OSMP - Entwurf und Implementierung einer Message Passing Umgebung für Interprozesskommunikation

Generated by Doxygen 1.9.1

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

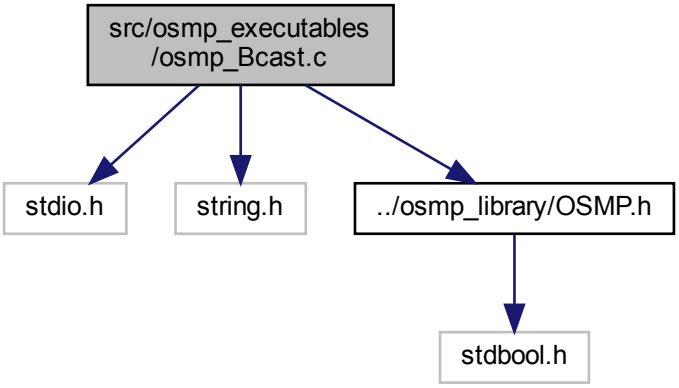
Chapter 2

File Documentation

2.1 src/osmp_executables/osmp_Bcast.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "../osmp_library/OSMP.h"
```

Include dependency graph for osmp_Bcast.c:



Functions

- int main (int argc, char ∗argv[ ])

2.1.1 Function Documentation

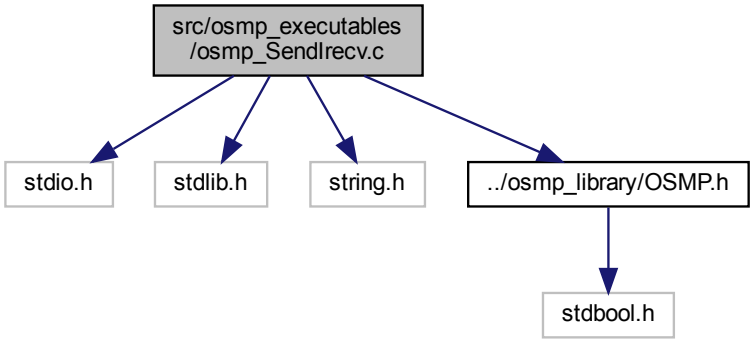
2.1.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

2.2 src/osmp_executables/osmp_SendIrecv.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../osmp_library/OSMP.h"
```

Include dependency graph for osmp_SendIrecv.c:



Functions

- int main (int argc, char ∗argv[ ])

2.2.1 Function Documentation

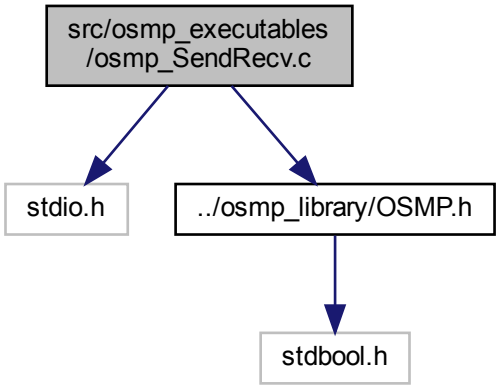
2.2.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

2.3 src/osmp_executables/osmp_SendRecv.c File Reference

```
#include <stdio.h>
#include "../osmp_library/OSMP.h"
```

Include dependency graph for osmp_SendRecv.c:



Functions

- int main (int argc, char ∗argv[ ])

2.3.1 Function Documentation

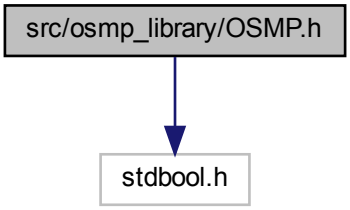
2.3.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

