

TEMU

OpenCores CAN Device Model Manual

Mattias Holm

Version 1.0, 2016-04-05

Table of Contents

| | |
|------------------------|---|
| 1. Introduction | 1 |
| 2. Configuration | 1 |
| 3. Attributes | 1 |
| 3.1. Properties | 1 |
| 3.2. Interfaces | 3 |
| 3.3. Ports | 3 |
| 4. Limitations | 3 |

Table 1. Record of Changes

| Rev | Date | Author | Note |
|-----|------------|--------|------------------|
| 1.0 | 2016-04-05 | MH | Initial version. |

1. Introduction

The CAN_OC is part of the OpenCores and the GRLIB IP libraries. It is available in libTEMUOpenCores.so.

2. Configuration

There are two configuration parameters in the CAN device, firstly the config.interrupt property can be set to influence the interrupt that is raised with the IRQ controller (setting this property also updates the AHB PnP info).

The second configuration property is config.infiniteSpeed, with this set, messages are sent immediately instead of being scheduled for the future based on the message length. This is the default option.

The device should be connected to an interrupt controller and a CAN bus, to work properly.

3. Attributes

3.1. Properties

| Name | Type | Description |
|----------------------|------------------|---|
| basiccan.acceptCode | uint8_t | Accept Code register for BasicCAN mode. |
| basiccan.acceptMask | uint8_t | Accept Mask register for BasicCAN mode. |
| basiccan.ctrl | uint8_t | Control register for BasicCAN mode. |
| basiccan.txID | [2 x uint8_t] | TxID registers for BasicCAN mode. |
| bus | iref / <unknown> | CAN bus the device is connected to. |
| busTiming | [2 x uint8_t] | Bus Timing registers. |
| clockDivider | uint8_t | Clock Divider register. |
| command | uint8_t | Command register. |
| config.infiniteSpeed | uint8_t | Enable infinite speed mode (no delays when sending messages). |

| Name | Type | Description |
|-------------------------|------------------|---|
| config.interrupt | uint8_t | External interrupt raised with IRQ controller. |
| fifo.data | [64 x uint8_t] | RX FIFO data buffer. |
| fifo.start | uint32_t | RX FIFO buffer start location. |
| fifo.usage | uint32_t | RX FIFO buffer usage. |
| interrupt | uint8_t | Interrupt register. |
| irqCtrl | iref / <unknown> | Interrupt controller. |
| object.timeSource | object | Time source object (a cpu or machine object) |
| pelican.acceptCode | [4 x uint8_t] | Accept Code registers for PeliCAN mode. |
| pelican.acceptMask | [4 x uint8_t] | Accept Mask registers for PeliCAN mode. |
| pelican.arbLostCaputure | uint8_t | Arbitration Lost Capture register for PeliCAN mode. |
| pelican.errCodeCapture | uint8_t | Error Code Capture register for PeliCAN mode. |
| pelican.errWarnLimit | uint8_t | Error Warning Limit register for PeliCAN mode. |
| pelican.interruptEnable | uint8_t | Interrupt Enable register for PeliCAN mode. |
| pelican.mode | uint8_t | Mode register for PeliCAN mode. |
| pelican.rxErrCounter | uint8_t | RX Error Counter register for PeliCAN mode. |
| pelican.rxMsgCounter | uint8_t | RX Message Counter register for PeliCAN mode. |
| pelican.txErrCounter | uint8_t | TX Error Counter register for PeliCAN mode. |
| pelican.txFI | uint8_t | TX Frame Info register for PeliCAN mode. |
| pelican.txID | [4 x uint8_t] | TxID registers for PeliCAN mode. |
| status | uint8_t | Status register. |
| txData | [8 x uint8_t] | TX data buffer (excluding TX FI and TX ID registers). |

3.2. Interfaces

| Name | Type | Description |
|----------------|----------------|--|
| AhbIface | AhbIface | AHB interface |
| CanDevIface | CanDevIface | CAN device interface. |
| DeviceIface | DeviceIface | Device interface. |
| MemAccessIface | MemAccessIface | Memory access interface for memory mapped registers. |
| ResetIface | ResetIface | |

3.3. Ports

| Prop | Iface | Description |
|------|-------|-------------|
| - | - | - |

4. Limitations

The following deviations from real hardware are known to exist with this model:

- The controller clears the RX and TX buffers on reset. This is not the proper behaviour and may have an impact on FDIR. Let us know if this is an issue.
- With all CAN models, there is no arbitration of messages in the simulated world and busses are not synchronised.
- The model does at present not register filters with the CAN bus model.
- The model currently ignores the error field in the CAN frame objects.
- The model currently assumes the CAN bus is running at 1 Mb/s (in non-infinite speed mode). This is arguably incorrect and the timing should be picked from the bus timing register, this has however not yet been done. Contact Terma if this is this is critical for your needs.