Under this assumption, the call to MPI_RSEND and the call to MPI_WAIT with an (active) ready send request handle are local.

18) Based on their semantics, when called using an intra-communicator, MPI_ALLGATHER, MPI_ALLTOALL, and their V and W variants, MPI_ALLREDUCE, MPI_REDUCE_SCATTER, and MPI_REDUCE_SCATTER_BLOCK must synchronize (i.e., S1/S2 instead of W1/W2) provided that all counts and the size of all datatypes are larger than zero.

19) MPI_COMM_FREE may return before any pending communication has finished and the communicator is deallocated. In contrast, MPI_COMM_DISCONNECT waits for pending communicaton to finish and deallocates the communicator before it returns.

20) The request handle is in the "active" state after MPI_START, i.e., MPI_REQUEST_FREE is now forbidden. But the starting stage is not yet finished, and the contents of the buffer are not yet "blocked." An additional MPI_PREADY and variants MPI_PREADY_RANGE MPI_PREADY_LIST are required to activate each partition of the send buffer to finish the starting stage.

21) As part of the completion stage, the user is allowed to read part of the output buffer after returning from MPI_PARRIVED with flag = TRUE before completing the whole operation with a MPI_WAIT/MPI_TEST procedure.

Procedure	Stages	Cpl	Loc	Blk	Op	Collective C	Collective sq	Collective S/W	Blocked resources and remarks
Chapter 3: Point-to-Point Communication									
MPI_SEND	i-s-c-f	c+f	nl	b	b-op	-			
MPI_SSEND	i-s-c-f	c+f	nl	b	b-op	-			1)
MPI_RSEND	i-s-c-f	c+f	l*	b	b-op	-			12)* 17)
MPI_BSEND	i-s-c-f	c+f	l*	b	b-op	-			12)*
MPI_RECV	i-s-c-f	c+f	nl	b	b-op	-			
MPI_ISEND, MPI_ISSEND	i-s----	ic	l	nb	nb-op	-			buffer
MPI_IRECV	i-s----	ic	l	nb	nb-op	-			buffer
corresponding MPI_WAIT	----c-f	c+f	nl		nb-op	-			
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	-			
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op	-			buffer cached on req
MPI_IBSEND, MPI_IRSEND	i-s----	ic	l	nb	nb-op	-			buffer
corresponding MPI_WAIT	----c-f	c+f	l*		nb-op	-			3)* 17)
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	-			
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op	-			buffer 7)
MPI_PROBE	-------	c	nl			-			2)
MPI_IPROBE	-------	c	l‡			-			2) 11)‡
MPI_MPROBE	i------	ic	nl*	b	b-op	-			message 8)*
MPI_IMPROBE	i------	ic	l	nb	b-op	-			message 15)
MPI_MRECV of a probed message	i-s-c-f	c+f	l	b	b-op	-			13)
MPI_IMRECV of a probed message	i-s----	ic	l	nb	nb-op	-			buffer 13)
corresponding MPI_WAIT	----c-f	c+f	l		nb-op	-			
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	-			
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op	-			buffer 7)
MPI_[S|B|R]SEND_INIT, MPI_RECV_INIT	i------	ic	l	nb‡	p-op	-			buffer address 9) 14)‡
corresponding MPI_START, MPI_STARTALL	--s----	ic	l	nb‡	p-op	-			buffer address+contents 7) 14)‡
corresponding MPI_WAIT (for MPI_(B|R)SEND_INIT req.)	----c--	c	l*		p-op	-			buffer address 3)* 7) 9) 17)
corresponding MPI_WAIT (for other request)	----c--	c	nl		p-op	-			buffer address 7) 9)
corresponding MPI_TEST returning flag=TRUE	----c--	c	l		p-op	-			buffer address 7) 9)
corresponding MPI_TEST returning flag=FALSE	-------		l		p-op	-			buffer address+contents 7)
corresponding MPI_REQUEST_FREE (for inactive request)	------f	f	l		p-op	-			
MPI_CANCEL of nonblocking/persistent pt-to-pt			l		p-op	-			
MPI_SENDRECV[_REPLACE]	i-s-c-f	c+f	nl	b	b-op	-			
MPI_ISENDRECV[_REPLACE]	i-s----	ic	l	nb	nb-op	-			buffer
corresponding MPI_WAIT	----c-f	c+f	nl		nb-op	-			
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	-			
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op	-			buffer cached on req
Chapter 4: Partitioned Point-to-Point Communication									
MPI_PSEND_INIT	i------	ic	l	nb‡	pp-op	-			buffer address 9) 14)‡
corresponding MPI_START, MPI_STARTALL	--s----	ic	l	nb‡	pp-op	-			buffer address 7) 9) 14)‡ 20)
corresponding MPI_PREADY and variants	--s----	ic	l	nb‡	pp-op	-			buffer address+contents 7) 9) 14)‡ 20)
corresponding MPI_WAIT	----c--	c	nl		pp-op	-			buffer address 7) 9)
corresponding MPI_TEST returning flag=TRUE	----c--	c	l		pp-op	-			buffer address 7) 9)
corresponding MPI_TEST returning flag=FALSE	-------		l		pp-op	-			buffer address+contents 7) 9)
corresponding MPI_REQUEST_FREE (for inactive request)	------f	f	l		pp-op	-			
MPI_PRECV_INIT	i------	ic	l	nb‡	pp-op	-			buffer address 9) 14)‡
corresponding MPI_START, MPI_STARTALL	--s----	ic	l	nb‡	pp-op	-			buffer address+contents 7) 9) 14)‡
MPI_PARRIVED returning flag=TRUE	----c--	ic	l		pp-op	-			buffer address+contents 7) 9) 21)
MPI_PARRIVED returning flag=FALSE	-------	ic	l		pp-op	-			buffer address+contents 7) 9)
corresponding MPI_WAIT	----c--	c	nl		pp-op	-			buffer address 7) 9) 21)
corresponding MPI_TEST returning flag=TRUE	----c--	c	l		pp-op	-			buffer address 7) 9) 21)
corresponding MPI_TEST returning flag=FALSE	-------		l		pp-op	-			buffer address+contents 7) 9)
corresponding MPI_REQUEST_FREE (for inactive request)	------f	f	l		pp-op	-			

