

OSMP - Entwurf und Implementierung einer Message Passing Umgebung für
Interprozesskommunikation

Erzeugt von Doxygen 1.9.1

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

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

Kapitel 1

Datei-Verzeichnis

1.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

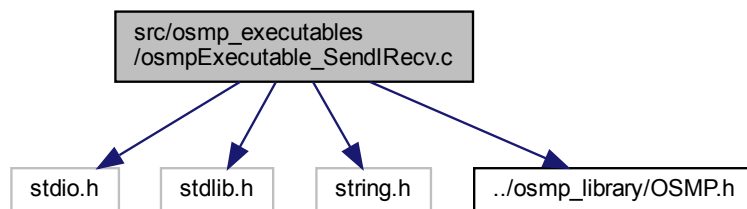
src/osmp_executables/ osmpExecutable_SendRecv.c	3
src/osmp_executables/ osmpExecutable_SendRecv.c	4
src/osmp_library/ OSMP.h	4
src/osmp_library/ osmplib.c	15
src/osmp_library/ osmplib.h	23
src/osmp_runner/ osmp_run.c	24
src/osmp_runner/ osmp_run.h	25

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[])

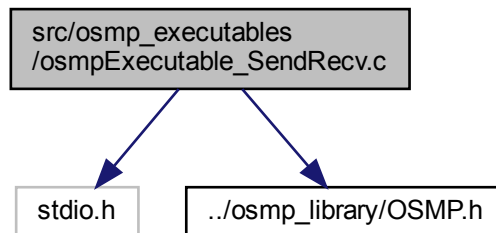
2.1.1 Dokumentation der Funktionen

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[])

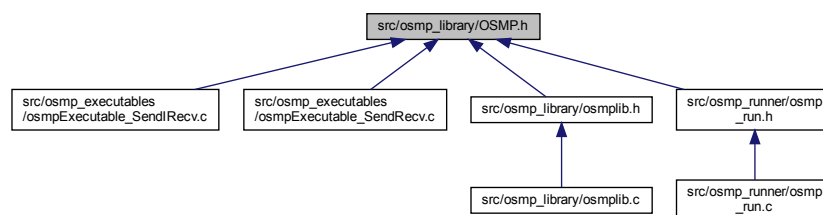
2.2.1 Dokumentation der Funktionen

2.2.1.1 main()

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

2.3 src/osmp_library/OSMP.h-Dateireferenz

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



Makrodefinitionen

- `#define OSMP_SUCCESS 0`
- `#define OSMP_FAILURE (!OSMP_SUCCESS)`
- `#define OSMP_MAX_MESSAGES_PROC 16`
- `#define OSMP_MAX_SLOTS 256`
- `#define OSMP_MAX_PAYLOAD_LENGTH 1024`

Typdefinitionen

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

Aufzählungen

- `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 }`

Funktionen

- `int get_OSMP_MAX_PAYLOAD_LENGTH ()`
- `int get_OSMP_MAX_SLOTS ()`
- `int get_OSMP_MAX_MESSAGES_PROC ()`
- `int get_OSMP_FAILURE ()`
- `int get_OSMP_SUCCESS ()`
- `int OSMP_SizeOf (OSMP_Datatype datatype, unsigned int *size)`
- `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, 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 OSMP_FAILURE

```
#define OSMP_FAILURE ( !OSMP_SUCCESS )
```

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).

2.3.1.2 OSMP_MAX_MESSAGES_PROC

```
#define OSMP_MAX_MESSAGES_PROC 16
```

Die maximale Zahl der Nachrichten pro Prozess

2.3.1.3 OSMP_MAX_PAYLOAD_LENGTH

```
#define OSMP_MAX_PAYLOAD_LENGTH 1024
```

Die maximale Länge der Nutzlast einer Nachricht

2.3.1.4 OSMP_MAX_SLOTS

```
#define OSMP_MAX_SLOTS 256
```

Die maximale Anzahl der Nachrichten, die insgesamt vorhanden sein dürfen

2.3.1.5 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 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.

