# CONV-TTL-BLO
# User Guide

Theodor-Adrian Stana
Carlos Gil Soriano
BE-CO-HT

February 25, 2013


**Abstract**

This document describes the CONV-TTL-BLO board, a Blocking pulse repetitor board in double height VME format. It replaces all the following boards:

- 8 channel repeater
- 16 channel repeater
- CTDAC
- LA-BLO-TTL
- LAF-BLO-TTL
- LASB-TTL-BLO
- LA-GATE
- LA-TTL-BLO
- LAPF-TTL-BLO[1]

[1]For replacing this board a pulse width of 4us must be set.

# Contents

# List of Figures



# List of Tables

# 1 Introduction

CONV-TTL-BLO is a board intended for replicating Blocking Pulses, offering six totally independent replication channels. The shape of the pulses is defined in [1]. CONV-TTL-BLO works together with two more boards: CONV-TTL-RTM and CONV-TTL-RTM-BLO.

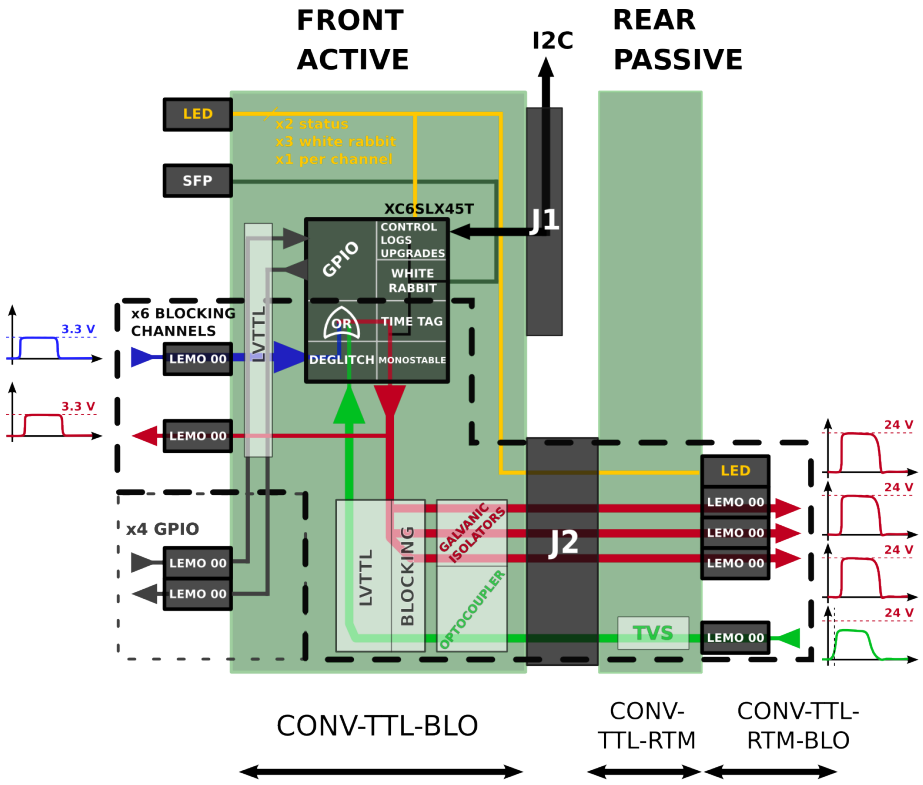


Figure 1: Pulse Repetition system

CONV-TTL-BLO holds all the active circuitry and it is connected as a Front Module to a VME64 backplane. CONV-TTL-RTM and CONV-TTL-RTM-BLO are both connected to the rear part of the crate and provide, in the rear panel, the connectivity of the I/O Blocking lines. Every channel offers, in the Rear Panel, three Blocking Pulse outputs and one Blocking Pulse input.

CONV-TTL-RTM is a motherboard attached to the Rear Transition Module of the P2 VME64 connector. It connects CONV-TTL-BLO to CONV-TTL-RTM-BLO and provides overvoltage protection for all the I/Os of all the channels.

CONV-TTL-RTM-BLO is a piggyback board mounted on CONV-TTL-RTM. It holds all the LEMO 00 connectors and channel LEDs that are offered in the Rear Panel.