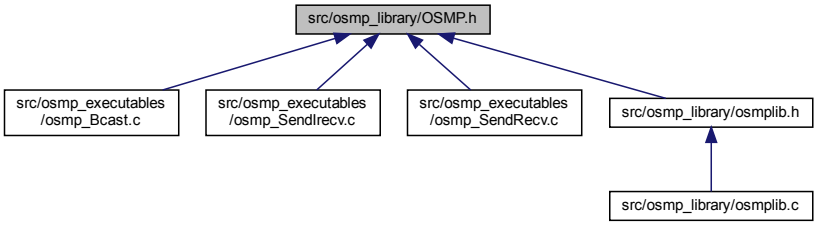
2.4 src/osmp_library/OSMP.h File Reference

```
#include <stdbool.h>
```

Include dependency graph for OSMP.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define OSMP_SUCCESS 0
- #define OSMP_FAILURE 1
- #define OSMP_CRITICAL_FAILURE 2
- #define OSMP_MAX_MESSAGES_PROC 16
- #define OSMP_MAX_SLOTS 256
- #define OSMP_MAX_PAYLOAD_LENGTH 1024

Typedefs

- typedef void * OSMP_Request
- typedef enum enum_OSMP_Datatype OSMP_Datatype

Enumerations

- enum enum_OSMP_Datatype {
  OSMP_SHORT , OSMP_INT , OSMP_LONG , OSMP_UNSIGNED_CHAR ,
  OSMP_UNSIGNED , OSMP_UNSIGNED_SHORT , OSMP_UNSIGNED_LONG , OSMP_FLOAT ,
  OSMP_DOUBLE , OSMP_BYTE }

Functions

- int get_OSMP_MAX_PAYLOAD_LENGTH ()
- int get_OSMP_MAX_SLOTS ()
- int get_OSMP_MAX_MESSAGES_PROC ()
- int get_OSMP_CRITICAL_FAILURE ()
- int get_OSMP_FAILURE ()
- int get_OSMP_SUCCESS ()
- size_t OSMP_sizeof (OSMP_Datatype datatype)
- int OSMP_Init (const int *argc, char ***argv)
- int OSMP_Size (int *size)
- int OSMP_Rank (int *rank)
- int OSMP_Send (const void *buf, int count, OSMP_Datatype datatype, int dest)
- int OSMP_Recv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len)
- int OSMP_Finalize (void)
- int OSMP_Barrier (void)
- int OSMP_Gather (void *sendbuf, int sendcount, OSMP_Datatype sendtype, void *recvbuf, int recvcount, OSMP_Datatype recvtype, bool recv)

  Sammelt Daten von allen processes in dem Empfänger-Prozess.
- int OSMP_Isend (const void *buf, int count, OSMP_Datatype datatype, int dest, OSMP_Request request)
- int OSMP_Irecv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len, OSMP_Request request)
- int OSMP_Test (OSMP_Request request, int *flag)
- int OSMP_Wait (OSMP_Request request)
- int OSMP_CreateRequest (OSMP_Request *request)
- int OSMP_RemoveRequest (OSMP_Request *request)
- int OSMP_GetShmName (char **name)

2.4.1 Macro Definition Documentation

2.4.1.1 OSMP_CRITICAL_FAILURE

```
#define OSMP_CRITICAL_FAILURE 2
```

Im Fehlerfall liefern die OSMP-Funktionen OSMP_CRITICAL_FAILURE zurück. Die Fehler sollten zum beenden des Programms führen (z. B.)

2.4.1.2 OSMP_FAILURE

```
#define OSMP_FAILURE 1
```

Im Fehlerfall liefern die OSMP-Funktionen den Wert OSMP_FAILURE zurück. Die Fehler führen aber nicht zum beenden des Programms (z. B. wenn ein Prozess eine Nachricht an einen nicht existierenden Prozess schickt).

