

qbox

Generated by Doxygen 1.8.17



<b>1 Main Page</b>	<b>1</b>
1.0.1 LIBQBOX	1
1.1 GreenSocs Build and make system	1
1.2 How to build	1
1.2.1 cmake version	1
1.2.2 details	1
1.2.2.1 Common CMake options	2
1.2.2.2 passwords for git.greensocs.com	2
1.2.3 More documentation	2
1.2.4 LIBGSSYNC	2
1.2.5 The GreenSocs SystemC simple components library.	2
1.2.6 LIBGSUTILS	2
1.2.7 LIBQEMU-CXX	2
1.2.8 Information about building and using the greensocs Qbox library	3
1.2.9 Information about building and using the base-components library	3
1.2.10 Information about building and using the libgssync library	3
1.2.11 Information about building and using the libgsutils library	3
1.2.12 Using yaml for configuration	4
1.2.13 Information about building and using the libqemu-cxx library	4
1.2.14 Instantiate Qemu	4
1.2.15 QEMU Arguments	5
1.2.16 Enabling GDB per CPU	5
1.2.17 The components of libqbox	5
1.2.17.1 CPU	5
1.2.17.2 IRQ-CTRL	6
1.2.17.3 UART	6
1.2.17.4 PORTS	6
1.2.18 The GreenSocs component library memory	6
1.2.19 The GreenSocs component library router	6
1.2.20 Functionality of the synchronization library	6
1.2.20.1 Suspend/Unsuspend interface	7
1.2.21 Using the ConfigurableBroker	7
1.2.22 Print out the available params	8
1.2.23 Halt Interface	8
<b>2 Hierarchical Index</b>	<b>9</b>
2.1 Class Hierarchy	9
<b>3 Class Index</b>	<b>11</b>
3.1 Class List	11
<b>4 Class Documentation</b>	<b>13</b>
4.1 CpuArmCortexM7 Class Reference	13

4.2 CpuArmCortexR5 Class Reference . . . . .	14
4.3 QemuInstanceDmiManager::DmiRegion Class Reference . . . . .	14
4.3.1 Detailed Description . . . . .	15
4.4 QemuInstanceDmiManager::DmiRegionAlias Class Reference . . . . .	15
4.4.1 Detailed Description . . . . .	16
4.4.2 Member Function Documentation . . . . .	16
4.4.2.1 invalidate_region() . . . . .	16
4.4.2.2 is_installed() . . . . .	16
4.4.2.3 is_valid() . . . . .	17
4.4.2.4 set_installed() . . . . .	17
4.5 LockedQemuInstanceDmiManager Class Reference . . . . .	17
4.5.1 Detailed Description . . . . .	18
4.5.2 Member Function Documentation . . . . .	18
4.5.2.1 get_new_region_alias() . . . . .	18
4.6 QemuInitiatorSocket< BUSWIDTH >::m_mem_obj Class Reference . . . . .	18
4.7 QboxException Class Reference . . . . .	18
4.8 QemuArmGicv2 Class Reference . . . . .	19
4.9 QemuArmGicv2m Class Reference . . . . .	20
4.10 QemuArmGicv3 Class Reference . . . . .	21
4.11 QemuArmSmmu Class Reference . . . . .	22
4.12 QemuInstanceDmiManager::QemuContainer Class Reference . . . . .	23
4.13 QemuCpu Class Reference . . . . .	23
4.14 QemuCpuArmCortexA53 Class Reference . . . . .	25
4.15 QemuCpuArmMax Class Reference . . . . .	26
4.16 QemuCpuArmNeoverseN1 Class Reference . . . . .	27
4.17 QemuCpuHexagon Class Reference . . . . .	28
4.18 QemuCpu::QemuCpuHintTlmExtension Class Reference . . . . .	29
4.19 QemuCpuHintTlmExtension Class Reference . . . . .	30
4.20 QemuCpuRiscv64 Class Reference . . . . .	30
4.21 QemuCpuRiscv64Rv64 Class Reference . . . . .	31
4.22 QemuCpuSifiveX280 Class Reference . . . . .	32
4.23 QemuDevice Class Reference . . . . .	33
4.23.1 Detailed Description . . . . .	34
4.23.2 Constructor & Destructor Documentation . . . . .	34
4.23.2.1 QemuDevice() . . . . .	34
4.24 QemuDeviceBaseIF Class Reference . . . . .	34
4.25 QemuHexagonL2vic Class Reference . . . . .	35
4.26 QemuHexagonQtimer Class Reference . . . . .	36
4.27 QemuInitiatorIface Class Reference . . . . .	37
4.28 QemuInitiatorSignalSocket Class Reference . . . . .	38
4.28.1 Detailed Description . . . . .	38
4.28.2 Member Function Documentation . . . . .	39

4.28.2.1 init()	39
4.28.2.2 init_named()	39
4.28.2.3 init_sbd()	39
4.29 QemuInitiatorSocket< BUSWIDTH > Class Template Reference	40
4.29.1 Detailed Description	41
4.30 QemuInstance Class Reference	41
4.30.1 Detailed Description	43
4.30.2 Member Function Documentation	43
4.30.2.1 add_arg()	43
4.30.2.2 create_quantum_keeper()	43
4.30.2.3 get()	43
4.30.2.4 get_dmi_manager()	43
4.30.2.5 get_tcg_mode()	44
4.30.2.6 init()	44
4.31 QemuInstanceDmiManager Class Reference	44
4.31.1 Detailed Description	45
4.32 QemuInstanceManager Class Reference	45
4.32.1 Detailed Description	46
4.32.2 Constructor & Destructor Documentation	46
4.32.2.1 QemuInstanceManager()	46
4.33 QemuMrHintTlmExtension Class Reference	46
4.34 CpuArmCortexM7::QemuNvicArmv7m Class Reference	47
4.35 QemuRiscvAclintMtimer Class Reference	48
4.36 QemuRiscvAclintSwi Class Reference	48
4.37 QemuRiscvSifiveL2pf Class Reference	49
4.38 QemuRiscvSifivePI2 Class Reference	50
4.39 QemuRiscvSifivePlic Class Reference	50
4.40 QemuSifiveUart Class Reference	51
4.41 QemuTargetSignalSocket Class Reference	52
4.41.1 Detailed Description	53
4.41.2 Member Function Documentation	53
4.41.2.1 get_gpio()	53
4.41.2.2 init()	53
4.41.2.3 init_named()	53
4.42 QemuTargetSocket< BUSWIDTH > Class Template Reference	54
4.43 QemuUart16550 Class Reference	55
4.44 QemuUartPI011 Class Reference	55
4.45 QemuVirtioMMIO Class Reference	56
4.46 QemuVirtioMMIOBlk Class Reference	57
4.47 QemuVirtioMMIONet Class Reference	57
4.48 TlmTargetToQemuBridge Class Reference	58



# Chapter 1

## Main Page

[//]: # DONT EDIT THIS FILE

### 1.0.1 LIBQBOX

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

## 1.1 GreenSocs Build and make system

### 1.2 How to build

This project may be built using cmake  
`cmake -B build; pushd build; make -j; popd`

cmake may ask for your git.greensocs.com credentials (see below for advice about passwords)

#### 1.2.1 cmake version

cmake version 3.14 or newer is required. This can be downloaded and used as follows

```
curl -L https://github.com/Kitware/CMake/releases/download/v3.20.0-rc4/cmake-3.20.0-rc4-linux-x86_64.tar.gz  
| tar -zxvf -  
./cmake-3.20.0-rc4-linux-x86_64/bin/cmake
```

#### 1.2.2 details

This project uses CPM <https://github.com/cpm-cmake/CPM.cmake> in order to find, and/or download missing components. In order to find locally installed SystemC, you may use the standard SystemC environment variables: `SYSTEMC_HOME` and `CCI_HOME`. CPM will use the standard CMAKE `find_package` mechanism to find installed packages [https://cmake.org/help/latest/command/find\\_package.html](https://cmake.org/help/latest/command/find_package.html) To specify a specific package location use `<package>_ROOT` CPM will also search along the `CMAKE_MODULE_PATH`

Sometimes it is convenient to have your own sources used, in this case, use the `CPM_<package>_SOURCE` and `E_DIR`. Hence you may wish to use your own copy of SystemC CCI `bash cmake -B build -DCPM_<package>_SOURCE=/path/to/your/cci/source`

It may also be convenient to have all the source files downloaded, you may do this by running `bash cmake -B build -DCPM_SOURCE_CACHE='pwd'/Packages`

This will populate the directory `Packages` Note that the cmake file system will automatically use the directory called `Packages` as source, if it exists.

NB, CMake holds a cache of compiled modules in `~/cmake/` Sometimes this can confuse builds. If you seem to be picking up the wrong version of a module, then it may be in this cache. It is perfectly safe to delete it.

### 1.2.2.1 Common CMake options

CMAKE\_INSTALL\_PREFIX : Install directory for the package and binaries. CMAKE\_BUILD\_TYPE : DEBUG or RELEASE

The library assumes the use of C++14, and is compatible with SystemC versions from SystemC 2.3.1a.

For a reference docker please use the following script from the top level of the Virtual Platform:

```
curl --header 'PRIVATE-TOKEN: W1Z9U8S_5BUEx1_Y29iS'  
      'https://git.greensocs.com/api/v4/projects/65/repository/files/docker_vp.sh/raw?ref=master' -o  
      docker_vp.sh  
chmod +x ./docker_vp.sh  
./docker_vp.sh  
> cmake -B build; cd build; make -j
```

### 1.2.2.2 passwords for git.greensocs.com

To avoid using passwords for git.greensocs.com please add a ssh key to your git account. You may also use a key-chain manager. As a last resort, the following script will populate ~/.git-credentials with your username and password (in plain text)

```
git config --global credential.helper store
```

## 1.2.3 More documentation

More documentation, including doxygen generated API documentation can be found in the `/docs` directory.

## 1.2.4 LIBGSSYNC

The GreenSocs Synchronization library provides a number of different policies for synchronizing between an external simulator (typically QEMU) and SystemC.

These are based on a proposed standard means to handle the SystemC simulator. This library provides a backwards compatibility layer, but the patched version of SystemC will perform better.

## 1.2.5 The GreenSocs SystemC simple components library.

This includes simple models such as routers, memories and exclusive monitor. The components are "Loosely timed" only. They support DMI where appropriate, and make use of CCI for configuration.

It also has several unit tests for memory, router and exclusive monitor.

## 1.2.6 LIBGSUTILS

The GreenSocs basic utilities library contains utility functions for CCI, simple logging and test functions. It also includes some basic tlm port types

## 1.2.7 LIBQEMU-CXX

Libqemu-cxx encapsulates QEMU as a C++ object, such that it can be instanced (for instance) within a SystemC simulation framework.



### 1.2.8 Information about building and using the greensocs Qbox library

The greensocs Qbox library depends on the libraries : base-components, libgssync, libqemu-cxx, libgsutils, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

### 1.2.9 Information about building and using the base-components library

The base-components library depends on the libraries : Libgsutils, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

### 1.2.10 Information about building and using the libgssync library

The libgssync library depends on the libraries : base-components, libgsutils, SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

### 1.2.11 Information about building and using the libgsutils library

The libgsutils library depends on the libraries : SystemC, RapidJSON, SystemCCI, Lua and GoogleTest.

The GreenSocs CCI libraries allows two options for setting configuration parameters

```
--gs_luafile <FILE.lua> this option will read the lua file to set parameters.
```

```
--param path.to.param=<value> this option will allow individual parameters to be set.
```

NOTE, order is important, the last option on the command line to set a parameter will take preference.

This library includes a Configurable Broker (gs::ConfigurableBroker) which provides additional functionality. Each broker can be configured separately, and has a parameter itself for the configuration file to read. This is `lua_file`. Hence

```
--param path.to.module.lua_file="\"/host/path/to/lua/file"
```

Note that a string parameter must be quoted.