Procedure	Stages	Cpl	Loc	Blk	Op	Collective C	Collective sq	Collective S/W	Blocked resources and remarks
Chapter 6: Collective Communication									
MPI_BCAST, MPI_BARRIER, MPI_GATHER, MPI_GATHERV, MPI_SCATTER, MPI_SCATTERV, MPI_ALLGATHER, MPI_ALLGATHERV, MPI_ALLTOALL, MPI_ALLTOALLV, MPI_ALLTOALLW, MPI_REDUCE, MPI_ALLREDUCE, MPI_REDUCE_SCATTER_BLOCK, MPI_REDUCE_SCATTER, MPI_SCAN, MPI_EXSCAN	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	4) 18)
MPI_IBCAST, MPI_IBARRIER, MPI_IGATHER, MPI_ISCATTER, MPI_IALLGATHER, MPI_IALLTOALL, MPI_IREDUCE, MPI_IALLREDUCE, MPI_IREDUCE_SCATTER_BLOCK, MPI_ISCAN, MPI_IEXSCAN	i-s----	ic	l	nb	nb-op	C	sq		buffer
MPI_IGATHERV, MPI_ISCATTERV, MPI_IALLGATHERV, MPI_IALLTOALLV, MPI_IALLTOALLW, MPI_IREDUCE_SCATTER	i-s----	ic	l	nb	nb-op	C	sq		buffer, array arguments
corresponding MPI_WAIT	----c-f	c+f	nl		nb-op	C		W2	4) 5) 18)
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	C		W2	4) 5) 18)
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op				buffer, array arguments 7)
MPI_BCAST_INIT, MPI_BARRIER_INIT, MPI_GATHER_INIT, MPI_SCATTER_INIT, MPI_ALLGATHER_INIT, MPI_ALLTOALL_INIT, MPI_REDUCE_INIT, MPI_ALLREDUCE_INIT, MPI_REDUCE_SCATTER_BLOCK_INIT, MPI_SCAN_INIT, MPI_EXSCAN_INIT	i------	ic	nl*	b	p-op	C	sq	W1	buffer address 8)* 9)
MPI_GATHERV_INIT, MPI_SCATTERV_INIT, MPI_ALLGATHERV_INIT, MPI_ALLTOALLV_INIT, MPI_ALLTOALLW_INIT, MPI_REDUCE_SCATTER_INIT	i------	ic	nl*	b	p-op	C	sq	W1	buffer address, array arguments 8)* 9)
corresponding MPI_START, MPI_STARTALL	--s----	ic	l	nb‡	p-op	C			buf.addr.+contents 7) 14)‡ 16)
corresponding MPI_WAIT	----c--	c	nl		p-op	C		W2	buffer address and array arguments cached on the request handle 4) 5) 7) 9) 18)
corresponding MPI_TEST returning flag=TRUE	----c--	c	l		p-op	C		W2	buf-addr & arr-args 4) 5) 7) 9) 18)
corresponding MPI_TEST returning flag=FALSE	-------		l		p-op				buf addr+contents & arr-args 7)
corresponding MPI_REQUEST_FREE	------f	f	l		p-op				
Chapter 7: Groups, Contexts, Communicators, and Caching									
MPI_COMM_CREATE, MPI_COMM_DUP, MPI_COMM_DUP_WITH_INFO, MPI_COMM_SPLIT, MPI_COMM_SPLIT_TYPE	i-s-c-f	c	nl	b	b-op	C	sq	W1	coll. over comm arg.
MPI_COMM_CREATE_GROUP	i-s-c-f	c	nl	b	b-op	C	sq	W1	coll. over group arg.
MPI_INTERCOMM_CREATE,MPI_INTERCOMM_MERGE	i-s-c-f	c	nl	b	b-op	C	sq	W1	coll. over union of local & remote group
MPI_COMM_IDUP	i-s----	ic	l	nb	nb-op	C	sq		communicator handle
corresponding MPI_WAIT	----c-f	c+f	nl		nb-op	C		W2	5)
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	C		W2	5)
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op				
MPI_COMM_FREE	i-s----	ic	nl	b	nb-op	C	sq	W1	19)
MPI_COMM_DISCONNECT	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	19)