2.4.1.3 OSMP_MAX_MESSAGES_PROC

```
#define OSMP_MAX_MESSAGES_PROC 16
```

2.4.1.4 OSMP_MAX_PAYLOAD_LENGTH

```
#define OSMP_MAX_PAYLOAD_LENGTH 1024
```

2.4.1.5 OSMP_MAX_SLOTS

```
#define OSMP_MAX_SLOTS 256
```

2.4.1.6 OSMP_SUCCESS

```
#define OSMP_SUCCESS 0
```

Diese Datei beinhaltet lediglich alle in der Anleitung angegeben Prototypen der OSMP Kernfunktionen wie z.B. OSMP_Test(), sowie vorgegebene Konstanten. Alle OSMP-Funktionen liefern im Erfolgsfall OSMP_SUCCESS als Rückgabewert. Weitere Rückgabewerte können mit Begründung (und Dokumentation!) definiert werden

2.4.2 Typedef Documentation

2.4.2.1 OSMP_Datatype

```
typedef enum enum_OSMP_Datatype OSMP_Datatype
```

Die OSMP-Datentypen entsprechen den C-Datentypen. Sie werden verwendet, um den Typ der Daten anzugeben, die mit den OSMP-Funktionen gesendet bzw. empfangen werden sollen. Mindestens folgende Datentypen *müssen* unterstützt werden:

2.4.2.2 OSMP_Request

```
typedef void* OSMP_Request
```

2.4.3 Enumeration Type Documentation

2.4.3.1 enum_OSMP_Datatype

```
enum enum_OSMP_Datatype
```

Die OSMP-Datentypen entsprechen den C-Datentypen. Sie werden verwendet, um den Typ der Daten anzugeben, die mit den OSMP-Funktionen gesendet bzw. empfangen werden sollen. Mindestens folgende Datentypen *müssen* unterstützt werden:

Enumerator

OSMP_SHORT	
OSMP_INT	
OSMP_LONG	
OSMP_UNSIGNED_CHAR	
OSMP_UNSIGNED	
OSMP_UNSIGNED_SHORT	
OSMP_UNSIGNED_LONG	
OSMP_FLOAT	
OSMP_DOUBLE	
OSMP_BYTE	

2.4.4 Function Documentation

2.4.4.1 get_OSMP_CRITICAL_FAILURE()

```
int get_OSMP_CRITICAL_FAILURE ( )
```

2.4.4.2 get_OSMP_FAILURE()

```
int get_OSMP_FAILURE ( )
```

2.4.4.3 get_OSMP_MAX_MESSAGES_PROC()

```
int get_OSMP_MAX_MESSAGES_PROC ( )
```

2.4.4.4 get_OSMP_MAX_PAYLOAD_LENGTH()

```
int get_OSMP_MAX_PAYLOAD_LENGTH ( )
```

2.4.4.5 get_OSMP_MAX_SLOTS()

```
int get_OSMP_MAX_SLOTS ( )
```

2.4.4.6 get_OSMP_SUCCESS()

```
int get_OSMP_SUCCESS ( )
```

2.4.4.7 OSMP_Barrier()

```
int OSMP_Barrier (
            void  )
```

Diese kollektive Funktion blockiert den aufrufenden Prozess. Erst wenn alle anderen Prozesse ebenfalls an der Barriere angekommen sind, laufen die Prozesse weiter.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.8 OSMP_CreateRequest()

```
int OSMP_CreateRequest (
            OSMP_Request * request )
```

Die Funktionen stellen den Speicher für einen Request zur Verfügung bzw. deallozieren den Speicher.

Parameters

out	*request*	Adresse eines Requests (input)

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.9 OSMP_Finalize()

```
int OSMP_Finalize (
            void  )
```

Alle OSMP-Prozesse müssen diese Funktion aufrufen, bevor sie sich beenden. Sie geben damit den Zugriff auf die gemeinsamen Ressourcen frei. Hierbei muss jeder Prozess zuvor alle noch vorhandenen Nachrichten abarbeiten.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.10 OSMP_Gather()

```
int OSMP_Gather (
            void * sendbuf,
            int sendcount,
            OSMP_Datatype sendtype,
            void * recvbuf,
            int recvcount,
            OSMP_Datatype recvtype,
            bool recv )
```

Sammelt Daten von allen processes in dem Empfänger-Prozess.

