



MPI_Sendrecv(3) man page (version 3.1.4)

[Home](#)[Support](#)[FAQ](#)

About

[Presentations](#)[Open MPI Team](#)[FAQ](#)[Videos](#)[Performance](#)

Open MPI Software

[Download](#)[Documentation](#)[Current](#)[v4.0 \(current stable\)](#)[Still supported](#)[v3.1 \(prior stable\)](#)[v3.0 \(prior stable\)](#)[v2.1 \(prior stable\)](#)[Older versions](#)[v2.0 \(retired\)](#)[v1.10 \(retired\)](#)[v1.8 \(ancient\)](#)[v1.7 \(ancient\)](#)[v1.6 \(ancient\)](#)[v1.5 \(ancient\)](#)[v1.4 \(ancient\)](#)[v1.3 \(ancient\)](#)[v1.2 \(ancient\)](#)[v1.1 \(ancient\)](#)[Source Code Access](#)[Bug Tracking](#)[Regression Testing](#)[Version Information](#)

Sub-Projects

[Hardware Locality](#)[Network Locality](#)[MPI Testing Tool](#)[Open MPI User Docs](#)[Open Tool for Parameter Optimization](#)

Community

[Mailing Lists](#)[Getting Help/Support](#)[Contribute](#)[Contact](#)[License](#)[« Return to documentation listing](#)[Table of Contents](#)

Name

MPI_Sendrecv - Sends and receives a message.

Syntax

C Syntax

```
#include <mpi.h>
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 int dest, int sendtag, void *recvbuf, int recvcount,
                 MPI_Datatype recvtype, int source, int recvtag,
                 MPI_Comm comm, MPI_Status *status)
```

Fortran Syntax

```
USE MPI
! or the older form: INCLUDE 'mpif.h'
MPI_SENDRCV(SENDBUF, SENDCOUNT, SENDTYPE, DEST, SENDTAG,
            RECVBUF, RECVCOUNT, RECVTYPE, SOURCE, RECVTAG, COMM,
            STATUS, IERROR)

<type>      SENDBUF(*), RECVBUF(*)
INTEGER     SENDCOUNT, SENDTYPE, DEST, SENDTAG
INTEGER     RECVCOUNT, RECVTYPE, SOURCE, RECVTAG, COMM
INTEGER     STATUS(MPI_STATUS_SIZE), IERROR
```

Fortran 2008 Syntax

```
USE mpi_f08
MPI_Sendrecv(sendbuf, sendcount, sendtype, dest, sendtag, recvbuf,
             recvcount, recvtype, source, recvtag, comm, status, ierror)
TYPE(*), DIMENSION(..), INTENT(IN) :: sendbuf
TYPE(*), DIMENSION(..) :: recvbuf
INTEGER, INTENT(IN) :: sendcount, dest, sendtag, recvcount, source,
recvtag
TYPE(MPI_Datatype), INTENT(IN) :: sendtype, recvtype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
```

Input Parameters

sendbuf

Initial address of send buffer (choice).

sendcount

Number of elements to send (integer).

sendtype

Type of elements in send buffer (handle).
 dest Rank of destination (integer).
 sendtag Send tag (integer).
 recvcount Maximum number of elements to receive (integer).
 recvtype Type of elements in receive buffer (handle).
 source Rank of source (integer).
 recvtag Receive tag (integer).
 comm Communicator (handle).

Output Parameters

recvbuf Initial address of receive buffer (choice).
 status Status object (status). This refers to the receive operation.
 IERROR Fortran only: Error status (integer).

Description

The send-receive operations combine in one call the sending of a message to one destination and the receiving of another message, from another process. The two (source and destination) are possibly the same. A send-receive operation is useful for executing a shift operation across a chain of processes. If blocking sends and receives are used for such a shift, then one needs to order the sends and receives correctly (for example, even processes send, then receive; odd processes receive first, then send) in order to prevent cyclic dependencies that may lead to deadlock. When a send-receive operation is used, the communication subsystem takes care of these issues. The send-receive operation can be used in conjunction with the functions described in Chapter 6 of the MPI-1 Standard, "Process Topologies," in order to perform shifts on various logical topologies. Also, a send-receive operation is useful for implementing remote procedure calls.

A message sent by a send-receive operation can be received by a regular receive operation or probed by a probe operation; a send-receive operation can receive a message sent by a regular send operation.

MPI_Sendrecv executes a blocking send and receive operation. Both send and receive use the same communicator, but possibly different tags. The send buffer and receive buffers must be disjoint, and may have different lengths and datatypes.

If your application does not need to examine the *status* field, you can save resources by using the predefined constant MPI_STATUS_IGNORE as a special value for the *status* argument.

Errors

Almost all MPI routines return an error value; C routines as the value of the function and Fortran routines in the last argument. C++ functions do not return errors. If the default error handler is set to MPI::ERRORS_THROW_EXCEPTIONS, then on error the C++ exception mechanism will be used to throw an MPI::Exception object.

Before the error value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job, except for I/O function errors. The error handler may be changed with [MPI_Comm_set_errhandler](#); the predefined error handler MPI_ERRORS_RETURN may be used to cause error values to be returned. Note that MPI does not guarantee that an MPI program can continue past an error.

See Also

Table of Contents

- [Name](#)
- [Syntax](#)
- [C Syntax](#)
- [Fortran Syntax](#)
- [Fortran 2008 Syntax](#)
- [Input Parameters](#)
- [Output Parameters](#)
- [Description](#)
- [Errors](#)
- [See Also](#)

[« Return to documentation listing](#)



**Open MPI is an
Associated Project
of the [Software in the
Public Interest](#) non-
profit organization**

**[Contact the Open MPI
webmaster](#)[Contact the Open MPI
webmaster](#)**

**Page last modified: 20-May-2019
©2004-2019 The Open MPI Project**