D R A F T

Document for a Standard Message-Passing Interface

Message Passing Interface Forum

February 5, 2015
This work was supported in part by NSF and ARPA under NSF contract CDA-9115428 and Esprit under project HPC Standards (21111).

This is the result of a LaTeX run of a draft of a single chapter of the MPIF Final Report document.

Chapter 19

Language Bindings Summary

In this section we summarize the specific bindings for C and Fortran. First we present the constants, type definitions, info values and keys. Then we present the routine prototypes separately for each binding. Listings are alphabetical within chapter.

19.1 Defined Values and Handles

19.1.1 Defined Constants

The C and Fortran names are listed below. Constants with the type `const int` may also be implemented as literal integer constants substituted by the preprocessor.

Error classes

C type: `const int` (or unnamed `enum`) Fortran type: `INTEGER`
MPI_SUCCESS
MPI_ERR_BUFFER
MPI_ERR_COUNT
MPI_ERR_TYPE
MPI_ERR_TAG
MPI_ERR_COMM
MPI_ERR_RANK
MPI_ERR_REQUEST
MPI_ERR_ROOT
MPI_ERR_GROUP
MPI_ERR_OP
MPI_ERR_TOPOLOGY
MPI_ERR_DIMS
MPI_ERR_ARG
MPI_ERR_UNKNOWN
MPI_ERR_TRUNCATE
MPI_ERR_OTHER
MPI_ERR_INTERN
MPI_ERR_PENDING

(Continued on next page)

Error classes (continued)
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_T_ERR_CANNOT_INIT
MPI_T_ERR_NOT_INITIALIZED
MPI_T_ERR_MEMORY
MPI_T_ERR_INVALID
MPI_T_ERR_INVALID_INDEX
MPI_T_ERR_INVALID_ITEM
MPI_T_ERR_INVALID_SESSION
MPI_T_ERR_INVALID_HANDLE
MPI_T_ERR_INVALID_NAME
MPI_T_ERR_OUT_OF_HANDLES
MPI_T_ERR_OUT_OF_SESSIONS
MPI_T_ERR_CVAR_SET_NOT_NOW
MPI_T_ERR_CVAR_SET_NEVER
MPI_T_ERR_PVAR_NO_WRITE
MPI_T_ERR_PVAR_NO_STARTSTOP
MPI_T_ERR_PVAR_NO_ATOMIC
MPI_ERR_LASTCODE

#400

#377

Buffer Address Constants
C type: `void * const`
Fortran type: (predefined memory location)[1]
MPI_BOTTOM
MPI_IN_PLACE

[1] Note that in Fortran these constants are not usable for initialization expressions or assignment. See Section 2.5.4.

Assorted Constants
C type: `const int` (or unnamed `enum`)
Fortran type: `INTEGER`
MPI_PROC_NULL
MPI_ANY_SOURCE
MPI_ANY_TAG
MPI_UNDEFINED
MPI_BSEND_OVERHEAD
MPI_KEYVAL_INVALID
MPI_LOCK_EXCLUSIVE
MPI_LOCK_SHARED
MPI_ROOT

No Process Message Handle
C type: `MPI_Message`
Fortran type: `INTEGER` or `TYPE(MPI_Message)`
MPI_MESSAGE_NO_PROC

C Constants Specifying Ignored Input (no Fortran)

C type: `MPI_Fint*`	equivalent to Fortran
MPI_F_STATUSES_IGNORE	MPI_STATUSES_IGNORE in `mpi` / `mpif.h`
MPI_F_STATUS_IGNORE	MPI_STATUS_IGNORE in `mpi` / `mpif.h`
C type: `MPI_F08_status*`	**equivalent to Fortran**
MPI_F08_STATUSES_IGNORE	MPI_STATUSES_IGNORE in `mpi_f08`
MPI_F08_STATUS_IGNORE	MPI_STATUS_IGNORE in `mpi_f08`

C preprocessor Constants and Fortran Parameters

C type: C-preprocessor macro that expands to an `int` value Fortran type: `INTEGER`
MPI_SUBVERSION
MPI_VERSION

