# qbox

Generated by Doxygen 1.9.1

# Chapter 1

# LIBQBOX

Libqbox encapsulates QEMU in SystemC such that it can be instanced as a SystemC TLM-2.0 model.

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

# Class Documentation

## 4.1 QemuInstanceDmiManager::DmiInfo Struct Reference

### Public Types

- using **Key** = uintptr_t

### Public Member Functions

- **DmiInfo** (const tlm::tlm_dmi &info)
- uint64_t **get_size** () const
- Key **get_key** () const

### Public Attributes

- uint64_t **start**
- uint64_t **end**
- void * **ptr**
- qemu::MemoryRegion **mr**

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/dmi-manager.h

## 4.2 LockedQemuInstanceDmiManager Class Reference

A locked QemuInstanceDmiManager.

`#include <dmi-manager.h>`

### Public Types

- using **DmiInfo** = QemuInstanceDmiManager::DmiInfo

### Public Member Functions

- **LockedQemuInstanceDmiManager** (QemuInstanceDmiManager &inst)
- **LockedQemuInstanceDmiManager** (const LockedQemuInstanceDmiManager &)=delete
- **LockedQemuInstanceDmiManager** (LockedQemuInstanceDmiManager &&)=default
- void get_global_mr (DmiInfo &info)

### Protected Attributes

- QemuInstanceDmiManager & **m_inst**
- std::unique_lock< std::mutex > **m_lock**

### 4.2.1 Detailed Description

A locked QemuInstanceDmiManager.

This class is a wrapper around QemuInstanceDmiManager that ensure safe accesses to it. As long as an instance of this class is live, the underlying QemuInstanceDmiManager is locked. It gets unlocked once the object goes out of scope.

### 4.2.2 Member Function Documentation

#### 4.2.2.1 get_global_mr()

```
void LockedQemuInstanceDmiManager::get_global_mr (
            DmiInfo & info )  [inline]
```

**See also**

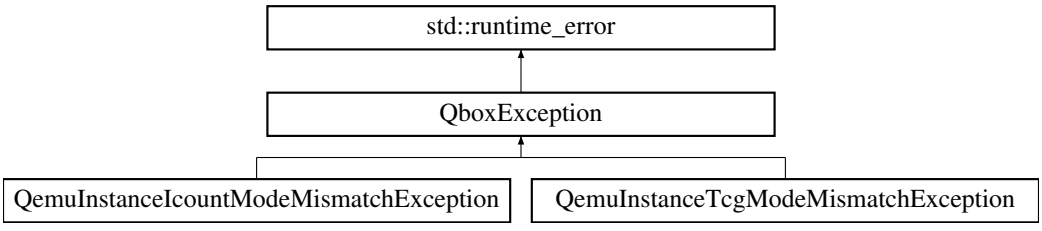
QemuInstanceDmiManager::get_global_mr

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/dmi-manager.h

## 4.3 QboxException Class Reference

Inheritance diagram for QboxException:



### Public Member Functions

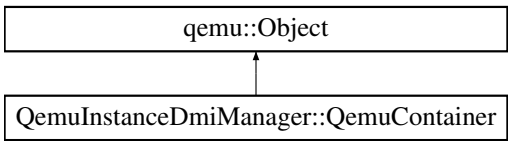
- **QboxException** (const char *what)

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/exceptions.h

## 4.4 QemuInstanceDmiManager::QemuContainer Class Reference

Inheritance diagram for QemuInstanceDmiManager::QemuContainer:




### Public Member Functions

- **QemuContainer** (const QemuContainer &o)=default
- **QemuContainer** (const Object &o)

### Static Public Attributes

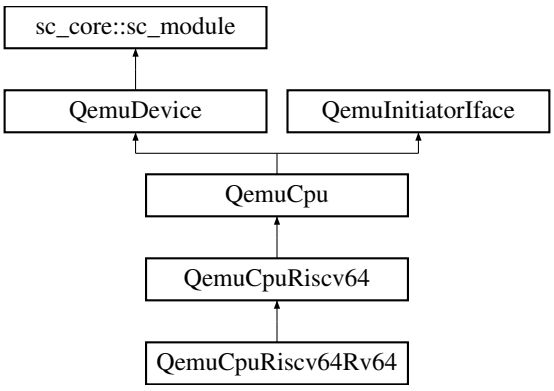
- static constexpr const char *const **TYPE** = "container"

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/dmi-manager.h

## 4.5 QemuCpu Class Reference

Inheritance diagram for QemuCpu:



## Public Member Functions

- **SC_HAS_PROCESS** (QemuCpu)
- **QemuCpu** (const sc_core::sc_module_name &name, QemuInstance &inst, const std::string &type_name)
- void **before_end_of_elaboration** () override
- virtual void **end_of_elaboration** () override
- virtual void **start_of_simulation** () override
- virtual void **initiator_customize_tlm_payload** (TlmPayload &payload) override
- virtual sc_core::sc_time **initiator_get_local_time** () override
- virtual void **initiator_set_local_time** (const sc_core::sc_time &t) override

## Public Attributes

- cci::cci_param< bool > **p_icount**
- cci::cci_param< int > **p_icount_mips**
- cci::cci_param< unsigned int > **p_gdb_port**
- cci::cci_param< std::string > **p_sync_policy**
- QemuInitiatorSocket **socket**

## Protected Member Functions

- void **create_quantum_keeper** ()
- void **set_qemu_instance_options** ()
- void **deadline_timer_cb** ()
- void **wait_for_work** ()
- void **rearm_deadline_timer** ()
- void **prepare_run_cpu** ()
- void **run_cpu_loop** ()
- void **sync_with_kernel** ()
- void **kick_cb** ()
- void **end_of_loop_cb** ()
- void **mainloop_thread_coroutine** ()

## Protected Attributes

- gs::RunOnSysC **m_on_sysc**
- std::shared_ptr< qemu::Timer > **m_deadline_timer**
- bool **m_coroutines**
- qemu::Cpu **m_cpu**
- gs::async_event **m_qemu_kick_ev**
- sc_core::sc_event_or_list **m_external_ev**
- int64_t **m_last_vclock**
- std::shared_ptr< gs::tlm_quantumkeeper_extended > **m_qk**

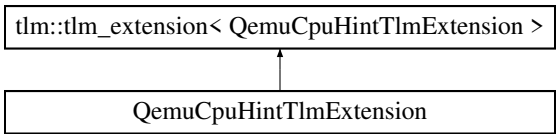
## Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/components/cpu/cpu.h

## 4.6 QemuCpuHintTlmExtension Class Reference

Inheritance diagram for QemuCpuHintTlmExtension:




### Public Member Functions

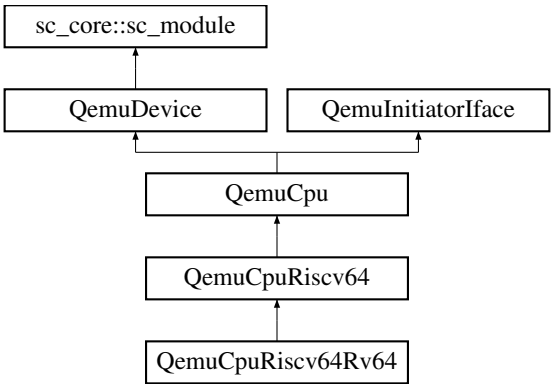
- **QemuCpuHintTlmExtension** (const QemuCpuHintTlmExtension &)=default
- **QemuCpuHintTlmExtension** (qemu::Cpu cpu)
- virtual tlm_extension_base * **clone** () const override
- virtual void **copy_from** (tlm_extension_base const &ext) override
- qemu::Cpu **get_cpu** () const

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/tlm-extensions/qemu-cpu-hint.h

## 4.7 QemuCpuRiscv64 Class Reference

Inheritance diagram for QemuCpuRiscv64:




### Public Member Functions

- **QemuCpuRiscv64** (const sc_core::sc_module_name &name, QemuInstance &inst, const char *model, uint64_t hartid)
- void **before_end_of_elaboration** ()

### Protected Member Functions

- void **mip_update_cb** (uint32_t value)

### Protected Attributes

- uint64_t **m_hartid**
- gs::async_event **m_irq_ev**

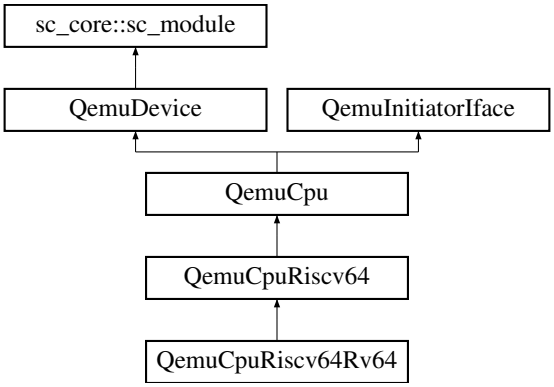
### Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

## 4.8 QemuCpuRiscv64Rv64 Class Reference

Inheritance diagram for QemuCpuRiscv64Rv64:




### Public Member Functions

- **QemuCpuRiscv64Rv64** (const sc_core::sc_module_name &n, QemuInstance &inst, uint64_t hartid)