2.3.2.2 OSMP_Request

```
typedef void* OSMP_Request
```

2.3.3 Dokumentation der Aufzählungstypen

2.3.3.1 OSMP_Datatype

```
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.

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_OSMP_FAILURE()**

```
int get_OSMP_FAILURE ( )
```

Gibt den Wert von OSMP_FAILURE zurück.

2.3.4.2 get_OSMP_MAX_MESSAGES_PROC()

```
int get_OSMP_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

2.3.4.3 get_OSMP_MAX_PAYLOAD_LENGTH()

```
int get_OSMP_MAX_PAYLOAD_LENGTH ( )
```

Gibt die maximale Länge der Nutzlast einer Nachricht zurück.

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

2.3.4.4 get_OSMP_MAX_SLOTS()

```
int get_OSMP_MAX_SLOTS ( )
```

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

2.3.4.5 get_OSMF_SUCCESS()

```
int get_OSMF_SUCCESS ( )
```

Gibt den Wert von OSMF_SUCCESS zurück.

2.3.4.6 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, sonst OSMF_FAILURE

2.3.4.7 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, sonst OSMF_FAILURE

2.3.4.8 OSMF_Finalize()

```
int OSMF_Finalize (
    void )
```

Alle OSMF-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 gelöscht werden.

Rückgabe

Im Erfolgsfall OSMF_SUCCESS, sonst OSMF_FAILURE

2.3.4.9 OSMP_Gather()

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

Diese Funktion ermöglicht die Gather-Kommunikation. Hierbei können mehrere Prozesse an einen Empfänger Prozess Daten schicken.

Parameter

in	<i>sendbuf</i>	Zeiger auf den Sendepuffer.
in	<i>sendcount</i>	Anzahl der Elemente im Sendepuffer.
in	<i>sendtype</i>	OSMP-Datentyp der Elemente im Sendepuffer.
out	<i>recvbuf</i>	Zeiger auf den Empfangspuffer.
in	<i>recvcount</i>	Anzahl der Elemente im Empfangspuffer.
in	<i>recvtype</i>	OSMP-Datentyp der Elemente im Empfangspuffer.
in	<i>recv</i>	1, falls der aufrufende Prozess der Empfänger ist, sonst 0.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.10 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, sonst OSMP_FAILURE

2.3.4.11 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, sonst OSMP_FAILURE

2.3.4.12 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, sonst OSMP_FAILURE

2.3.4.13 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, sonst OSMP_FAILURE

2.3.4.14 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 0,...,np-1 des aktuellen OSMP-Prozesse
-----	-------------	--

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.3.4.15 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 0, ... ,np-1
out	<i>len</i>	tatsächliche Länge der empfangenen Nachricht in Byte

Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, sonst `OSMP_FAILURE`

2.3.4.16 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`, sonst `OSMP_FAILURE`

2.3.4.17 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`, sonst `OSMP_FAILURE`

2.3.4.18 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`, sonst `OSMP_FAILURE`

2.3.4.19 OSMP_SizeOf()

```
int OSMP_SizeOf (
    OSMP_Datatype datatype,
    unsigned int * size )
```

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

Parameter

in	<i>datatype</i>	OSMP-Datentyp
out	<i>size</i>	Größe des Datentyps in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS; falls der OSMP_Datatype nicht existiert, OSMP_FAILURE

2.3.4.20 OSMP_Test()

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

Die Funktion testet, ob die mit der Request verknüpften 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, sonst OSMP_FAILURE

2.3.4.21 OSMP_Wait()

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

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

Parameter

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

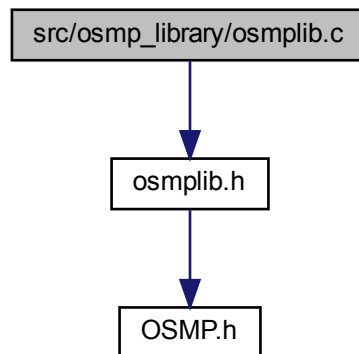
Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.4 src/osmp_library/osmplib.c-Dateireferenz

