## NAME

Constants − Meaning of MPI *s defined constants*

## DATA TYPES

Note that the Fortran types should only be used in Fortran programs, and the C types should only be used in C programs. For example, it is in error to use MPI_INT *for a Fortran INTEGER.* Datatypes are of type MPI_Datatype *in C and of tyep* INTEGER *in Fortran.*

## C DATATYPES

**MPI_CHAR**
- char

**MPI_BYTE**
- See standard; like unsigned char

**MPI_SHORT**
- short

**MPI_INT**
- int

**MPI_LONG**
- long

**MPI_FLOAT**
- float

**MPI_DOUBLE**
- double

**MPI_UNSIGNED_CHAR**
- unsigned char

**MPI_UNSIGNED_SHORT**
- unsigned short

**MPI_UNSIGNED**
- unsigned int

**MPI_UNSIGNED_LONG**
- unsigned long

**MPI_LONG_DOUBLE**
- long double (some systems may not implement)

**MPI_LONG_LONG_INT**
- long long (some systems may not implement)

The following are datatypes for the MPI functions MPI_MAXLOC *and* MPI_MINLOC .

**MPI_FLOAT_INT**
- struct { float, int }

**MPI_LONG_INT**
- struct { long, int }

**MPI_DOUBLE_INT**
- struct { double, int }

**MPI_SHORT_INT**
- struct { short, int }

**MPI_2INT**
- struct { int, int }

**MPI_LONG_DOUBLE_INT**
- struct { long double, int }; this is an OPTIONAL type, and may be set to NULL

Note that there is no *MPI_LONG_LONG_INT* type corresponding to a *struct { long long, int }* .

Special datatypes for C and Fortran

**MPI_PACKED**
- For *MPI_Pack* and *MPI_Unpack*



**MPI_UB**
- For *MPI_Type_struct* ; an upper-bound indicator

**MPI_LB**
- For *MPI_Type_struct* ; a lower-bound indicator

## FORTRAN DATATYPES

**MPI_REAL**
- *REAL*

**MPI_INTEGER**
- *INTEGER*

**MPI_LOGICAL**
- *LOGICAL*

**MPI_DOUBLE_PRECISION**
- *DOUBLE PRECISION*

**MPI_COMPLEX**
- *COMPLEX*

**MPI_DOUBLE_COMPLEX**
- *complex*16* (or *complex*32* ) where supported.

The following datatypes are optional

**MPI_INTEGER1**
- *integer*1* if supported

**MPI_INTEGER2**
- *integer*2* if supported

**MPI_INTEGER4**
- *integer*4* if supported

**MPI_REAL4**
- *real*4* if supported

**MPI_REAL8**
- *real*8* if supported

The following are datatypes for the MPI functions *MPI_MAXLOC* and *MPI_MINLOC* . In Fortran, these datatype always consist of two elements of the same Fortran type.

**MPI_2INTEGER**
- *INTEGER,INTEGER*

**MPI_2REAL**
- *REAL, REAL*

**MPI_2DOUBLE_PRECISION**
- *DOUBLE PRECISION, DOUBLE PRECISION*

**MPI_2COMPLEX**
- *COMPLEX, COMPLEX*

**MPI_2DOUBLE_COMPLEX**
- *complex*16, complex*16*

## COMMUNICATORS

Communicators are of type *MPI_Comm* in C and *INTEGER* in Fortran



**MPI_COMM_WORLD**
- Contains all of the processes

**MPI_COMM_SELF**
- Contains only the calling process

## GROUPS

Groups are of type *MPI_Group* in C and *INTEGER* in Fortran

**MPI_GROUP_EMPTY**
- A group containing no members.

## RESULTS OF THE COMPARE OPERATIONS

**MPI_IDENT**
- Identical

**MPI_CONGRUENT**
- (only for *MPI_COMM_COMPARE* ) The groups are identical

**MPI_SIMILAR**
- Same members, but in a different order

**MPI_UNEQUAL**
- Different

## COLLECTIVE OPERATIONS

The collective combination operations ( *MPI_REDUCE* , *MPI_ALLREDUCE* , *MPI_REDUCE_SCATTER* , and *MPI_SCAN* ) take a combination operation. This operation is of type *MPI_Op* in C and of type *INTEGER* in Fortran. The predefined operations are

**MPI_MAX**
- return the maximum

**MPI_MIN**
- return the minumum

**MPI_SUM**
- return the sum

**MPI_PROD**
- return the product

**MPI_LAND**
- return the logical and

**MPI_BAND**
- return the bitwise and

**MPI_LOR**
- return the logical or

**MPI_BOR**
- return the bitwise of

**MPI_LXOR**
- return the logical exclusive or

**MPI_BXOR**
- return the bitwise exclusive or

**MPI_MINLOC**
- return the minimum and the location (actually, the value of the second element of the structure where the minimum of the first is found)

**MPI_MAXLOC**
- return the maximum and the location



## NOTES ON COLLECTIVE OPERATIONS

The reduction functions ( *MPI_Op* ) do not return an error value. As a result, if the functions detect an error, all they can do is either call *MPI_Abort* or silently skip the problem. Thus, if you change the error handler from *MPI_ERRORS_ARE_FATAL* to something else, for example, *MPI_ERRORS_RETURN* , then no error may be indicated.