This function gathers data from all processes in the communicator and delivers it to the root process. Each process can provide a different send buffer and send count, but the receive buffer and receive count must be the same on all processes.

Parameters

in	*sendbuf*	Pointer to the send buffer.
in	*sendcount*	Number of elements in the send buffer.
in	*sendtype*	MPI datatype of the send buffer elements.
out	*recvbuf*	Pointer to the receive buffer.
in	*recvcount*	Number of elements in the receive buffer.
in	*recvtype*	MPI datatype of the receive buffer elements.
in	*recv*	Ist der aufrufende Prozess der Sender

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.11 OSMP_GetShmName()

```
int OSMP_GetShmName (
            char ** name )
```

Diese Funktion gibt den Namen des Shared Memory Bereichs im Parameter name zurück.

Parameters

out	*name*	Der Name des Shared Memory Bereichs

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.12 OSMP_Init()

```
int OSMP_Init (
            const int * argc,
            char *** argv )
```

Die Funktion OSMP_Init() initialisiert die OSMP-Umgebung und ermöglicht den Zugang zu den gemeinsamen Ressourcen der OSMP-Prozesse. Sie muss von jedem OSMP-Prozess zu Beginn aufgerufen werden.

Parameters

in	*argc*	Adresse der Argumentzahl
in	*argv*	Adresse des Argumentvektors

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

In dieser Quelltext-Datei sind Implementierungen der OSMP Bibliothek zu finden.

2.4.4.13 OSMP_Irecv()

```
int OSMP_Irecv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len,
            OSMP_Request request )
```

Die Funktion empfängt eine Nachricht analog zu OSMP_Recv(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Empfangen).

Parameters

out	*buf*	
in	*count*	
in	*datatype*	
out	*source*	
out	*len*	
in,out	*request*	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.14 OSMP_Isend()

```
int OSMP_Isend (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest,
            OSMP_Request request )
```

Die Funktion sendet eine Nachricht analog zu OSMP_Send(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Senden).

Parameters

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	PID des Empfängers zwischen 0, . . . , np-1
in,out	*request*	Adresse einer eigenen Datenstruktur, die später verwendet werden kann, um abzufragen, ob die Operation abgeschlossen ist.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.15 OSMP_Rank()

```
int OSMP_Rank (
            int * rank )
```

Die Funktion OSMP_Rank() liefert in ∗rank die OSMP-Prozessnummer des aufrufenden OSMP-Prozesses von 0,. . . ,np-1 zurück.

Parameters

out	*rank*	Prozessnummer 0,. . . ,np-1 des aktuellen OSMP-Prozesse

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.16 OSMP_Recv()

```
int OSMP_Recv (
            void * buf,
```

```
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len )
```

Der aufrufende Prozess empfängt eine Nachricht mit maximal count Elementen des angegebenen Datentyps datatype. Die Nachricht wird an die Adresse buf des aufrufenden Prozesses geschrieben. Unter source wird die OSMP-Prozessnummer des sendenden Prozesses und unter len die tatsächliche Länge der gelesenen Nachricht abgelegt. Die Funktion ist blockierend, d.h. sie wartet, bis eine Nachricht für den Prozess vorhanden ist. Wenn die Funktion zurückkehrt, ist der Kopierprozess abgeschlossen. Die Nachricht gilt nach dem Aufruf dieser Funktion als abgearbeitet.

Parameters

out	*buf*	Startadresse des Puffers im lokalen Speicher des aufrufenden Prozesses, in den die Nachricht kopiert werden soll.
in	*count*	maximale Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	Nummer des Senders zwischen 0,. . . ,np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.17 OSMP_RemoveRequest()

```
int OSMP_RemoveRequest (
            OSMP_Request * request )
```

Die Funktionen stellen den Speicher für einen Request zur Verfügung bzw. deallozieren den Speicher.