Table 1: Boards for Blocking repetition

| Board | Connection | Ports |
|---|---|---|
| *CONV-TTL-BLO* | Front | SFP<br>TTL Blocking triggers<br>TTL Blocking output replica inverters |
| *CONV-TTL-RTM* | Back | - |
| *CONV-TTL-RTM-BLO* | Back | Blocking Pulse input<br>Blocking Pulse outputs |

# 2 Main board front panel

The front panel of CONV-TTL-BLO boards is shown in Figure 2). It consists of status leds and several ports, divided in three sections from top to bottom:

- SFP connector: [**A**].
- Blocking connectors: [**B**] and [**C**].
- General Purpose connectors: [**D**] and [**E**].

## 2.1 Status LEDs

In the current version of the CONV-TTL-BLO boards, only several of the status LEDs present on the board are being used, due to limited firmware support in the FPGA. The implemented status LEDs are presented in Table 2. Unimplemented status LEDs are off by default.

## 2.2 SFP connector

This connector is used to add White Rabbit support to the CONV-TTL-BLO boards. If an optic fibre cable is connected to this socket, White Rabbit precise time-stamping can be added to CONV-TTL-BLO. Three LEDs above the connector are provisioned to show the status of the White Rabbit link.

White Rabbit is currently not supported in the CONV-TTL-BLO firmware.

