

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, 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 the following memory-mapped registers:

- **BLOCK_DEVICE_BUFFER** Physical address of the buffer in SoC memory
- **BLOCK_DEVICE_COUNT** Count of blocks to transfer
- **BLOCK_DEVICE_LBA** Base sector for transfer
- **BLOCK_DEVICE_OP** Type of operation, writing here initiates the operation.
This register goes back to **BLOCK_DEVICE_NOOP** when operation is finished. (write only)
- **BLOCK_DEVICE_STATUS** State of the transfer. Reading this register while not busy resets its value to **IDLE**. Value may be one of
 - ◆ **BLOCK_DEVICE_IDLE**
 - ◆ **BLOCK_DEVICE_BUSY**
 - ◆ **BLOCK_DEVICE_READ_SUCCESS**
 - ◆ **BLOCK_DEVICE_WRITE_SUCCESS**
 - ◆ **BLOCK_DEVICE_READ_ERROR**
 - ◆ **BLOCK_DEVICE_WRITE_ERROR**
 - ◆ **BLOCK_DEVICE_ERROR**
- **BLOCK_DEVICE_IRQ_ENABLE** Boolean enabling the IRQ line
- **BLOCK_DEVICE_SIZE** Number of blocks addressable in the controller (read-only)
- **BLOCK_DEVICE_BLOCK_SIZE** Block size (in bytes) (read-only)

The following operations codes are defined:

- **BLOCK_DEVICE_NOOP** Nothing
- **BLOCK_DEVICE_READ** `read()`
- **BLOCK_DEVICE_WRITE** `write()`

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

(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?

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
- implementation :
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-T Implementation

The TLM-T implementation is not yet available.