Parameters

in	*request*	Adresse eines Requests

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.18 OSMP_Send()

```
int OSMP_Send (
            const void * buf,
            int count,
```

```
            OSMP_Datatype datatype,
            int dest )
```

Die Funktion OSMP_Send() sendet eine Nachricht an den Prozess mit der Nummer dest. Die Nachricht besteht aus count Elementen vom Typ datatype. Die zu sendende Nachricht beginnt im aufrufenden Prozess bei der Adresse buf. Die Funktion ist blockierend, d.h. wenn sie in das aufrufende Programm zurückkehrt, ist der Kopiervorgang abgeschlossen.

Parameters

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	Nummer des Empfängers zwischen 0,...,np-1

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.19 OSMP_Size()

```
int OSMP_Size (
            int * size )
```

Die Funktion OSMP_Size() liefert in ∗size die Zahl der OSMP-Prozesse ohne den OSMP-Starter Prozess zurück. Sollte mit der Zahl übereinstimmen, die in der Kommandozeile dem OSMP-Starter übergeben wird.

Parameters

out	*rank*	Zahl der OSMP-Prozesse

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.20 OSMP_sizeof()

```
size_t OSMP_sizeof (
            OSMP_Datatype datatype )
```

Die Funktion OSMP_sizeof() liefert die Größe des Datentyps datatype in Byte zurück.

Parameters

in	*datatype*	OSMP-Datentyp

Returns

Größe des Datentyps in Byte

2.4.4.21 OSMP_Test()

```
int OSMP_Test (
            OSMP_Request request,
            int * flag )
```

Die Funktion testet, ob die mit dem request verknüpfte Operation abgeschlossen ist. Sie ist nicht blockierend, d.h. sie wartet nicht auf das Ende der mit request verknüpften Operation.

Parameters

in	*request*	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	*flag*	Gibt den Status der Operation an.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.4.4.22 OSMP_Wait()

```
int OSMP_Wait (
            OSMP_Request request )
```

Die Funktion prüft, ob die mit dem request verknüpfte, nicht blockierende Operation abgeschlossen ist. Sie ist so lange blockiert, bis dies der Fall ist.

Parameters

in	*request*	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert

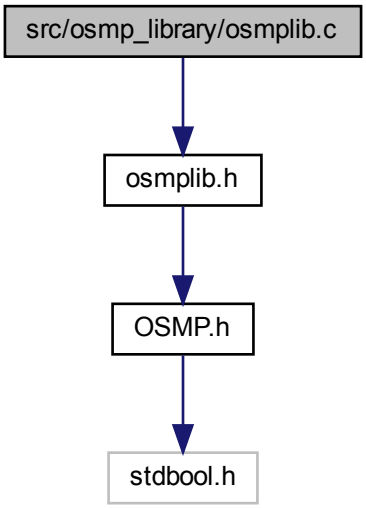
Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5 src/osmp_library/osmplib.c File Reference

```
#include "osmplib.h"
```

Include dependency graph for osmplib.c:



Functions

- int OSMP_Init (const int *argc, char ***argv)
- int OSMP_Size (int *size)
- int OSMP_Rank (int *rank)
- int OSMP_Send (const void *buf, int count, OSMP_Datatype datatype, int dest)
- int OSMP_Recv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len)
- int OSMP_Finalize (void)
- int OSMP_Barrier (void)
- int OSMP_Bcast (void *buf, int count, OSMP_Datatype datatype, bool send, int *source, int *len)
- int OSMP_Isend (const void *buf, int count, OSMP_Datatype datatype, int dest, OSMP_Request request)
- int OSMP_Irecv (void *buf, int count, OSMP_Datatype datatype, int *source, int *len, OSMP_Request request)
- int OSMP_Test (OSMP_Request request, int *flag)
- int OSMP_Wait (OSMP_Request request)
- int OSMP_CreateRequest (OSMP_Request *request)
- int OSMP_RemoveRequest (OSMP_Request *request)
- int OSMP_GetShmName (char **name)

2.5.1 Function Documentation

2.5.1.1 OSMP_Barrier()

```
int OSMP_Barrier (
            void  )
```

Diese kollektive Funktion blockiert den aufrufenden Prozess. Erst wenn alle anderen Prozesse ebenfalls an der Barriere angekommen sind, laufen die Prozesse weiter.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.2 OSMP_Bcast()

```
int OSMP_Bcast (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            bool send,
            int * source,
            int * len )
```

2.5.1.3 OSMP_CreateRequest()

```
int OSMP_CreateRequest (
            OSMP_Request * request )
```

Die Funktionen stellen den Speicher für einen Request zur Verfügung bzw. deallozieren den Speicher.

