SoCPlatform

v1.0

Generated by Doxygen 1.11.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

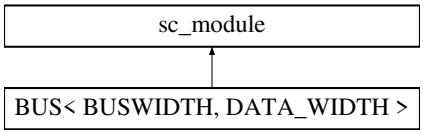
Chapter 4

Class Documentation

4.1 BUS< BUSWIDTH, DATA_WIDTH >

`#include "bus.h"`

Inheritance diagram for BUS< BUSWIDTH, DATA_WIDTH >:



Classes

- struct **address**

Public Member Functions

- **BUS** (sc_core::sc_module_name name, bool message=false)

  BUS *(p. 7) constructor.*
- ~**BUS** ()
- tlm_utils::multi_passthrough_initiator_socket< **BUS**, BUSWIDTH > & **mapping_target_sockets** (unsigned int _addr, unsigned int _size)

  mapping_target_sockets Implement the registration socket address range for target socket

Public Attributes

- **sc_clk_in m_clk**
- sc_core::sc_in< bool > **m_rst**
- tlm_utils::multi_passthrough_target_socket< **BUS**, BUSWIDTH > **target_sockets**

Private Member Functions

- void **copy_tlm_generic_payload** (tlm::tlm_generic_payload &des, tlm::tlm_generic_payload &src)

  copy_tlm_generic_payload Impelmentation the copy operation from source TLM generic payload to destination TLM generic payload
- void **foward_transaction_process** ()

  foward_transaction_process Implementation the thread to synchronize with clock cycles and forward the transaction to slave through the corresponding initiator
- void **mth_reset** ()

  mth_reset Impelmentation of the method when reset is active
- tlm::tlm_sync_enum **nb_transport_bw** (int id, tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc↩ _core::sc_time &delay)

  nb_transport_bw Implements the non-blocking backward transport interface for the nitiator.
- tlm::tlm_sync_enum **nb_transport_fw** (int id, tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc↩ _core::sc_time &delay)

  nb_transport_fw Implements the non-blocking forward transport interface for the target.
- void **TS_handle_begin_req** (int id, tlm::tlm_generic_payload &trans, sc_core::sc_time &delay)

  TS_handle_begin_req Implementation for decoding address from transaction payload and selecting the suitable initiator socket with the corresponding id.

Private Attributes

- std::vector< **address** > **address_mapping**
- tlm::tlm_generic_payload **current_trans**
- sc_core::sc_event **e_forward_tran**
- tlm_utils::multi_passthrough_initiator_socket< **BUS**, BUSWIDTH > **initiator_sockets**
- unsigned int **m_bind_id**
- bool **m_bus_lock**
- unsigned int **m_cur_socket**
- unsigned int **m_current_ts_id**
- bool **m_message**
- std::string **m_name**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 BUS()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
BUS (
            sc_core::sc_module_name name,
            bool message = false)  [inline]
```

BUS (p. 7) constructor.

Parameters

name	Reference to sc_module name
message	To enable message log

References **BUS< BUSWIDTH, DATA_WIDTH >::e_forward_tran**, **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::initiator_sockets**, **BUS< BUSWIDTH, DATA_WIDTH >::m_rst**, **BUS< BUSWIDTH, DATA_WIDTH >::mth_reset()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**, and **BUS< BUSWIDTH, DATA_WIDTH >::target_sockets**.

4.1.1.2 ~BUS()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
~ BUS ()  [inline]
```

4.1.2 Member Function Documentation

4.1.2.1 copy_tlm_generic_payload()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
void copy_tlm_generic_payload (
             tlm::tlm_generic_payload & des,
             tlm::tlm_generic_payload & src)  [inline], [private]
```

copy_tlm_generic_payload Impelmentation the copy operation from source TLM generic payload to destination TLM generic payload

Parameters

des	Reference to destination TLM generic payload
src	Reference to source TLM generic payload

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

4.1.2.2 foward_transaction_process()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
void foward_transaction_process ()  [inline], [private]
```

foward_transaction_process Implementation the thread to synchronize with clock cycles and forward the transaction to slave through the corresponding initiator

References **BUS< BUSWIDTH, DATA_WIDTH >::address_mapping**, **BUS< BUSWIDTH, DATA_WIDTH >::current_trans**, **BUS< BUSWIDTH, DATA_WIDTH >::e_forward_tran**, **BUS< BUSWIDTH, DATA_WIDTH >::initiator_sockets**, **BUS< BUSWIDTH, DATA_WIDTH >::m_clk**, **BUS< BUSWIDTH, DATA_WIDTH >::m_cur_socket**, **BUS< BUSWIDTH, DATA_WIDTH >::m_message**, and **BUS< BUSWIDTH, DATA_WIDTH >::m_name**.

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**.

4.1.2.3 mapping_target_sockets()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
tlm_utils::multi_passthrough_initiator_socket<  BUS, BUSWIDTH > & mapping_target_sockets (
             unsigned int _addr,
             unsigned int _size)  [inline]
```

mapping_target_sockets Implement the registration socket address range for target socket

Parameters

_id	The id number of target socket in bus memory mapping I/O
_addr	The base address of target socket that is registered into bus memory mapping I/O
_size	the range of address space

Returns

tlm_utils::simple_initiator_socket is the initiator socket with id registration used to bind with the corresponding target socket

References **BUS< BUSWIDTH, DATA_WIDTH >::address_mapping**, **BUS< BUSWIDTH, DATA_WIDTH >::initiator_sockets**, and **BUS< BUSWIDTH, DATA_WIDTH >::m_bind_id**.

4.1.2.4 mth_reset()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
void mth_reset ()  [inline], [private]
```

mth_reset Impelmentation of the method when reset is active

References **BUS< BUSWIDTH, DATA_WIDTH >::current_trans**, **BUS< BUSWIDTH, DATA_WIDTH >::e_forward_tran**, **BUS< BUSWIDTH, DATA_WIDTH >::m_bus_lock**, **BUS< BUSWIDTH, DATA_WIDTH >::m_cur_socket**, and **BUS< BUSWIDTH, DATA_WIDTH >::m_rst**.

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**.

4.1.2.5 nb_transport_bw()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
tlm::tlm_sync_enum nb_transport_bw (
             int id,
             tlm::tlm_generic_payload & trans,
             tlm::tlm_phase & phase,
             sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_bw Implements the non-blocking backward transport interface for the nitiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **BUS< BUSWIDTH, DATA_WIDTH >::m_bus_lock**, **BUS< BUSWIDTH, DATA_WIDTH >::m_cur_socket**, **BUS< BUSWIDTH, DATA_WIDTH >::m_current_ts_id**, **BUS< BUSWIDTH, DATA_WIDTH >::m_message**, **BUS< BUSWIDTH, DATA_WIDTH >::m_name**, and **BUS< BUSWIDTH, DATA_WIDTH >::target_sockets**.

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**.

4.1.2.6 nb_transport_fw()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
tlm::tlm_sync_enum nb_transport_fw (
             int id,
             tlm::tlm_generic_payload & trans,
             tlm::tlm_phase & phase,
             sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_fw Implements the non-blocking forward transport interface for the target.

Parameters

id	Integer identifier to distinguish between multiple initiators or channels.
trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **BUS< BUSWIDTH, DATA_WIDTH >::initiator_sockets**, **BUS< BUSWIDTH, DATA_WIDTH >::m_bus_lock**, **BUS< BUSWIDTH, DATA_WIDTH >::m_cur_socket**, **BUS< BUSWIDTH, DATA_WIDTH >::m_message**, **BUS< BUSWIDTH, DATA_WIDTH >::m_name**, and **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**.

4.1.2.7 TS_handle_begin_req()

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
void TS_handle_begin_req (
             int id,
             tlm::tlm_generic_payload & trans,
             sc_core::sc_time & delay)  [inline], [private]
```

TS_handle_begin_req Implementation for decoding address from transaction payload and selecting the suitable initiator socket with the corresponding id.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

References **BUS< BUSWIDTH, DATA_WIDTH >::address_mapping**, **BUS< BUSWIDTH, DATA_WIDTH >::copy_tlm_generic_payload()**, **BUS< BUSWIDTH, DATA_WIDTH >::current_trans**, **BUS< BUSWIDTH, DATA_WIDTH >::e_forward_tran**, **BUS< BUSWIDTH, DATA_WIDTH >::m_cur_socket**, and **BUS< BUSWIDTH, DATA_WIDTH >::m_current_ts_id**.

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**.

4.1.3 Member Data Documentation

4.1.3.1 address_mapping

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
std::vector< address> address_mapping [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::mapping_target_sockets()**, and **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

4.1.3.2 current_trans

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
tlm::tlm_generic_payload current_trans [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::mth_reset()**, and **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

4.1.3.3 e_forward_tran

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
sc_core::sc_event e_forward_tran [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**, **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::mth_reset()**, and **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

4.1.3.4 initiator_sockets

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
tlm_utils::multi_passthrough_initiator_socket< BUS, BUSWIDTH> initiator_sockets [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**, **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::mapping_target_sockets()**, and **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**.

4.1.3.5 m_bind_id

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int m_bind_id [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::mapping_target_sockets()**.

4.1.3.6 m_bus_lock

```
template<unsigned int BUSWIDTH = 32, DATAWIDTH DATA_WIDTH = D8BIT>
bool m_bus_lock [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::mth_reset()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, and **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**.

