

OSMP - Entwurf und Implementierung einer Message Passing Umgebung für  
Interprozesskommunikation

Erzeugt von Doxygen 1.9.1



<b>1 Datei-Verzeichnis</b>	<b>1</b>
1.1 Auflistung der Dateien	1
<b>2 Datei-Dokumentation</b>	<b>3</b>
2.1 src/osmp_executables/osmpExecutable_SendRecv.c-Dateireferenz	3
2.1.1 Dokumentation der Funktionen	3
2.1.1.1 main()	4
2.2 src/osmp_executables/osmpExecutable_SendRecv.c-Dateireferenz	4
2.2.1 Dokumentation der Funktionen	4
2.2.1.1 main()	4
2.3 src/osmp_library/OSMP.h-Dateireferenz	5
2.3.1 Makro-Dokumentation	6
2.3.1.1 OSMP_FAILURE	6
2.3.1.2 OSMP_MAX_MESSAGES_PROC	6
2.3.1.3 OSMP_MAX_PAYLOAD_LENGTH	7
2.3.1.4 OSMP_MAX_SLOTS	7
2.3.1.5 OSMP_NOT_IMPLEMENTED_YET	7
2.3.1.6 OSMP_SUCCESS	7
2.3.2 Dokumentation der benutzerdefinierten Typen	7
2.3.2.1 OSMP_Datatype	7
2.3.2.2 OSMP_Request	7
2.3.3 Dokumentation der Aufzählungstypen	7
2.3.3.1 enum_OSMP_Datatype	7
2.3.4 Dokumentation der Funktionen	8
2.3.4.1 get_OSMP_FAILURE()	8
2.3.4.2 get_OSMP_MAX_MESSAGES_PROC()	8
2.3.4.3 get_OSMP_MAX_PAYLOAD_LENGTH()	8
2.3.4.4 get_OSMP_MAX_SLOTS()	8
2.3.4.5 get_OSMP_NOT_IMPLEMENTED_YET()	9
2.3.4.6 get_OSMP_SUCCESS()	9
2.3.4.7 OSMP_Barrier()	9
2.3.4.8 OSMP_CreateRequest()	9
2.3.4.9 OSMP_Finalize()	10
2.3.4.10 OSMP_Gather()	10
2.3.4.11 OSMP_GetSharedMemoryName()	10
2.3.4.12 OSMP_Init()	11
2.3.4.13 OSMP_IRecv()	11
2.3.4.14 OSMP_ISend()	12
2.3.4.15 OSMP_Rank()	12
2.3.4.16 OSMP_Recv()	13
2.3.4.17 OSMP_RemoveRequest()	13
2.3.4.18 OSMP_Send()	14

2.3.4.19 OSMP_Size()	14
2.3.4.20 OSMP_SizeOf()	15
2.3.4.21 OSMP_Test()	15
2.3.4.22 OSMP_Wait()	15
2.4 src/osmp_library/osmplib.c-Dateireferenz	16
2.4.1 Dokumentation der Funktionen	17
2.4.1.1 get_OSMP_FAILURE()	17
2.4.1.2 get_OSMP_MAX_MESSAGES_PROC()	17
2.4.1.3 get_OSMP_MAX_PAYLOAD_LENGTH()	17
2.4.1.4 get_OSMP_MAX_SLOTS()	17
2.4.1.5 get_OSMP_NOT_IMPLEMENTED_YET()	17
2.4.1.6 get_OSMP_SUCCESS()	18
2.4.1.7 OSMP_Barrier()	18
2.4.1.8 OSMP_Bcast()	18
2.4.1.9 OSMP_CreateRequest()	18
2.4.1.10 OSMP_Finalize()	19
2.4.1.11 OSMP_GetShmName()	19
2.4.1.12 OSMP_Init()	19
2.4.1.13 OSMP_IRecv()	20
2.4.1.14 OSMP_ISend()	20
2.4.1.15 OSMP_Rank()	21
2.4.1.16 OSMP_Recv()	21
2.4.1.17 OSMP_RemoveRequest()	22
2.4.1.18 OSMP_Send()	22
2.4.1.19 OSMP_Size()	23
2.4.1.20 OSMP_Test()	23
2.4.1.21 OSMP_Wait()	23
2.5 src/osmp_library/osmplib.h-Dateireferenz	24
2.5.1 Makro-Dokumentation	25
2.5.1.1 UNUSED	25
2.6 src/osmp_runner/osmp_run.c-Dateireferenz	25
2.6.1 Dokumentation der Funktionen	25
2.6.1.1 main()	25
2.7 src/osmp_runner/osmp_run.h-Dateireferenz	26
<b>Index</b>	<b>27</b>