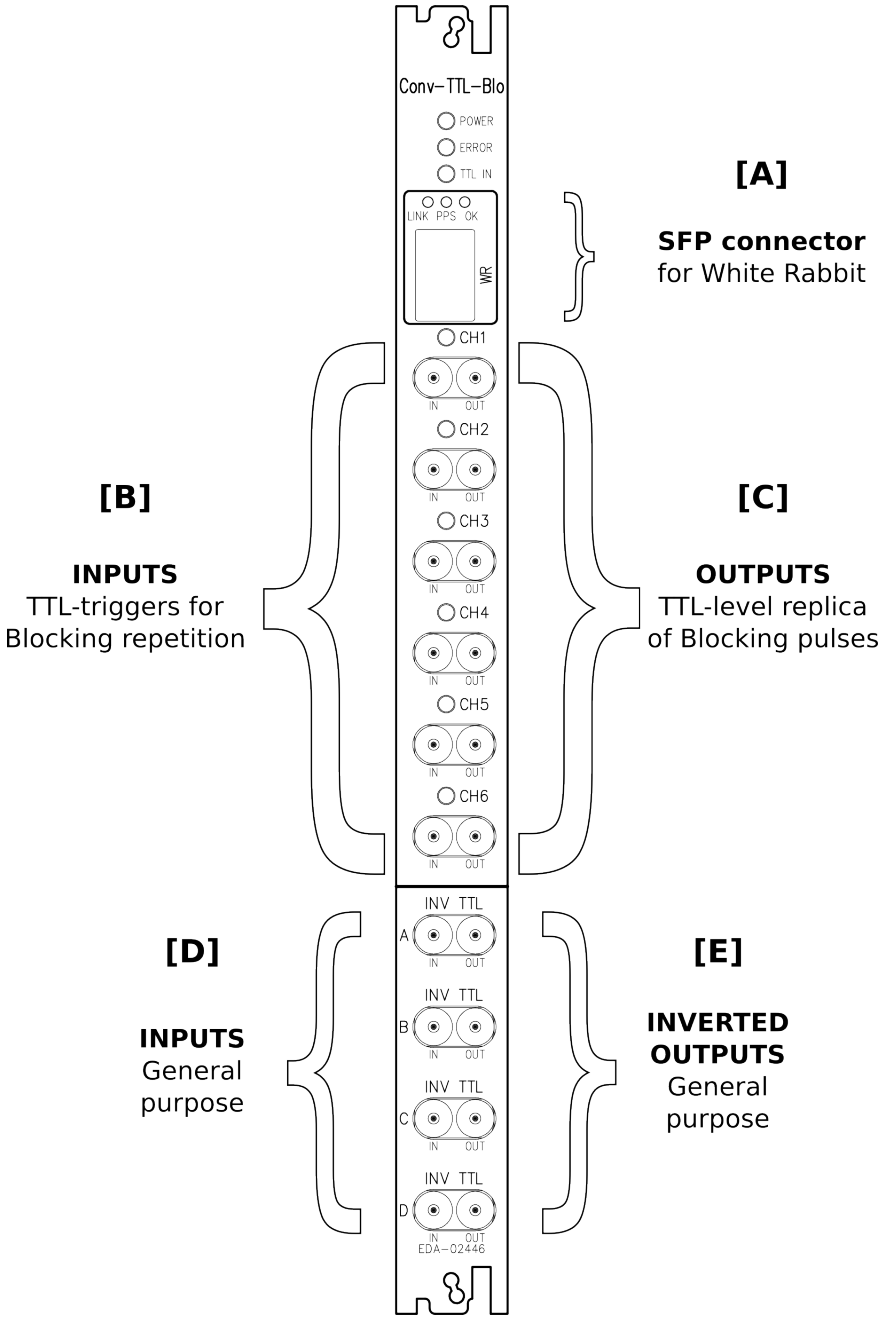


Figure 2: CONV-TTL-BLO Front Panel

Table 2: Status LEDs on CONV-TTL-BLO front panels

| LED | Description |
|---|---|
| PW | Power LED. Lights *green* when a valid CONV-TTL-BLO firmware is loaded to the FPGA. |
| ERR | Error LED. Lights *red* when no rear transition module board is present. |
| TTL_N | Negated-TTL status LED. Lights *green* when negated TTL logic is selected via the $8^{th}$ position of the on-board selection switch. |
| I2C | $I^2C$ status LED. Lights *red* until an $I^2C$ transfer has taken place. Once either a read or a write is successfully completed, the $I^2C$ status LED lights *green* to signal the communication is up. |

### 2.3 TTL triggers

The TTL triggers correspond to block [**B**] in Figure 2. The connectors are LEMO 00 (type EPY). By connecting an external trigger source to one of the connectors a pulse is replicated in a Blocking Pulse level in the Rear Panel and in a TTL panel in the Front Panel. All input channels are 50Ω-terminated.

### 2.4 Repeated TTL pulses

These correspond to block [**C**] in Figure 2. The connectors are LEMO 00 (type EPA). From these connectors a TTL level Blocking Pulse replica of the Rear Panel outputs if offered to the Front Panel. The pulse width of this output is similar to the pulse outputted in the Rear Panel; the rise time and top pulse level are however different from the Blocking output.

When the pulse is outputted, the LED of the corresponding channel blinks for 125 ms.

The TTL output lines are not internally terminated.

### 2.5 General pupose

Four dedicated inverters can be found in the lower part of the Front Panel ([**D**] and [**E**] in Figure 2). The output is a TTL inverted version of the TTL input. The TTL_N outputs are not internally terminated.

## 3 Functional Description

The task of CONV-TTL-BLO is to output a Blocking Pulse upon a reception of a trigger is received. As stated before, CONV-TTL-BLO works together

with CONV-TTL-RTM and CONV-TTL-RTM-BLO in the rear part of the crate. The system formed by these three boards offers three independent channels for outputting Blocking Pulses. Refer to Figure 1 for a visual, whole-system, description.

There are two sources for triggering in every channel: one coming from the Front Panel (CONV-TTL-BLO) and other coming from the Rear Panel (CONV-TTL-RTM through CONV-TTL-RTM-BLO).

The trigger policy is that a Blocking pulse will be outputted whenever either a trigger in the Front Panel or the Rear Panel is detected.

Table 3: Trigger sources

| **Trigger** | **Board** | **Connection** |
|---|---|---|
| TTL | CONV-TTL-BLO | Front Panel |
| Blocking | CONV-TTL-RTM-BLO | Rear Panel |

**Global trigger is OR function of the two sources above**

All the control logic of the system is implemented in a Xilinx Spartan-6 FPGA. Apart from pulse repetition, an $I^2C$ to Wishbone bridge is also implemented. This module translates $I^2C$ accesses as presented in Section 4.2 into accesses to internal memory-mapped registers. The status of the various components in the system can in this way be checked and controlled.

# 4 Accessing internal registers

## 4.1 ELMA crates

CONV-TTL-BLO boards have been designed to operate in ELMA crates. These crates provide a back plane with VME64x connectors which boards can be plugged into. A dedicated board inside the ELMA crates called the SysMon (System Monitor) monitors overall system status and provides access to boards plugged into the VME back plane.

The user can connect to SysMon boards either through a simple RS-232 interface, or through Telnet. In order to send commands to a board plugged into an ELMA crate, the user would connect to the SysMon over one of these two interfaces and board-specific registers to control their functioning.

Since it is the interface most experimented with up to the point of writing of this document, only the Telnet interface is used throughout this document.