Null handles used in the MPI tool information interface

MPI_T_ENUM_NULL
    `MPI_T_enum`
MPI_T_CVAR_HANDLE_NULL
    `MPI_T_cvar_handle`
MPI_T_PVAR_HANDLE_NULL
    `MPI_T_pvar_handle`
MPI_T_PVAR_SESSION_NULL
    `MPI_T_pvar_session`

Verbosity Levels in the MPI tool information interface

C type: `const int` (or unnamed `enum`) No Fortran
MPI_T_VERBOSITY_USER_BASIC
MPI_T_VERBOSITY_USER_DETAIL
MPI_T_VERBOSITY_USER_ALL
MPI_T_VERBOSITY_TUNER_BASIC
MPI_T_VERBOSITY_TUNER_DETAIL
MPI_T_VERBOSITY_TUNER_ALL
MPI_T_VERBOSITY_MPIDEV_BASIC
MPI_T_VERBOSITY_MPIDEV_DETAIL
MPI_T_VERBOSITY_MPIDEV_ALL

#354

Constants to identify associations of variables in the MPI tool information interface
C type: `const int` (or unnamed `enum`) ← No Fortran
MPI_T_BIND_NO_OBJECT
MPI_T_BIND_MPI_COMM
MPI_T_BIND_MPI_DATATYPE
MPI_T_BIND_MPI_ERRHANDLER
MPI_T_BIND_MPI_FILE
MPI_T_BIND_MPI_GROUP
MPI_T_BIND_MPI_OP
MPI_T_BIND_MPI_REQUEST
MPI_T_BIND_MPI_WIN
MPI_T_BIND_MPI_MESSAGE
MPI_T_BIND_MPI_INFO

#354

Constants describing the scope of a control variable in the MPI tool information interface
C type: `const int` (or unnamed `enum`) ← No Fortran
MPI_T_SCOPE_CONSTANT
MPI_T_SCOPE_READONLY
MPI_T_SCOPE_LOCAL
MPI_T_SCOPE_GROUP
MPI_T_SCOPE_GROUP_EQ
MPI_T_SCOPE_ALL
MPI_T_SCOPE_ALL_EQ

#354

Additional constants used by the MPI tool information interface
C type: `MPI_T_pvar_handle`
MPI_T_PVAR_ALL_HANDLES

Performance variables classes used by the MPI tool information interface
C type: `const int` (or unnamed `enum`) ← No Fortran
MPI_T_PVAR_CLASS_STATE
MPI_T_PVAR_CLASS_LEVEL
MPI_T_PVAR_CLASS_SIZE
MPI_T_PVAR_CLASS_PERCENTAGE
MPI_T_PVAR_CLASS_HIGHWATERMARK
MPI_T_PVAR_CLASS_LOWWATERMARK
MPI_T_PVAR_CLASS_COUNTER
MPI_T_PVAR_CLASS_AGGREGATE
MPI_T_PVAR_CLASS_TIMER
MPI_T_PVAR_CLASS_GENERIC

#354

19.1.2 Types

The following are defined C type definitions, included in the file `mpi.h`.

```
/* C opaque types */
MPI_Aint
MPI_Count
MPI_Fint
MPI_Offset
MPI_Status
MPI_F08_status

/* C handles to assorted structures */
MPI_Comm
MPI_Datatype
MPI_Errhandler
MPI_File
MPI_Group
MPI_Info
MPI_Message
MPI_Op
MPI_Request
MPI_Win

/* Types for the MPI_T interface */
MPI_T_enum
MPI_T_cvar_handle
MPI_T_pvar_handle
MPI_T_pvar_session
```

#345

The following are defined Fortran type definitions, included in the mpi_f08 and mpi modules.

```
!  Fortran opaque types in the mpi_f08 and mpi modules
TYPE(MPI_Status)

!  Fortran handles in the mpi_f08 and mpi modules
TYPE(MPI_Comm)
TYPE(MPI_Datatype)
TYPE(MPI_Errhandler)
TYPE(MPI_File)
TYPE(MPI_Group)
TYPE(MPI_Info)
TYPE(MPI_Message)
TYPE(MPI_Op)
TYPE(MPI_Request)
TYPE(MPI_Win)
```