The lua file read by the ConfigurableBroker has relative paths - this means that in the example above the `path.to.module` portion of the absolute path should not appear in the (local) configuration file. (Hence changes in the hierarchy will not need changes to the configuration file).

### 1.2.12 Using yaml for configuration

If you would prefer to use yaml as a configuration language, `lyaml` provides a link. This can be downloaded from <https://github.com/gvvaughan/lyaml>

The following lua code will load "conf.yaml".

```
local lyaml = require "lyaml"
function readAll(file)
    local f = assert(io.open(file, "rb"))
    local content = f:read("*all")
    f:close()
    return content
end
print "Loading conf.yaml"
yamldata=readAll("conf.yaml")
ytab=lyaml.load(yamldata)
for k,v in pairs(ytab) do
    _G[k]=v
end
yamldata=nil
ytab=nil
```

### 1.2.13 Information about building and using the libqemu-cxx library

The libgsutils library does not depend on any library.

### 1.2.14 Instantiate Qemu

A `QemuManager` is required in order to instantiate a `Qemu` instance. A `QemuManager` will hold, and maintain the instance until the end of execution. The `QemuInstance` can contain one or many CPU's and other devices. To create a new instance you can do this:

```
{c++}
    QemuInstanceManager m_inst_mgr;
```

then you can initialize it by providing the `QemuInstance` object with the `QemuInstanceManager` object which will call the `new_instance` method to create a new instance.

```
{c++}
    QemuInstance m_qemu_inst(m_inst_mgr.new_instance(QemuInstance::Target::AARCH64))
```

In order to add a CPU device to an instance they can be constructed as follows:

```
{c++}
    sc_core::sc_vector<QemuCpuArmCortexA53> m_cpus
    m_cpus("cpu", 32, [this] (const char *n, size_t i) { return new QemuCpuArmCortexA53(n, m_qemu_inst); })
```

You can change the CPUs to those listed below in the "CPU" section

Interrupt Controllers and others devices also need a QEMU instance and can be set up as follows:

```
{c++}
    QemuArmGicv3 m_gic("gic", m_qemu_inst);
    QemuUartPl011 m_uart("uart", m_qemu_inst)
```

### 1.2.15 QEMU Arguments

QEMU arguments can be added to an entire instance using the configuration mechanism. The argument name should be in a form `"name.of.your.qemu.instance.args.-ARG" = "value"`.

The QEMU instance provides the following default arguments :

```
"-M", "none", /* no machine */
"-m", "2048", /* used by QEMU to set some internal buffer sizes */
"-monitor", "null", /* no monitor */
"-serial", "null", /* no serial backend */
"-display", "none", /* no GUI */
```

Example : Using the lua file configuration mechanism to set `-monitor` to enable telnet communication with QEMU, with the QEMU instance "platform.QemuInstance" the lua file should contain :

```
["platform.QemuInstance.args.-monitor"] = "tcp:127.0.0.1:55555,server,nowait",
```

To check that the QEMU argument has been added QEMU will report : Added QEMU argument: "name of the argument" "value of the argument"

In the example it's : Added QEMU argument : `-monitor tcp:127.0.0.1:55555,server,nowait`

Telnet can be used to connector to the monitor as follows:

```
$ telnet 127.0.0.1 55555
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
QEMU 5.1.0 monitor - type 'help' for more information
(qemu) quit
quit
Connection closed by foreign host.
```

NOTE :

This should not be used to enable GDB.

### 1.2.16 Enabling GDB per CPU

In order to connect a GDB the CCI parameter `name.of.cpu.gdb-port` must be set a none zero value.

For instance

```
$ ./build/vp --gs_luafile conf.lua -p platform.cpu_1.gdb-port=1234
```

Will open a gdb server on port 1234, for `cpu_1`, and the virtual platform will wait for GDB to connect.

### 1.2.17 The components of libqbox

#### 1.2.17.1 CPU

The libqbox library supports several CPU architectures such as ARM and RISC-V.

- In ARM architectures the library supports the cortex-a53 and the Neoverse-N1 which is based on the cortex-a76 architecture which itself derives from the cortex-a75/73/72.
- In RISC-V architecture, the library manages only the riscv64.

### 1.2.17.2 IRQ-CTRL

The library also manages interrupts by providing :

- ARM GICv2
- ARM GICv3 which are Arm Generic Interrupt Controller.

Then :

- SiFive CLINT
- SiFive PLIC which are also Interrupt controller but for SiFive.

### 1.2.17.3 UART

Finally, it has 2 uarts:

- pl011 for ARM
- 16550 for more general use

### 1.2.17.4 PORTS

The library also provides socket initiators and targets for Qemu

## 1.2.18 The GreenSocs component library memory

The memory component allows you to add memory when creating an object of type `Memory("name", size)`.

The memory component consists of a simple target socket `:tlm_utils::simple_target_socket<Memory>socket`

## 1.2.19 The GreenSocs component library router

The router offers `add_target(socket, base_address, size)` as an API to add components into the address map for routing. (It is recommended that the addresses and size are CCI parameters).

It also allows to bind multiple initiators with `add_initiator(socket)` to send multiple transactions. So there is no need for the `bind()` method offered by sockets because the `add_initiator` method already takes care of that.

## 1.2.20 Functionality of the synchronization library

In addition the library contains utilities such as an thread safe event (`async_event`) and a real time speed limited for SystemC.

### 1.2.20.1 Suspend/Unsuspend interface

This patch adds four new basic functions to SystemC:

```
void sc_suspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspendable()
void sc_suspendable()
```

**suspend\_all/unsuspend\_all** : This pair of functions requests the kernel to 'atomically suspend' all processes (using the same semantics as the thread `suspend()` call). This is atomic in that the kernel will only suspend all the processes together, such that they can be suspended and unsuspended without any side effects. Calling `suspend_all()`, and subsequently calling `unsuspend_all()` will have no effect on the suspended status of an individual process. A process may call `suspend_all()` followed by `unsuspend_all()`, the calls should be 'paired', (multiple calls to either `suspend_all()` or `unsuspend_all()` will be ignored). Outside of the context of a process, it is the programmers responsibility to ensure that the calls are paired. As a consequence, multiple calls to `suspend_all()` may be made (within separate process, or from within `sc_main`). So long as there have been more calls to `suspend_all()` than to `unsuspend_all()`, the kernel will suspend all processes.

*[note, this patch set does not add convenience functions, including those to find out if suspension has happened, these are expected to be layered ontop]*

**unsuspendable()/suspendable()**: This pair of functions provides an 'opt-out' for specific process to the `suspend_all()`. The consequence is that if there is a process that has opted out, the kernel will not be able to `suspend_all` (as it would no longer be atomic). These functions can only be called from within a process. A process should only call `suspendable/unsuspendable` in pairs (multiple calls to either will be ignored). *Note that the default is that a process is marked as suspendable.*

**Use cases:** 1 : *Save and Restore* For Save and Restore, the expectation is that when a save is requested, 'suspend\_all' will be called. If there are models that are in an unsuspendable state, the entire simulation will be allowed to continue until such a time that there are no unsuspendable processes.

2 : *External sync* When an external model injects events into a SystemC model (for instance, using an 'async\_request\_update()'), time can drift between the two simulators. In order to maintain time, SystemC can be prevented from advancing by calling `suspend_all()`. If there are process in an unsuspendable state (for instance, processing on behalf of the external model), then the simulation will be allowed to continue. NOTE, an event injected into the kernel by an `async_request_update` will cause the kernel to execute the associated `update()` function (leaving the suspended state). The update function should arrange to mark any processes that it requires as unsuspendable before the end of the current delta cycle, to ensure that they are scheduled.

### 1.2.21 Using the ConfigurableBroker

The broker will self register in the SystemC CCI hierarchy. All brokers have a parameter `lua_file` which will be read and used to configure parameters held within the broker. This file is read at the *local* level, and paths are *relative* to the location where the ConfigurableBroker is instantiated.

These brokers can be used as global brokers.

The `gs::ConfigurableBroker` can be instantiated in 3 ways:

1. `ConfigurableBroker()` This will instance a 'Private broker' and will hide **ALL** parameters held within this broker.

A local `lua_file` can be read and will set parameters in the private broker. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`).

2. `ConfigurableBroker({{"key1", "value1"}, {"key2", "value2"} ...})` This will instance a broker that sets and hides the listed keys. All other keys are passed through (exported). Hence the broker is 'invisible' for parameters that are not listed. This is specifically useful for structural parameters.

It is also possible to instance a 'pass through' broker using `ConfigurationBroker({})`. This is useful to provide a *local* configuration broker than can, for instance, read a local configuration file.

A local `lua_file` can be read and will set parameters in the private broker (exported or not). This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(false)`). The `lua_file` will be read **AFTER** the construction key-value list and hence can be used to over-right default values in the code.

3. `ConfigurableBroker(argc, argv)` This will instance a broker that is typically a global broker. The `argc/argv` values should come from the command line. The command line will be parsed to find:

> -p, --param path.to.param=<value> this option will allow individual parameters to be set.

> -l, --gs\_luafile <FILE.lua> this option will read the lua file to set parameters. Similar functionality can be achieved using `--param lua_file="<FILE.lua>".`

A `{{key, value}}` list can also be provided, otherwise it is assumed to be empty. Such a list will set parameter values within this broker. These values will be read and used **BEFORE** the command line is read.

Finally **AFTER** the command line is read, if the `lua_file` parameter has been set, the configuration file that it indicates will also be read. This can be prevented by passing 'false' as a construction parameter (`ConfigurableBroker(argc, argv, false)`). The `lua_file` will be read **AFTER** the construction key-value list, and after the command like, so it can be used to over-right default values in either.

### 1.2.22 Print out the available params

It is possible to display the list of available cci parameters with the `-h` option when launching the virtual platform.

CAUTION:

This will only print the parameters at the begining of simulation.

### 1.2.23 Halt Interface

The `libqbox` library allows to manage the halt state of the different CPUs that are used. The halt will allow a cpu to be in "standby".

Indeed, by default the halt state is released (state 0). It is important to note that the halt does not work with the `power_off` set (parameter `p_power_off` set to true).

## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

QemuInstanceDmiManager::DmiRegion . . . . .	14
QemuInstanceDmiManager::DmiRegionAlias . . . . .	15
InitiatorSignalSocket	
QemuInitiatorSignalSocket . . . . .	38
LockedQemuInstanceDmiManager . . . . .	17
QemuInitiatorSocket< BUSWIDTH >::m_mem_obj . . . . .	18
Object	
QemuInstanceDmiManager::QemuContainer . . . . .	23
QemuDeviceBaseIF . . . . .	34
QemuDevice . . . . .	33
CpuArmCortexM7::QemuNvicArmv7m . . . . .	47
QemuArmGicv2 . . . . .	19
QemuArmGicv2m . . . . .	20
QemuArmGicv3 . . . . .	21
QemuArmSmmu . . . . .	22
QemuCpu . . . . .	23
CpuArmCortexM7 . . . . .	13
CpuArmCortexR5 . . . . .	14
QemuCpuArmCortexA53 . . . . .	25
QemuCpuArmMax . . . . .	26
QemuCpuArmNeoverseN1 . . . . .	27
QemuCpuHexagon . . . . .	28
QemuCpuRiscv64 . . . . .	30
QemuCpuRiscv64Rv64 . . . . .	31
QemuCpuSifiveX280 . . . . .	32
QemuHexagonL2vic . . . . .	35
QemuHexagonQtimer . . . . .	36
QemuRiscvAclintMtimer . . . . .	48
QemuRiscvAclintSwi . . . . .	48
QemuRiscvSifiveL2pf . . . . .	49
QemuRiscvSifivePI2 . . . . .	50
QemuRiscvSifivePlic . . . . .	50
QemuSifiveUart . . . . .	51
QemuUart16550 . . . . .	55
QemuUartPIO11 . . . . .	55

QemuVirtioMMIO . . . . .	56
QemuVirtioMMIOBlk . . . . .	57
QemuVirtioMMIONet . . . . .	57
QemuInitiatorIface . . . . .	37
QemuArmSmmu . . . . .	22
QemuCpu . . . . .	23
QemuInstanceDmiManager . . . . .	44
QemuInstanceManager . . . . .	45
runtime_error	
QboxException . . . . .	18
sc_module	
QemuDevice . . . . .	33
QemuInstance . . . . .	41
TargetSignalSocket	
QemuTargetSignalSocket . . . . .	52
t1m_bw_transport_if	
QemuInitiatorSocket< BUSWIDTH > . . . . .	40
t1m_extension	
QemuCpuHintT1mExtension . . . . .	30
QemuCpu::QemuCpuHintT1mExtension . . . . .	29
QemuMrHintT1mExtension . . . . .	46
t1m_fw_transport_if	
T1mTargetToQemuBridge . . . . .	58
t1m_initiator_socket	
QemuInitiatorSocket< BUSWIDTH > . . . . .	40
t1m_target_socket	
QemuTargetSocket< BUSWIDTH > . . . . .	54



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">CpuArmCortexM7</a>	13
<a href="#">CpuArmCortexR5</a>	14
<a href="#">QemuInstanceDmiManager::DmiRegion</a>	
DMI region	14
<a href="#">QemuInstanceDmiManager::DmiRegionAlias</a>	
An alias to a DMI region	15
<a href="#">LockedQemuInstanceDmiManager</a>	
A locked <a href="#">QemuInstanceDmiManager</a>	17
<a href="#">QemuInitiatorSocket&lt; BUSWIDTH &gt;::m_mem_obj</a>	18
<a href="#">QboxException</a>	18
<a href="#">QemuArmGicv2</a>	19
<a href="#">QemuArmGicv2m</a>	20
<a href="#">QemuArmGicv3</a>	21
<a href="#">QemuArmSmmu</a>	22
<a href="#">QemuInstanceDmiManager::QemuContainer</a>	23
<a href="#">QemuCpu</a>	23
<a href="#">QemuCpuArmCortexA53</a>	25
<a href="#">QemuCpuArmMax</a>	26
<a href="#">QemuCpuArmNeoverseN1</a>	27
<a href="#">QemuCpuHexagon</a>	28
<a href="#">QemuCpu::QemuCpuHintTlmExtension</a>	29
<a href="#">QemuCpuHintTlmExtension</a>	30
<a href="#">QemuCpuRiscv64</a>	30
<a href="#">QemuCpuRiscv64Rv64</a>	31
<a href="#">QemuCpuSifiveX280</a>	32
<a href="#">QemuDevice</a>	
QEMU device abstraction as a SystemC module	33
<a href="#">QemuDeviceBaseIF</a>	34
<a href="#">QemuHexagonL2vic</a>	35
<a href="#">QemuHexagonQtimer</a>	36
<a href="#">QemuInitiatorIface</a>	37
<a href="#">QemuInitiatorSignalSocket</a>	
A QEMU output GPIO exposed as a <a href="#">InitiatorSignalSocket&lt;bool&gt;</a>	38
<a href="#">QemuInitiatorSocket&lt; BUSWIDTH &gt;</a>	
TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping	40

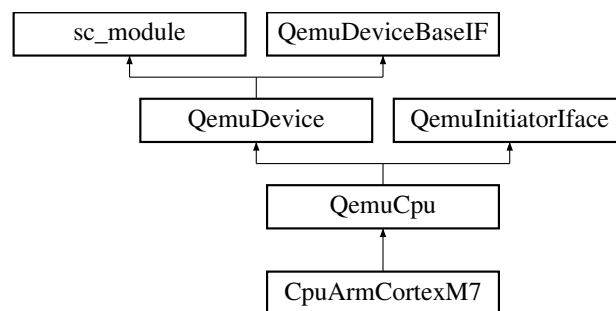
<a href="#">QemuInstance</a>	
This class encapsulates a libqemu-cxx qemu::LibQemu instance. It handles QEMU parameters and instance initialization . . . . .	41
<a href="#">QemuInstanceDmiManager</a>	
Handles the DMI regions at the QEMU instance level . . . . .	44
<a href="#">QemuInstanceManager</a>	
QEMU instance manager class . . . . .	45
<a href="#">QemuMrHintTlmExtension</a>	46
<a href="#">CpuArmCortexM7::QemuNvicArmv7m</a>	47
<a href="#">QemuRiscvAclintMtimer</a>	48
<a href="#">QemuRiscvAclintSwi</a>	48
<a href="#">QemuRiscvSifiveL2pf</a>	49
<a href="#">QemuRiscvSifivePI2</a>	50
<a href="#">QemuRiscvSifivePlic</a>	50
<a href="#">QemuSifiveUart</a>	51
<a href="#">QemuTargetSignalSocket</a>	
A QEMU input GPIO exposed as a TargetSignalSocket<bool> . . . . .	52
<a href="#">QemuTargetSocket&lt; BUSWIDTH &gt;</a>	54
<a href="#">QemuUart16550</a>	55
<a href="#">QemuUartPIO11</a>	55
<a href="#">QemuVirtioMMIO</a>	56
<a href="#">QemuVirtioMMIOBlk</a>	57
<a href="#">QemuVirtioMMIONet</a>	57
<a href="#">TlmTargetToQemuBridge</a>	58

## Chapter 4

# Class Documentation

### 4.1 CpuArmCortexM7 Class Reference

Inheritance diagram for CpuArmCortexM7:



#### Classes

- class [QemuNvicArmv7m](#)

#### Public Member Functions

- **CpuArmCortexM7** (sc\_core::sc\_module\_name name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override

#### Public Attributes

- cci::cci\_param< bool > **p\_start\_powered\_off**
- [QemuNvicArmv7m](#) **m\_nvic**
- cci::cci\_param< uint64\_t > **p\_init\_nsvtor**

#### Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

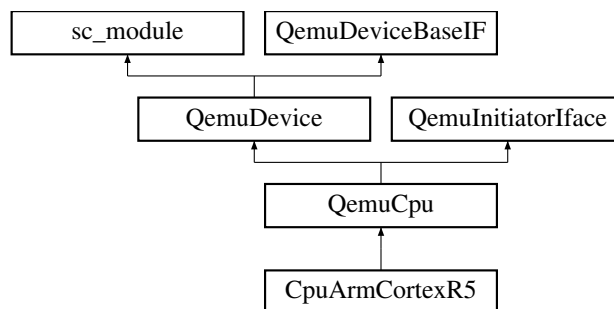
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/cortex-m7.h

## 4.2 CpuArmCortexR5 Class Reference

Inheritance diagram for CpuArmCortexR5:



## Public Member Functions

- **CpuArmCortexR5** (sc\_core::sc\_module\_name name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override

## Public Attributes

- cci::cci\_param< bool > **p\_start\_powered\_off**

## Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/cortex-r5.h

## 4.3 QemuInstanceDmiManager::DmiRegion Class Reference

a DMI region

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

## Public Types

- using **Key** = uintptr\_t
- using **Ptr** = std::shared\_ptr< [DmiRegion](#) >

## Public Member Functions

- **DmiRegion** (const tlm::tlm\_dmi &info, qemu::LibQemu &inst)
- uint64\_t **get\_size** () const
- qemu::MemoryRegion **get\_mr** ()
- Key **get\_key** () const
- bool **is\_valid** () const
- void **invalidate** ()

## Static Public Member Functions

- static Key **key\_from\_tlm\_dmi** (const tlm::tlm\_dmi &info)

### 4.3.1 Detailed Description

a DMI region

@detail Represent a DMI region with a size and an host pointer. It also embeds the QEMU memory region mapping to this host pointer. Note that it does not have start and end addresses as it is totally address space agnostic. Two initiators with two different views of the address space can map the same DMI region.

Note: The `get_key` method is used to index the map in which the regions are stored. Currently, we use the host memory address itself to index the map. This makes a strong assumption on the fact that two consecutive DMI region requests for the same region will return the same host address. This is not clearly stated in the TLM-2.0 standard but is quite reasonable to assume.

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

## 4.4 QemuInstanceDmiManager::DmiRegionAlias Class Reference

An alias to a DMI region.

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

## Public Member Functions

- **DmiRegionAlias** (DmiRegion::Ptr region, const tlm::tlm\_dmi &info, qemu::LibQemu &inst)
- uint64\_t **get\_start** () const
- uint64\_t **get\_end** () const
- uint64\_t **get\_size** () const
- qemu::MemoryRegion **get\_alias\_mr** () const
- bool **is\_valid** () const  
*Return true if the alias and its underlying DMI region are valid.*
- void **invalidate\_region** ()  
*Invalidate the underlying DMI region.*
- void **set\_installed** ()  
*Mark the alias as mapped onto QEMU root MR.*
- bool **is\_installed** () const  
*Return true if the alias is mapped onto QEMU root MR.*

### 4.4.1 Detailed Description

An alias to a DMI region.

@detail An object of this class represents an alias to a DMI region a CPU can map on its own address space. Contrary to a [DmiRegion](#), it has a start and an end address as it is requested from the point of view of an initiator's address map.

It embeds a shared pointer of the underlying DMI region. The DMI region gets destroyed once all aliases referencing it have been destroyed.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 invalidate\_region()

```
void QemuInstanceDmiManager::DmiRegionAlias::invalidate_region ( ) [inline]
```

Invalidate the underlying DMI region.

##### Note

Must be called with the DMI manager lock held

#### 4.4.2.2 is\_installed()

```
bool QemuInstanceDmiManager::DmiRegionAlias::is_installed ( ) const [inline]
```

Return true if the alias is mapped onto QEMU root MR.

##### Note

Must be called with the DMI manager lock held

#### 4.4.2.3 is\_valid()

```
bool QemuInstanceDmiManager::DmiRegionAlias::is_valid ( ) const [inline]
```

Return true if the alias and its underlying DMI region are valid.

##### Note

Must be called with the DMI manager lock held

#### 4.4.2.4 set\_installed()

```
void QemuInstanceDmiManager::DmiRegionAlias::set_installed ( ) [inline]
```

Mark the alias as mapped onto QEMU root MR.

##### Note

Must be called with the DMI manager lock held

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

## 4.5 LockedQemuInstanceDmiManager Class Reference

A locked [QemuInstanceDmiManager](#).