4.1.3.7 m_clk

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
sc_clk_in m_clk
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**.

4.1.3.8 m_cur_socket

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int m_cur_socket  [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::mth_reset()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**, and **BUS< BUSWIDTH, DATA_WIDTH >::TS_handle_begin_req()**.

4.1.3.9 m_current_ts_id

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int m_current_ts_id  [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, and **BUS< BUSWIDTH, DATA_↩ WIDTH >::TS_handle_begin_req()**.

4.1.3.10 m_message

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
bool m_message  [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, and **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**.

4.1.3.11 m_name

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
std::string m_name  [private]
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::foward_transaction_process()**, **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_bw()**, and **BUS< BUSWIDTH, DATA_WIDTH >::nb_transport_fw()**.

4.1.3.12 m_rst

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
sc_core::sc_in<bool> m_rst
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**, and **BUS< BUSWIDTH, DATA_WIDTH >::mth_↩ reset()**.

4.1.3.13 target_sockets

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
tlm_utils::multi_passthrough_target_socket< BUS, BUSWIDTH> target_sockets
```

Referenced by **BUS< BUSWIDTH, DATA_WIDTH >::BUS()**, and **BUS< BUSWIDTH, DATA_WIDTH >::nb_↩ transport_bw()**.

The documentation for this class was generated from the following file:

- common/ **bus.h**

4.2 BUS< BUSWIDTH, DATA_WIDTH >::address

Public Attributes

- unsigned int **addr**
- unsigned int **id**
- unsigned int **size**

4.2.1 Member Data Documentation

4.2.1.1 addr

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int addr
```

4.2.1.2 id

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int id
```

4.2.1.3 size

```
template<unsigned int BUSWIDTH = 32,  DATAWIDTH DATA_WIDTH = D8BIT>
unsigned int size
```

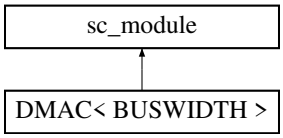
The documentation for this struct was generated from the following file:

- common/ **bus.h**

4.3 DMAC< BUSWIDTH >

`#include "DMAC.h"`

Inheritance diagram for DMAC< BUSWIDTH >:



Public Member Functions

- **DMAC** (sc_core::sc_module_name name, bool message=false)
  DMAC *(p. 15) constructor.*
- ~**DMAC** ()
  DMAC *(p. 15) destructor.*

Public Attributes

- sc_core::sc_in< bool > **clk**
- sc_core::sc_out< bool > **DMA_ack** [**DMA_MAX_CH**]
- sc_core::sc_out< bool > **DMA_int** [**DMA_MAX_CH**]
- sc_core::sc_in< bool > **DMA_req** [**DMA_MAX_CH**]
- tlm_utils::simple_initiator_socket< **DMAC**, BUSWIDTH > **initiator_socket**
- sc_core::sc_in< bool > **rst**
- tlm_utils::simple_target_socket< **DMAC**, BUSWIDTH > **target_socket**

Private Member Functions

- void **cb_DMAREQ** (const std::string &name, uint32_t value, uint32_t old_value, uint32_t mask, uint32_t ch)
  cb_DMAREQ
- void **copy_tlm_generic_payload** (tlm::tlm_generic_payload &des, tlm::tlm_generic_payload &src)
  copy_tlm_generic_payload
- void **init_registers** ()
  init_registers
- void **mth_request_signals** ()
  mth_request_signals
- void **mth_reset** ()
  mth_reset
- tlm::tlm_sync_enum **nb_transport_bw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core↩ ::sc_time &delay)
  nb_transport_bw
- tlm::tlm_sync_enum **nb_transport_fw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core↩ ::sc_time &delay)
  nb_transport_fw
- void **thr_DMA_forward_process** ()
  thr_DMA_forward_process
- void **thr_DMA_run_process** ()
  thr_DMA_run_process
- void **thr_priority_process** ()
  thr_priority_process

Private Attributes

- tlm::tlm_generic_payload **current_trans**
- sc_core::sc_event **e_DMA_forward**
- sc_core::sc_event **e_DMA_request**
- sc_core::sc_event **e_DMA_run**
- sc_core::sc_event **e_DMA_run_done**
- unsigned int **m_cur_reg_ch**
- std::string **m_cur_reg_name**
- unsigned int **m_current_ch**
- bool **m_message**
- std::string **m_name**
- bool **m_running**
- bool **m_testmode**
- std::list< unsigned int > **port_req_ids**
- **RegisterInterface regs**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 DMAC()

```
template<unsigned int BUSWIDTH = 32>
DMAC (
            sc_core::sc_module_name name,
            bool message = false)  [inline]
```

DMAC (p. 15) constructor.

Parameters

name	Reference to sc_module name
message	To enable message log

References **DMA_MAX_CH**, **DMAC< BUSWIDTH >::DMA_req**, **DMAC< BUSWIDTH >::e_DMA_forward**, **DMAC< BUSWIDTH >::e_DMA_request**, **DMAC< BUSWIDTH >::e_DMA_run**, **DMAC< BUSWIDTH >::init_registers()**, **DMAC< BUSWIDTH >::initiator_socket**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, **DMAC< BUSWIDTH >::nb_transport_fw()**, **DMAC< BUSWIDTH >::rst**, **DMAC< BUSWIDTH >::target_socket**, **DMAC< BUSWIDTH >::thr_DMA_forward_process()**, **DMAC< BUSWIDTH >::thr_DMA_run_process()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.1.2 ~DMAC()

```
template<unsigned int BUSWIDTH = 32>
~ DMAC ()  [inline]
```

DMAC (p. 15) destructor.

4.3.2 Member Function Documentation

4.3.2.1 cb_DMAREQ()

```
template<unsigned int BUSWIDTH = 32>
void cb_DMAREQ (
            const std::string & name,
            uint32_t value,
            uint32_t old_value,
            uint32_t mask,
            uint32_t ch)  [inline], [private]
```

cb_DMAREQ

The call back register function for DMAREQ register

References **DMAC< BUSWIDTH >::e_DMA_request**, **DMAC< BUSWIDTH >::m_cur_reg_ch**, **DMAC< BUSWIDTH >::m_cur_reg_name**, **DMAC< BUSWIDTH >::m_message**, **DMAC< BUSWIDTH >::m_name**, **DMAC< BUSWIDTH >::m_running**, and **DMAC< BUSWIDTH >::m_testmode**.

Referenced by **DMAC< BUSWIDTH >::init_registers()**.

4.3.2.2 copy_tlm_generic_payload()

```
template<unsigned int BUSWIDTH = 32>
void copy_tlm_generic_payload (
            tlm::tlm_generic_payload & des,
            tlm::tlm_generic_payload & src)  [inline], [private]
```

copy_tlm_generic_payload

Impelmentation the copy operation from source TLM generic payload to destination TLM generic payload

Parameters

des	Reference to destination TLM generic payload
src	Reference to source TLM generic payload

Referenced by **DMAC< BUSWIDTH >::nb_transport_bw()**.

4.3.2.3 init_registers()

```
template<unsigned int BUSWIDTH = 32>
void init_registers ()  [inline], [private]
```

init_registers

To initialize registers for **DMAC** (p. 15) model

References **RegisterInterface::add_register()**, **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMA_MAX_CH**, **DMAACK**, **DMACHEN**, **DMADATALENGTH**, **DMADESADDR**, **DMAINT**, **DMAREQ**, **DMASRCADDR**, **READWRITE**, **DMAC< BUSWIDTH >::regs**, and **RegisterInterface::set_register_callback()**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.4 mth_request_signals()