# Kapitel 1

## Datei-Verzeichnis

### 1.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

src/osmp_executables/ <a href="#">osmpExecutable_SendRecv.c</a> . . . . .	3
src/osmp_executables/ <a href="#">osmpExecutable_SendRecv.c</a> . . . . .	4
src/osmp_library/ <a href="#">OSMP.h</a> . . . . .	5
src/osmp_library/ <a href="#">osmplib.c</a> . . . . .	16
src/osmp_library/ <a href="#">osmplib.h</a> . . . . .	24
src/osmp_runner/ <a href="#">osmp_run.c</a> . . . . .	25
src/osmp_runner/ <a href="#">osmp_run.h</a> . . . . .	26



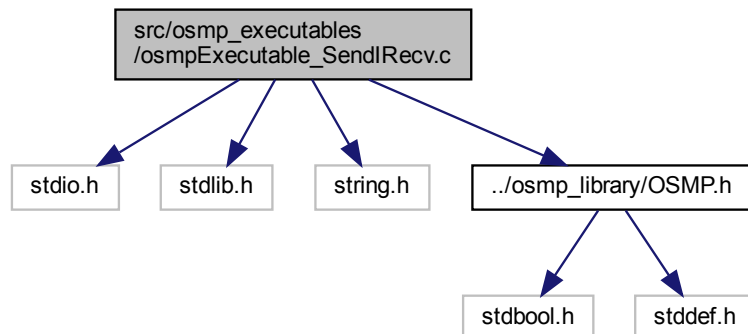
## Kapitel 2

# Datei-Dokumentation

### 2.1 src/osmp\_executables/osmpExecutable\_SendRecv.c-Dateireferenz

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

Include-Abhängigkeitsdiagramm für osmpExecutable\_SendRecv.c:



#### Funktionen

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

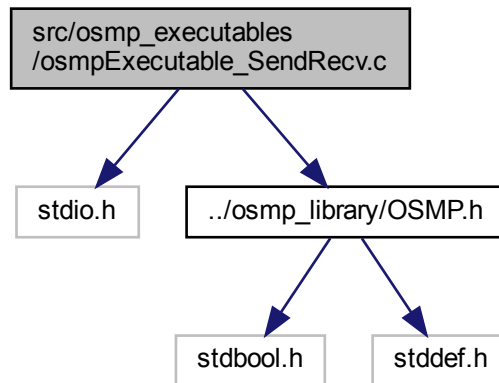
#### 2.1.1 Dokumentation der Funktionen

### 2.1.1.1 main()

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

## 2.2 src/osmp\_executables/osmpExecutable\_SendRecv.c-Dateireferenz

```
#include <stdio.h>
#include "../osmp_library/OSMP.h"
Include-Abhängigkeitsdiagramm für osmpExecutable_SendRecv.c:
```



### Funktionen

- int [main](#) (int argc, char \*argv[])

## 2.2.1 Dokumentation der Funktionen

### 2.2.1.1 main()

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

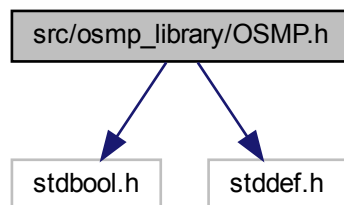


## 2.3 src/osmp\_library/OSMP.h-Dateireferenz

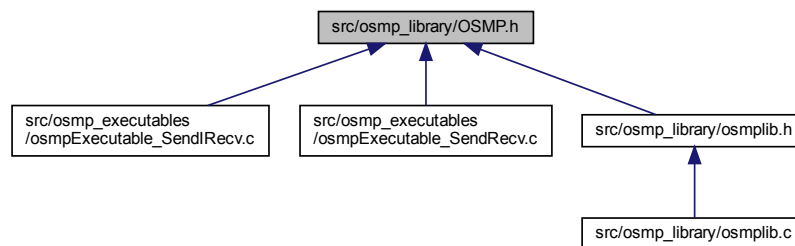
```
#include <stdbool.h>
```

```
#include <stddef.h>
```