### Additional Inherited Members

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

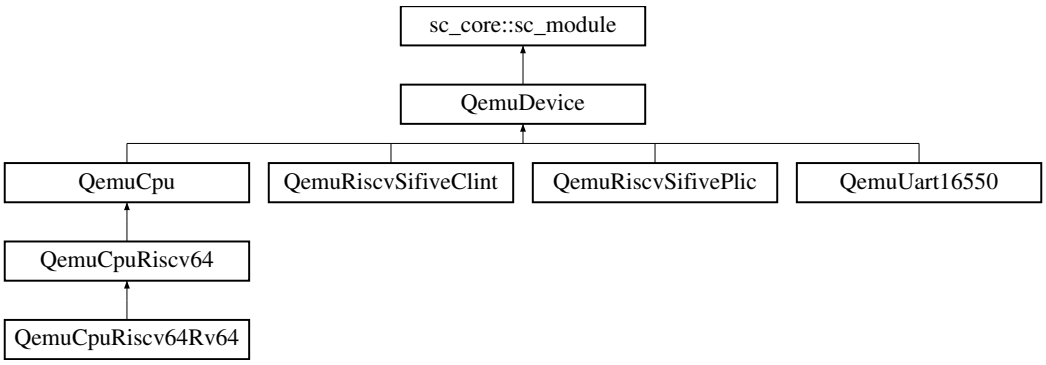
- /Users/mark/work/libqbox/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

## 4.9 QemuDevice Class Reference

QEMU device abstraction as a SystemC module.

`#include <device.h>`

Inheritance diagram for QemuDevice:




### Public Member Functions

- QemuDevice (const sc_core::sc_module_name &name, QemuInstance &inst, const char *qom_type)
  *Construct a QEMU device.*
- virtual void **before_end_of_elaboration** () override
- virtual void **end_of_elaboration** () override
- const char * **get_qom_type** () const
- qemu::Device **get_qemu_dev** ()
- QemuInstance & **get_qemu_inst** ()

### Protected Member Functions

- void **realize** ()

### Protected Attributes

- QemuInstance & **m_inst**
- qemu::Device **m_dev**
- bool **m_realized** = false

### 4.9.1 Detailed Description

QEMU device abstraction as a SystemC module.

This class abstract a QEMU device as a SystemC module. It is constructed using the QEMU instance it will lie in, and the QOM type name corresponding to the device. This class is meant to be inherited from by children classes that implement a given device.

The elaboration flow is as follows:

- At construct time, nothing happen on the QEMU side.
- When before_end_of_elaboration is called, the QEMU object correponding to this component is created. Children classes should always call the parent method when overriding it. Usually, they start by calling it and then set the QEMU properties on the device.
- When end_of_elaboration is called, the device is realized. No more property can be set (unless particular cases such as some link properties) and the device can now be connected to busses and GPIO.

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 QemuDevice()

```
QemuDevice::QemuDevice (
            const sc_core::sc_module_name & name,
            QemuInstance & inst,
            const char * qom_type )  [inline]
```

Construct a QEMU device.

**Parameters**

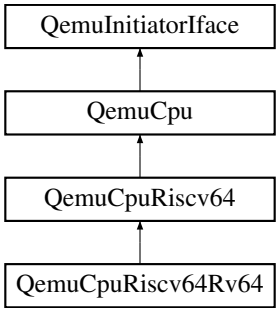
| | | |
|---|---|---|
| in | *name* | SystemC module name |
| in | *inst* | QEMU instance the device will be created in |
| in | *qom_type* | Device QOM type name |

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/components/device.h

## 4.10 QemuInitiatorIface Class Reference

Inheritance diagram for QemuInitiatorIface:




### Public Types

- using **TlmPayload** = tlm::tlm_generic_payload

### Public Member Functions

- virtual void **initiator_customize_tlm_payload** (TlmPayload &payload)=0
- virtual sc_core::sc_time **initiator_get_local_time** ()=0
- virtual void **initiator_set_local_time** (const sc_core::sc_time &)=0

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

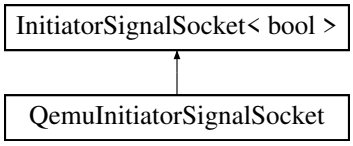
- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/initiator.h

## 4.11 QemuInitiatorSignalSocket Class Reference

A QEMU output GPIO exposed as a InitiatorSignalSocket<bool>

`#include <initiator-signal-socket.h>`

Inheritance diagram for QemuInitiatorSignalSocket:




### Public Member Functions

- **QemuInitiatorSignalSocket** (const char *name)
- void init (qemu::Device dev, int gpio_idx)

  *Initialize this socket with a device and a GPIO index.*
- void init_named (qemu::Device dev, const char *gpio_name, int gpio_idx)

  *Initialize this socket with a device, a GPIO namespace, and a GPIO index.*
- void init_sbd (qemu::SysBusDevice sbd, int gpio_idx)