```
template<unsigned int BUSWIDTH = 32>
void mth_request_signals ()  [inline], [private]
```

mth_request_signals

Impelmentation of the method when requests signal are triggered by peripherals

References **DMA_MAX_CH**, **DMAC< BUSWIDTH >::DMA_req**, **DMACHEN**, **DMAC< BUSWIDTH >::e_↩ DMA_request**, **DMAC< BUSWIDTH >::m_message**, **DMAC< BUSWIDTH >::m_name**, **DMAC< BUSWIDTH >::m_running**, **DMAC< BUSWIDTH >::port_req_ids**, and **DMAC< BUSWIDTH >::regs**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.5 mth_reset()

```
template<unsigned int BUSWIDTH = 32>
void mth_reset ()  [inline], [private]
```

mth_reset

Impelmentation of the method when reset is active

References **DMAC< BUSWIDTH >::e_DMA_forward**, **DMAC< BUSWIDTH >::e_DMA_request**, **DMAC< BUSWIDTH >::e_DMA_run**, **DMAC< BUSWIDTH >::e_DMA_run_done**, **DMAC< BUSWIDTH >::m_cur↩ _reg_ch**, **DMAC< BUSWIDTH >::m_cur_reg_name**, **DMAC< BUSWIDTH >::m_current_ch**, **DMAC< BUSWIDTH >::m_running**, **DMAC< BUSWIDTH >::m_testmode**, **DMAC< BUSWIDTH >::regs**, **Register↩ Interface::reset_regs()**, and **DMAC< BUSWIDTH >::rst**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.6 nb_transport_bw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_bw (
             tlm::tlm_generic_payload & trans,
             tlm::tlm_phase & phase,
             sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_bw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **DMAC< BUSWIDTH >::copy_tlm_generic_payload()**, **DMAC< BUSWIDTH >::current_trans**, **DMADESADDR**, **DMASRCADDR**, **DMAC< BUSWIDTH >::e_DMA_forward**, **DMAC< BUSWIDTH >::e↩ _DMA_run_done**, **DMAC< BUSWIDTH >::m_current_ch**, **DMAC< BUSWIDTH >::m_message**, **DMAC< BUSWIDTH >::m_name**, and **DMAC< BUSWIDTH >::regs**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.7 nb_transport_fw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_fw (
            tlm::tlm_generic_payload & trans,
            tlm::tlm_phase & phase,
            sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_fw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **DMAC< BUSWIDTH >::m_message**, **DMAC< BUSWIDTH >::m_name**, **DMAC< BUSWIDTH >::regs**, **DMAC< BUSWIDTH >::target_socket**, and **RegisterInterface::update_register()**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.8 thr_DMA_forward_process()

```
template<unsigned int BUSWIDTH = 32>
void thr_DMA_forward_process ()  [inline], [private]
```

thr_DMA_forward_process

Impelmentation of the thread to forward data from the source to the destination

References **DMAC< BUSWIDTH >::current_trans**, **DMAC< BUSWIDTH >::e_DMA_forward**, and **DMAC< BUSWIDTH >::initiator_socket**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.9 thr_DMA_run_process()

```
template<unsigned int BUSWIDTH = 32>
void thr_DMA_run_process ()  [inline], [private]
```

thr_DMA_run_process

Impelmentation of the thread to start DMA operation

References **DMADATALENGTH**, **DMASRCADDR**, **DMAC< BUSWIDTH >::e_DMA_run**, **DMAC< BUSWIDTH >::initiator_socket**, **DMAC< BUSWIDTH >::m_current_ch**, and **DMAC< BUSWIDTH >::regs**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.2.10 thr_priority_process()

```
template<unsigned int BUSWIDTH = 32>
void thr_priority_process ()  [inline], [private]
```

thr_priority_process

Impelmentation of the thread to handle priority of DMA operation

References **DMAC< BUSWIDTH >::clk**, **DMAC< BUSWIDTH >::DMA_ack**, **DMAC< BUSWIDTH >::DMA_int**, **DMAACK**, **DMACHEN**, **DMAINT**, **DMAC< BUSWIDTH >::e_DMA_request**, **DMAC< BUSWIDTH >::e_DMA_run**, **DMAC< BUSWIDTH >::e_DMA_run_done**, **DMAC< BUSWIDTH >::m_cur_reg_ch**, **DMAC< BUSWIDTH >::m_cur_reg_name**, **DMAC< BUSWIDTH >::m_current_ch**, **DMAC< BUSWIDTH >::m_message**, **DMAC< BUSWIDTH >::m_name**, **DMAC< BUSWIDTH >::m_running**, **DMAC< BUSWIDTH >::m_testmode**, **DMAC< BUSWIDTH >::port_req_ids**, and **DMAC< BUSWIDTH >::regs**.

Referenced by **DMAC< BUSWIDTH >::DMAC()**.

4.3.3 Member Data Documentation

4.3.3.1 clk

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> clk
```

Referenced by **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.2 current_trans

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_generic_payload current_trans  [private]
```

Referenced by **DMAC< BUSWIDTH >::nb_transport_bw()**, and **DMAC< BUSWIDTH >::thr_DMA_forward_process()**.

4.3.3.3 DMA_ack

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_out<bool> DMA_ack[ DMA_MAX_CH]
```

Referenced by **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.4 DMA_int

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_out<bool> DMA_int[ DMA_MAX_CH]
```

Referenced by **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.5 DMA_req

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> DMA_req[ DMA_MAX_CH]
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, and **DMAC< BUSWIDTH >::mth_request_signals()**.

4.3.3.6 e_DMA_forward

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_DMA_forward  [private]
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, and **DMAC< BUSWIDTH >::thr_DMA_forward_process()**.

4.3.3.7 e_DMA_request

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_DMA_request  [private]
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::mth_reset()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.8 e_DMA_run

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_DMA_run  [private]
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::thr_DMA_run_process()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.9 e_DMA_run_done

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_DMA_run_done  [private]
```

Referenced by **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.10 initiator_socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_initiator_socket< DMAC, BUSWIDTH> initiator_socket
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, **DMAC< BUSWIDTH >::thr_DMA_forward_process()**, and **DMAC< BUSWIDTH >::thr_DMA_run_process()**.

4.3.3.11 m_cur_reg_ch

```
template<unsigned int BUSWIDTH = 32>
unsigned int m_cur_reg_ch  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_reset()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.12 m_cur_reg_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_cur_reg_name  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_reset()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.13 m_current_ch

```
template<unsigned int BUSWIDTH = 32>
unsigned int m_current_ch  [private]
```

Referenced by **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, **DMAC< BUSWIDTH >::thr_DMA_run_process()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.14 m_message

```
template<unsigned int BUSWIDTH = 32>
bool m_message  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, **DMAC< BUSWIDTH >::nb_transport_fw()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.15 m_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_name  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, **DMAC< BUSWIDTH >::nb_transport_fw()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.16 m_running

```
template<unsigned int BUSWIDTH = 32>
bool m_running  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::mth_reset()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.17 m_testmode

```
template<unsigned int BUSWIDTH = 32>
bool m_testmode  [private]
```

Referenced by **DMAC< BUSWIDTH >::cb_DMAREQ()**, **DMAC< BUSWIDTH >::mth_reset()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.18 port_req_ids

```
template<unsigned int BUSWIDTH = 32>
std::list<unsigned int> port_req_ids  [private]
```

Referenced by **DMAC< BUSWIDTH >::mth_request_signals()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.19 regs

```
template<unsigned int BUSWIDTH = 32>
RegisterInterface regs  [private]
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, **DMAC< BUSWIDTH >::mth_request_signals()**, **DMAC< BUSWIDTH >::mth_reset()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, **DMAC< BUSWIDTH >::nb_transport_fw()**, **DMAC< BUSWIDTH >::thr_DMA_run_process()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

4.3.3.20 rst

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> rst
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, and **DMAC< BUSWIDTH >::mth_reset()**.

4.3.3.21 target_socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_target_socket< DMAC, BUSWIDTH> target_socket
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, and **DMAC< BUSWIDTH >::nb_transport_fw()**.

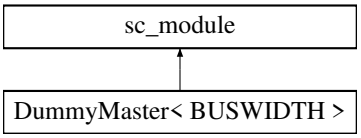
The documentation for this class was generated from the following file:

- common/ **DMAC.h**

4.4 DummyMaster< BUSWIDTH >

```
#include "DummyMaster.h"
```

Inheritance diagram for DummyMaster< BUSWIDTH >:



Classes

- struct **localdata**

Public Member Functions

- **DummyMaster** (sc_core::sc_module_name name, bool message=false)

  DummyMaster *(p. 24).*
- unsigned int **get_received_32bit_data** ()

  get_received_data
- unsigned char * **get_received_data** ()

  get_received_data
- void **Sentcustomtransaction** (unsigned int addr, unsigned char *data, unsigned int data_length, tlm::tlm↩ _command cmd)

  Sentcustomtransaction.
- void **SentTransaction** (unsigned int addr, uint32_t data, tlm::tlm_command cmd)

  SentTransaction.