Include-Abhängigkeitsdiagramm für OSMP.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



### Makrodefinitionen

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

### Typdefinitionen

- typedef void \* `OSMP_Request`
- typedef enum `enum_OSMP_Datatype` `OSMP_Datatype`

## Aufzählungen

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

## Funktionen

- int `get_OSM_P_MAX_PAYLOAD_LENGTH` ()
- int `get_OSM_P_MAX_SLOTS` ()
- int `get_OSM_P_MAX_MESSAGES_PROC` ()
- int `get_OSM_P_NOT_IMPLEMENTED_YET` ()
- int `get_OSM_P_FAILURE` ()
- int `get_OSM_P_SUCCESS` ()
- int `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` recvttype, int recv)
- 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_GetSharedMemoryName` (char \*\*name)

## 2.3.1 Makro-Dokumentation

### 2.3.1.1 OSM\_P\_FAILURE

```
#define OSM_P_FAILURE 1
```

Im Fehlerfall liefern die OSM\_P-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).

### 2.3.1.2 OSM\_P\_MAX\_MESSAGES\_PROC

```
#define OSM_P_MAX_MESSAGES_PROC 16
```

### 2.3.1.3 OSMP\_MAX\_PAYLOAD\_LENGTH

```
#define OSMP_MAX_PAYLOAD_LENGTH 1024
```

### 2.3.1.4 OSMP\_MAX\_SLOTS

```
#define OSMP_MAX_SLOTS 256
```

### 2.3.1.5 OSMP\_NOT\_IMPLEMENTED\_YET

```
#define OSMP_NOT_IMPLEMENTED_YET 2
```

Falls eine Funktion noch nicht implementiert ist, geben diese OSMP\_NOT\_IMPLEMENTED\_YET zurück.

### 2.3.1.6 OSMP\_SUCCESS

```
#define OSMP_SUCCESS 0
```

Alle OSMP-Funktionen liefern im Erfolgsfall OSMP\_SUCCESS als Rückgabewert. Weitere Rückgabewerte können mit Begründung (und Dokumentation!) definiert werden

## 2.3.2 Dokumentation der benutzerdefinierten Typen

### 2.3.2.1 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:

### 2.3.2.2 OSMP\_Request

```
typedef void* OSMP\_Request
```

## 2.3.3 Dokumentation der Aufzählungstypen

### 2.3.3.1 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:

**Aufzählungswerte**

OSMP_SHORT	
OSMP_INT	
OSMP_LONG	
OSMP_UNSIGNED_CHAR	
OSMP_UNSIGNED	
OSMP_UNSIGNED_SHORT	
OSMP_UNSIGNED_LONG	
OSMP_FLOAT	
OSMP_DOUBLE	
OSMP_BYTE	

## 2.3.4 Dokumentation der Funktionen

### 2.3.4.1 `get_OSMF_FAILURE()`

```
int get_OSMF_FAILURE ( )
```

Gibt den Wert von OSMF\_FAILURE zurück.

### 2.3.4.2 `get_OSMF_MAX_MESSAGES_PROC()`

```
int get_OSMF_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

### 2.3.4.3 `get_OSMF_MAX_PAYLOAD_LENGTH()`

```
int get_OSMF_MAX_PAYLOAD_LENGTH ( )
```

Gibt die maximale Länge der Nutzlast einer Nachricht zurück.

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

### 2.3.4.4 `get_OSMF_MAX_SLOTS()`

```
int get_OSMF_MAX_SLOTS ( )
```

Gibt die Maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen zurück.

#### 2.3.4.5 get\_OSMP\_NOT\_IMPLEMENTED\_YET()

```
int get_OSMP_NOT_IMPLEMENTED_YET ( )
```

Gibt den Wert von OSMP\_NOT\_IMPLEMENTED\_YET zurück.

#### 2.3.4.6 get\_OSMP\_SUCCESS()