  *Initialize this socket with a QEMU SysBusDevice, and a GPIO index.*

### Protected Member Functions

- void **event_cb** (bool val)
- void **init_qemu_to_sysc_gpio_proxy** (qemu::Device &dev)
- void **init_internal** (qemu::Device &dev)

### Protected Attributes

- qemu::Gpio **m_proxy**
- gs::RunOnSysC **m_on_sysc**
- QemuTargetSignalSocket * **m_qemu_remote** = nullptr

### 4.11.1 Detailed Description

A QEMU output GPIO exposed as a InitiatorSignalSocket<bool>

This class exposes an output GPIO of a QEMU device as a InitiatorSignalSocket<bool>. It can be connected to an sc_core::sc_port<bool> or a TargetSignalSocket<bool>. Modifications to the interal QEMU GPIO will be propagated through the socket.

If this socket happens to be connected to a QemuTargetSignalSocket, the propagation is done directly within QEMU and do not go through the SystemC kernel. Note that this is only true if the GPIOs wrapped by both this socket and the remote socket lie in the same QEMU instance.

### 4.11.2 Member Function Documentation

#### 4.11.2.1 init()

```
void QemuInitiatorSignalSocket::init (
            qemu::Device dev,
            int gpio_idx )  [inline]
```

Initialize this socket with a device and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO index in this device. See the QEMU API and the device you want to wrap to know what index to use here.

**Parameters**

| | | |
|---|---|---|
| in | *dev* | The QEMU device |
| in | *gpio_idx* | The GPIO index within the device |

#### 4.11.2.2 init_named()

```
void QemuInitiatorSignalSocket::init_named (
            qemu::Device dev,
            const char * gpio_name,
            int gpio_idx )  [inline]
```

Initialize this socket with a device, a GPIO namespace, and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO (namespace, index) pair in this device. See the QEMU API and the device you want to wrap to know what namespace/index to use here.

**Parameters**

| | | |
|---|---|---|
| in | *dev* | The QEMU device |
| in | *gpio_name* | The GPIO namespace within the device |
| in | *gpio_idx* | The GPIO index within the device |

#### 4.11.2.3 init_sbd()

```
void QemuInitiatorSignalSocket::init_sbd (
            qemu::SysBusDevice sbd,
            int gpio_idx )  [inline]
```

Initialize this socket with a QEMU SysBusDevice, and a GPIO index.

This method initializes the socket using the given QEMU SysBusDevice (SBD) and the corresponding GPIO index) in this SBD. See the QEMU API and the SBD you want to wrap to know what index to use here.

**Parameters**

| | | |
|---|---|---|
| in | *sbd* | The QEMU SysBusDevice |
| in | *gpio_idx* | The GPIO index within the SBD |

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

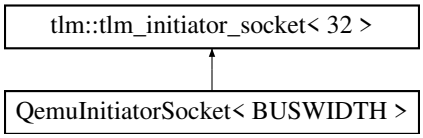
- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/initiator-signal-socket.h

## 4.12 QemuInitiatorSocket< BUSWIDTH > Class Template Reference

TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping.

`#include <initiator.h>`

Inheritance diagram for QemuInitiatorSocket< BUSWIDTH >:




### Public Types

- using **TlmInitiatorSocket** = tlm::tlm_initiator_socket< BUSWIDTH >
- using **TlmPayload** = tlm::tlm_generic_payload
- using **MemTxResult** = qemu::MemoryRegionOps::MemTxResult
- using **MemTxAttrs** = qemu::MemoryRegionOps::MemTxAttrs
- using **DmiInfo** = QemuInstanceDmiManager::DmiInfo

### Public Member Functions

- **QemuInitiatorSocket** (const char *name, QemuInitiatorIface &initiator, QemuInstance &inst)
- void **init** (qemu::Device &dev, const char *prop)

### Protected Member Functions

- void **init_payload** (TlmPayload &trans, tlm::tlm_command command, uint64_t addr, uint64_t *val, unsigned int size)
- void **add_dmi_mr_alias** (DmiInfo &info)
- void **check_dmi_hint** (TlmPayload &trans)
- void **check_qemu_mr_hint** (TlmPayload &trans)
- void **do_regular_access** (TlmPayload &trans)
- void **do_debug_access** (TlmPayload &trans)
- MemTxResult **qemu_io_access** (tlm::tlm_command command, uint64_t addr, uint64_t *val, unsigned int size, MemTxAttrs attrs)
- MemTxResult **qemu_io_read** (uint64_t addr, uint64_t *val, unsigned int size, MemTxAttrs attrs)
- MemTxResult **qemu_io_write** (uint64_t addr, uint64_t val, unsigned int size, MemTxAttrs attrs)

### Protected Attributes

- QemuToTlmInitiatorBridge **m_bridge**
- QemuInstance & **m_inst**
- QemuInitiatorIface & **m_initiator**
- qemu::Device **m_dev**
- gs::RunOnSysC **m_on_sysc**
- qemu::MemoryRegion **m_root**

### 4.12.1 Detailed Description

**template<unsigned int BUSWIDTH = 32>**
**class QemuInitiatorSocket< BUSWIDTH >**

TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping.