Public Attributes

- sc_core::sc_in< bool > **clk**
- tlm_utils::simple_initiator_socket< **DummyMaster**, BUSWIDTH > **initiator_socket**
- sc_core::sc_out< bool > **rst**

Private Member Functions

- tlm::tlm_sync_enum **nb_transport_bw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core::sc_time &delay)

  nb_transport_bw
- void **reset_process** ()

  resetsystem Reset thread handling
- void **resetsystem** ()

  resetsystem To triggered reset signal

Private Attributes

- sc_core::sc_event **e_reset**
- bool **m_message**
- std::string **m_name**
- **localdata tempdata**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 DummyMaster()

```
template<unsigned int BUSWIDTH = 32>
DummyMaster (
             sc_core::sc_module_name name,
             bool message = false)  [inline]
```

DummyMaster (p. 24).

The constructor of **DummyMaster** (p. 24)

References **DummyMaster< BUSWIDTH >::e_reset**, **DummyMaster< BUSWIDTH >::initiator_socket**, **DummyMaster< BUSWIDTH >::nb_transport_bw()**, and **DummyMaster< BUSWIDTH >::reset_process()**.

4.4.2 Member Function Documentation

4.4.2.1 get_received_32bit_data()

```
template<unsigned int BUSWIDTH = 32>
unsigned int get_received_32bit_data ()  [inline]
```

get_received_data

Using to get data from the returned transaction

Returns

the 32-bit data

References **DummyMaster< BUSWIDTH >::localdata::m_data**, and **DummyMaster< BUSWIDTH >::tempdata**.

4.4.2.2 get_received_data()

```
template<unsigned int BUSWIDTH = 32>
unsigned char * get_received_data ()  [inline]
```

get_received_data

Using to get data from the returned transaction

Returns

the array of data

References **DummyMaster< BUSWIDTH >::localdata::m_data**, and **DummyMaster< BUSWIDTH >::tempdata**.

4.4.2.3 nb_transport_bw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_bw (
             tlm::tlm_generic_payload & trans,
             tlm::tlm_phase & phase,
             sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_bw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **DummyMaster< BUSWIDTH >::localdata::m_data**, **DummyMaster< BUSWIDTH >::m_message**, **DummyMaster< BUSWIDTH >::m_name**, and **DummyMaster< BUSWIDTH >::tempdata**.

Referenced by **DummyMaster< BUSWIDTH >::DummyMaster()**.

4.4.2.4 reset_process()

```
template<unsigned int BUSWIDTH = 32>
void reset_process ()  [inline], [private]
```

resetsystem Reset thread handling

References **DummyMaster< BUSWIDTH >::clk**, **DummyMaster< BUSWIDTH >::e_reset**, and **DummyMaster< BUSWIDTH >::rst**.

Referenced by **DummyMaster< BUSWIDTH >::DummyMaster()**.

4.4.2.5 resetsystem()

```
template<unsigned int BUSWIDTH = 32>
void resetsystem ()  [inline], [private]
```

resetsystem To triggered reset signal

References **DummyMaster< BUSWIDTH >::e_reset**.

4.4.2.6 Sentcustomtransaction()

```
template<unsigned int BUSWIDTH = 32>
void Sentcustomtransaction (
            unsigned int addr,
            unsigned char * data,
            unsigned int data_length,
            tlm::tlm_command cmd)  [inline]
```

Sentcustomtransaction.

Implements the sent a custom transaction to bus MMIO

Parameters

addr	Reference to the address of the slave
data	Reference to the pointer of array data
data_length	Reference to the length of data
cmd	Reference to tlm command -TLM_READ_COMMAND : TLM read requset -TLM_WRITE_COMMAND : TLM write requset -TLM_IGNORE_COMMAND: TLM ignore requset

References **DummyMaster< BUSWIDTH >::initiator_socket**, **DummyMaster< BUSWIDTH >::m_message**, and **DummyMaster< BUSWIDTH >::m_name**.

4.4.2.7 SentTransaction()

```
template<unsigned int BUSWIDTH = 32>
void SentTransaction (
            unsigned int addr,
            uint32_t data,
            tlm::tlm_command cmd)  [inline]
```

SentTransaction.

Implements the sent a transaction with 32 bit data to bus MMIO

Parameters

addr	Reference to the address of the slave
data	Reference to 32-bit data
cmd	Reference to tlm command -TLM_READ_COMMAND : TLM read requset -TLM_WRITE_COMMAND : TLM write requset -TLM_IGNORE_COMMAND: TLM ignore requset

References **DummyMaster< BUSWIDTH >::initiator_socket**, **DummyMaster< BUSWIDTH >::m_message**, and **DummyMaster< BUSWIDTH >::m_name**.

4.4.3 Member Data Documentation

4.4.3.1 clk

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> clk
```

Referenced by **DummyMaster< BUSWIDTH >::reset_process()**.

4.4.3.2 e_reset

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_reset  [private]
```

Referenced by **DummyMaster< BUSWIDTH >::DummyMaster()**, **DummyMaster< BUSWIDTH >::reset_process()**, and **DummyMaster< BUSWIDTH >::resetsystem()**.

4.4.3.3 initiator_socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_initiator_socket< DummyMaster, BUSWIDTH> initiator_socket
```

Referenced by **DummyMaster< BUSWIDTH >::DummyMaster()**, **DummyMaster< BUSWIDTH >::Sentcustomtransaction()**, and **DummyMaster< BUSWIDTH >::SentTransaction()**.

4.4.3.4 m_message

```
template<unsigned int BUSWIDTH = 32>
bool m_message  [private]
```

Referenced by **DummyMaster< BUSWIDTH >::nb_transport_bw()**, **DummyMaster< BUSWIDTH >::Sentcustomtransaction()**, and **DummyMaster< BUSWIDTH >::SentTransaction()**.

4.4.3.5 m_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_name  [private]
```

Referenced by **DummyMaster< BUSWIDTH >::nb_transport_bw()**, **DummyMaster< BUSWIDTH >::Sentcustomtransaction()**, and **DummyMaster< BUSWIDTH >::SentTransaction()**.

4.4.3.6 rst

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_out<bool> rst
```

Referenced by **DummyMaster< BUSWIDTH >::reset_process()**.

4.4.3.7 tempdata

```
template<unsigned int BUSWIDTH = 32>
localdata tempdata  [private]
```

Referenced by **DummyMaster< BUSWIDTH >::get_received_32bit_data()**, **DummyMaster< BUSWIDTH >::get_received_data()**, and **DummyMaster< BUSWIDTH >::nb_transport_bw()**.

The documentation for this class was generated from the following file:

- common/ **DummyMaster.h**

4.5 DummyMaster< BUSWIDTH >::localdata

Public Attributes

- unsigned char * **m_data**
- unsigned int **m_length**

4.5.1 Member Data Documentation

4.5.1.1 m_data

```
template<unsigned int BUSWIDTH = 32>
unsigned char* m_data
```

Referenced by **DummyMaster< BUSWIDTH >::get_received_32bit_data()**, **DummyMaster< BUSWIDTH >::get_received_data()**, and **DummyMaster< BUSWIDTH >::nb_transport_bw()**.

4.5.1.2 m_length