The reason for this is the performance problems in ensuring that all collective routines return the same error value.

Note that not all datatypes are valid for these functions. For example, *MPI_COMPLEX* is not valid for *MPI_MAX* and *MPI_MIN* . In addition, the MPI 1.1 standard did not include the C types *MPI_CHAR* and *MPI_UNSIGNED_CHAR* among the lists of arithmetic types for operations like *MPI_SUM* . However, since the C type *char* is an integer type (like *short* ), it should have been included. The MPI Forum will probably include *char* and *unsigned char* as a clarification to MPI 1.1; until then, users are advised that MPI implementations may not accept *MPI_CHAR* and *MPI_UNSIGNED_CHAR* as valid datatypes for *MPI_SUM* , *MPI_PROD* , etc. MPICH does allow these datatypes.

## PERMANENT KEY VALUES

These are the same in C and Fortran

**MPI_TAG_UB**
- Largest tag value

**MPI_HOST**
- Rank of process that is host, if any

**MPI_IO**
- Rank of process that can do I/O

**MPI_WTIME_IS_GLOBAL**
- Has value 1 if *MPI_WTIME* is globally synchronized.

## NULL OBJECTS

**MPI_COMM_NULL**
- Null communicator

**MPI_OP_NULL**
- Null operation

**MPI_GROUP_NULL**
- Null group

**MPI_DATATYPE_NULL**
- Null datatype

**MPI_REQUEST_NULL**
- Null request

**MPI_ERRHANDLER_NULL**
- Null error handler

## PREDEFINED CONSTANTS

**MPI_MAX_PROCESSOR_NAME**
- Maximum length of name returned by *MPI_GET_PROCESSOR_NAME*

**MPI_MAX_ERROR_STRING**
- Maximum length of string return by *MPI_ERROR_STRING*

**MPI_UNDEFINED**
- Used by many routines to indicated undefined or unknown integer value

**MPI_UNDEFINED_RANK**
- Unknown rank



**MPI_KEYVAL_INVALID**
- Special keyval that may be used to detect uninitialized keyvals.

**MPI_BSEND_OVERHEAD**
- Add this to the size of a *MPI_BSEND* buffer for each outstanding message

**MPI_PROC_NULL**
- This rank may be used to send or receive from no-one.

**MPI_ANY_SOURCE**
- In a receive, accept a message from anyone.

**MPI_ANY_TAG**
- In a receive, accept a message with any tag value.

**MPI_BOTTOM**
- May be used to indicate the bottom of the address space

## TOPOLOGY TYPES

**MPI_GRAPH**
- General graph

**MPI_CART**
- Cartesian grid

## MPI STATUS

The *MPI_Status* datatype is a structure. The three elements for use by programmers are

**MPI_SOURCE**
- Who sent the message

**MPI_TAG**
- What tag the message was sent with

**MPI_ERROR**
- Any error return

## SPECIAL MPI TYPES AND FUNCTIONS

**MPI_Aint**
- C type that holds any valid address.

**MPI_Handler_function**
- C function for handling errors (see *MPI_Errhandler_create* ) .

**MPI_User_function**
- C function to combine values (see collective operations and *MPI_Op_create* )

**MPI_Copy_function**
- Function to copy attributes (see *MPI_Keyval_create* )

**MPI_NULL_COPY_FN**
- Predefined copy function

**MPI_Delete_function**
- Function to delete attributes (see *MPI_Keyval_create* )

**MPI_NULL_DELETE_FN**
- Predefined delete function

**MPI_DUP_FN**
- Predefined duplication function

**MPI_ERRORS_ARE_FATAL**
- Error handler that forces exit on error

**MPI_ERRORS_RETURN**
- Error handler that returns error codes (as value of MPI routine in C and through last argument in Fortran)

## MPI ERROR CLASSES



**MPI_SUCCESS**
- Successful return code

**MPI_ERR_BUFFER**
- Invalid buffer pointer

**MPI_ERR_COUNT**
- Invalid count argument

**MPI_ERR_TYPE**
- Invalid datatype argument

**MPI_ERR_TAG**
- Invalid tag argument

**MPI_ERR_COMM**
- Invalid communicator

**MPI_ERR_RANK**
- Invalid rank

**MPI_ERR_ROOT**
- Invalid root

**MPI_ERR_GROUP**
- Null group passed to function

**MPI_ERR_OP**
- Invalid operation

**MPI_ERR_TOPOLOGY**
- Invalid topology

**MPI_ERR_DIMS**
- Illegal dimension argument

**MPI_ERR_ARG**
- Invalid argument

**MPI_ERR_UNKNOWN**
- Unknown error

**MPI_ERR_TRUNCATE**
- message truncated on receive

**MPI_ERR_OTHER**
- Other error; use Error_string

**MPI_ERR_INTERN**
- internal error code

**MPI_ERR_IN_STATUS**
- Look in status for error value

**MPI_ERR_PENDING**
- Pending request

**MPI_ERR_REQUEST**
- illegal mpi_request handle

**MPI_ERR_LASTCODE**
- Last error code -- always at end

## LOCATION

/home/MPI/mansrc/mpiconsts.txt