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

### Public Types

- using **DmiRegion** = [QemuInstanceDmiManager::DmiRegion](#)

### Public Member Functions

- **LockedQemuInstanceDmiManager** ([QemuInstanceDmiManager](#) &inst)
- **LockedQemuInstanceDmiManager** (const [LockedQemuInstanceDmiManager](#) &)=delete
- **LockedQemuInstanceDmiManager** ([LockedQemuInstanceDmiManager](#) &&)=default
- [QemuInstanceDmiManager::DmiRegionAlias](#) **get\_new\_region\_alias** (const tlm::tlm\_dmi &info)

### Protected Attributes

- [QemuInstanceDmiManager](#) & **m\_inst**
- std::unique\_lock< std::mutex > **m\_lock**

### 4.5.1 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.

### 4.5.2 Member Function Documentation

#### 4.5.2.1 get\_new\_region\_alias()

```
QemuInstanceDmiManager::DmiRegionAlias LockedQemuInstanceDmiManager::get_new_region_alias (
    const tlm::tlm_dmi & info ) [inline]
```

See also

[QemuInstanceDmiManager::get\\_new\\_region\\_alias](#)

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

## 4.6 QemuInitiatorSocket< BUSWIDTH >::m\_mem\_obj Class Reference

### Public Member Functions

- [m\\_mem\\_obj](#) (qemu::LibQemu &inst)

### Public Attributes

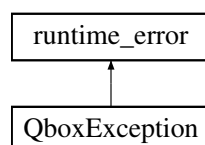
- [qemu::MemoryRegion m\\_root](#)

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator.h

## 4.7 QboxException Class Reference

Inheritance diagram for QboxException:





## Public Member Functions

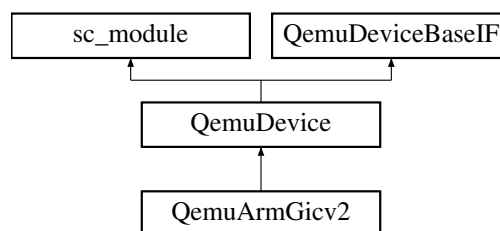
- **QboxException** (const char \*what)

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/exceptions.h

## 4.8 QemuArmGicv2 Class Reference

Inheritance diagram for QemuArmGicv2:



## Public Member Functions

- **QemuArmGicv2** (const sc\_core::sc\_module\_name &name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** ()
- void **end\_of\_elaboration** ()

## Public Attributes

- [QemuArmGicv2m](#) \* **m\_gicv2m**
- [QemuTargetSocket](#) **dist\_iface**
- [QemuTargetSocket](#) **cpu\_iface**
- [QemuTargetSocket](#) **virt\_iface**
- [QemuTargetSocket](#) **vcpu\_iface**
- [QemuTargetSocket](#) ::TlmTargetSocket **v2m\_iface**
- sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > **spi\_in**
- sc\_core::sc\_vector< sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > > **ppi\_in**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **irq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **fiq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **virq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **vfiq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **maintenance\_out**

## Static Public Attributes

- static const uint32\_t **NUM\_PPI** = 32

## Protected Attributes

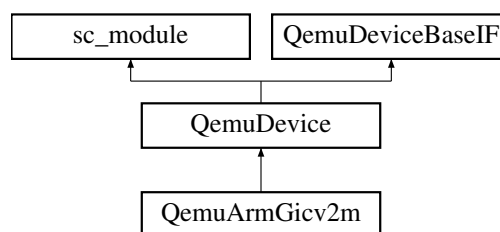
- cci::cci\_param< unsigned int > **p\_num\_cpu**
- cci::cci\_param< unsigned int > **p\_num\_spi**
- cci::cci\_param< unsigned int > **p\_revision**
- cci::cci\_param< bool > **p\_has\_virt\_extensions**
- cci::cci\_param< bool > **p\_has\_security\_extensions**
- cci::cci\_param< unsigned int > **p\_num\_prio\_bits**
- cci::cci\_param< bool > **p\_has\_msi\_support**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv2.h

## 4.9 QemuArmGicv2m Class Reference

Inheritance diagram for QemuArmGicv2m:



## Public Member Functions

- **QemuArmGicv2m** (const sc\_core::sc\_module\_name &name, [QemuInstance](#) &inst)
- unsigned int **get\_base\_spi** ()
- unsigned int **get\_num\_spis** ()
- void **before\_end\_of\_elaboration** ()
- void **end\_of\_elaboration** ()

## Public Attributes

- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **spi\_out**
- [QemuTargetSocket](#) **iface**

## Protected Attributes

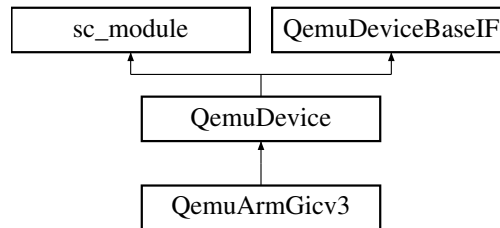
- cci::cci\_param< unsigned int > **p\_base\_spi**
- cci::cci\_param< unsigned int > **p\_num\_spis**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv2.h

## 4.10 QemuArmGicv3 Class Reference

Inheritance diagram for QemuArmGicv3:



### Public Member Functions

- **QemuArmGicv3** (const sc\_core::sc\_module\_name &name, [QemuInstance](#) &inst, unsigned num\_cpus=0)
- void **before\_end\_of\_elaboration** ()
- void **end\_of\_elaboration** ()

### Public Attributes

- [QemuTargetSocket](#) **dist\_iface**
- sc\_core::sc\_vector< [QemuTargetSocket](#)<> > **redist\_iface**
- sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > **spi\_in**
- sc\_core::sc\_vector< sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > > **ppi\_in**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **irq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **fiq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **virq\_out**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **vfiq\_out**

### Static Public Attributes

- static const uint32\_t **NUM\_PPI** = 32

### Protected Attributes

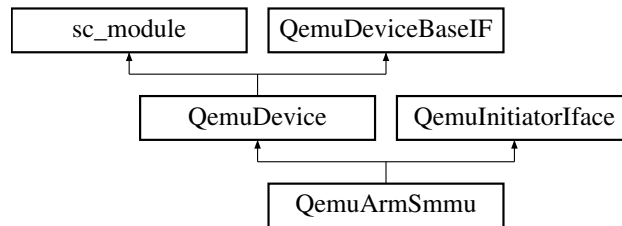
- cci::cci\_param< unsigned int > **p\_num\_cpu**
- cci::cci\_param< unsigned int > **p\_num\_spi**
- cci::cci\_param< unsigned int > **p\_revision**
- cci::cci\_param< std::vector< unsigned int > > **p\_redist\_region**
- cci::cci\_param< bool > **p\_has\_security\_extensions**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/arm-gicv3.h

## 4.11 QemuArmSmmu Class Reference

Inheritance diagram for QemuArmSmmu:



### Public Member Functions

- **QemuArmSmmu** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override
- virtual void **initiator\_customize\_tlm\_payload** (TlmPayload &payload) override
- virtual void **initiator\_tidy\_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< uint32\_t > **p\_pamax**
- cci::cci\_param< uint16\_t > **p\_num\_smr**
- cci::cci\_param< uint16\_t > **p\_num\_cb**
- cci::cci\_param< uint16\_t > **p\_num\_pages**
- cci::cci\_param< bool > **p\_ato**
- cci::cci\_param< uint8\_t > **p\_version**
- cci::cci\_param< uint8\_t > **p\_num\_tbu**
- [QemuTargetSocket](#) **register\_socket**
- sc\_core::sc\_vector< [QemuTargetSocket](#)<> > **upstream\_socket**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **irq\_context**
- [QemuInitiatorSignalSocket](#) **irq\_global**
- sc\_core::sc\_vector< [QemuInitiatorSocket](#)<> > **downstream\_socket**

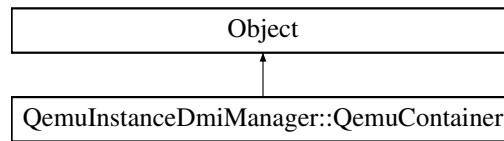
### Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/mmu/arm-smmu.h

## 4.12 QemuInstanceDmiManager::QemuContainer Class Reference

Inheritance diagram for QemuInstanceDmiManager::QemuContainer:



### Public Member Functions

- **QemuContainer** (const [QemuContainer](#) &o)=default
- **QemuContainer** (const Object &o)

### Static Public Attributes

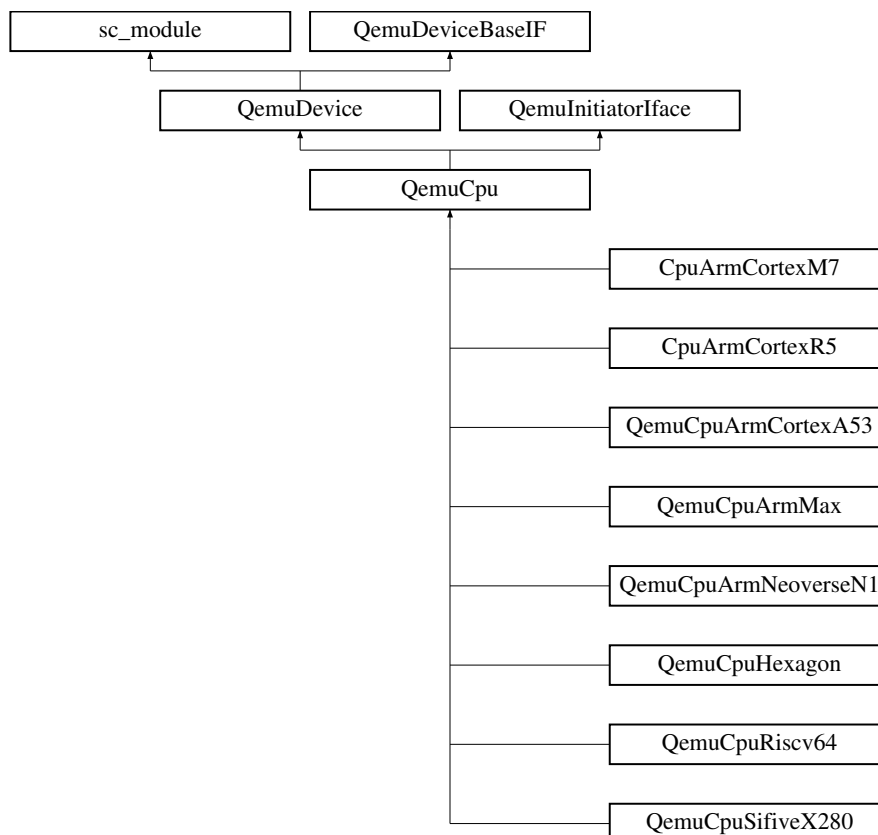
- static constexpr const char \*const **TYPE** = "container"

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h

## 4.13 QemuCpu Class Reference

Inheritance diagram for QemuCpu:



## Classes

- class [QemuCpuHintTlmExtension](#)

## Public Member Functions

- **SC\_HAS\_PROCESS** ([QemuCpu](#))
- **QemuCpu** (const sc\_core::sc\_module\_name &name, [QemuInstance](#) &inst, const std::string &type\_name)
- bool **can\_run** () override
- void **before\_end\_of\_elaboration** () override
- void **halt\_cb** (const bool &val)
- virtual void **end\_of\_elaboration** () override
- virtual void **start\_of\_simulation** () override
- virtual void **initiator\_customize\_tlm\_payload** (TlmPayload &payload) override
- virtual void **initiator\_tidy\_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< unsigned int > **p\_gdb\_port**
- cci::cci\_param< bool > **p\_start\_halted**
- [QemuInitiatorSocket](#) **socket**
- TargetSignalSocket< bool > **halt**

## Protected Member Functions

- void **create\_quantum\_keeper** ()
- void **set\_coroutine\_mode** ()
- void **set\_signaled** ()
- void **watch\_external\_ev** ()
- void **kick\_cb** ()
- 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 **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**
- bool **m\_signaled**
- std::mutex **m\_signaled\_lock**
- std::condition\_variable **m\_signaled\_cond**
- int64\_t **m\_last\_vclock**
- std::shared\_ptr< gs::tlm\_quantumkeeper\_extended > **m\_qk**
- bool **finished** =false
- [QemuCpuHintTlmExtension](#) **m\_cpu\_hint\_ext**

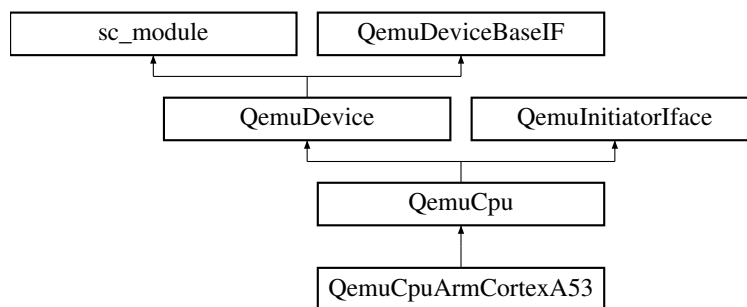
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/cpu.h

## 4.14 QemuCpuArmCortexA53 Class Reference

Inheritance diagram for QemuCpuArmCortexA53:



## Public Member Functions

- **QemuCpuArmCortexA53** (sc\_core::sc\_module\_name name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override
- void **initiator\_customize\_tlm\_payload** (TlmPayload &payload) override
- void **initiator\_tidy\_tlm\_payload** (TlmPayload &payload) override

## Public Attributes

- cci::cci\_param< unsigned int > **p\_mp\_affinity**
- cci::cci\_param< bool > **p\_has\_el2**
- cci::cci\_param< bool > **p\_has\_el3**
- cci::cci\_param< bool > **p\_start\_powered\_off**
- cci::cci\_param< std::string > **p\_psci\_conduit**
- cci::cci\_param< uint64\_t > **p\_rvbar**
- [QemuTargetSignalSocket](#) **irq\_in**
- [QemuTargetSignalSocket](#) **fiq\_in**
- [QemuTargetSignalSocket](#) **virq\_in**
- [QemuTargetSignalSocket](#) **vfiq\_in**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_phys\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_virt\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_hyp\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_sec\_out**

## Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

## Protected Member Functions

- int **get\_psci\_conduit\_val** () const
- void **add\_exclusive\_ext** (TlmPayload &pl)

## Static Protected Member Functions

- static uint64\_t **extract\_data\_from\_payload** (const TlmPayload &pl)

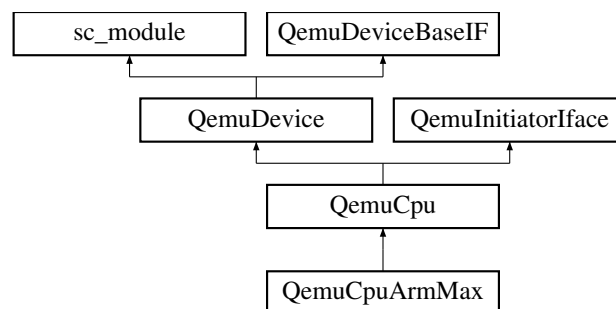
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/cortex-a53.h

## 4.15 QemuCpuArmMax Class Reference

Inheritance diagram for QemuCpuArmMax:



## Public Member Functions

- **QemuCpuArmMax** (sc\_core::sc\_module\_name name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override
- void **initiator\_customize\_tlm\_payload** (TlmPayload &payload) override
- void **initiator\_tidy\_tlm\_payload** (TlmPayload &payload) override

## Public Attributes

- cci::cci\_param< unsigned int > **p\_mp\_affinity**
- cci::cci\_param< bool > **p\_has\_el2**
- cci::cci\_param< bool > **p\_has\_el3**
- cci::cci\_param< bool > **p\_start\_powered\_off**
- cci::cci\_param< std::string > **p\_psci\_conduit**
- cci::cci\_param< uint64\_t > **p\_rvbar**
- [QemuTargetSignalSocket](#) **irq\_in**
- [QemuTargetSignalSocket](#) **fiq\_in**
- [QemuTargetSignalSocket](#) **virq\_in**
- [QemuTargetSignalSocket](#) **vfiq\_in**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_phys\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_virt\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_hyp\_out**
- [QemuInitiatorSignalSocket](#) **irq\_timer\_sec\_out**
- [QemuInitiatorSignalSocket](#) **irq\_maintenance\_out**
- [QemuInitiatorSignalSocket](#) **irq\_pmu\_out**



## Static Public Attributes

- static constexpr qemu::Target **ARCH** = qemu::Target::AARCH64

## Protected Member Functions

- int **get\_psci\_conduit\_val** () const
- void **add\_exclusive\_ext** (TlmPayload &pl)

## Static Protected Member Functions

- static uint64\_t **extract\_data\_from\_payload** (const TlmPayload &pl)

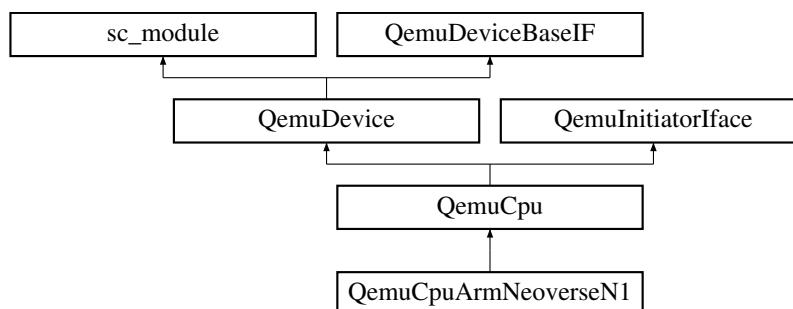
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/max.h

## 4.16 QemuCpuArmNeoverseN1 Class Reference

Inheritance diagram for QemuCpuArmNeoverseN1:



## Public Member Functions

- **QemuCpuArmNeoverseN1** (sc\_core::sc\_module\_name name, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override
- void **initiator\_customize\_tlm\_payload** (TlmPayload &payload) override
- void **initiator\_tidy\_tlm\_payload** (TlmPayload &payload) override

## Public Attributes

- `cci::cci_param< unsigned int > p_mp_affinity`
- `cci::cci_param< bool > p_has_el2`
- `cci::cci_param< bool > p_has_el3`
- `cci::cci_param< bool > p_start_powered_off`
- `cci::cci_param< std::string > p_psci_conduit`
- `cci::cci_param< uint64_t > p_rvbar`
- [QemuTargetSignalSocket](#) `irq_in`
- [QemuTargetSignalSocket](#) `fiq_in`
- [QemuTargetSignalSocket](#) `virq_in`
- [QemuTargetSignalSocket](#) `vfiq_in`
- [QemuInitiatorSignalSocket](#) `irq_timer_phys_out`
- [QemuInitiatorSignalSocket](#) `irq_timer_virt_out`
- [QemuInitiatorSignalSocket](#) `irq_timer_hyp_out`
- [QemuInitiatorSignalSocket](#) `irq_timer_sec_out`

## Static Public Attributes

- static constexpr `qemu::Target ARCH = qemu::Target::AARCH64`

## Protected Member Functions

- int `get_psci_conduit_val ()` const
- void `add_exclusive_ext (TlmPayload &pl)`

## Static Protected Member Functions

- static `uint64_t extract_data_from_payload (const TlmPayload &pl)`

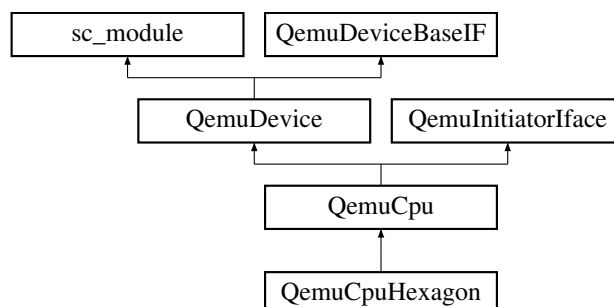
## Additional Inherited Members

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/neoverse-n1.h`

## 4.17 QemuCpuHexagon Class Reference

Inheritance diagram for QemuCpuHexagon:



## Public Types

- enum **Rev\_t** { **v66\_rev** = 0xa666, **v68\_rev** = 0x8d68, **v69\_rev** = 0x8c69, **v73\_rev** = 0x8c73 }

## Public Member Functions

- QemuCpuHexagon** (const sc\_core::sc\_module\_name &name, [QemuInstance](#) &inst, uint32\_t cfgbase, Rev\_t rev, uint32\_t l2vic\_base\_addr, uint32\_t qtimer\_base\_addr, uint32\_t exec\_start\_addr)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

## Public Attributes

- sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > **irq\_in**
- cci::cci\_param< bool > **p\_start\_powered\_off**

## Protected Attributes

- uint32\_t **m\_cfgbase**
- Rev\_t **m\_rev**
- uint32\_t **m\_l2vic\_base\_addr**
- uint32\_t **m\_qtimer\_base\_addr**
- uint32\_t **m\_exec\_start\_addr**

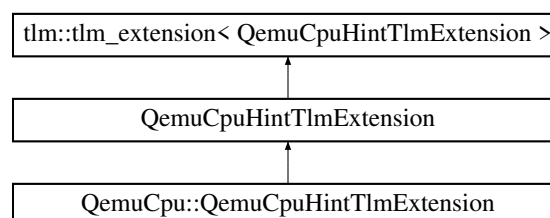
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/hexagon/hexagon.↔h

## 4.18 QemuCpu::QemuCpuHintTlmExtension Class Reference

Inheritance diagram for QemuCpu::QemuCpuHintTlmExtension:



## Public Member Functions

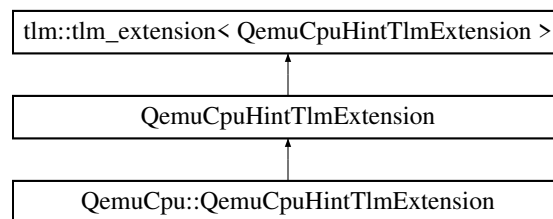
- void **free** () override

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/cpu.h

## 4.19 QemuCpuHintTlmExtension Class Reference

Inheritance diagram for QemuCpuHintTlmExtension:



## Public Member Functions

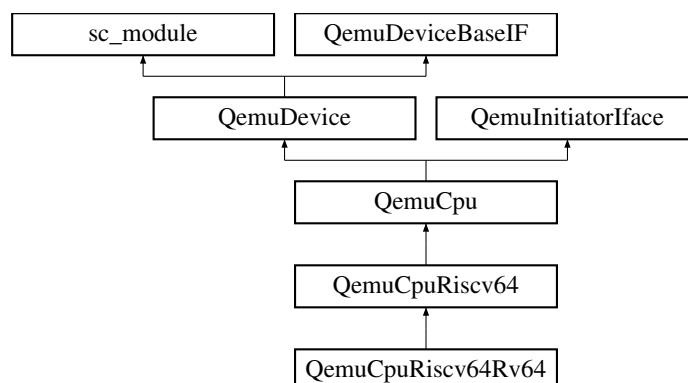
- **QemuCpuHintTlmExtension** (const [QemuCpuHintTlmExtension](#) &)=default
- **QemuCpuHintTlmExtension** (qemu::Cpu cpu)
- virtual tlm\_extension\_base \* **clone** () const override
- virtual void **copy\_from** (tlm\_extension\_base const &ext) override
- void **set\_cpu** (qemu::Cpu cpu)
- qemu::Cpu **get\_cpu** () const

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/tlm-extensions/qemu-cpu-hint.h

## 4.20 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**

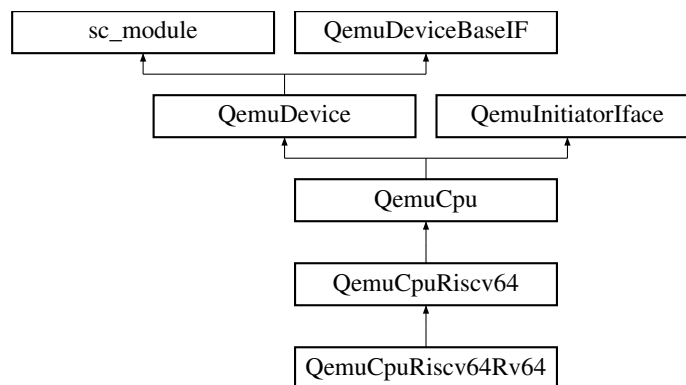
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

## 4.21 QemuCpuRiscv64Rv64 Class Reference

Inheritance diagram for QemuCpuRiscv64Rv64:



## Public Member Functions

- **QemuCpuRiscv64Rv64** (const sc\_core::sc\_module\_name &n, [QemuInstance](#) &inst, uint64\_t hartid)