```
template<unsigned int BUSWIDTH = 32>
unsigned int m_length
```

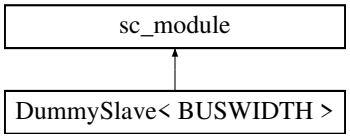
The documentation for this struct was generated from the following file:

- common/ **DummyMaster.h**

4.6 DummySlave< BUSWIDTH >

`#include "DummySlave.h"`

Inheritance diagram for DummySlave< BUSWIDTH >:



Public Member Functions

- **DummySlave** (sc_core::sc_module_name name, bool message=false)
  DummySlave *(p. 30).*
- sc_core::sc_in< bool > * **add_input_port** (const std::string &name)
  add_input_port
- sc_core::sc_out< bool > * **add_output_port** (const std::string &name)
  add_output_port
- void **enable_monitor_clock** (bool is_enable)
  enable_monitor_clock
- void **monitor_ports** (bool is_enable)
  monitor_ports
- bool **read_input_ports** (const std::string &name)
  read_input_ports
- void **set_output_ports** (const std::string &name, bool value)
  set_output_ports
- void **trigger_output_ports** (const std::string &name, bool high_level, bool is_pos)
  set_output_ports

Public Attributes

- sc_core::sc_in< bool > **clk**
- sc_core::sc_in< bool > **rst**
- tlm_utils::simple_target_socket< **DummySlave**, BUSWIDTH > **target_socket**

Private Member Functions

- void **cb_DUMMYRESULT** (const std::string &name, uint32_t value, uint32_t old_value, uint32_t mask, uint32_t ch)
  cb_DUMMYRESULT
- void **end_of_elaboration** () override
  end_of_elaboration
- void **init_registers** ()
  init_registers
- void **monitor_inputs** ()
  monitor_inputs
- void **mth_reset** ()

mth_reset

- void **mth_synchronize_cycles** ()

  mth_synchronize_cycles
- tlm::tlm_sync_enum **nb_transport_fw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core↩ ::sc_time &delay)

  nb_transport_fw
- void **thr_triggered_port_process** ()

  thr_triggered_port_process

Private Attributes

- sc_core::sc_event **e_triggerd_port**
- std::map< std::string, sc_core::sc_in< bool > * > **input_ports**
- std::map< std::string, bool > **input_val_ports**
- bool **m_clkmonitor**
- bool **m_cur_is_pos**
- std::string **m_cur_port_name**
- bool **m_cur_triggered_val**
- bool **m_message**
- std::string **m_name**
- bool **m_portmonitor**
- std::map< std::string, sc_core::sc_out< bool > * > **output_ports**
- std::map< std::string, bool > **output_val_ports**
- **RegisterInterface regs**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 DummySlave()

```
template<unsigned int BUSWIDTH = 32>
DummySlave (
            sc_core::sc_module_name name,
            bool message = false)  [inline]
```

DummySlave (p. 30).

DummySlave (p. 30) Constructure

Parameters

name	Reference to sc_module name
message	To enable message log

References **DummySlave< BUSWIDTH >::clk**, **DummySlave< BUSWIDTH >::e_triggerd_port**, **Dummy↩ Slave< BUSWIDTH >::init_registers()**, **DummySlave< BUSWIDTH >::mth_reset()**, **DummySlave< BUSWIDTH >::mth_synchronize_cycles()**, **DummySlave< BUSWIDTH >::nb_transport_fw()**, **Dummy↩ Slave< BUSWIDTH >::rst**, **DummySlave< BUSWIDTH >::target_socket**, and **DummySlave< BUSWIDTH >::thr_triggered_port_process()**.

4.6.2 Member Function Documentation

4.6.2.1 add_input_port()

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in< bool > * add_input_port (
            const std::string & name)  [inline]
```

add_input_port

To add input specific input ports to **DummySlave** (p. 30)

Parameters

name	Reference to the port name

References **DummySlave**< **BUSWIDTH** >**::input_ports**, and **DummySlave**< **BUSWIDTH** >**::input_val_ports**.

4.6.2.2 add_output_port()

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_out< bool > * add_output_port (
            const std::string & name)  [inline]
```

add_output_port

To add ouput specific output ports to **DummySlave** (p. 30)

Parameters

name	Reference to the port name

References **DummySlave**< **BUSWIDTH** >**::output_ports**, and **DummySlave**< **BUSWIDTH** >**::output_val_↩ ports**.

4.6.2.3 cb_DUMMYRESULT()

```
template<unsigned int BUSWIDTH = 32>
void cb_DUMMYRESULT (
            const std::string & name,
            uint32_t value,
            uint32_t old_value,
            uint32_t mask,
            uint32_t ch)  [inline], [private]
```

cb_DUMMYRESULT

The callback function for DUMMYRESULT register

Parameters

name	The name of register
value	The value of register
old_value	the previous value
mask	the mask of register
ch	the register channel

Referenced by **DummySlave< BUSWIDTH >::init_registers()**.

4.6.2.4 enable_monitor_clock()

```
template<unsigned int BUSWIDTH = 32>
void enable_monitor_clock (
             bool is_enable)  [inline]
```

enable_monitor_clock

Using to enable or disable clock monitor operation

Parameters

is_enable	Indicating whether enabling or disabling clock monitor

References **DummySlave< BUSWIDTH >::m_clkmonitor**.

4.6.2.5 end_of_elaboration()

```
template<unsigned int BUSWIDTH = 32>
void end_of_elaboration ()  [inline], [override], [private]
```

end_of_elaboration

The function end_of_elaboration is called before starting simulation

References **DummySlave< BUSWIDTH >::input_ports**, and **DummySlave< BUSWIDTH >::monitor_↩ inputs()**.

4.6.2.6 init_registers()

```
template<unsigned int BUSWIDTH = 32>
void init_registers ()  [inline], [private]
```

init_registers

Initialization registers

References **RegisterInterface::add_register()**, **DummySlave< BUSWIDTH >::cb_DUMMYRESULT()**, **DUMMYRESULT**, **READWRITE**, **DummySlave< BUSWIDTH >::regs**, and **RegisterInterface::set_register_↩ callback()**.

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.2.7 monitor_inputs()

```
template<unsigned int BUSWIDTH = 32>
void monitor_inputs ()  [inline], [private]
```

monitor_inputs

The method uses to monitor input ports

References **DummySlave< BUSWIDTH >::input_ports**, **DummySlave< BUSWIDTH >::input_val_ports**, **DummySlave< BUSWIDTH >::m_name**, and **DummySlave< BUSWIDTH >::m_portmonitor**.

Referenced by **DummySlave< BUSWIDTH >::end_of_elaboration()**.

4.6.2.8 monitor_ports()

```
template<unsigned int BUSWIDTH = 32>
void monitor_ports (
            bool is_enable)  [inline]
```

monitor_ports

Using to enable or disable port monitor operation

Parameters

is_enable	Indicating whether enabling or disabling port monitor

References **DummySlave< BUSWIDTH >::m_portmonitor**.

4.6.2.9 mth_reset()

```
template<unsigned int BUSWIDTH = 32>
void mth_reset ()  [inline], [private]
```

mth_reset

Implementation the method to handle reset operation

References **DummySlave< BUSWIDTH >::regs**, and **RegisterInterface::reset_regs()**.

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.2.10 mth_synchronize_cycles()

```
template<unsigned int BUSWIDTH = 32>
void mth_synchronize_cycles ()  [inline], [private]
```

mth_synchronize_cycles

Implementation the method to monitor clock cycles

References **DummySlave< BUSWIDTH >::m_clkmonitor**.

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.2.11 nb_transport_fw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_fw (
            tlm::tlm_generic_payload & trans,
            tlm::tlm_phase & phase,
            sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_fw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **DummySlave< BUSWIDTH >::m_message**, **DummySlave< BUSWIDTH >::m_name**, **DummySlave< BUSWIDTH >::regs**, **DummySlave< BUSWIDTH >::target_socket**, and **RegisterInterface::update_register()**.

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.2.12 read_input_ports()

```
template<unsigned int BUSWIDTH = 32>
bool read_input_ports (
            const std::string & name)  [inline]
```

read_input_ports

Using to read the value of specific port

Parameters

name	Reference to the port name

References **DummySlave< BUSWIDTH >::input_ports**.

4.6.2.13 set_output_ports()

```
template<unsigned int BUSWIDTH = 32>
void set_output_ports (
            const std::string & name,
            bool value)  [inline]
```

set_output_ports

Using to set specific output port

Parameters

name	Reference to the port name
value	the value of the output port

References **DummySlave< BUSWIDTH >::output_ports**.

4.6.2.14 thr_triggered_port_process()

```
template<unsigned int BUSWIDTH = 32>
void thr_triggered_port_process ()  [inline], [private]
```

thr_triggered_port_process

Implementation the process to trigger specific ports

References **DummySlave< BUSWIDTH >::clk**, **DummySlave< BUSWIDTH >::e_triggerd_port**, **DummySlave< BUSWIDTH >::m_cur_is_pos**, **DummySlave< BUSWIDTH >::m_cur_port_name**, **DummySlave< BUSWIDTH >::m_cur_triggered_val**, and **DummySlave< BUSWIDTH >::output_ports**.

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.2.15 trigger_output_ports()

```
template<unsigned int BUSWIDTH = 32>
void trigger_output_ports (
            const std::string & name,
            bool high_level,
            bool is_pos)  [inline]
```

set_output_ports

Using to trigger specific output port

Parameters

name	Reference to the port name
high_level	Indicating the triggered level
is_pos	Indicating the clock edge synchronization is positive or negative

References **DummySlave< BUSWIDTH >::m_cur_is_pos**, **DummySlave< BUSWIDTH >::m_cur_port_name**, **DummySlave< BUSWIDTH >::m_cur_triggered_val**, and **DummySlave< BUSWIDTH >::output_ports**.

4.6.3 Member Data Documentation

4.6.3.1 clk

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> clk
```

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**, and **DummySlave< BUSWIDTH >::thr_triggered_port_process()**.