19.1.3 Prototype Definitions

C Bindings

The following are defined C typedefs for user-defined functions, also included in the file `mpi.h`.

```
/* prototypes for user-defined functions */
typedef void MPI_User_function(void *invec, void *inoutvec, int *len,
              MPI_Datatype *datatype);

typedef int MPI_Comm_copy_attr_function(MPI_Comm oldcomm,
              int comm_keyval, void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag);
typedef int MPI_Comm_delete_attr_function(MPI_Comm comm,
              int comm_keyval, void *attribute_val, void *extra_state);

typedef int MPI_Win_copy_attr_function(MPI_Win oldwin, int win_keyval,
              void *extra_state, void *attribute_val_in,
              void *attribute_val_out, int *flag);
typedef int MPI_Win_delete_attr_function(MPI_Win win, int win_keyval,
              void *attribute_val, void *extra_state);

typedef int MPI_Type_copy_attr_function(MPI_Datatype oldtype,
              int type_keyval, void *extra_state,
              void *attribute_val_in, void *attribute_val_out, int *flag);
typedef int MPI_Type_delete_attr_function(MPI_Datatype datatype,
              int type_keyval, void *attribute_val, void *extra_state);

typedef void MPI_Comm_errhandler_function(MPI_Comm *, int *, ...);
typedef void MPI_Win_errhandler_function(MPI_Win *, int *, ...);
typedef void MPI_File_errhandler_function(MPI_File *, int *, ...);

typedef int MPI_Grequest_query_function(void *extra_state,
            MPI_Status *status);
typedef int MPI_Grequest_free_function(void *extra_state);
typedef int MPI_Grequest_cancel_function(void *extra_state, int complete);

typedef int MPI_Datarep_extent_function(MPI_Datatype datatype,
            MPI_Aint *file_extent, void *extra_state);
typedef int MPI_Datarep_conversion_function(void *userbuf,
            MPI_Datatype datatype, int count, void *filebuf,
            MPI_Offset position, void *extra_state);
```

Fortran 2008 Bindings with the mpi_f08 Module

The callback prototypes when using the Fortran `mpi_f08` module are shown below:

The user-function argument to MPI_Op_create should be declared according to:

```
ABSTRACT INTERFACE
```

#388

BIND(C) removed in all ABSTRACT INTERFACE definitions

```
  SUBROUTINE MPI_User_function(invec, inoutvec, len, datatype)
      USE, INTRINSIC :: ISO_C_BINDING, ONLY : C_PTR
      TYPE(C_PTR), VALUE :: invec, inoutvec
      INTEGER :: len
      TYPE(MPI_Datatype) :: datatype
```

The copy and delete function arguments to MPI_Comm_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_copy_attr_function(oldcomm, comm_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Comm) :: oldcomm
      INTEGER :: comm_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,
      attribute_val_out
      LOGICAL :: flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_delete_attr_function(comm, comm_keyval,
  attribute_val, extra_state, ierror)
      TYPE(MPI_Comm) :: comm
      INTEGER :: comm_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state
```

The copy and delete function arguments to MPI_Win_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_copy_attr_function(oldwin, win_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Win) :: oldwin
      INTEGER :: win_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,
      attribute_val_out
      LOGICAL :: flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_delete_attr_function(win, win_keyval, attribute_val,
  extra_state, ierror)
      TYPE(MPI_Win) :: win
      INTEGER :: win_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state
```

The copy and delete function arguments to MPI_Type_create_keyval should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Type_copy_attr_function(oldtype, type_keyval, extra_state,
  attribute_val_in, attribute_val_out, flag, ierror)
      TYPE(MPI_Datatype) :: oldtype
      INTEGER :: type_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state, attribute_val_in,
      attribute_val_out
      LOGICAL :: flag

ABSTRACT INTERFACE
  SUBROUTINE MPI_Type_delete_attr_function(datatype, type_keyval,
  attribute_val, extra_state, ierror)
      TYPE(MPI_Datatype) :: datatype
      INTEGER :: type_keyval, ierror
      INTEGER(KIND=MPI_ADDRESS_KIND) :: attribute_val, extra_state
```

The handler-function argument to MPI_Comm_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Comm_errhandler_function(comm, error_code)
        TYPE(MPI_Comm) :: comm
        INTEGER :: error_code
```