This class is used to expose a QEMU AddressSpace object as a standard TLM-2.0 initiator socket. It creates a root memory region to map the whole address space, receives I/O accesses to it and forwards them as standard TLM-2.0 transactions.

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/initiator.h

## 4.13 QemuInstance Class Reference

This class encapsulates a libqemu-cxx qemu::LibQemu instance. It handles QEMU parameters and instance initialization.

`#include <qemu-instance.h>`

### Public Types

- enum **TcgMode** { **TCG_UNSPECIFIED** , **TCG_SINGLE** , **TCG_SINGLE_COROUTINE** , **TCG_MULTI** }
- enum **IcountMode** { **ICOUNT_UNSPECIFIED** , **ICOUNT_OFF** , **ICOUNT_ON** }
- using **Target** = qemu::Target
- using **LibLoader** = qemu::LibraryLoaderIface

### Public Member Functions

- **QemuInstance** (LibLoader &loader, Target t)
- **QemuInstance** (const QemuInstance &)=delete
- **QemuInstance** (QemuInstance &&)=default
- void set_tcg_mode (TcgMode m)

  *Set the desired TCG mode for this instance.*
- void set_icount_mode (IcountMode m, int mips_shift)

  *Set the desired icount mode for this instance.*
- void init ()

  *Initialize the QEMU instance.*
- bool is_inited () const

  *Returns true if the instance is initialized.*
- qemu::LibQemu & get ()

  *Returns the underlying qemu::LibQemu instance.*
- LockedQemuInstanceDmiManager get_dmi_manager ()

  *Returns the locked QemuInstanceDmiManager instance.*

### Protected Member Functions

- void **push_default_args** ()
- void **push_icount_mode_args** ()
- void **push_tcg_mode_args** ()

### Protected Attributes

- qemu::LibQemu **m_inst**
- QemuInstanceDmiManager **m_dmi_mgr**
- TcgMode **m_tcg_mode** = TCG_UNSPECIFIED
- IcountMode **m_icount_mode** = ICOUNT_UNSPECIFIED
- int **m_icount_mips** = 0

## 4.13.1 Detailed Description

This class encapsulates a libqemu-cxx qemu::LibQemu instance. It handles QEMU parameters and instance initialization.

## 4.13.2 Member Function Documentation

### 4.13.2.1 get()

```
qemu::LibQemu& QemuInstance::get ( )  [inline]
```

Returns the underlying qemu::LibQemu instance.

Returns the underlying qemu::LibQemu instance. If the instance hasn't been initialized, init is called just before returning the instance.

### 4.13.2.2 get_dmi_manager()

```
LockedQemuInstanceDmiManager QemuInstance::get_dmi_manager ( )  [inline]
```

Returns the locked QemuInstanceDmiManager instance.

Note: we rely on RVO here so no copy happen on return (this is enforced by the fact that the LockedQemuInstanceDmiManager copy constructor is deleted).

### 4.13.2.3 init()

```
void QemuInstance::init ( )  [inline]
```

Initialize the QEMU instance.

Initialize the QEMU instance with the set TCG and icount mode. If the TCG mode hasn't been set, it defaults to TCG_SINGLE. If icount mode hasn't been set, it defaults to ICOUNT_OFF.

The instance should not already be initialized when calling this method.

#### 4.13.2.4 set_icount_mode()

```
void QemuInstance::set_icount_mode (
            IcountMode m,
            int mips_shift )  [inline]
```

Set the desired icount mode for this instance.

This method is called by CPU instances to specify the desired icount mode according to the synchronization policy in use. All CPUs should use the same mode.

This method should be called before the instance is initialized.

**Parameters**

| | | |
|---|---|---|
| in | *m* | The desired icount mode |
| in | *mips_shift* | The QEMU icount shift parameter. It sets the virtual time an instruction takes to execute to 2^(mips_shift) ns. |

#### 4.13.2.5 set_tcg_mode()

```
void QemuInstance::set_tcg_mode (
            TcgMode m )  [inline]
```

Set the desired TCG mode for this instance.

This method is called by CPU instances to specify the desired TCG mode according to the synchronization policy in use. All CPUs should use the same mode (meaning they should all use synchronization policies compatible one with the other).

This method should be called before the instance is initialized.