4.6.3.2 e_triggerd_port

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_event e_triggerd_port  [private]
```

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**, and **DummySlave< BUSWIDTH >::thr_triggered_port_process()**.

4.6.3.3 input_ports

```
template<unsigned int BUSWIDTH = 32>
std::map<std::string, sc_core::sc_in<bool>*> input_ports  [private]
```

Referenced by **DummySlave< BUSWIDTH >::add_input_port()**, **DummySlave< BUSWIDTH >::end_of_elaboration()**, **DummySlave< BUSWIDTH >::monitor_inputs()**, and **DummySlave< BUSWIDTH >::read_input_ports()**.

4.6.3.4 input_val_ports

```
template<unsigned int BUSWIDTH = 32>
std::map<std::string, bool> input_val_ports  [private]
```

Referenced by **DummySlave< BUSWIDTH >::add_input_port()**, and **DummySlave< BUSWIDTH >::monitor_inputs()**.

4.6.3.5 m_clkmonitor

```
template<unsigned int BUSWIDTH = 32>
bool m_clkmonitor  [private]
```

Referenced by **DummySlave< BUSWIDTH >::enable_monitor_clock()**, and **DummySlave< BUSWIDTH >::mth_synchronize_cycles()**.

4.6.3.6 m_cur_is_pos

```
template<unsigned int BUSWIDTH = 32>
bool m_cur_is_pos  [private]
```

Referenced by **DummySlave< BUSWIDTH >::thr_triggered_port_process()**, and **DummySlave< BUSWIDTH >::trigger_output_ports()**.

4.6.3.7 m_cur_port_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_cur_port_name  [private]
```

Referenced by **DummySlave< BUSWIDTH >::thr_triggered_port_process()**, and **DummySlave< BUSWIDTH >::trigger_output_ports()**.

4.6.3.8 m_cur_triggered_val

```
template<unsigned int BUSWIDTH = 32>
bool m_cur_triggered_val  [private]
```

Referenced by **DummySlave< BUSWIDTH >::thr_triggered_port_process()**, and **DummySlave< BUSWIDTH >::trigger_output_ports()**.

4.6.3.9 m_message

```
template<unsigned int BUSWIDTH = 32>
bool m_message  [private]
```

Referenced by **DummySlave< BUSWIDTH >::nb_transport_fw()**.

4.6.3.10 m_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_name  [private]
```

Referenced by **DummySlave< BUSWIDTH >::monitor_inputs()**, and **DummySlave< BUSWIDTH >::nb_transport_fw()**.

4.6.3.11 m_portmonitor

```
template<unsigned int BUSWIDTH = 32>
bool m_portmonitor  [private]
```

Referenced by **DummySlave< BUSWIDTH >::monitor_inputs()**, and **DummySlave< BUSWIDTH >::monitor_ports()**.

4.6.3.12 output_ports

```
template<unsigned int BUSWIDTH = 32>
std::map<std::string, sc_core::sc_out<bool>*> output_ports  [private]
```

Referenced by **DummySlave< BUSWIDTH >::add_output_port()**, **DummySlave< BUSWIDTH >::set_output_ports()**, **DummySlave< BUSWIDTH >::thr_triggered_port_process()**, and **DummySlave< BUSWIDTH >::trigger_output_ports()**.

4.6.3.13 output_val_ports

```
template<unsigned int BUSWIDTH = 32>
std::map<std::string, bool> output_val_ports  [private]
```

Referenced by **DummySlave< BUSWIDTH >::add_output_port()**.

4.6.3.14 regs

```
template<unsigned int BUSWIDTH = 32>
RegisterInterface regs  [private]
```

Referenced by **DummySlave< BUSWIDTH >::init_registers()**, **DummySlave< BUSWIDTH >::mth_reset()**, and **DummySlave< BUSWIDTH >::nb_transport_fw()**.

4.6.3.15 rst

```
template<unsigned int BUSWIDTH = 32>
sc_core::sc_in<bool> rst
```

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**.

4.6.3.16 target_socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_target_socket< DummySlave, BUSWIDTH> target_socket
```

Referenced by **DummySlave< BUSWIDTH >::DummySlave()**, and **DummySlave< BUSWIDTH >::nb_transport_fw()**.

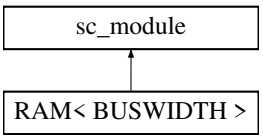
The documentation for this class was generated from the following file:

- common/ **DummySlave.h**

4.7 RAM< BUSWIDTH >

```
#include "memory.h"
```

Inheritance diagram for RAM< BUSWIDTH >:



Public Member Functions

- **RAM** (sc_core::sc_module_name name, sc_dt::uint64 **size**, bool message=false)
  RAM *(p. 39).*
- ~**RAM** ()
  RAM *(p. 39).*
- void **dump_memory** (sc_dt::uint64 offset, unsigned int len)
  dump_memory
- std::string **get_name** ()
  get_name
- tlm::tlm_sync_enum **nb_transport_fw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core::sc_time &delay)
  nb_transport_fw

Public Attributes

- tlm_utils::simple_target_socket< **RAM**, BUSWIDTH > **socket**

Private Attributes

- unsigned char * **data**
- bool **m_message**
- std::string **m_name**
- sc_dt::uint64 **size**

4.7.1 Constructor & Destructor Documentation

4.7.1.1 RAM()

```
template<unsigned int BUSWIDTH = 32>
RAM (
            sc_core::sc_module_name name,
            sc_dt::uint64 size,
            bool message = false)  [inline]
```

RAM (p. 39).

RAM (p. 39) Constructure

Parameters

name	Reference to sc_module name
size	Reference to the ram size
message	To enable message log

References **RAM< BUSWIDTH >::data**, **RAM< BUSWIDTH >::nb_transport_fw()**, **RAM< BUSWIDTH >::size**, and **RAM< BUSWIDTH >::socket**.

4.7.1.2 ~RAM()

```
template<unsigned int BUSWIDTH = 32>
~ RAM ()  [inline]
```

RAM (p. 39).

RAM (p. 39) Destructure

References **RAM< BUSWIDTH >::data**.

4.7.2 Member Function Documentation

4.7.2.1 dump_memory()

```
template<unsigned int BUSWIDTH = 32>
void dump_memory (
            sc_dt::uint64 offset,
            unsigned int len)  [inline]
```

dump_memory

Show memory regions

Parameters

offset	The ram offset
len	The ram size

References **RAM< BUSWIDTH >::data**, **RAM< BUSWIDTH >::m_name**, and **RAM< BUSWIDTH >::size**.

4.7.2.2 get_name()

```
template<unsigned int BUSWIDTH = 32>
std::string get_name ()  [inline]
```

get_name

Using to get the name of Ram.

References **RAM< BUSWIDTH >::m_name**.

4.7.2.3 nb_transport_fw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_fw (
             tlm::tlm_generic_payload & trans,
             tlm::tlm_phase & phase,
             sc_core::sc_time & delay)  [inline]
```

nb_transport_fw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **RAM< BUSWIDTH >::data**, **RAM< BUSWIDTH >::m_message**, **RAM< BUSWIDTH >::m_name**, **RAM< BUSWIDTH >::size**, and **RAM< BUSWIDTH >::socket**.

Referenced by **RAM< BUSWIDTH >::RAM()**.

4.7.3 Member Data Documentation

4.7.3.1 data

```
template<unsigned int BUSWIDTH = 32>
unsigned char* data  [private]
```

Referenced by **RAM< BUSWIDTH >::RAM()**, **RAM< BUSWIDTH >::~RAM()**, **RAM< BUSWIDTH >::dump_memory()**, and **RAM< BUSWIDTH >::nb_transport_fw()**.

4.7.3.2 m_message

```
template<unsigned int BUSWIDTH = 32>
bool m_message  [private]
```

Referenced by **RAM< BUSWIDTH >::nb_transport_fw()**.

4.7.3.3 m_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_name  [private]
```

Referenced by **RAM< BUSWIDTH >::dump_memory()**, **RAM< BUSWIDTH >::get_name()**, and **RAM< BUSWIDTH >::nb_transport_fw()**.