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

Include-Abhängigkeitsdiagramm für osmplib.c:



Funktionen

- int [get_OSMF_MAX_PAYLOAD_LENGTH](#) ()
- int [get_OSMF_MAX_SLOTS](#) ()
- int [get_OSMF_MAX_MESSAGES_PROC](#) ()
- int [get_OSMF_FAILURE](#) ()
- int [get_OSMF_SUCCESS](#) ()
- int [OSMF_Init](#) (const int *argc, char ***argv)
- int [OSMF_SizeOf](#) ([OSMF_Datatype](#) datatype, unsigned int *size)
- int [OSMF_Size](#) (int *size)
- int [OSMF_Rank](#) (int *rank)
- int [OSMF_Send](#) (const void *buf, int count, [OSMF_Datatype](#) datatype, int dest)
- int [OSMF_Recv](#) (void *buf, int count, [OSMF_Datatype](#) datatype, int *source, int *len)
- int [OSMF_Finalize](#) (void)
- int [OSMF_Barrier](#) (void)
- int [OSMF_Gather](#) (void *sendbuf, int sendcount, [OSMF_Datatype](#) sendtype, void *recvbuf, int recvcount, [OSMF_Datatype](#) recvtype, int recv)
- int [OSMF_ISend](#) (const void *buf, int count, [OSMF_Datatype](#) datatype, int dest, [OSMF_Request](#) request)
- int [OSMF_IRecv](#) (void *buf, int count, [OSMF_Datatype](#) datatype, int *source, int *len, [OSMF_Request](#) request)
- int [OSMF_Test](#) ([OSMF_Request](#) request, int *flag)
- int [OSMF_Wait](#) ([OSMF_Request](#) request)
- int [OSMF_CreateRequest](#) ([OSMF_Request](#) *request)
- int [OSMF_RemoveRequest](#) ([OSMF_Request](#) *request)
- int [OSMF_GetSharedMemoryName](#) (char **name)

2.4.1 Dokumentation der Funktionen

2.4.1.1 `get_OSMF_FAILURE()`

```
int get_OSMF_FAILURE ( )
```

Gibt den Wert von `OSMP_FAILURE` zurück.

2.4.1.2 `get_OSMF_MAX_MESSAGES_PROC()`

```
int get_OSMF_MAX_MESSAGES_PROC ( )
```

Gibt die maximale Zahl der Nachrichten pro Prozess zurück.

2.4.1.3 `get_OSMF_MAX_PAYLOAD_LENGTH()`

```
int get_OSMF_MAX_PAYLOAD_LENGTH ( )
```

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

2.4.1.4 `get_OSMF_MAX_SLOTS()`

```
int get_OSMF_MAX_SLOTS ( )
```

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

2.4.1.5 `get_OSMF_SUCCESS()`

```
int get_OSMF_SUCCESS ( )
```

Gibt den Wert von `OSMP_SUCCESS` zurück.

2.4.1.6 `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`, sonst `OSMP_FAILURE`

2.4.1.7 `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, sonst OSMP_FAILURE

2.4.1.8 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 gelöscht werden.

Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

2.4.1.9 OSMP_Gather()

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

Diese Funktion ermöglicht die Gather-Kommunikation. Hierbei können mehrere Prozesse an einen Empfänger Prozess Daten schicken.

Parameter

in	<i>sendbuf</i>	Zeiger auf den Sendepuffer.
in	<i>sendcount</i>	Anzahl der Elemente im Sendepuffer.
in	<i>sendtype</i>	OSMP-Datentyp der Elemente im Sendepuffer.
out	<i>recvbuf</i>	Zeiger auf den Empfangspuffer.
in	<i>recvcount</i>	Anzahl der Elemente im Empfangspuffer.
in	<i>recvtype</i>	OSMP-Datentyp der Elemente im Empfangspuffer.
in	<i>recv</i>	1, falls der aufrufende Prozess der Empfänger ist, sonst 0.

Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, sonst `OSMP_FAILURE`

2.4.1.10 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`, sonst `OSMP_FAILURE`

2.4.1.11 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`, sonst `OSMP_FAILURE`

2.4.1.12 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, sonst OSMP_FAILURE

2.4.1.13 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, sonst OSMP_FAILURE

2.4.1.14 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 0,...,np-1 des aktuellen OSMP-Prozesse
-----	-------------	--

Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, sonst `OSMP_FAILURE`

2.4.1.15 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 0,...,np-1
out	<i>len</i>	tatsächliche Länge der empfangenen Nachricht in Byte

Rückgabe

Im Erfolgsfall `OSMP_SUCCESS`, sonst `OSMP_FAILURE`

2.4.1.16 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, sonst OSMP_FAILURE

2.4.1.17 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, sonst OSMP_FAILURE

2.4.1.18 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, sonst OSMP_FAILURE

2.4.1.19 OSMP_SizeOf()

```
int OSMP_SizeOf (
    OSMP_Datatype datatype,
    unsigned int * size )
```

Die Funktion [OSMP_SizeOf\(\)](#) liefert in *size* die Größe des Datentyps *datatype* in Byte zurück.

Parameter

in	<i>datatype</i>	OSMP-Datentyp
out	<i>size</i>	Größe des Datentyps in Byte

Rückgabe

Im Erfolgsfall OSMP_SUCCESS; falls der OSMP_Datatype nicht existiert, OSMP_FAILURE

2.4.1.20 OSMP_Test()

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

Die Funktion testet, ob die mit der Request verknüpften 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, sonst OSMP_FAILURE