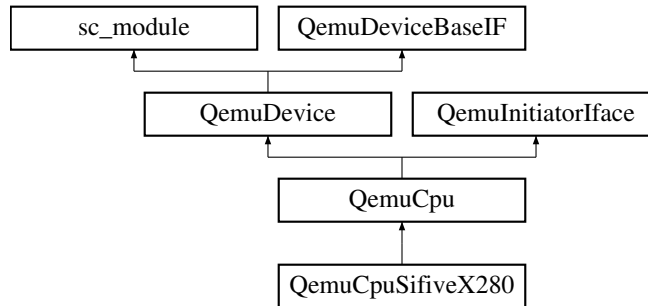
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/riscv64/riscv64.h

## 4.22 QemuCpuSifiveX280 Class Reference

Inheritance diagram for QemuCpuSifiveX280:



### Public Member Functions

- **QemuCpuSifiveX280** (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:

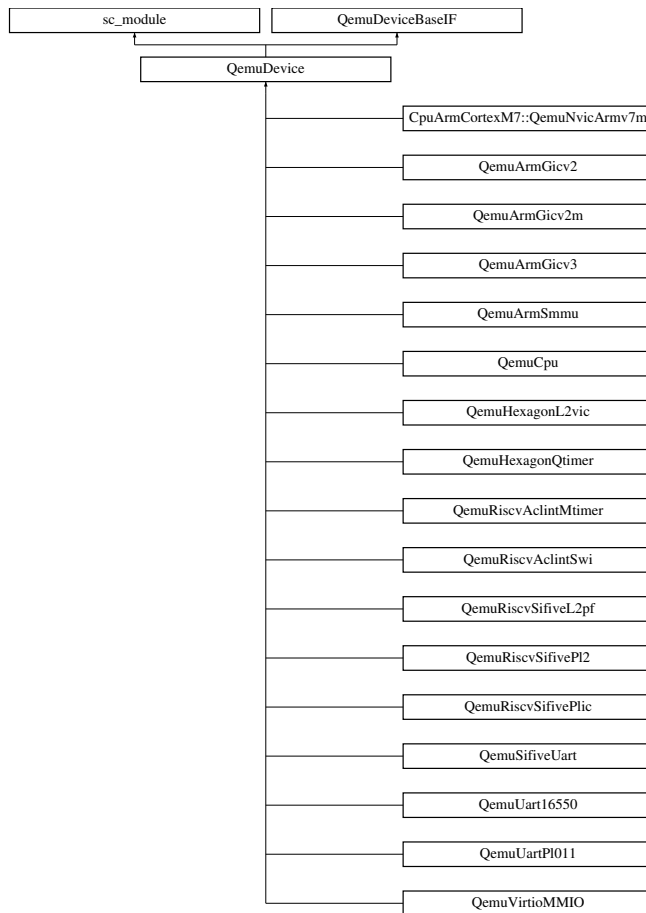
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/riscv64/sifive-x280-rtl.h

## 4.23 QemuDevice Class Reference

QEMU device abstraction as a SystemC module.

```
#include <device.h>
```

Inheritance diagram for QemuDevice:



### Public Member Functions

- void **instantiate** ()
- void **realize** ()
- [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 Attributes

- [QemuInstance](#) & **m\_inst**
- qemu::Device **m\_dev**
- bool **m\_instanciated** = false
- bool **m\_realized** = false

### 4.23.1 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 corresponding 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.

### 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 QemuDevice()

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

Construct a QEMU device.

#### Parameters

in	<i>name</i>	SystemC module name
in	<i>inst</i>	QEMU instance the device will be created in
in	<i>qom_type</i>	Device QOM type name

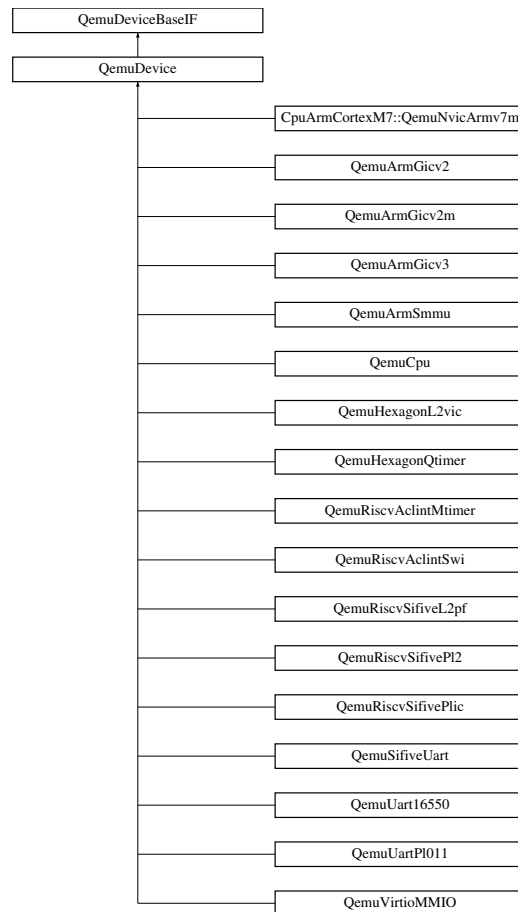
The documentation for this class was generated from the following file:

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/device.h`

## 4.24 QemuDeviceBaseIF Class Reference

Inheritance diagram for QemuDeviceBaseIF:





## Public Member Functions

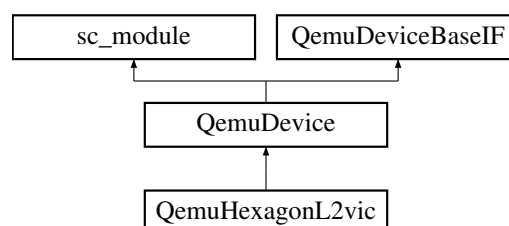
- virtual bool **can\_run** ()

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h

## 4.25 QemuHexagonL2vic Class Reference

Inheritance diagram for QemuHexagonL2vic:



## Public Member Functions

- **QemuHexagonL2vic** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

## Public Attributes

- const unsigned int **p\_num\_sources**
- const unsigned int **p\_num\_outputs**
- [QemuTargetSocket](#) **socket**
- [QemuTargetSocket](#) **socket\_fast**
- sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > **irq\_in**
- sc\_core::sc\_vector< [QemuInitiatorSignalSocket](#) > **irq\_out**

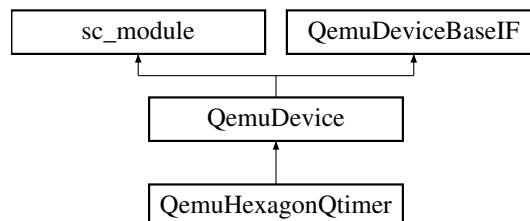
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/hexagon-l2vic.h

## 4.26 QemuHexagonQtimer Class Reference

Inheritance diagram for QemuHexagonQtimer:



## Public Member Functions

- **QemuHexagonQtimer** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

## Public Attributes

- [QemuTargetSocket](#) **socket**
- [QemuTargetSocket](#) **timer0\_socket**
- [QemuTargetSocket](#) **timer1\_socket**
- [QemuInitiatorSignalSocket](#) **timer0\_irq**
- [QemuInitiatorSignalSocket](#) **timer1\_irq**

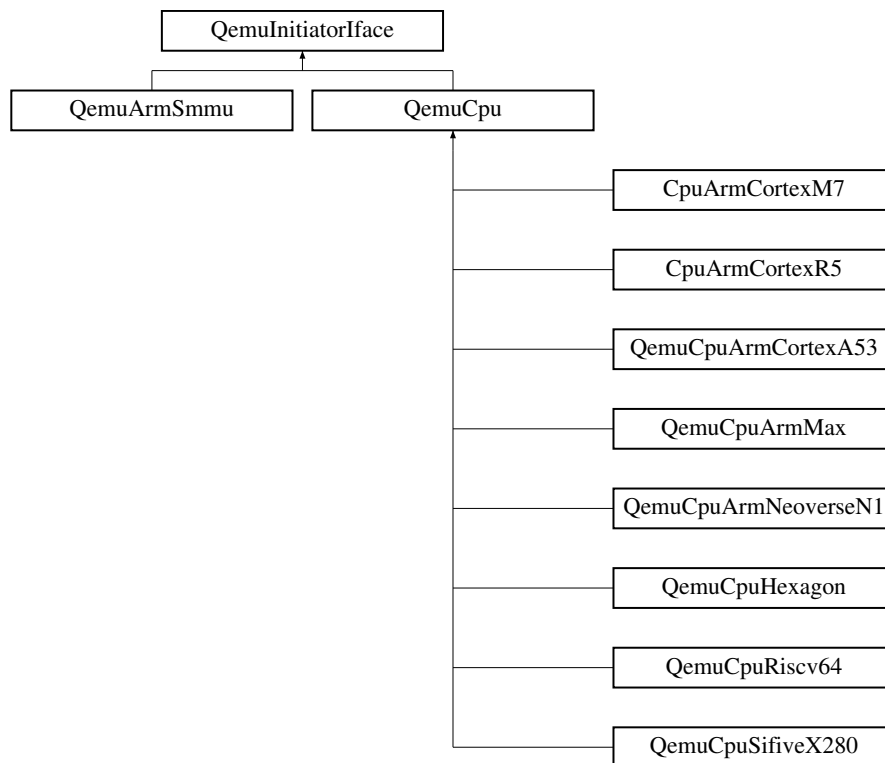
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/timer/hexagon-qtimer.h

## 4.27 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 void **initiator\_tidy\_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:

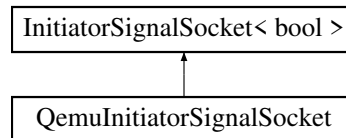
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator.h

## 4.28 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

#### 4.28.1 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 internal 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.

## 4.28.2 Member Function Documentation

### 4.28.2.1 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	<i>dev</i>	The QEMU device
in	<i>gpio_idx</i>	The GPIO index within the device

### 4.28.2.2 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	<i>dev</i>	The QEMU device
in	<i>gpio_name</i>	The GPIO namespace within the device
in	<i>gpio_idx</i>	The GPIO index within the device

### 4.28.2.3 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. This is only for "sysbus\_irq", if you want to wrap a normal gpio, just use init or init\_named.