Procedure	Stages	Cpl	Loc	Blk	Op	Collective C	Collective sq	Collective S/W	Blocked resources and remarks
Chapter 8: Process Topologies									
MPI_CART_CREATE, MPI_CART_SUB, MPI_GRAPH_CREATE, MPI_DIST_GRAPH_CREATE_ADJACENT, MPI_DIST_GRAPH_CREATE	i-s-c-f	c	nl	b	b-op	C	sq	W1	coll. over comm arg.
MPI_NEIGHBOR_ALLGATHER, MPI_NEIGHBOR_ALLGATHERV, MPI_NEIGHBOR_ALLTOALL, MPI_NEIGHBOR_ALLTOALLV, MPI_NEIGHBOR_ALLTOALLW	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	4) 18)
MPI_INEIGHBOR_ALLGATHER, MPI_INEIGHBOR_ALLTOALL	i-s----	ic	l	nb	nb-op	C	sq		buffer
MPI_INEIGHBOR_ALLGATHERV, MPI_INEIGHBOR_ALLTOALLV, MPI_INEIGHBOR_ALLTOALLW	i-s----	ic	l	nb	nb-op	C	sq		buffer, array arguments
corresponding MPI_WAIT	----c-f	c+f	nl		nb-op	C		W2	4) 5) 18)
corresponding MPI_TEST returning flag=TRUE	----c-f	c+f	l		nb-op	C		W2	4) 5) 18)
corresponding MPI_TEST returning flag=FALSE	-------		l		nb-op				buffer, array arguments 7)
MPI_NEIGHBOR_ALLGATHER_INIT, MPI_NEIGHBOR_ALLTOALL_INIT	i------	ic	nl*	b	p-op	C	sq	W1	buffer address 8)* 9)
MPI_NEIGHBOR_ALLGATHERV_INIT, MPI_NEIGHBOR_ALLTOALLV_INIT, MPI_NEIGHBOR_ALLTOALLW_INIT	i------	ic	nl*	b	p-op	C	sq	W1	buffer address, array arguments 8)* 9)
corresponding MPI_START, MPI_STARTALL	--s----	ic	l	nb‡	p-op	C			buf.addr.+contents 7) 14)‡ 16)
corresponding MPI_WAIT	----c--	c	nl		p-op	C		W2	buffer address and array arguments cached on the request handle 4) 5) 7) 9) 18)
corresponding MPI_TEST returning flag=TRUE	----c--	c	l		p-op	C		W2	buf-addr & arr-args 4) 5) 7) 9) 18)
corresponding MPI_TEST returning flag=FALSE	-------		l		p-op				buf addr+contents & arr-args 7)
corresponding MPI_REQUEST_FREE	------f	f	l		p-op				
Chapter 11: Process Initialization, Creation, and Management									
MPI_INIT, MPI_INIT_THREAD	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	collective over MPI_COMM_WORLD
MPI_FINALIZE	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	collective over all connected processes
MPI_SESSION_INIT	-------		l						2)
MPI_SESSION_FINALIZE	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	collective over connected processes scoped by the session
MPI_COMM_SPAWN, ..._MULTIPLE	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	collective over comm 6)
MPI_COMM_ACCEPT, MPI_COMM_CONNECT	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	collective over comm
Chapter 14: I/O									
MPI_FILE_READ/WRITE[_AT|_SHARED], MPI_FILE_DELETE/SEEK/GET_VIEW	i-s-c-f	c+f	l*	b	b-op	-			12)*
MPI_FILE_READ/WRITE_AT_ALL, MPI_FILE_READ/WRITE_ALL|ORDERED, MPI_FILE_OPEN/CLOSE/SEEK_SHARED, MPI_FILE_PREALLOCATE/SYNC, MPI_FILE_SET_VIEW/SIZE/INFO/ATOMICITY	i-s-c-f	c+f	nl	b	b-op	C	sq	W1	
MPI_FILE_IREAD/IWRITE[_AT|_SHARED]	i-s----	ic	l	nb	nb-op	-			buffer 10a)
MPI_FILE_IREAD/IWRITE[_AT]_ALL	i-s----	ic	l	nb	nb-op	C	sq		buffer 10b)
MPI_FILE_READ/WRITE[_AT]_ALL_BEGIN MPI_FILE_READ/WRITE_ORDERED_BEGIN	i-s----	ic	nl*	b	b-op	C	sq	W1	buffer 8)*
MPI_FILE_READ/WRITE[_AT]_ALL_END MPI_FILE_READ/WRITE_ORDERED_END	----c-f	c+f	nl	b	b-op	C	sq	W1	

Index