```
int get_OSMP_SUCCESS ( )
```

Gibt den Wert von OSMP\_SUCCESS zurück.

#### 2.3.4.7 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.

##### Rückgabe

Im Erfolgsfall OSMP\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMP\_NOT\_IMPLEMENTED\_YET, sonst OSMP\_FAILURE

#### 2.3.4.8 OSMP\_CreateRequest()

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

Erstellt eine OSMP\_Request. Eine OSMP\_Request wird dazu verwendet, um nicht blockierende Operationen zu überwachen.

##### Parameter

out	<i>request</i>	Adresse eines Requests (input)
-----	----------------	--------------------------------

##### Rückgabe

Im Erfolgsfall OSMP\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMP\_NOT\_IMPLEMENTED\_YET, sonst OSMP\_FAILURE

### 2.3.4.9 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. Dies bedeutet, dass der Posteingang gesperrt wird und alle noch vorhandenen Nachrichten werden gelöscht.

#### Rückgabe

Im Erfolgsfall OSMP\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMP\_NOT\_IMPLEMENTED\_YET, sonst OSMP\_FAILURE

### 2.3.4.10 OSMP\_Gather()

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

Sammelt Daten von allen aufrufenden Prozessen und liefert sie an den Empfängerprozess. Jeder Prozess kann einen anderen Sendebuffer und eine andere Sendeanzahl bereitstellen, der Empfängerbuffer und die Empfängeranzahl müssen jedoch auf allen Prozessen gleich sein.

#### Parameter

in	<i>sendbuf</i>	Pointer to the send buffer.
in	<i>sendcount</i>	Number of elements in the send buffer.
in	<i>sendtype</i>	MPI datatype of the send buffer elements.
out	<i>recvbuf</i>	Pointer to the receive buffer.
in	<i>recvcount</i>	Number of elements in the receive buffer.
in	<i>recvtype</i>	MPI datatype of the receive buffer elements.
in	<i>recv</i>	1, Falls der aufrufende Prozess der Sender ist, sonst 0.

#### Rückgabe

Im Erfolgsfall OSMP\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMP\_NOT\_IMPLEMENTED\_YET, sonst OSMP\_FAILURE

### 2.3.4.11 OSMP\_GetSharedMemoryName()

```
int OSMP_GetSharedMemoryName (
    char ** name )
```

Gibt den Namen des Shared Memory Bereichs zurück.

#### Parameter

out	<i>name</i>	Der Name des Shared Memory Bereichs
-----	-------------	-------------------------------------

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.3.4.12 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. Durch diesen Aufruf wird außerdem der Posteingang des Prozesses freigegeben.

#### Parameter

in	<i>argc</i>	Adresse der Argumentzahl
in	<i>argv</i>	Adresse des Argumentvektors

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.3.4.13 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).

**Parameter**

out	<i>buf</i>	Startadresse des Speicherbereichs, wo die zu empfangende Nachricht gespeichert werden soll.
in	<i>count</i>	Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	<i>datatype</i>	OSMP-Typ der Daten im Puffer
out	<i>source</i>	PID des Senders zwischen 0, ..., np-1
out	<i>len</i>	tatsächliche Länge der empfangenen Nachricht in Byte
in, out	<i>request</i>	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.3.4.14 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).

**Parameter**

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

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.3.4.15 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.

#### Parameter

out	<i>rank</i>	Prozessnummer <code>0, ..., np-1</code> des aktuellen OSMP-Prozesse
-----	-------------	---

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.3.4.16 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.

#### Parameter

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

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.3.4.17 OSMP\_RemoveRequest()

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

Löscht eine `OSMP_Request`.

**Parameter**

in	<i>request</i>	Adresse eines Requests
----	----------------	------------------------

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.3.4.18 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.

**Parameter**

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

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.3.4.19 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.

**Parameter**

out	<i>rank</i>	Zahl der OSMP-Prozesse
-----	-------------	------------------------

### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

#### 2.3.4.20 OSMP\_SizeOf()

