

# VciBlockDevice

## 1) Functional Description

This VCI component is both a target and an initiator.

- It is addressed as a target to be configured for a transfer.
- It is acting as an initiator to do the transfer

There is only one block device handled by this component. It can be seen as one single *file*, that has a storage capacity limited to  $2^{41}$  bytes. An IRQ is optionally asserted when transfer is finished.

This hardware component checks for segmentation violation, and can be used as a default target.

It contains 8 memory-mapped registers:

- **BLOCK\_DEVICE\_BUFFER** (read/write)

Physical address of the source (or destination) buffer in SoC memory.

- **BLOCK\_DEVICE\_COUNT** (read/write)

Number of blocks to be transfered.

- **BLOCK\_DEVICE\_LBA** (read/write)

Logical Base Address (index of the first block in the block device)

- **BLOCK\_DEVICE\_OP** (write only)

Type of operation, writing here initiates the operation. This register goes back to BLOCK\_DEVICE\_NOOP when operation is finished.

- **BLOCK\_DEVICE\_STATUS** (read only)

State of the transfer. Reading this register while not busy resets its value to IDLE, and acknowledge the IRQ. Value may be one of :

1. BLOCK\_DEVICE\_IDLE
2. BLOCK\_DEVICE\_BUSY
3. BLOCK\_DEVICE\_READ\_SUCCESS
4. BLOCK\_DEVICE\_WRITE\_SUCCESS
5. BLOCK\_DEVICE\_READ\_ERROR
6. BLOCK\_DEVICE\_WRITE\_ERROR
7. BLOCK\_DEVICE\_ERROR

- **BLOCK\_DEVICE\_IRQ\_ENABLE** (read/write)

Boolean enabling the IRQ line

- **BLOCK\_DEVICE\_SIZE** (read only)

Number of blocks addressable in the block device

- **BLOCK\_DEVICE\_BLOCK\_SIZE** (read only)

Block size (in bytes)

The following operations codes are defined:

1. **BLOCK\_DEVICE\_NOOP** Nothing
2. **BLOCK\_DEVICE\_READ** from flock device to memory
3. **BLOCK\_DEVICE\_WRITE** from memory to block device

For extensibility issues, you should access this component using globally-defined offsets. You should include file `soclib/block_device.h` from your software, it defines `BLOCK_DEVICE_COUNT`, `BLOCK_DEVICE_READ`, ...

Sample code: Please see reference implementation in [source:trunk/soclib/soclib/platform/topcells/caba-vgmn-block\\_device-mips32el](source:trunk/soclib/soclib/platform/topcells/caba-vgmn-block_device-mips32el)

(add `-I/path/to/soclib/include` to your compilation command-line)

## 2) Component definition & usage

[source:trunk/soclib/soclib/module/connectivity\\_component/vci\\_block\\_device/caba/metadata/vci\\_block\\_device.sd?](source:trunk/soclib/soclib/module/connectivity_component/vci_block_device/caba/metadata/vci_block_device.sd?)

See [SoclibCc/VciParameters](#)

```
Uses( 'vci_block_device', **vci_parameters )
```

## 3) CABA Implementation

### CABA sources

- interface :  
[source:trunk/soclib/soclib/module/connectivity\\_component/vci\\_block\\_device/caba/source/include/vci\\_block\\_device.h](source:trunk/soclib/soclib/module/connectivity_component/vci_block_device/caba/source/include/vci_block_device.h)
- implementation :  
[source:trunk/soclib/soclib/module/connectivity\\_component/vci\\_block\\_device/caba/source/src/vci\\_block\\_device.cpp?](source:trunk/soclib/soclib/module/connectivity_component/vci_block_device/caba/source/src/vci_block_device.cpp?)

### CABA Constructor parameters

```
VciBlockDevice(
    sc_module_name name,    // Component Name
    const soclib::common::MappingTable &mt, // MappingTable
    const soclib::common::IntTab &srcid,    // Initiator index
    const soclib::common::IntTab &tgtid,    // Target index
    const std::string &filename, // mapped file, may be a host block device
    const uint32_t block_size = 512 ); // one-block size
```

## CABA Ports

- `sc_in<bool> p_resetn` : Global system reset
- `sc_in<bool> p_clk` : Global system clock
- `soclib::caba::VciTarget<vci_param> p_vci_target` : The VCI target port
- `soclib::caba::VciInitiator<vci_param> p_vci_initiator` : The VCI initiator port
- `sc_out<bool> p_irq` : Interrupt port

## 4) TLM-DT Implementation

The TLM-DT implementation is not yet available.