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/qemu-instance.h

## 4.14 QemuInstanceDmiManager Class Reference

Handles the DMI regions at the QEMU instance level.

```
#include <dmi-manager.h>
```

### Classes

- struct DmiInfo
- class QemuContainer

### Public Member Functions

- **QemuInstanceDmiManager** (qemu::LibQemu &inst)
- **QemuInstanceDmiManager** (const QemuInstanceDmiManager &)=delete
- **QemuInstanceDmiManager** (QemuInstanceDmiManager &&a)
- void **init** ()
- void get_global_mr (DmiInfo &info)

  *Fill DMI info with the corresponding global memory region.*

### Protected Member Functions

- void **create_global_mr** (DmiInfo &info)

### Protected Attributes

- qemu::LibQemu & **m_inst**
- QemuContainer **m_mr_container**
- std::mutex **m_mutex**
- std::map< DmiInfo::Key, DmiInfo > **m_dmis**

### Friends

- class **LockedQemuInstanceDmiManager**

## 4.14.1 Detailed Description

Handles the DMI regions at the QEMU instance level.

This class handles the DMI regions at the level of a QEMU instance. For a given DMI region, we need to use a unique memory region (called the global memory region, in a sense that it is global to all the CPUs in the instance). Each CPU is then supposed to create an alias to this region to be able to access it. This is required to ensure QEMU sees this region as a unique piece of memory. Creating multiple regions mapping to the same host address leads QEMU into thinking that those are different data, and it won't properly invalidate corresponding TBs if CPUs do SMC (self modifying code).

## 4.14.2 Member Function Documentation

### 4.14.2.1 get_global_mr()

```
void QemuInstanceDmiManager::get_global_mr (
            DmiInfo & info )  [inline]
```

Fill DMI info with the corresponding global memory region.

Fill the DMI info 'info' with the global memory region matching the rest of 'info'. If the global MR does not exist yet, it is created. Otherwise the already existing one is returned.

**Parameters**

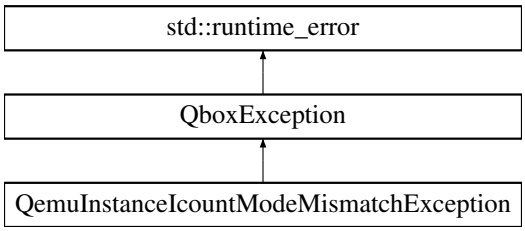
| | | |
|---|---|---|
| in,out | *info* | The DMI info to use and to fill with the global MR |

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/dmi-manager.h

## 4.15 QemuInstanceIcountModeMismatchException Class Reference

Inheritance diagram for QemuInstanceIcountModeMismatchException:




### Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/qemu-instance.h

## 4.16 QemuInstanceManager Class Reference

QEMU instance manager class.

`#include <qemu-instance.h>`

### Public Types

- using **Target** = qemu::Target
- using **LibLoader** = qemu::LibraryLoaderIface

### Public Member Functions

- QemuInstanceManager ()

  *Construct a QemuInstanceManager. The manager will use the default library loader provided by libqemu-cxx.*
- QemuInstanceManager (LibLoader ∗loader)

  *Construct a QemuInstanceManager by providing a custom library loader.*
- QemuInstance & new_instance (Target t)

  *Returns a new QEMU instance for target t.*

### Protected Attributes

- LibLoader * **m_loader**
- std::vector< QemuInstance > **m_insts**

## 4.16.1 Detailed Description

QEMU instance manager class.

This class manages QEMU instances. It allows to create instances using the same library loader, thus allowing multiple instances of the same library being loaded.

## 4.16.2 Constructor & Destructor Documentation

### 4.16.2.1 QemuInstanceManager()

```
QemuInstanceManager::QemuInstanceManager (
            LibLoader * loader )  [inline]
```

Construct a QemuInstanceManager by providing a custom library loader.

**Parameters**

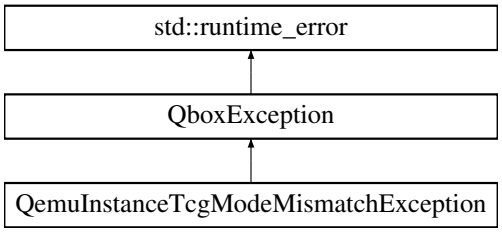
| | | |
|---|---|---|
| in | *loader* | The custom loader |

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/qemu-instance.h

# 4.17 QemuInstanceTcgModeMismatchException Class Reference

Inheritance diagram for QemuInstanceTcgModeMismatchException:



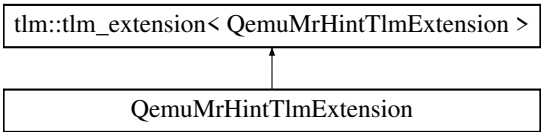

### Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/qemu-instance.h

## 4.18 QemuMrHintTlmExtension Class Reference

Inheritance diagram for QemuMrHintTlmExtension:




### Public Member Functions

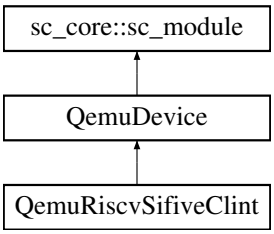
- **QemuMrHintTlmExtension** (const QemuMrHintTlmExtension &)=default
- **QemuMrHintTlmExtension** (qemu::MemoryRegion mr, uint64_t offset)
- virtual tlm_extension_base * **clone** () const override
- virtual void **copy_from** (tlm_extension_base const &ext) override
- qemu::MemoryRegion **get_mr** () const
- uint64_t **get_offset** () const

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/tlm-extensions/qemu-mr-hint.h

## 4.19 QemuRiscvSifiveClint Class Reference

Inheritance diagram for QemuRiscvSifiveClint:




### Public Member Functions

- **QemuRiscvSifiveClint** (sc_core::sc_module_name nm, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

### Public Attributes

- cci::cci_param< unsigned int > **p_num_harts**
- cci::cci_param< uint64_t > **p_sip_base**
- cci::cci_param< uint64_t > **p_timecmp_base**
- cci::cci_param< uint64_t > **p_time_base**
- cci::cci_param< bool > **p_provide_rdtime**
- cci::cci_param< uint64_t > **p_aperture_size**
- QemuTargetSocket **socket**

### Protected Attributes

- uint64_t **m_aperture_size**
- int **m_num_harts**

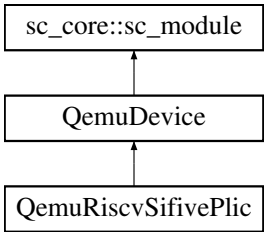
### Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/components/irq-ctrl/clint-sifive.h

## 4.20 QemuRiscvSifivePlic Class Reference

Inheritance diagram for QemuRiscvSifivePlic:




### Public Member Functions

- **QemuRiscvSifivePlic** (sc_core::sc_module_name nm, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

### Public Attributes

- cci::cci_param< unsigned int > **p_num_sources**
- cci::cci_param< unsigned int > **p_num_priorities**
- cci::cci_param< uint64_t > **p_priority_base**
- cci::cci_param< uint64_t > **p_pending_base**
- cci::cci_param< uint64_t > **p_enable_base**
- cci::cci_param< uint64_t > **p_enable_stride**
- cci::cci_param< uint64_t > **p_context_base**
- cci::cci_param< uint64_t > **p_context_stride**
- cci::cci_param< uint64_t > **p_aperture_size**
- cci::cci_param< std::string > **p_hart_config**
- QemuTargetSocket **socket**
- sc_core::sc_vector< QemuTargetSignalSocket > **irq_in**

### Additional Inherited Members

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

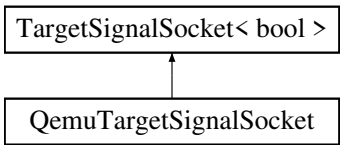
- /Users/mark/work/libqbox/libqbox/include/libqbox/components/irq-ctrl/plic-sifive.h

## 4.21 QemuTargetSignalSocket Class Reference

A QEMU input GPIO exposed as a TargetSignalSocket<bool>

`#include <target-signal-socket.h>`

Inheritance diagram for QemuTargetSignalSocket:




### Public Member Functions

- **QemuTargetSignalSocket** (const char *name)
- void init (qemu::Device dev, int gpio_idx)

  *Initialize this socket with a device and a GPIO index.*
- void init_named (qemu::Device dev, const char *gpio_name, int gpio_idx)

  *Initialize this socket with a device, a GPIO namespace, and a GPIO index.*
- qemu::Gpio get_gpio ()

  *Returns the GPIO wrapped by this socket.*

### Protected Member Functions

- void **value_changed_cb** (const bool &val)
- void **init_with_gpio** (qemu::Gpio gpio)

### Protected Attributes

- qemu::Gpio **m_gpio_in**

### 4.21.1 Detailed Description

A QEMU input GPIO exposed as a TargetSignalSocket<bool>

This class exposes an input GPIO of a QEMU device as a TargetSignalSocket<bool>. It can be connected to an sc_core::sc_port<bool> or a TargetInitiatorSocket<bool>. Modifications to this socket will be reported to the wrapped GPIO.

## 4.21.2 Member Function Documentation

### 4.21.2.1 get_gpio()

```
qemu::Gpio QemuTargetSignalSocket::get_gpio ( )  [inline]
```

Returns the GPIO wrapped by this socket.

**Returns**

the GPIO wrapped by this socket

### 4.21.2.2 init()

```
void QemuTargetSignalSocket::init (
            qemu::Device dev,
            int gpio_idx )  [inline]
```

Initialize this socket with a device and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO index in this device. See the QEMU API and the device you want to wrap to know what index to use here.