Parameters

out	*request*	Adresse eines Requests (input)

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.4 OSMP_Finalize()

```
int OSMP_Finalize (
            void  )
```

Alle OSMP-Prozesse müssen diese Funktion aufrufen, bevor sie sich beenden. Sie geben damit den Zugriff auf die gemeinsamen Ressourcen frei. Hierbei muss jeder Prozess zuvor alle noch vorhandenen Nachrichten abarbeiten.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.5 OSMP_GetShmName()

```
int OSMP_GetShmName (
            char ** name )
```

Diese Funktion gibt den Namen des Shared Memory Bereichs im Parameter name zurück.

Parameters

out	*name*	Der Name des Shared Memory Bereichs

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.6 OSMP_Init()

```
int OSMP_Init (
            const int * argc,
            char *** argv )
```

In dieser Quelltext-Datei sind Implementierungen der OSMP Bibliothek zu finden.

2.5.1.7 OSMP_Irecv()

```
int OSMP_Irecv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len,
            OSMP_Request request )
```

Die Funktion empfängt eine Nachricht analog zu OSMP_Recv(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Empfangen).

Parameters

out	*buf*	
in	*count*	
in	*datatype*	
out	*source*	
out	*len*	
in,out	*request*	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.8 OSMP_Isend()

```
int OSMP_Isend (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest,
            OSMP_Request request )
```

Die Funktion sendet eine Nachricht analog zu OSMP_Send(). Die Funktion kehrt jedoch sofort zurück, ohne dass das Kopieren der Nachricht sichergestellt ist (nicht blockierendes Senden).

Parameters

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	PID des Empfängers zwischen 0, . . . , np-1
in,out	*request*	Adresse einer eigenen Datenstruktur, die später verwendet werden kann, um abzufragen, ob die Operation abgeschlossen ist.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.9 OSMP_Rank()

```
int OSMP_Rank (
            int * rank )
```

Die Funktion OSMP_Rank() liefert in ∗rank die OSMP-Prozessnummer des aufrufenden OSMP-Prozesses von 0,. . . ,np-1 zurück.

Parameters

out	*rank*	Prozessnummer 0,. . . ,np-1 des aktuellen OSMP-Prozesse

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.10 OSMP_Recv()

```
int OSMP_Recv (
            void * buf,
            int count,
            OSMP_Datatype datatype,
            int * source,
            int * len )
```

Der aufrufende Prozess empfängt eine Nachricht mit maximal count Elementen des angegebenen Datentyps datatype. Die Nachricht wird an die Adresse buf des aufrufenden Prozesses geschrieben. Unter source wird die OSMP-Prozessnummer des sendenden Prozesses und unter len die tatsächliche Länge der gelesenen Nachricht abgelegt. Die Funktion ist blockierend, d.h. sie wartet, bis eine Nachricht für den Prozess vorhanden ist. Wenn die Funktion zurückkehrt, ist der Kopierprozess abgeschlossen. Die Nachricht gilt nach dem Aufruf dieser Funktion als abgearbeitet.

Parameters

out	*buf*	Startadresse des Puffers im lokalen Speicher des aufrufenden Prozesses, in den die Nachricht kopiert werden soll.
in	*count*	maximale Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	*datatype*	OSMP-Typ der Daten im Puffer
out	*source*	Nummer des Senders zwischen 0,. . . ,np-1
out	*len*	tatsächliche Länge der empfangenen Nachricht in Byte

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.11 OSMP_RemoveRequest()

```
int OSMP_RemoveRequest (
            OSMP_Request * request )
```

Die Funktionen stellen den Speicher für einen Request zur Verfügung bzw. deallozieren den Speicher.