The handler-function argument to MPI_Win_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Win_errhandler_function(win, error_code)
        TYPE(MPI_Win) :: win
        INTEGER :: error_code
```

The handler-function argument to MPI_File_create_errhandler should be declared like this:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_File_errhandler_function(file, error_code)
        TYPE(MPI_File) :: file
        INTEGER :: error_code
```

The query, free, and cancel function arguments to MPI_Grequest_start should be declared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_query_function(extra_state, status, ierror)
        TYPE(MPI_Status) :: status
        INTEGER :: ierror
        INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state

ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_free_function(extra_state, ierror)
        INTEGER :: ierror
        INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state

ABSTRACT INTERFACE
  SUBROUTINE MPI_Grequest_cancel_function(extra_state, complete, ierror)
        INTEGER :: ierror
        INTEGER(KIND=MPI_ADDRESS_KIND) :: extra_state
        LOGICAL :: complete
```

The extent and conversion function arguments to MPI_Register_datarep should be de-

#388

clared according to:

```
ABSTRACT INTERFACE
  SUBROUTINE MPI_Datarep_extent_function(datatype, extent, extra_state,
  ierror)
      TYPE(MPI_Datatype) ::  datatype
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extent, extra_state
      INTEGER ::  ierror

ABSTRACT INTERFACE
  SUBROUTINE MPI_Datarep_conversion_function(userbuf, datatype, count,
  filebuf, position, extra_state, ierror)
      USE, INTRINSIC ::  ISO_C_BINDING, ONLY : C_PTR
      TYPE(C_PTR), VALUE ::  userbuf, filebuf
      TYPE(MPI_Datatype) ::  datatype
      INTEGER ::  count, ierror
      INTEGER(KIND=MPI_OFFSET_KIND) ::  position
      INTEGER(KIND=MPI_ADDRESS_KIND) ::  extra_state
```

Fortran Bindings with mpif.h or the mpi Module

With the Fortran `mpi` module or `mpif.h`, here are examples of how each of the user-defined subroutines should be declared.

The user-function argument to MPI_OP_CREATE should be declared like this:

```
SUBROUTINE USER_FUNCTION(INVEC, INOUTVEC, LEN, DATATYPE)
   <type> INVEC(LEN), INOUTVEC(LEN)
   INTEGER LEN, DATATYPE
```

The copy and delete function arguments to MPI_COMM_CREATE_KEYVAL should be declared like these:

```
SUBROUTINE COMM_COPY_ATTR_FUNCTION(OLDCOMM, COMM_KEYVAL, EXTRA_STATE,
             ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
   INTEGER OLDCOMM, COMM_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) EXTRA_STATE, ATTRIBUTE_VAL_IN,
             ATTRIBUTE_VAL_OUT
   LOGICAL FLAG

SUBROUTINE COMM_DELETE_ATTR_FUNCTION(COMM, COMM_KEYVAL, ATTRIBUTE_VAL,
             EXTRA_STATE, IERROR)
   INTEGER COMM, COMM_KEYVAL, IERROR
   INTEGER(KIND=MPI_ADDRESS_KIND) ATTRIBUTE_VAL, EXTRA_STATE
```

The copy and delete function arguments to MPI_WIN_CREATE_KEYVAL should be declared like these:

```
SUBROUTINE WIN_COPY_ATTR_FUNCTION(OLDWIN, WIN_KEYVAL, EXTRA_STATE,
             ATTRIBUTE_VAL_IN, ATTRIBUTE_VAL_OUT, FLAG, IERROR)
   INTEGER OLDWIN, WIN_KEYVAL, IERROR
```

19.1.5 Info Keys

The following info keys are reserved. They are strings.

access_style
red bar: #347
accumulate_ops
accumulate_ordering
alloc_shared_noncontig
appnum
arch
cb_block_size
cb_buffer_size
cb_nodes
chunked_item
chunked_size
chunked
collective_buffering
file_perm
filename
file
host
io_node_list
ip_address
ip_port
nb_proc
no_locks
num_io_nodes
path
#369 same_disp_unit
#347 same_size
soft
striping_factor
striping_unit
wdir

19.1.6 Info Values

The following info values are reserved. They are strings.

false
random
rar
raw
read_mostly
read_once
reverse_sequential
same_op
same_op_no_op
sequential

true
#347 war
waw
write_mostly
write_once