## 4.2 Board Addressing

Communication with the CONV-TTL-BLO FPGA is done via $I^2C$ interface through the SERA and SERB pins in P1 VME64x connectors. In order to access a CONV-TTL-BLO board, it is necessary to send:

- The board's 7-bit $I^2C$ address. Every CONV-TTL-BLO has an address that prepends two bits of value *10* to the Geographical Address of the slot according to VME64x specifications.
- An internal CONV-TTL-BLO register address. It is a 16-bit integer in *little endian* format (most significant byte is sent first).

After this, four bytes of data are read/written from/to the internal CONV-TTL-BLO register. These four bytes of data are written in *big endian* format (least significant byte is sent first).

The addressing protocol is thoroughly described in [2].

## 4.3 CONV-TTL-BLO memory map

Table 4 summarizes the registers mapping in the current version of the CONV-TTL-BLO firmware.

Table 4: Memory map of the CONV-TTL-BLO design

| Address | Name | Access | Description |
|---|---|---|---|
| 0x00 | STAT_L | R | Lower 32 bits of system status register |
| 0x04 | STAT_H | R | Upper 32 bits of system status register |
| 0x40 | I2C_CTR0 | R | $I^2C$ control register |
| 0x44 | I2C_LT | R | $I^2C$ line timing register, provides the current $I^2C$ line speed |
| 0x48 | I2C_DTX | R/W | Data to transmit through the $I^2C$ interface |
| 0x4C | I2C_DRXA | R | Lower 32 bits of data received through the $I^2C$ interface |
| 0x50 | I2C_DRXB | R | Upper 32 bits of data received through the $I^2C$ interface |

## 4.4 Register description

### 4.4.1 STAT_L

### 4.4.2 STAT_H

Table 5: STAT_L register

| Bit | Field | Description |
|---|---|---|
| 31..0 | IDENT_L | Lower 32 bits of board identity, as provided by Maxim DS18B20U+ thermometer |

Table 6: STAT_H register

| Bit | Field | Description |
|---|---|---|
| 15..0 | IDENT_H | Upper 32 bits of board identity, as provided by Maxim DS18B20U+ thermometer |
| 18..16 | RTMM | Rear transition module mainboard (RTMM) identification [3] |
| 21..19 | RTMP | Rear transition module piggyback (RTMP) identification [3] |

# 5 Register Description

## 5.1 Access Registers

### 5.1.1 CTR0_ACCESS

The *CTR0* register is a read-write register that enable and disable access to Wishbone devices in the memory map.

| **CTR0_ACCESS** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 0 | EN0 | ENable access to device 0 in memory map |
| 1 | EN1 | ENable access to device 1 in memory map |
| 2 | EN2 | ENable access to device 2 in memory map |
| 3 | EN3 | ENable access to device 3 in memory map |
| 4 | EN4 | ENable access to device 4 in memory map |
| 5 | EN5 | ENable access to device 5 in memory map |
| 6 | EN6 | ENable access to device 6 in memory map |
| 7 | EN7 | ENable access to device 7 in memory map |
| 8 | EN8 | ENable access to device 8 in memory map |
| 9 | EN9 | ENable access to device 9 in memory map |
| 10 | EN10 | ENable access to device 10 in memory map |
| 11 | EN11 | ENable access to device 11 in memory map |
| 31-13 | x | Reserved |

Table 7: CTR0_ACCESS Register

### 5.1.2 CTR1_ACCESS

The *CTR1* register is a read-write register that enable and disable read and write permissions to Wishbone devices in the memory map.

| **CTR1_ACCESS** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 0 | GWR | General write permissions |
| 1 | GRD | General read permissions |
| 2 | WR0 | WRite permissions in memory map 0 |
| 3 | RD0 | ReaD permissions in memory map 0 |
| 4 | WR1 | WRite permissions in memory map 1 |
| 5 | RD1 | ReaD permissions in memory map 1 |
| 6 | WR2 | WRite permissions in memory map 2 |
| 7 | RD2 | ReaD permissions in memory map 2 |
| 8 | WR3 | WRite permissions in memory map 3 |
| 9 | RD3 | ReaD permissions in memory map 3 |
| 10 | WR4 | WRite permissions in memory map 4 |
| 11 | RD4 | ReaD permissions in memory map 4 |
| 12 | WR5 | WRite permissions in memory map 5 |
| 13 | RD5 | ReaD permissions in memory map 5 |
| 14 | WR6 | WRite permissions in memory map 6 |
| 15 | RD6 | ReaD permissions in memory map 6 |
| 16 | WR7 | WRite permissions in memory map 7 |
| 17 | RD7 | ReaD permissions in memory map 7 |
| 18 | WR8 | WRite permissions in memory map 8 |
| 19 | RD8 | ReaD permissions in memory map 8 |
| 20 | WR9 | WRite permissions in memory map 9 |
| 21 | RD9 | ReaD permissions in memory map 9 |
| 22 | WR10 | WRite permissions in memory map 10 |
| 23 | RD10 | ReaD permissions in memory map 10 |
| 24 | WR11 | WRite permissions in memory map 11 |
| 25 | RD11 | ReaD permissions in memory map 11 |
| 31-26 | x | Reserved |