```
int OSMP_SizeOf (
    OSMP_Datatype datatype )
```

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

### Parameter

in	<i>datatype</i>	OSMP-Datentyp
----	-----------------	---------------

### Rückgabe

Größe des Datentyps in Byte

#### 2.3.4.21 OSMP\_Test()

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

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

### Parameter

in	<i>request</i>	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	<i>flag</i>	Gibt den Status der Operation an.

### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

#### 2.3.4.22 OSMP\_Wait()

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

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

#### Parameter

<code>in</code>	<code>request</code>	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert
-----------------	----------------------	--

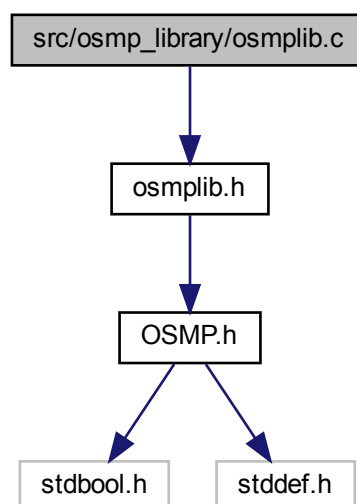
#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

## 2.4 src/osmp\_library/osmplib.c-Dateireferenz

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

Include-Abhängigkeitsdiagramm für `osmplib.c`:



## Funktionen

- `int get_OSMP_MAX_PAYLOAD_LENGTH ()`
- `int get_OSMP_MAX_SLOTS ()`
- `int get_OSMP_MAX_MESSAGES_PROC ()`
- `int get_OSMP_NOT_IMPLEMENTED_YET ()`
- `int get_OSMP_FAILURE ()`
- `int get_OSMP_SUCCESS ()`
- `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)

## 2.4.1 Dokumentation der Funktionen

### 2.4.1.1 `get_OSMP_FAILURE()`

```
int get_OSMP_FAILURE ( )
```

Gibt den Wert von `OSMP_FAILURE` zurück.

### 2.4.1.2 `get_OSMP_MAX_MESSAGES_PROC()`

```
int get_OSMP_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

### 2.4.1.3 `get_OSMP_MAX_PAYLOAD_LENGTH()`

```
int get_OSMP_MAX_PAYLOAD_LENGTH ( )
```

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

### 2.4.1.4 `get_OSMP_MAX_SLOTS()`

```
int get_OSMP_MAX_SLOTS ( )
```

Gibt die Maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen zurück.

### 2.4.1.5 `get_OSMP_NOT_IMPLEMENTED_YET()`

```
int get_OSMP_NOT_IMPLEMENTED_YET ( )
```

Gibt den Wert von `OSMP_NOT_IMPLEMENTED_YET` zurück.

#### 2.4.1.6 get\_OSMF\_SUCCESS()

```
int get_OSMF_SUCCESS ( )
```

Gibt den Wert von OSMF\_SUCCESS zurück.

#### 2.4.1.7 OSMF\_Barrier()

```
int OSMF_Barrier (
    void )
```

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

##### Rückgabe

Im Erfolgsfall OSMF\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMF\_NOT\_IMPLEMENTED\_YET, sonst OSMF\_FAILURE

#### 2.4.1.8 OSMF\_Bcast()

```
int OSMF_Bcast (
    void * buf,
    int count,
    OSMF_Datatype datatype,
    bool send,
    int * source,
    int * len )
```

#### 2.4.1.9 OSMF\_CreateRequest()

```
int OSMF_CreateRequest (
    OSMF_Request * request )
```

Erstellt eine OSMF\_Request. Eine OSMF\_Request wird dazu verwendet, um nicht blockierende Operationen zu überwachen.

##### Parameter

out	<i>request</i>	Adresse eines Requests (input)
-----	----------------	--------------------------------

##### Rückgabe

Im Erfolgsfall OSMF\_SUCCESS, falls die Funktion noch nicht implementiert ist OSMF\_NOT\_IMPLEMENTED\_YET, sonst OSMF\_FAILURE

#### 2.4.1.10 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. Dies bedeutet, dass der Posteingang gesperrt wird und alle noch vorhandenen Nachrichten werden gelöscht.

##### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

#### 2.4.1.11 OSMP\_GetShmName()

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

#### 2.4.1.12 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. Durch diesen Aufruf wird außerdem der Posteingang des Prozesses freigegeben.

##### Parameter

in	<i>argc</i>	Adresse der Argumentzahl
in	<i>argv</i>	Adresse des Argumentvektors

##### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.4.1.13 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).

#### Parameter

out	<i>buf</i>	Startadresse des Speicherbereichs, wo die zu empfangende Nachricht gespeichert werden soll.
in	<i>count</i>	Zahl der Elemente vom angegebenen Typ, die empfangen werden können
in	<i>datatype</i>	OSMP-Typ der Daten im Puffer
out	<i>source</i>	PID des Senders zwischen 0, ..., np-1
out	<i>len</i>	tatsächliche Länge der empfangenen Nachricht in Byte
in, out	<i>request</i>	Adresse einer Datenstruktur, die später verwendet werden kann, um abzufragen, ob die die Operation abgeschlossen ist.

