width and encode address values in the same manner such that address values in one language may be passed directly to another language without conversion. There is the MPI constant MPI_BOTTOM to indicate the start of the address range.

2.5.7 File Offsets

For I/O there is a need to give the size, displacement, and offset into a file. These quantities can easily be larger than 32 bits which can be the default size of a Fortran integer. To overcome this, these quantities are declared to be `INTEGER (KIND=MPI_OFFSET_KIND)` in Fortran. In C one uses MPI_Offset whereas in C++ one uses MPI::Offset. These types must have the same width and encode address values in the same manner such that offset values in one language may be passed directly to another language without conversion. ticket265.

2.5.8 Counts

Derived datatypes can be created representing more elements than can be encoded in a C int or Fortran `INTEGER`. MPI_GET_COUNT, MPI_GET_ELEMENTS , and associated functions cannot properly express these quantities. To overcome this limitation, these quantities are declared to be `INTEGER (KIND=MPI_COUNT_KIND)` in Fortran. In C, one uses MPI_Count. These types must have the same width and encode values in the same manner such that count values in one language may be passed directly to another language without conversion. The size of the MPI_Count type is determined by the MPI implementation with the restriction that it must be minimally capable of encoding a C int, Fortran `INTEGER`, and any value that may be stored in a variable of type MPI_Aint.

> *Rationale.* MPI_Count explicitly specifies the number of elements in a datatype, and therefore implicitly specifies the bounds of that datatype. The number of elements in a datatype is specified at creation time using a C `int` or Fortran `INTEGER`. The extent of a datatype is expressed using an MPI_Aint. (*End of rationale.*)

2.6 Language Binding

This section defines the rules for MPI language binding in general and for Fortran, ISO C, and C++, in particular. (Note that ANSI C has been replaced by ISO C.) The C++ language bindings have been deprecated. Defined here are various object representations, as well as the naming conventions used for expressing this standard. The actual calling sequences are defined elsewhere.

MPI bindings are for Fortran 90, though they are designed to be usable in Fortran 77 environments.

Since the word `PARAMETER` is a keyword in the Fortran language, we use the word "argument" to denote the arguments to a subroutine. These are normally referred to as parameters in C and C++, however, we expect that C and C++ programmers will understand the word "argument" (which has no specific meaning in C/C++), thus allowing us to avoid unnecessary confusion for Fortran programmers.

Since Fortran is case insensitive, linkers may use either lower case or upper case when resolving Fortran names. Users of case sensitive languages should avoid the "mpi_" and "pmpi_" prefixes.