#### Parameters

in	<i>sbd</i>	The QEMU SysBusDevice
in	<i>gpio_idx</i>	The GPIO index within the SBD

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

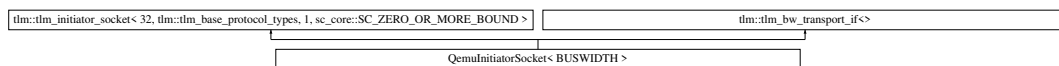
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator-signal-socket.h

## 4.29 QemuInitiatorSocket< BUSWIDTH > Class Template Reference

TLM-2.0 initiator socket specialisation for QEMU AddressSpace mapping.

```
#include <initiator.h>
```

Inheritance diagram for QemuInitiatorSocket< BUSWIDTH >:



### Classes

- class [m\\_mem\\_obj](#)

### Public Types

- using **TlmInitiatorSocket** = tlm::tlm\_initiator\_socket< BUSWIDTH, tlm::tlm\_base\_protocol\_types, 1, sc\_core::SC\_ZERO\_OR\_MORE\_BOUND >
- using **TlmPayload** = tlm::tlm\_generic\_payload
- using **MemTxResult** = qemu::MemoryRegionOps::MemTxResult
- using **MemTxAttrs** = qemu::MemoryRegionOps::MemTxAttrs
- using **DmiRegion** = [QemuInstanceDmiManager::DmiRegion](#)
- using **DmiRegionAlias** = [QemuInstanceDmiManager::DmiRegionAlias](#)
- using **DmiRegionAliasKey** = uint64\_t

### Public Member Functions

- **QemuInitiatorSocket** (const char \*name, [QemuInitiatorIface](#) &initiator, [QemuInstance](#) &inst)
- void **init** (qemu::Device &dev, const char \*prop)
- void **end\_of\_simulation** ()
- void **init\_global** (qemu::Device &dev)
- void **cancel\_all** ()
- 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)

## Protected Member Functions

- void **init\_payload** (TlmPayload &trans, tlm::tlm\_command command, uint64\_t addr, uint64\_t \*val, unsigned int size)
- DmiRegionAliasKey **get\_dmi\_region\_alias\_key** (const tlm::tlm\_dmi &info)
- DmiRegionAliasKey **get\_dmi\_region\_alias\_key** (const [DmiRegionAlias](#) &alias)
- void **add\_dmi\_mr\_alias** ([DmiRegionAlias](#) &alias)
- void **del\_dmi\_mr\_alias** (const [DmiRegionAlias](#) &alias)
- [DmiRegionAlias](#) \* **request\_dmi\_region** (TlmPayload &trans)
- 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

- [QemuInstance](#) & **m\_inst**
- [QemuInitiatorIface](#) & **m\_initiator**
- qemu::Device **m\_dev**
- gs::RunOnSysC **m\_on\_sysc**
- [m\\_mem\\_obj](#) \* **m\_r** = nullptr
- std::map< DmiRegionAliasKey, [DmiRegionAlias](#) > **m\_dmi\_aliases**

### 4.29.1 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:

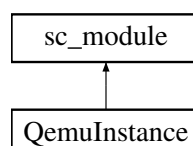
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/initiator.h

## 4.30 QemuInstance Class Reference

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

```
#include <qemu-instance.h>
```

Inheritance diagram for QemuInstance:



## Public Types

- enum **TcgMode** { **TCG\_UNSPECIFIED**, **TCG\_SINGLE**, **TCG\_COROUTINE**, **TCG\_MULTI** }
- using **Target** = qemu::Target
- using **LibLoader** = qemu::LibraryLoaderIface

## Public Member Functions

- void **add\_dev** (QemuDeviceBaseIF \*d)
- void **del\_dev** (QemuDeviceBaseIF \*d)
- bool **can\_run** ()
- TcgMode **StringToTcgMode** (std::string s)
- **QemuInstance** (const sc\_core::sc\_module\_name &n, LibLoader &loader, Target t)
- **QemuInstance** (const QemuInstance &)=delete
- **QemuInstance** (QemuInstance &&)=delete
- bool **operator==** (const QemuInstance &b) const
- bool **operator!=** (const QemuInstance &b) const
- void **add\_arg** (const char \*arg)  
*Add a command line argument to the qemu instance.*
- TcgMode **get\_tcg\_mode** ()  
*Get the TCG mode for this instance.*
- std::shared\_ptr< gs::tlm\_quantumkeeper\_extended > **create\_quantum\_keeper** ()  
*Get the TCG 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**
- cci::cci\_param< std::string > **p\_tcg\_mode**
- cci::cci\_param< std::string > **p\_sync\_policy**
- TcgMode **m\_tcg\_mode**
- cci::cci\_param< bool > **p\_icount**
- cci::cci\_param< int > **p\_icount\_mips**
- cci::cci\_param< std::string > **p\_args**



### 4.30.1 Detailed Description

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

### 4.30.2 Member Function Documentation

#### 4.30.2.1 `add_arg()`

```
void QemuInstance::add_arg (
    const char * arg ) [inline]
```

Add a command line argument to the qemu instance.

This method may only be called before the instance is initialized.

#### 4.30.2.2 `create_quantum_keeper()`

```
std::shared_ptr<gs::tlm_quantumkeeper_extended> QemuInstance::create_quantum_keeper ( ) [inline]
```

Get the TCG mode for this instance.

This method is called by CPU instances determin if to use coroutines or not.

#### 4.30.2.3 `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.

#### 4.30.2.4 `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).

#### 4.30.2.5 `get_tcg_mode()`

```
TcgMode QemuInstance::get_tcg_mode ( ) [inline]
```

Get the TCG mode for this instance.

This method is called by CPU instances determin if to use coroutines or not.

#### 4.30.2.6 `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.

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h`

## 4.31 QemuInstanceDmiManager Class Reference

Handles the DMI regions at the QEMU instance level.

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

### Classes

- class [DmiRegion](#)  
*a DMI region*
- class [DmiRegionAlias](#)  
*An alias to a DMI region.*
- class [QemuContainer](#)

### Public Member Functions

- **QemuInstanceDmiManager** (qemu::LibQemu &inst)
- **QemuInstanceDmiManager** (const [QemuInstanceDmiManager](#) &)=delete
- **QemuInstanceDmiManager** ([QemuInstanceDmiManager](#) &&a)
- [DmiRegionAlias](#) **get\_new\_region\_alias** (const tlm::tlm\_dmi &info)  
*Create a new alias for the DMI region designated by `info`*

### Protected Member Functions

- DmiRegion::Ptr **create\_region** (const tlm::tlm\_dmi &info)
- DmiRegion::Ptr **get\_region** (const tlm::tlm\_dmi &info)

## Protected Attributes

- `qemu::LibQemu & m_inst`
- `std::mutex m_mutex`
- `std::map< DmiRegion::Key, std::weak_ptr< DmiRegion > > m_regions`

## Friends

- class `LockedQemuInstanceDmiManager`

### 4.31.1 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).

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/dmi-manager.h`

## 4.32 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 (const std::string &n, Target t)`  
*Returns a new QEMU instance for target t.*
- `QemuInstance & new_instance (Target t)`

## Protected Attributes

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

### 4.32.1 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.

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 QemuInstanceManager()

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

Construct a [QemuInstanceManager](#) by providing a custom library loader.

#### Parameters

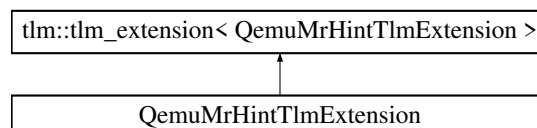
in	<i>loader</i>	The custom loader
----	---------------	-------------------

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/qemu-instance.h`

## 4.33 QemuMrHintTlmExtension Class Reference

Inheritance diagram for QemuMrHintTlmExtension:



## Public Member Functions

- **QemuMrHintTlmExtension** (const [QemuMrHintTlmExtension](#) &)=default
- **QemuMrHintTlmExtension** (qemu::MemoryRegion mr, uint64\_t offset)

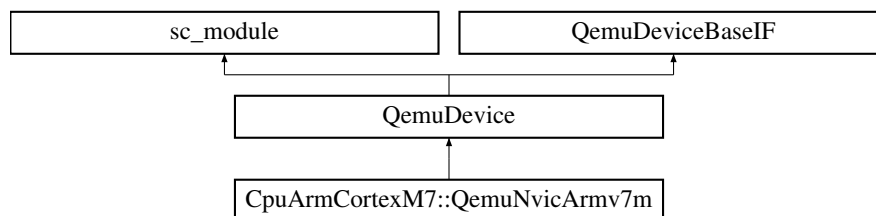
- virtual tlm\_extension\_base \* **clone** () const override
- virtual void **copy\_from** (tlm\_extension\_base const &ext) override
- qemu::MemoryRegion **get\_mr** () const
- uint64\_t **get\_offset** () const

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/tlm-extensions/qemu-mr-hint.h

## 4.34 CpuArmCortexM7::QemuNvicArmv7m Class Reference

Inheritance diagram for CpuArmCortexM7::QemuNvicArmv7m:



### Public Member Functions

- **QemuNvicArmv7m** (const sc\_core::sc\_module\_name &n, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

### Public Attributes

- cci::cci\_param< unsigned int > **p\_num\_irq**
- [QemuTargetSocket](#) **socket**
- sc\_core::sc\_vector< [QemuTargetSignalSocket](#) > **irq\_in**

### Protected Attributes

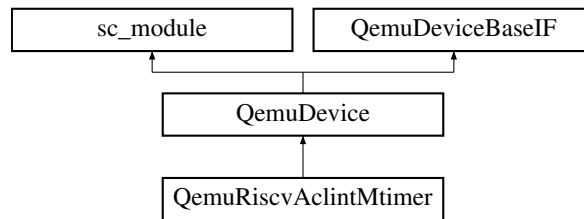
- bool **before\_end\_of\_elaboration\_done**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cpu/arm/cortex-m7.h

## 4.35 QemuRiscvAclintMtimer Class Reference

Inheritance diagram for QemuRiscvAclintMtimer:



### Public Member Functions

- **QemuRiscvAclintMtimer** (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\_timecmp\_base**
- cci::cci\_param< uint64\_t > **p\_time\_base**
- cci::cci\_param< uint64\_t > **p\_aperture\_size**
- cci::cci\_param< uint32\_t > **p\_timebase\_freq**
- cci::cci\_param< bool > **p\_provide\_rdtm**
- [QemuTargetSocket](#) **socket**

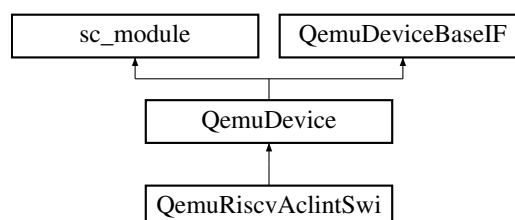
### Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/timer/riscv-aclint-mtimer.h↩

## 4.36 QemuRiscvAclintSwi Class Reference

Inheritance diagram for QemuRiscvAclintSwi:



## Public Member Functions

- **QemuRiscvAclintSwi** (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**
- [QemuTargetSocket](#) **socket**

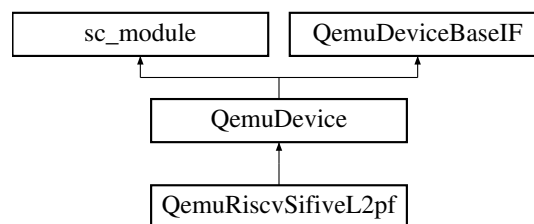
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/riscv-aclint-swi.h

## 4.37 QemuRiscvSifiveL2pf Class Reference

Inheritance diagram for QemuRiscvSifiveL2pf:



## Public Member Functions

- **QemuRiscvSifiveL2pf** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

## Public Attributes

- [QemuTargetSocket](#) **socket**

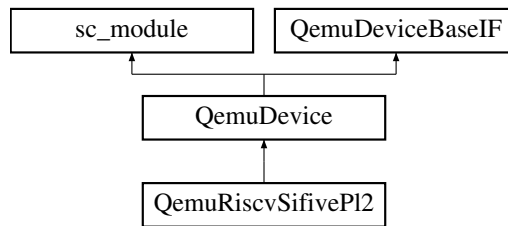
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cache-ctrl/sifive-l2pf.h

## 4.38 QemuRiscvSifivePI2 Class Reference

Inheritance diagram for QemuRiscvSifivePI2:



### Public Member Functions

- **QemuRiscvSifivePI2** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

### Public Attributes

- [QemuTargetSocket](#) socket

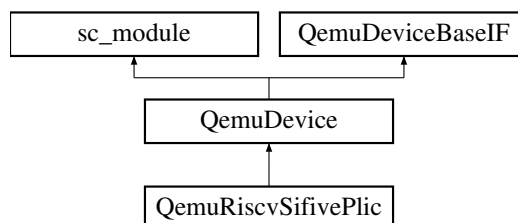
### Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/cache-ctrl/sifive-pl2.h

## 4.39 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**

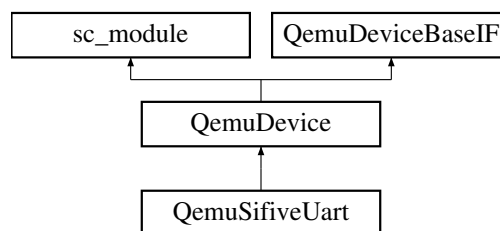
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/irq-ctrl/plic-sifive.h

## 4.40 QemuSifiveUart Class Reference

Inheritance diagram for QemuSifiveUart:



## Public Member Functions

- **QemuSifiveUart** (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**
- gs::async\_event **m\_ext\_ev**

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

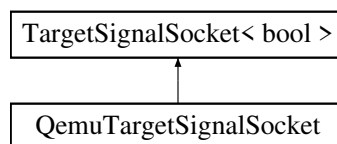
- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/uart/sifive-uart.h

## 4.41 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.*
- void **notify** ()  
*Force a notification on the default event.*

## Protected Member Functions

- void **value\_changed\_cb** (const bool &val)
- void **init\_with\_gpio** (qemu::Gpio gpio)

## Protected Attributes

- qemu::Gpio **m\_gpio\_in**

### 4.41.1 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.

### 4.41.2 Member Function Documentation

#### 4.41.2.1 `get_gpio()`

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

Returns the GPIO wrapped by this socket.