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.4.1.14 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).

#### Parameter

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



**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.4.1.15 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.

**Parameter**

out	<i>rank</i>	Prozessnummer <code>0, ..., np-1</code> des aktuellen OSMP-Prozesse
-----	-------------	---

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.4.1.16 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.

**Parameter**

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

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.4.1.17 OSMP\_RemoveRequest()**

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

Löscht eine `OSMP_Request`.

**Parameter**

in	<i>request</i>	Adresse eines Requests
----	----------------	------------------------

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

**2.4.1.18 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.

**Parameter**

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

**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.4.1.19 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.

#### Parameter

out	<i>rank</i>	Zahl der OSMP-Prozesse
-----	-------------	------------------------

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.4.1.20 OSMP\_Test()

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

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

#### Parameter

in	<i>request</i>	Adresse der Struktur, die eine blockierende Operation spezifiziert
out	<i>flag</i>	Gibt den Status der Operation an.

#### Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

### 2.4.1.21 OSMP\_Wait()

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

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

**Parameter**

<code>in</code>	<code>request</code>	Adresse der Struktur, die eine nicht blockierende Operation spezifiziert
-----------------	----------------------	--

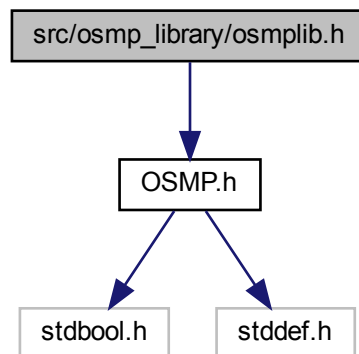
**Rückgabe**

Im Erfolgsfall `OSMP_SUCCESS`, falls die Funktion noch nicht implementiert ist `OSMP_NOT_IMPLEMENTED_YET`, sonst `OSMP_FAILURE`

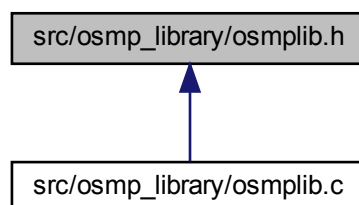
## 2.5 src/osmp\_library/osmplib.h-Dateireferenz

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

Include-Abhängigkeitsdiagramm für `osmplib.h`:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:

**Makrodefinitionen**

- `#define UNUSED(x) { (void)(x); }`

## 2.5.1 Makro-Dokumentation

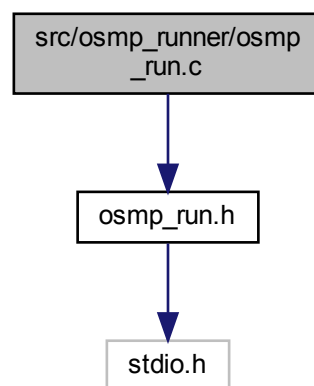
### 2.5.1.1 UNUSED

```
#define UNUSED(  
    x ) { (void) (x); }
```

## 2.6 src/osmp\_runner/osmp\_run.c-Dateireferenz

```
#include "osmp_run.h"
```

Include-Abhängigkeitsdiagramm für osmp\_run.c:



## Funktionen

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

## 2.6.1 Dokumentation der Funktionen

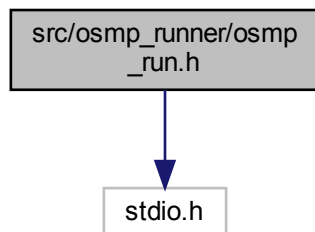
### 2.6.1.1 main()

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

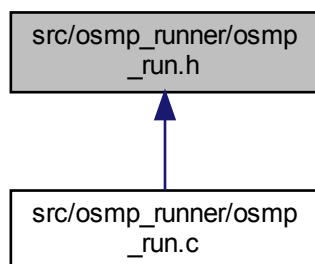
## 2.7 src/osmp\_runner/osmp\_run.h-Dateireferenz

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

Include-Abhängigkeitsdiagramm für osmp\_run.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