2.4.1.21 OSMP_Wait()

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

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

Parameter

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

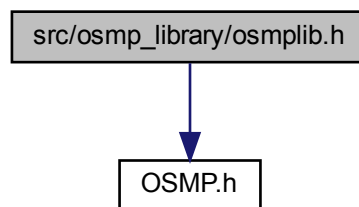
Rückgabe

Im Erfolgsfall OSMP_SUCCESS, sonst OSMP_FAILURE

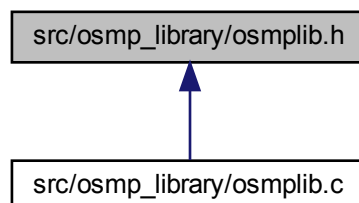
2.5 src/osmp_library/osmplib.h-Dateireferenz

```
#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

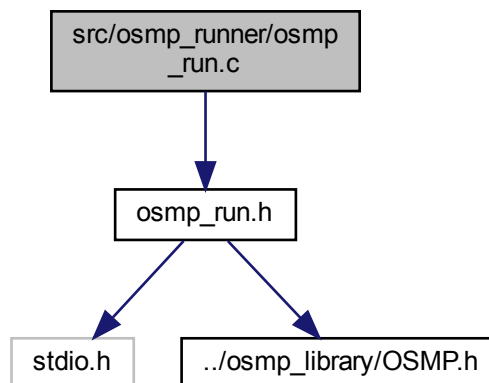
```
#define UNUSED(  
    x ) { (void) (x); }
```

Dieses Makro wird verwendet, um den Compiler davon zu überzeugen, dass eine Variable verwendet wird.

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

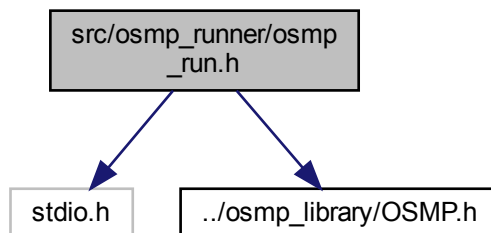
2.6.1.1 main()

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

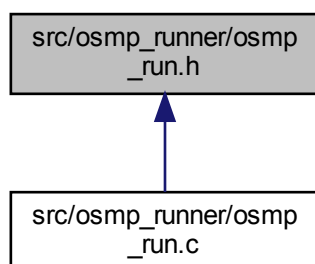
2.7 src/osmp_runner/osmp_run.h-Dateireferenz

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

Include-Abhängigkeitsdiagramm für osmp_run.h:



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



Index

get_OSMF_FAILURE
 OSMP.h, [7](#)
 osmplib.c, [15](#)
get_OSMF_MAX_MESSAGES_PROC
 OSMP.h, [7](#)
 osmplib.c, [16](#)
get_OSMF_MAX_PAYLOAD_LENGTH
 OSMP.h, [7](#)
 osmplib.c, [16](#)
get_OSMF_MAX_SLOTS
 OSMP.h, [7](#)
 osmplib.c, [16](#)
get_OSMF_SUCCESS
 OSMP.h, [7](#)
 osmplib.c, [16](#)

main
 osmp_run.c, [24](#)
 osmpExecutable_SendIRecv.c, [3](#)
 osmpExecutable_SendRecv.c, [4](#)

OSMP.h
 get_OSMF_FAILURE, [7](#)
 get_OSMF_MAX_MESSAGES_PROC, [7](#)
 get_OSMF_MAX_PAYLOAD_LENGTH, [7](#)
 get_OSMF_MAX_SLOTS, [7](#)
 get_OSMF_SUCCESS, [7](#)
 OSMP_Barrier, [8](#)
 OSMP_BYTE, [7](#)
 OSMP_CreateRequest, [8](#)
 OSMP_Datatype, [6](#)
 OSMP_DOUBLE, [7](#)
 OSMP_FAILURE, [5](#)
 OSMP_Finalize, [8](#)
 OSMP_FLOAT, [7](#)
 OSMP_Gather, [8](#)
 OSMP_GetSharedMemoryName, [9](#)
 OSMP_Init, [9](#)
 OSMP_INT, [7](#)
 OSMP_IRecv, [10](#)
 OSMP_ISend, [10](#)
 OSMP_LONG, [7](#)
 OSMP_MAX_MESSAGES_PROC, [6](#)
 OSMP_MAX_PAYLOAD_LENGTH, [6](#)
 OSMP_MAX_SLOTS, [6](#)
 OSMP_Rank, [11](#)
 OSMP_Recv, [11](#)
 OSMP_RemoveRequest, [12](#)
 OSMP_Request, [6](#)
 OSMP_Send, [12](#)

 OSMP_SHORT, [7](#)
 OSMP_Size, [13](#)
 OSMP_SizeOf, [13](#)
 OSMP_SUCCESS, [6](#)
 OSMP_Test, [14](#)
 OSMP_UNSIGNED, [7](#)
 OSMP_UNSIGNED_CHAR, [7](#)
 OSMP_UNSIGNED_LONG, [7](#)
 OSMP_UNSIGNED_SHORT, [7](#)
 OSMP_Wait, [14](#)
OSMP_Barrier
 OSMP.h, [8](#)
 osmplib.c, [16](#)