4.7.3.4 size

```
template<unsigned int BUSWIDTH = 32>
sc_dt::uint64 size  [private]
```

Referenced by **RAM< BUSWIDTH >::RAM()**, **RAM< BUSWIDTH >::dump_memory()**, and **RAM< BUSWIDTH >::nb_transport_fw()**.

4.7.3.5 socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_target_socket< RAM, BUSWIDTH> socket
```

Referenced by **RAM< BUSWIDTH >::RAM()**, and **RAM< BUSWIDTH >::nb_transport_fw()**.

The documentation for this class was generated from the following file:

- common/ **memory.h**

4.8 Register

```
#include "Registerif.h"
```

Public Types

- using **Callback** = std::function<void(const std::string&, uint32_t, uint32_t, uint32_t, uint32_t)>

Public Member Functions

- **Register** ()

  Register *(*p. *42).*
- **Register** (std::string **name**, uint64_t **address**, uint32_t init, uint32_t **mask**, uint32_t **ch**, **REGPERMISSION permission**)

  Register *(*p. *42).*
- ~**Register** ()

  ~Register Destructure
- uint64_t **get_address** ()

  get_address Return the register base address
- std::string **get_name** ()

  get_name Return the register name
- uint32_t **get_value** () const

  get_value Return the register value
- **Register** & **operator=** (uint32_t new_value)

  operator= To set new value for the rigster
- void **reset** ()

  reset To reset register to initialization value
- void **set_callback** (**Callback** cb)

  operator= Using to register call back function for the register
- void **set_readonly_value** (uint32_t new_value)

  set_value Using for developer to set read only register value
- void **set_value** (uint32_t new_value)

  set_value Using to set register value

Private Attributes

- uint64_t **address**
- **Callback callback**
- const uint32_t **ch**
- const uint32_t **init_val**
- uint32_t **mask**
- std::string **name**
- const **REGPERMISSION permission**
- uint32_t **value**

4.8.1 Member Typedef Documentation

4.8.1.1 Callback

```
using Callback = std::function<void(const std::string&, uint32_t, uint32_t, uint32_t, uint32↩
_t)>
```

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Register() [1/2]

```
Register () [inline]
```

Register (p. 42).

Default Constructure (this function does not refer to use)

4.8.2.2 Register() [2/2]

```
Register (
            std::string name,
            uint64_t address,
            uint32_t init,
            uint32_t mask,
            uint32_t ch,
             REGPERMISSION permission) [inline]
```

Register (p. 42).

Constructure

Parameters

name	The register name
address	The register base address
init	The register initialization value
mask	The register mask
ch	The register channel
permission	Indicating this register is read /write or read only permission

4.8.2.3 ~Register()

```
~ Register () [inline]
```

~Register Destructure

4.8.3 Member Function Documentation

4.8.3.1 get_address()

```
uint64_t get_address () [inline]
```

get_address Return the register base address

References **address**.

4.8.3.2 get_name()

```
std::string get_name ()  [inline]
```

get_name Return the register name

References **name**.

4.8.3.3 get_value()

```
uint32_t get_value () const  [inline]
```

get_value Return the register value

References **value**.

4.8.3.4 operator=()

```
Register & operator= (
            uint32_t new_value)  [inline]
```

operator= To set new value for the rigster

Parameters

new_value	The new value that is wrote into this register

References **callback**, **ch**, **mask**, **name**, **permission**, **READWRITE**, and **value**.

4.8.3.5 reset()

```
void reset ()  [inline]
```

reset To reset register to initialization value

References **init_val**, and **value**.

4.8.3.6 set_callback()

```
void set_callback (
             Callback cb)  [inline]
```

operator= Using to register call back function for the register

Parameters

cb	The address of call back function

References **callback**.

4.8.3.7 set_readonly_value()

```
void set_readonly_value (
            uint32_t new_value)  [inline]
```

set_value Using for developer to set read only register value

Parameters

new_value	The new value that is wrote into this register

References **mask**, **permission**, **READONLY**, and **value**.

4.8.3.8 set_value()

```
void set_value (
            uint32_t new_value)  [inline]
```

set_value Using to set register value

Parameters

new_value	The new value that is wrote into this register

References **callback**, **ch**, **mask**, **name**, **permission**, **READWRITE**, and **value**.

4.8.4 Member Data Documentation

4.8.4.1 address

```
uint64_t address  [private]
```

Referenced by **get_address()**.

4.8.4.2 callback

```
Callback callback  [private]
```

Referenced by **operator=()**, **set_callback()**, and **set_value()**.

4.8.4.3 ch

```
const uint32_t ch  [private]
```

Referenced by **operator=()**, and **set_value()**.

4.8.4.4 init_val

```
const uint32_t init_val  [private]
```

Referenced by **reset()**.

4.8.4.5 mask

```
uint32_t mask  [private]
```

Referenced by **operator=()**, **set_readonly_value()**, and **set_value()**.

4.8.4.6 name

```
std::string name  [private]
```

Referenced by **get_name()**, **operator=()**, and **set_value()**.

4.8.4.7 permission

```
const  REGPERMISSION permission  [private]
```

Referenced by **operator=()**, **set_readonly_value()**, and **set_value()**.

4.8.4.8 value

```
uint32_t value  [private]
```

Referenced by **get_value()**, **operator=()**, **reset()**, **set_readonly_value()**, and **set_value()**.

The documentation for this class was generated from the following file:

- common/ **Registerif.h**

4.9 RegisterInterface

```
#include "Registerif.h"
```

Public Member Functions

- void **add_register** (std::string name, uint64_t address, uint32_t init, uint32_t mask, uint32_t ch, **REGPERMISSION** permission)

  add_register
- void **dump_registers** ()

  update_register Using to show all register informations
- **Register** & **operator[ ]** (std::string name)

  add_register The operator to get register by name for example this->reg[name]
- **Register** & **operator[ ]** (uint64_t address)

  operator[] The operator to get register by address for example this->reg[addr]
- void **reset_regs** ()

  update_register Using to reset registers
- void **set_register_callback** (const std::string &name, **Register::Callback** cb)

  update_register Using to register register callback function
- void **update_register** (uint64_t address, uint32_t value)

  update_register Using to update the value of register with specific address

Private Attributes

- std::map< std::string, **Register** > **registers**

4.9.1 Member Function Documentation

4.9.1.1 add_register()

```
void add_register (
            std::string name,
            uint64_t address,
            uint32_t init,
            uint32_t mask,
            uint32_t ch,
             REGPERMISSION permission)  [inline]
```

add_register

Constructure

Parameters

name	The register name
address	The register base address
init	The register initialization value
mask	The register mask
ch	The register channel
permission	Indicating this register is read /write or read only permission

References **registers**.

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DummySlave< BUSWIDTH >::init_registers()**.

4.9.1.2 dump_registers()

```
void dump_registers ()  [inline]
```

update_register Using to show all register informations

References **registers**.

4.9.1.3 operator[]() [1/2]

```
Register & operator[] (
            std::string name)  [inline]
```

add_register The operator to get register by name for example this->reg[name]

Parameters

name	The register name

Returns

Register (p. 42) object with corresponding name

References **registers**.

4.9.1.4 operator[]() [2/2]

```
Register & operator[] (
            uint64_t address)  [inline]
```

operator[] The operator to get register by address for example this->reg[addr]

Parameters

address	The register address

Returns

Register (p. 42) object with corresponding address

References **registers**.

4.9.1.5 reset_regs()

```
void reset_regs ()  [inline]
```

update_register Using to reset registers

References **registers**.

Referenced by **DMAC< BUSWIDTH >::mth_reset()**, and **DummySlave< BUSWIDTH >::mth_reset()**.

4.9.1.6 set_register_callback()

```
void set_register_callback (
            const std::string & name,
             Register::Callback cb)  [inline]
```

update_register Using to register register callback function

Parameters

name	The register name
cb	The address of call back function

References **registers**.

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DummySlave< BUSWIDTH >::init_registers()**.

4.9.1.7 update_register()

```
void update_register (
            uint64_t address,
            uint32_t value)  [inline]
```

update_register Using to update the value of register with specific address

Parameters

address	The register address

Returns

value The new value that is wrote into this register

References **registers**.

Referenced by **DMAC< BUSWIDTH >::nb_transport_fw()**, and **DummySlave< BUSWIDTH >::nb_transport_fw()**.

4.9.2 Member Data Documentation

4.9.2.1 registers

```
std::map<std::string, Register> registers [private]
```

Referenced by **add_register()**, **dump_registers()**, **operator[]()**, **operator[]()**, **reset_regs()**, **set_register_callback()**, and **update_register()**.

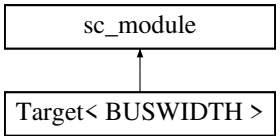
The documentation for this class was generated from the following file:

- common/ **Registerif.h**

4.10 Target< BUSWIDTH >