# Index

- enum\_OSM\_P\_Datatype
  - OSMP.h, [7](#)
- get\_OSM\_P\_FAILURE
  - OSMP.h, [8](#)
  - osmplib.c, [17](#)
- get\_OSM\_P\_MAX\_MESSAGES\_PROC
  - OSMP.h, [8](#)
  - osmplib.c, [17](#)
- get\_OSM\_P\_MAX\_PAYLOAD\_LENGTH
  - OSMP.h, [8](#)
  - osmplib.c, [17](#)
- get\_OSM\_P\_MAX\_SLOTS
  - OSMP.h, [8](#)
  - osmplib.c, [17](#)
- get\_OSM\_P\_NOT\_IMPLEMENTED\_YET
  - OSMP.h, [8](#)
  - osmplib.c, [17](#)
- get\_OSM\_P\_SUCCESS
  - OSMP.h, [9](#)
  - osmplib.c, [17](#)
- main
  - osmp\_run.c, [25](#)
  - osmpExecutable\_SendRecv.c, [3](#)
  - osmpExecutable\_SendRecv.c, [4](#)
- OSMP.h
  - enum\_OSM\_P\_Datatype, [7](#)
  - get\_OSM\_P\_FAILURE, [8](#)
  - get\_OSM\_P\_MAX\_MESSAGES\_PROC, [8](#)
  - get\_OSM\_P\_MAX\_PAYLOAD\_LENGTH, [8](#)
  - get\_OSM\_P\_MAX\_SLOTS, [8](#)
  - get\_OSM\_P\_NOT\_IMPLEMENTED\_YET, [8](#)
  - get\_OSM\_P\_SUCCESS, [9](#)
  - OSMP\_Barrier, [9](#)
  - OSMP\_BYTE, [8](#)
  - OSMP\_CreateRequest, [9](#)
  - OSMP\_Datatype, [7](#)
  - OSMP\_DOUBLE, [8](#)
  - OSMP\_FAILURE, [6](#)
  - OSMP\_Finalize, [9](#)
  - OSMP\_FLOAT, [8](#)
  - OSMP\_Gather, [10](#)
  - OSMP\_GetSharedMemoryName, [10](#)
  - OSMP\_Init, [11](#)
  - OSMP\_INT, [8](#)
  - OSMP\_IRecv, [11](#)
  - OSMP\_ISend, [12](#)
  - OSMP\_LONG, [8](#)
  - OSMP\_MAX\_MESSAGES\_PROC, [6](#)
  - OSMP\_MAX\_PAYLOAD\_LENGTH, [6](#)
  - OSMP\_MAX\_SLOTS, [7](#)
  - OSMP\_NOT\_IMPLEMENTED\_YET, [7](#)
  - OSMP\_Rank, [12](#)
  - OSMP\_Recv, [13](#)
  - OSMP\_RemoveRequest, [13](#)
  - OSMP\_Request, [7](#)
  - OSMP\_Send, [14](#)
  - OSMP\_SHORT, [8](#)
  - OSMP\_Size, [14](#)
  - OSMP\_SizeOf, [15](#)
  - OSMP\_SUCCESS, [7](#)
  - OSMP\_Test, [15](#)
  - OSMP\_UNSIGNED, [8](#)
  - OSMP\_UNSIGNED\_CHAR, [8](#)
  - OSMP\_UNSIGNED\_LONG, [8](#)
  - OSMP\_UNSIGNED\_SHORT, [8](#)
  - OSMP\_Wait, [15](#)
- OSMP\_Barrier
  - OSMP.h, [9](#)
  - osmplib.c, [18](#)
- OSMP\_Bcast
  - osmplib.c, [18](#)
- OSMP\_BYTE
  - OSMP.h, [8](#)
- OSMP\_CreateRequest
  - OSMP.h, [9](#)
  - osmplib.c, [18](#)
- OSMP\_Datatype
  - OSMP.h, [7](#)
- OSMP\_DOUBLE
  - OSMP.h, [8](#)
- OSMP\_FAILURE
  - OSMP.h, [6](#)
- OSMP\_Finalize
  - OSMP.h, [9](#)
  - osmplib.c, [19](#)
- OSMP\_FLOAT
  - OSMP.h, [8](#)
- OSMP\_Gather
  - OSMP.h, [10](#)
- OSMP\_GetSharedMemoryName
  - OSMP.h, [10](#)
- OSMP\_GetShmName
  - osmplib.c, [19](#)
- OSMP\_Init
  - OSMP.h, [11](#)
  - osmplib.c, [19](#)