Parameters

in	*request*	Adresse eines Requests

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.12 OSMP_Send()

```
int OSMP_Send (
            const void * buf,
            int count,
            OSMP_Datatype datatype,
            int dest )
```

Die Funktion OSMP_Send() sendet eine Nachricht an den Prozess mit der Nummer dest. Die Nachricht besteht aus count Elementen vom Typ datatype. Die zu sendende Nachricht beginnt im aufrufenden Prozess bei der Adresse buf. Die Funktion ist blockierend, d.h. wenn sie in das aufrufende Programm zurückkehrt, ist der Kopiervorgang abgeschlossen.

Parameters

in	*buf*	Startadresse des Puffers mit der zu sendenden Nachricht
in	*count*	Zahl der Elemente vom angegebenen Typ im Puffer
in	*datatype*	OSMP-Typ der Daten im Puffer
in	*dest*	Nummer des Empfängers zwischen 0,...,np-1

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.13 OSMP_Size()

```
int OSMP_Size (
            int * size )
```

Die Funktion OSMP_Size() liefert in ∗size die Zahl der OSMP-Prozesse ohne den OSMP-Starter Prozess zurück. Sollte mit der Zahl übereinstimmen, die in der Kommandozeile dem OSMP-Starter übergeben wird.

Parameters

out	*rank*	Zahl der OSMP-Prozesse

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.14 OSMP_Test()

```
int OSMP_Test (
            OSMP_Request request,
            int * flag )
```

Die Funktion testet, ob die mit dem request verknüpfte Operation abgeschlossen ist. Sie ist nicht blockierend, d.h. sie wartet nicht auf das Ende der mit request verknüpften Operation.

Parameters

in	*request*	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	*flag*	Gibt den Status der Operation an.

Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

2.5.1.15 OSMP_Wait()

```
int OSMP_Wait (
            OSMP_Request request )
```

Die Funktion prüft, ob die mit dem request verknüpfte, nicht blockierende Operation abgeschlossen ist. Sie ist so lange blockiert, bis dies der Fall ist.

Parameters

in	*request*	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert

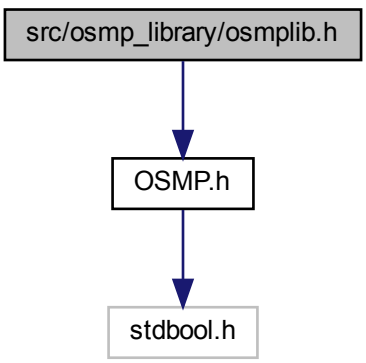
Returns

OSMP_SUCCESS or OSMP_FAILURE or OSMP_CRITICAL_FAILURE

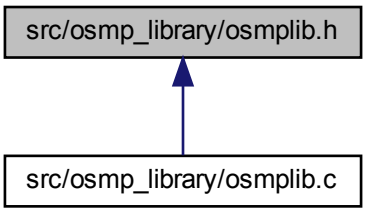
2.6 src/osmp_library/osmplib.h File Reference

```
#include "OSMP.h"
```

Include dependency graph for osmplib.h:



This graph shows which files directly or indirectly include this file:



2.7 src/osmp_runner/osmp_run.c File Reference

Functions

- int main (int argc, char **argv)

2.7.1 Function Documentation

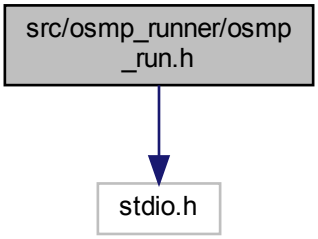
2.7.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

2.8 src/osmp_runner/osmp_run.h File Reference

```
#include <stdio.h>
```

Include dependency graph for osmp_run.h:



Index