**Parameters**

| | | |
|---|---|---|
| in | *dev* | The QEMU device |
| in | *gpio_idx* | The GPIO index within the device |

### 4.21.2.3 init_named()

```
void QemuTargetSignalSocket::init_named (
            qemu::Device dev,
            const char * gpio_name,
            int gpio_idx )  [inline]
```

Initialize this socket with a device, a GPIO namespace, and a GPIO index.

This method initializes the socket using the given QEMU device and the corresponding GPIO (namespace, index) pair in this device. See the QEMU API and the device you want to wrap to know what namespace/index to use here.

**Parameters**

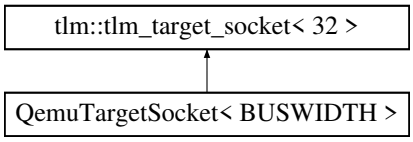
| | | |
|---|---|---|
| in | *dev* | The QEMU device |
| in | *gpio_name* | The GPIO namespace within the device |
| in | *gpio_idx* | The GPIO index within the device |

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/target-signal-socket.h

## 4.22 QemuTargetSocket< BUSWIDTH > Class Template Reference

Inheritance diagram for QemuTargetSocket< BUSWIDTH >:




### Public Types

- using **TlmTargetSocket** = tlm::tlm_target_socket< BUSWIDTH >
- using **TlmPayload** = tlm::tlm_generic_payload

### Public Member Functions

- **QemuTargetSocket** (const char *name, QemuInstance &inst)
- void **init** (qemu::SysBusDevice sbd, int mmio_idx)

### Protected Attributes

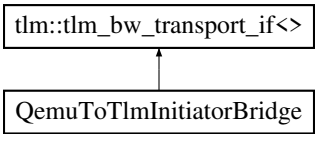
- TlmTargetToQemuBridge **m_bridge**
- QemuInstance & **m_inst**
- qemu::SysBusDevice **m_sbd**

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/target.h

## 4.23 QemuToTlmInitiatorBridge Class Reference

Inheritance diagram for QemuToTlmInitiatorBridge:



### Public Member Functions

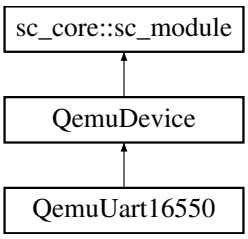
- virtual tlm::tlm_sync_enum **nb_transport_bw** (tlm::tlm_generic_payload &trans, tlm::tlm_phase &phase, sc_core::sc_time &t)
- virtual void **invalidate_direct_mem_ptr** (sc_dt::uint64 start_range, sc_dt::uint64 end_range)

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/initiator.h

## 4.24 QemuUart16550 Class Reference

Inheritance diagram for QemuUart16550:




### Public Member Functions

- **QemuUart16550** (const sc_core::sc_module_name &n, QemuInstance &inst)
- void **before_end_of_elaboration** () override
- void **end_of_elaboration** () override

### Public Attributes

- QemuTargetSocket **socket**
- QemuInitiatorSignalSocket **irq_out**

### Protected Attributes

- qemu::Chardev **m_chardev**
- cci::cci_param< unsigned int > **p_baudbase**
- cci::cci_param< unsigned int > **p_regshift**

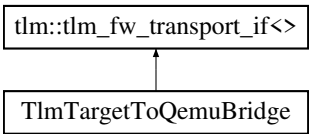
### Additional Inherited Members

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/components/uart/16550.h

## 4.25 TlmTargetToQemuBridge Class Reference

Inheritance diagram for TlmTargetToQemuBridge:




### Public Types

- using **MemTxAttrs** = qemu::MemoryRegion::MemTxAttrs
- using **MemTxResult** = qemu::MemoryRegion::MemTxResult
- using **TlmPayload** = tlm::tlm_generic_payload

### Public Member Functions

- void **init** (qemu::SysBusDevice sbd, int mmio_idx)
- virtual void **b_transport** (TlmPayload &trans, sc_core::sc_time &t)
- virtual tlm::tlm_sync_enum **nb_transport_fw** (TlmPayload &trans, tlm::tlm_phase &phase, sc_core::sc_time &t)
- virtual bool **get_direct_mem_ptr** (TlmPayload &trans, tlm::tlm_dmi &dmi_data)
- virtual unsigned int **transport_dbg** (TlmPayload &trans)

### Protected Member Functions

- qemu::Cpu **push_current_cpu** (TlmPayload &trans)
- void **pop_current_cpu** (qemu::Cpu cpu)

### Protected Attributes

- qemu::MemoryRegion **m_mr**

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

- /Users/mark/work/libqbox/libqbox/include/libqbox/ports/target.h

# Index