OSMP\_INT  
     OSMP.h, [8](#)  
 OSMP\_IRecv  
     OSMP.h, [11](#)  
     osmplib.c, [19](#)  
 OSMP\_ISend  
     OSMP.h, [12](#)  
     osmplib.c, [20](#)  
 OSMP\_LONG  
     OSMP.h, [8](#)  
 OSMP\_MAX\_MESSAGES\_PROC  
     OSMP.h, [6](#)  
 OSMP\_MAX\_PAYLOAD\_LENGTH  
     OSMP.h, [6](#)  
 OSMP\_MAX\_SLOTS  
     OSMP.h, [7](#)  
 OSMP\_NOT\_IMPLEMENTED\_YET  
     OSMP.h, [7](#)  
 OSMP\_Rank  
     OSMP.h, [12](#)  
     osmplib.c, [21](#)  
 OSMP\_Recv  
     OSMP.h, [13](#)  
     osmplib.c, [21](#)  
 OSMP\_RemoveRequest  
     OSMP.h, [13](#)  
     osmplib.c, [22](#)  
 OSMP\_Request  
     OSMP.h, [7](#)  
 osmp\_run.c  
     main, [25](#)  
 OSMP\_Send  
     OSMP.h, [14](#)  
     osmplib.c, [22](#)  
 OSMP\_SHORT  
     OSMP.h, [8](#)  
 OSMP\_Size  
     OSMP.h, [14](#)  
     osmplib.c, [22](#)  
 OSMP\_SizeOf  
     OSMP.h, [15](#)  
 OSMP\_SUCCESS  
     OSMP.h, [7](#)  
 OSMP\_Test  
     OSMP.h, [15](#)  
     osmplib.c, [23](#)  
 OSMP\_UNSIGNED  
     OSMP.h, [8](#)  
 OSMP\_UNSIGNED\_CHAR  
     OSMP.h, [8](#)  
 OSMP\_UNSIGNED\_LONG  
     OSMP.h, [8](#)  
 OSMP\_UNSIGNED\_SHORT  
     OSMP.h, [8](#)  
 OSMP\_Wait  
     OSMP.h, [15](#)  
     osmplib.c, [23](#)  
 osmpExecutable\_SendIRecv.c  
     main, [3](#)  
 osmpExecutable\_SendRecv.c  
     main, [4](#)  
 osmplib.c  
     get\_OSMF\_FAILURE, [17](#)  
     get\_OSMF\_MAX\_MESSAGES\_PROC, [17](#)  
     get\_OSMF\_MAX\_PAYLOAD\_LENGTH, [17](#)  
     get\_OSMF\_MAX\_SLOTS, [17](#)  
     get\_OSMF\_NOT\_IMPLEMENTED\_YET, [17](#)  
     get\_OSMF\_SUCCESS, [17](#)  
     OSMP\_Barrier, [18](#)  
     OSMP\_Bcast, [18](#)  
     OSMP\_CreateRequest, [18](#)  
     OSMP\_Finalize, [19](#)  
     OSMP\_GetShmName, [19](#)  
     OSMP\_Init, [19](#)  
     OSMP\_IRecv, [19](#)  
     OSMP\_ISend, [20](#)  
     OSMP\_Rank, [21](#)  
     OSMP\_Recv, [21](#)  
     OSMP\_RemoveRequest, [22](#)  
     OSMP\_Send, [22](#)  
     OSMP\_Size, [22](#)  
     OSMP\_Test, [23](#)  
     OSMP\_Wait, [23](#)  
 osmplib.h  
     UNUSED, [25](#)  
  
 src/osmp\_executables/osmpExecutable\_SendIRecv.c, [3](#)  
 src/osmp\_executables/osmpExecutable\_SendRecv.c, [4](#)  
 src/osmp\_library/OSMP.h, [5](#)  
 src/osmp\_library/osmplib.c, [16](#)  
 src/osmp\_library/osmplib.h, [24](#)  
 src/osmp\_runner/osmp\_run.c, [25](#)  
 src/osmp\_runner/osmp\_run.h, [26](#)  
  
 UNUSED  
     osmplib.h, [25](#)