```
#include "target.h"
```

Inheritance diagram for Target< BUSWIDTH >:



Public Member Functions

- **Target** (sc_core::sc_module_name name)
  Target *(p. 50).*

Public Attributes

- tlm_utils::simple_target_socket< **Target**, BUSWIDTH > **target_socket**

Private Member Functions

- tlm::tlm_sync_enum **nb_transport_fw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core↩ ::sc_time &delay)

  nb_transport_fw

Private Attributes

- std::string **m_name**

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Target()

```
template<unsigned int BUSWIDTH = 32>
Target (
            sc_core::sc_module_name name)  [inline]
```

Target (p. 50).

Target (p. 50) Constructure

Parameters

name	Reference to sc_module name

References **Target**< **BUSWIDTH** >**::nb_transport_fw()**, and **Target**< **BUSWIDTH** >**::target_socket**.

4.10.2 Member Function Documentation

4.10.2.1 nb_transport_fw()

```
template<unsigned int BUSWIDTH = 32>
tlm::tlm_sync_enum nb_transport_fw (
            tlm::tlm_generic_payload & trans,
            tlm::tlm_phase & phase,
            sc_core::sc_time & delay)  [inline], [private]
```

nb_transport_fw

Implements the non-blocking backward transport interface for the initiator.

Parameters

trans	Reference to the generic payload object containing the transaction details such as command, address, and data.
phase	Reference to the transaction phase. The current phase of the transaction, which may be updated by the function.
delay	Reference to the annotated delay. Specifies the timing delay for the transaction and may be updated by the function.

Returns

tlm::tlm_sync_enum Enumeration indicating the synchronization state of the transaction:

- TLM_ACCEPTED: Transaction is accepted, and no immediate further action is required.
- TLM_UPDATED: Transaction phase has been updated. The initiator should check the new phase.
- TLM_COMPLETED: Transaction is completed immediately, and no further phases will occur.

References **Target< BUSWIDTH >::m_name**, and **Target< BUSWIDTH >::target_socket**.

Referenced by **Target< BUSWIDTH >::Target()**.

4.10.3 Member Data Documentation

4.10.3.1 m_name

```
template<unsigned int BUSWIDTH = 32>
std::string m_name  [private]
```

Referenced by **Target< BUSWIDTH >::nb_transport_fw()**.

4.10.3.2 target_socket

```
template<unsigned int BUSWIDTH = 32>
tlm_utils::simple_target_socket< Target, BUSWIDTH> target_socket
```

Referenced by **Target< BUSWIDTH >::Target()**, and **Target< BUSWIDTH >::nb_transport_fw()**.

The documentation for this class was generated from the following file:

- common/ **target.h**

Chapter 5

File Documentation

5.1 common/bus.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <tlm_utils/peq_with_cb_and_phase.h>
#include <map>
#include <vector>
#include <mutex>
```

Classes

- class **BUS< BUSWIDTH, DATA_WIDTH >**
- struct **BUS< BUSWIDTH, DATA_WIDTH >::address**

Typedefs

- typedef sc_core::sc_in< bool > **sc_clk_in**

Enumerations

- enum **BUS_TYPE** { **AXI32** = 32 , **AXI64** = 64 , **APB** = 32 }
- enum **DATAWIDTH** {
  D8BIT = 8 , **D16BIT** = 16 , **D32BIT** = 32 , **D64BIT** = 64 ,
  D128BIT = 128 }

5.1.1 Typedef Documentation

5.1.1.1 sc_clk_in

```
typedef sc_core::sc_in<bool>  sc_clk_in
```

5.1.2 Enumeration Type Documentation

5.1.2.1 BUS_TYPE

enum **BUS_TYPE**

Enumerator

AXI32	
AXI64	
APB	

5.1.2.2 DATAWIDTH

enum **DATAWIDTH**

Enumerator

D8BIT	
D16BIT	
D32BIT	
D64BIT	
D128BIT	

5.2 common/DMAC.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <map>
#include <vector>
#include <iostream>
#include <list>
#include <cstdint>
#include "Registerif.h"
```

Classes

- class **DMAC**< **BUSWIDTH** >

Macros

- #define **DMA_MAX_CH** 256
- #define **DMAACK**(i) (0xC20 + 0x04∗(i))
- #define **DMACHEN**(i) (0xC60 + 0x04∗(i))
- #define **DMADATALENGTH**(i) (0x800 + 0x04∗(i))
- #define **DMADESADDR**(i) (0x00 + 0x04∗(i))
- #define **DMAINT**(i) (0xC40 + 0x04∗(i))
- #define **DMAREQ**(i) (0xC00 + 0x04∗(i))
- #define **DMASRCADDR**(i) (0x400 + 0x04∗(i))

5.2.1 Macro Definition Documentation

5.2.1.1 DMA_MAX_CH

```
#define DMA_MAX_CH 256
```

Referenced by **DMAC< BUSWIDTH >::DMAC()**, **DMAC< BUSWIDTH >::init_registers()**, and **DMAC< BUSWIDTH >::mth_request_signals()**.

5.2.1.2 DMAACK

```
#define DMAACK(
              i) (0xC20 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

5.2.1.3 DMACHEN

```
#define DMACHEN(
              i) (0xC60 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, **DMAC< BUSWIDTH >::mth_request_signals()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

5.2.1.4 DMADATALENGTH

```
#define DMADATALENGTH(
              i) (0x800 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DMAC< BUSWIDTH >::thr_DMA_run_process()**.

5.2.1.5 DMADESADDR

```
#define DMADESADDR(
              i) (0x00 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DMAC< BUSWIDTH >::nb_transport_bw()**.

5.2.1.6 DMAINT

```
#define DMAINT(
              i) (0xC40 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, and **DMAC< BUSWIDTH >::thr_priority_process()**.

5.2.1.7 DMAREQ

```
#define DMAREQ(
              i) (0xC00 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**.

5.2.1.8 DMASRCADDR

```
#define DMASRCADDR(
              i) (0x400 + 0x04*(i))
```

Referenced by **DMAC< BUSWIDTH >::init_registers()**, **DMAC< BUSWIDTH >::nb_transport_bw()**, and **DMAC< BUSWIDTH >::thr_DMA_run_process()**.

5.3 common/DummyMaster.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <map>
#include <vector>
#include <iostream>
#include <cstdint>
#include "Registerif.h"
#include "bus.h"
```

Classes

- class **DummyMaster< BUSWIDTH >**
- struct **DummyMaster< BUSWIDTH >::localdata**

5.4 common/DummySlave.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <map>
#include <vector>
#include <iostream>
#include <cstdint>
#include "Registerif.h"
```

Classes

- class **DummySlave**< **BUSWIDTH** >

Macros

- #define **DUMMYRESULT** 0x00

5.4.1 Macro Definition Documentation

5.4.1.1 DUMMYRESULT

```
#define DUMMYRESULT 0x00
```

Referenced by **DummySlave**< **BUSWIDTH** >**::init_registers()**.

5.5 common/Inverter.h File Reference

```
#include <systemc.h>
```

Functions

- **SC_MODULE** (Inverter)

5.5.1 Function Documentation

5.5.1.1 SC_MODULE()

```
SC_MODULE (
            Inverter )
```

5.6 common/memory.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <map>
#include <vector>
#include <iostream>
#include <cstdint>
#include <iomanip>
```

Classes

- class **RAM**< **BUSWIDTH** >

5.7 common/Registerif.h File Reference

```
#include <unordered_map>
#include <map>
#include <string>
#include <cstdint>
#include <stdexcept>
#include <memory>
#include <stdio.h>
#include <iostream>
```

Classes

- class **Register**
- class **RegisterInterface**

Enumerations

- enum **REGPERMISSION** { **READWRITE** = 0 , **READONLY** = 1 }

5.7.1 Enumeration Type Documentation

5.7.1.1 REGPERMISSION

enum **REGPERMISSION**

Enumerator

READWRITE	
READONLY	

5.8 common/target.h File Reference

```
#include <systemc>
#include <tlm>
#include <tlm_utils/simple_initiator_socket.h>
#include <tlm_utils/simple_target_socket.h>
#include <tlm_utils/multi_passthrough_initiator_socket.h>
#include <tlm_utils/multi_passthrough_target_socket.h>
#include <map>
#include <vector>
#include <iostream>
#include <cstdint>
```

Classes

- class **Target**< **BUSWIDTH** >

Index