Table 8: CTR1_ACCESS Register

## 5.2 I2C Registers

### 5.2.1 CTR0_I2C

The CTR0 register is a write-read register. It controls the indirect addres and holds the I2C address (which in the case of *CONV-TTL-BLO* will be connected to VME64x geographical address pins).

| **CTR0_I2C** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 0 | EN | general ENable |
| 1 | RST | general ReSeT |
| 2 | PEN | Prescaler ENable |
| 5-3 | x | Reserved |
| 7-6 | BIA | Bytes Indirect Addressing |
| 14-8 | A[6:0] | I2C address |
| 15 | x | Reserved |
| 31-16 | - | Not used |

Table 9: CTR0_I2C Register

### 5.2.2 CTR1_I2C

The CTR1 register is a write-read register. It shows the fsm of the separate *read* and *write* fsms.

| **CTR1_I2C** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 7-0 | RDS | fsm status: ReaD Status |
| 15-8 | WDS | fsm status: WRite Status |
| 31-16 | - | Not used |

Table 10: CTR1_I2C Register

## 5.3 Channel Registers

### 5.3.1 CTR0_CH[x]

The CTR0_CH[x] register is a read-write register. It allows setting up the basic configuration of the trigger HDL core, its RAM blocks and both the deglitching mask and output pulse length to be used.

| **CTR0_CH[x]** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 0 | EN | General ENable |
| 1 | CLR | General CLeaR |
| 3-2 | x | Reserved |
| 4 | EN_TT | Enable Time-Tagging |
| 5 | CLR_TT | Clear Time-Tagging |
| 7-6 | RDM | time-tagging ReaD Mode |
| 15-8 | CGM | Current Glitch Mask |
| 31-16 | CPL | Current Pulse Length |

Table 11: CTR0_CH[x] Register

### 5.3.2 CTR1_CH[x]

The CTR1_CH[x] register is a read-write register. It allows setting up the boundaries that override invalid values of CPL field in CTR0.

| **CTR1_CH[x]** | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 15-0 | MinPL | Minimum Pulse Length |
| 31-16 | MaxPL | Maximum Pulse Length |

Table 12: CTR1_CH[x] Register

### 5.3.3 RAM0_CH[x]

The RAM0 register is a read-write register. It allows setting up the RAM and the read request to it.

| RAM0_CH[x] | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 0 | EN_TT | Enable Time-Tagging |
| 1 | CLR_TT | Clear Time-Tagging |
| 3-2 | RDM | time-tagging ReaD mode |
| 4 | EMPTY | RAM empty |
| 5 | FULL | RAM full |
| 6 | WA | RAM Wrapped Around |
| 7 | RQT | ReQuesT read |
| 31-8 | x | Reserved |

Table 13: RAM0_CH[x] Register

### 5.3.4 RAM1_CH[x]

The RAM1 register is a read-only register. It shows the current read and write address configured to be accessed.

| RAM1_CH[x] | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 15-0 | CRA | Current Read Address |
| 31-16 | CWA | Current Write Address |

Table 14: RAM1_CH[x] Register

### 5.3.5 RAM2_CH[x]

The RAM2 register is a read-write register. It allows setting up the RAM adress range to be read.

| RAM2_CH[x] | | |
|---|---|---|
| **Bits** | **Field** | **Meaning** |
| 15-0 | SA | Starting read Address |
| 31-16 | EA | Ending read Address |

Table 15: RAM2_CH[x] Register

### 5.4 Multiboot Registers

### 5.5 White Rabbit Registers

# 6 Installation

# References

[1] C. G. Soriano, "Standard Blocking Output Signal Definition for CTDAH board," Sept. 2011. http://www.ohwr.org/documents/109.

[2] ELMA, "Access to board data using SNMP and I2C." http://www.ohwr.org/documents/227.

[3] "RTM detection." http://www.ohwr.org/projects/conv-ttl-blo/wiki/RTM_board_detection.