OSMP_BYTE
 OSMP.h, [7](#)
OSMP_CreateRequest
 OSMP.h, [8](#)
 osmplib.c, [16](#)
OSMP_Datatype
 OSMP.h, [6](#)
OSMP_DOUBLE
 OSMP.h, [7](#)
OSMP_FAILURE
 OSMP.h, [5](#)
OSMP_Finalize
 OSMP.h, [8](#)
 osmplib.c, [17](#)
OSMP_FLOAT
 OSMP.h, [7](#)
OSMP_Gather
 OSMP.h, [8](#)
 osmplib.c, [17](#)
OSMP_GetSharedMemoryName
 OSMP.h, [9](#)
 osmplib.c, [18](#)
OSMP_Init
 OSMP.h, [9](#)
 osmplib.c, [18](#)
OSMP_INT
 OSMP.h, [7](#)
OSMP_IRecv
 OSMP.h, [10](#)
 osmplib.c, [18](#)
OSMP_ISend
 OSMP.h, [10](#)
 osmplib.c, [19](#)
OSMP_LONG
 OSMP.h, [7](#)
OSMP_MAX_MESSAGES_PROC

- OSMP.h, [6](#)
- OSMP_MAX_PAYLOAD_LENGTH
 - OSMP.h, [6](#)
- OSMP_MAX_SLOTS
 - OSMP.h, [6](#)
- OSMP_Rank
 - OSMP.h, [11](#)
 - osmplib.c, [19](#)
- OSMP_Recv
 - OSMP.h, [11](#)
 - osmplib.c, [20](#)
- OSMP_RemoveRequest
 - OSMP.h, [12](#)
 - osmplib.c, [20](#)
- OSMP_Request
 - OSMP.h, [6](#)
- osmp_run.c
 - main, [24](#)
- OSMP_Send
 - OSMP.h, [12](#)
 - osmplib.c, [21](#)
- OSMP_SHORT
 - OSMP.h, [7](#)
- OSMP_Size
 - OSMP.h, [13](#)
 - osmplib.c, [21](#)
- OSMP_SizeOf
 - OSMP.h, [13](#)
 - osmplib.c, [22](#)
- OSMP_SUCCESS
 - OSMP.h, [6](#)
- OSMP_Test
 - OSMP.h, [14](#)
 - osmplib.c, [22](#)
- OSMP_UNSIGNED
 - OSMP.h, [7](#)
- OSMP_UNSIGNED_CHAR
 - OSMP.h, [7](#)
- OSMP_UNSIGNED_LONG
 - OSMP.h, [7](#)
- OSMP_UNSIGNED_SHORT
 - OSMP.h, [7](#)
- OSMP_Wait
 - OSMP.h, [14](#)
 - osmplib.c, [22](#)
- osmpExecutable_SendIRecv.c
 - main, [3](#)
- osmpExecutable_SendRecv.c
 - main, [4](#)
- osmplib.c
 - get_OSMF_FAILURE, [15](#)
 - get_OSMF_MAX_MESSAGES_PROC, [16](#)
 - get_OSMF_MAX_PAYLOAD_LENGTH, [16](#)
 - get_OSMF_MAX_SLOTS, [16](#)
 - get_OSMF_SUCCESS, [16](#)
 - OSMP_Barrier, [16](#)
 - OSMP_CreateRequest, [16](#)
 - OSMP_Finalize, [17](#)
 - OSMP_Gather, [17](#)
 - OSMP_GetSharedMemoryName, [18](#)
 - OSMP_Init, [18](#)
 - OSMP_IRecv, [18](#)
 - OSMP_ISend, [19](#)
 - OSMP_Rank, [19](#)
 - OSMP_Recv, [20](#)
 - OSMP_RemoveRequest, [20](#)
 - OSMP_Send, [21](#)
 - OSMP_Size, [21](#)
 - OSMP_SizeOf, [22](#)
 - OSMP_Test, [22](#)
 - OSMP_Wait, [22](#)
- osmplib.h
 - UNUSED, [24](#)
- src/osmp_executables/osmpExecutable_SendIRecv.c, [3](#)
- src/osmp_executables/osmpExecutable_SendRecv.c, [4](#)
- src/osmp_library/OSMP.h, [4](#)
- src/osmp_library/osmplib.c, [15](#)
- src/osmp_library/osmplib.h, [23](#)
- src/osmp_runner/osmp_run.c, [24](#)
- src/osmp_runner/osmp_run.h, [25](#)
- UNUSED
 - osmplib.h, [24](#)