##### Returns

the GPIO wrapped by this socket

#### 4.41.2.2 `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	<i>dev</i>	The QEMU device
in	<i>gpio_idx</i>	The GPIO index within the device

#### 4.41.2.3 `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

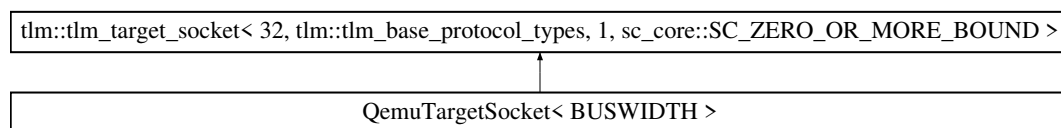
in	<i>dev</i>	The QEMU device
in	<i>gpio_name</i>	The GPIO namespace within the device
in	<i>gpio_idx</i>	The GPIO index within the device

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target-signal-socket.h

## 4.42 QemuTargetSocket< BUSWIDTH > Class Template Reference

Inheritance diagram for QemuTargetSocket< BUSWIDTH >:



### Public Types

- using **TlmTargetSocket** = tlm::tlm\_target\_socket< BUSWIDTH, tlm::tlm\_base\_protocol\_types, 1, sc\_core::SC\_ZERO\_OR\_MORE\_BOUND >
- using **TlmPayload** = tlm::tlm\_generic\_payload

### Public Member Functions

- **QemuTargetSocket** (const char \*name, [QemuInstance](#) &inst)
- void **init** (qemu::SysBusDevice sbd, int mmio\_idx)
- void **init\_with\_mr** (qemu::MemoryRegion mr)

### Protected Attributes

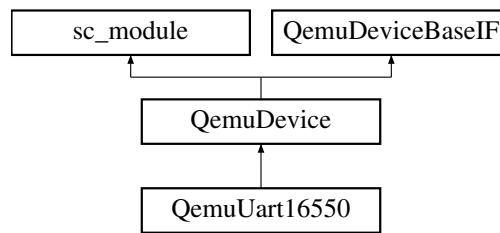
- [TlmTargetToQemuBridge](#) **m\_bridge**
- [QemuInstance](#) & **m\_inst**
- qemu::SysBusDevice **m\_sbd**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target.h

## 4.43 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

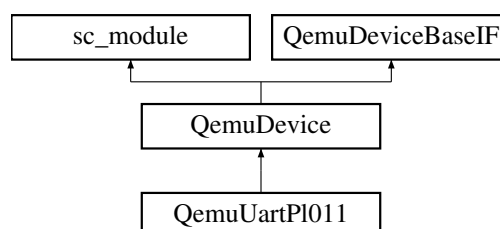
- `qemu::Chardev` **m\_chardev**
- `cci::cci_param< unsigned int >` **p\_baudbase**
- `cci::cci_param< unsigned int >` **p\_regshift**

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/uart/16550.h`

## 4.44 QemuUartPI011 Class Reference

Inheritance diagram for QemuUartPI011:



## Public Member Functions

- **QemuUartPI011** (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

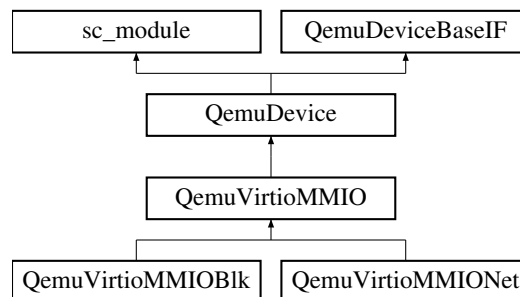
- qemu::Chardev **m\_chardev**
- gs::async\_event **m\_ext\_ev**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/uart/pl011.h

## 4.45 QemuVirtioMMIO Class Reference

Inheritance diagram for QemuVirtioMMIO:



## Public Member Functions

- **QemuVirtioMMIO** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst, const char \*device\_type)
- void **before\_end\_of\_elaboration** () override
- void **end\_of\_elaboration** () override

## Public Attributes

- [QemuTargetSocket](#) **socket**
- [QemuInitiatorSignalSocket](#) **irq\_out**
- [QemuDevice](#) **virtio\_mmio\_device**

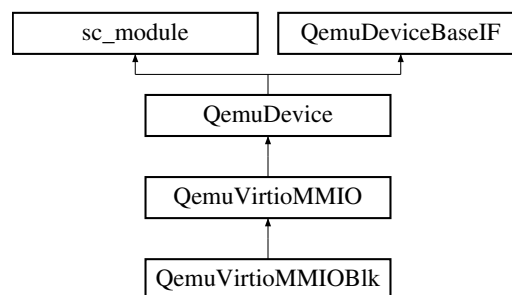
## Additional Inherited Members

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/virtio/virtio-mmio.h`

## 4.46 QemuVirtioMMIOBlk Class Reference

Inheritance diagram for QemuVirtioMMIOBlk:



## Public Member Functions

- **QemuVirtioMMIOBlk** (`sc_core::sc_module_name nm`, [QemuInstance](#) &inst)
- `void before_end_of_elaboration ()` override

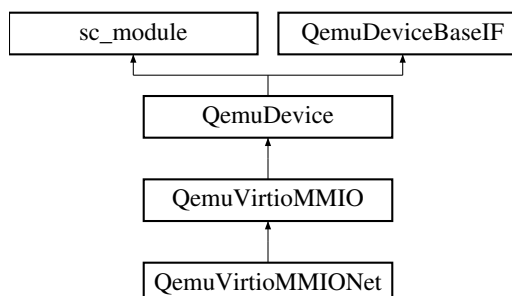
## Additional Inherited Members

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

- `/home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/blk/virtio-mmio-blk.h`

## 4.47 QemuVirtioMMIONet Class Reference

Inheritance diagram for QemuVirtioMMIONet:



## Public Member Functions

- **QemuVirtioMMIONet** (sc\_core::sc\_module\_name nm, [QemuInstance](#) &inst)
- void **before\_end\_of\_elaboration** () override

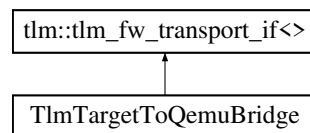
## Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/components/net/virtio-mmio-net.h

## 4.48 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)
- void **init\_with\_mr** (qemu::MemoryRegion mr)
- 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

- void **init\_as** ()
- qemu::Cpu **push\_current\_cpu** (TlmPayload &trans)
- void **pop\_current\_cpu** (qemu::Cpu cpu)

## Protected Attributes

- qemu::MemoryRegion **m\_mr**
- std::shared\_ptr< qemu::AddressSpace > **m\_as**

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

- /home/thomas/Documents/GreenSocs/build-lib/libqbox/include/libqbox/ports/target.h



# Index

- add\_arg
  - QemuInstance, [43](#)
- CpuArmCortexM7, [13](#)
- CpuArmCortexM7::QemuNvicArmv7m, [47](#)
- CpuArmCortexR5, [14](#)
- create\_quantum\_keeper
  - QemuInstance, [43](#)
- get
  - QemuInstance, [43](#)
- get\_dmi\_manager
  - QemuInstance, [43](#)
- get\_gpio
  - QemuTargetSignalSocket, [53](#)
- get\_new\_region\_alias
  - LockedQemuInstanceDmiManager, [18](#)
- get\_tcg\_mode
  - QemuInstance, [43](#)
- init
  - QemuInitiatorSignalSocket, [39](#)
  - QemuInstance, [44](#)
  - QemuTargetSignalSocket, [53](#)
- init\_named
  - QemuInitiatorSignalSocket, [39](#)
  - QemuTargetSignalSocket, [53](#)
- init\_sbd
  - QemuInitiatorSignalSocket, [39](#)
- invalidate\_region
  - QemuInstanceDmiManager::DmiRegionAlias, [16](#)
- is\_installed
  - QemuInstanceDmiManager::DmiRegionAlias, [16](#)
- is\_valid
  - QemuInstanceDmiManager::DmiRegionAlias, [16](#)
- LockedQemuInstanceDmiManager, [17](#)
  - get\_new\_region\_alias, [18](#)
- QboxException, [18](#)
- QemuArmGicv2, [19](#)
- QemuArmGicv2m, [20](#)
- QemuArmGicv3, [21](#)
- QemuArmSmmu, [22](#)
- QemuCpu, [23](#)
- QemuCpu::QemuCpuHintTlmExtension, [29](#)
- QemuCpuArmCortexA53, [25](#)
- QemuCpuArmMax, [26](#)
- QemuCpuArmNeoverseN1, [27](#)
- QemuCpuHexagon, [28](#)
- QemuCpuHintTlmExtension, [30](#)
- QemuCpuRiscv64, [30](#)
- QemuCpuRiscv64Rv64, [31](#)
- QemuCpuSifiveX280, [32](#)
- QemuDevice, [33](#)
  - QemuDevice, [34](#)
- QemuDeviceBaseIF, [34](#)
- QemuHexagonL2vic, [35](#)
- QemuHexagonQtimer, [36](#)
- QemuInitiatorIface, [37](#)
- QemuInitiatorSignalSocket, [38](#)
  - init, [39](#)
  - init\_named, [39](#)
  - init\_sbd, [39](#)
- QemuInitiatorSocket< BUSWIDTH >, [40](#)
- QemuInitiatorSocket< BUSWIDTH >::m\_mem\_obj, [18](#)
- QemuInstance, [41](#)
  - add\_arg, [43](#)
  - create\_quantum\_keeper, [43](#)
  - get, [43](#)
  - get\_dmi\_manager, [43](#)
  - get\_tcg\_mode, [43](#)
  - init, [44](#)
- QemuInstanceDmiManager, [44](#)
- QemuInstanceDmiManager::DmiRegion, [14](#)
- QemuInstanceDmiManager::DmiRegionAlias, [15](#)
  - invalidate\_region, [16](#)
  - is\_installed, [16](#)
  - is\_valid, [16](#)
  - set\_installed, [17](#)
- QemuInstanceDmiManager::QemuContainer, [23](#)
- QemuInstanceManager, [45](#)
  - QemuInstanceManager, [46](#)
- QemuMrHintTlmExtension, [46](#)
- QemuRiscvAclintMtimer, [48](#)
- QemuRiscvAclintSwi, [48](#)
- QemuRiscvSifiveL2pf, [49](#)
- QemuRiscvSifivePI2, [50](#)
- QemuRiscvSifivePlic, [50](#)
- QemuSifiveUart, [51](#)
- QemuTargetSignalSocket, [52](#)
  - get\_gpio, [53](#)
  - init, [53](#)
  - init\_named, [53](#)
- QemuTargetSocket< BUSWIDTH >, [54](#)
- QemuUart16550, [55](#)
- QemuUartPIO11, [55](#)
- QemuVirtioMMIO, [56](#)
- QemuVirtioMMIOBlk, [57](#)
- QemuVirtioMMIONet, [57](#)

set\_installed

    QemuInstanceDmiManager::DmiRegionAlias, [17](#)

TlmTargetToQemuBridge, [58](#)