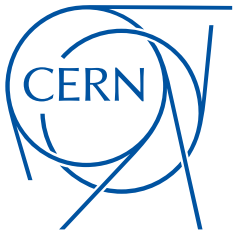
CONV-TTL-BLO HDL Guide

Gateware v0.2
September 29, 2014



Theodor-Adrian Stana (CERN/BE-CO-HT)

Licensing information

Revision history

Date	Version	Change
07-01-2014	1.0	Created document from original HDL guide as the Golden Firmware HDL guide
29-09-2014	2.0	Updated with **new memory map** and using *conv-common-gw*, and added licensing information

Contents

List of Figures

List of Tables

List of abbreviations

FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
LSR	Line Status Register
SR	Status Register

1 Introduction

This document is the HDL guide for the CONV-TTL-BLO board [1]. The HDL for the CONV-TTL-BLO board uses the converter board common gateware [2] as a subproject and adds some external logic to it to adapt for peculiarities on the CONV-TTL-BLO. This short HDL guide explains these peculiarities and the corresponding logic implemented.

1.1 Additional documentation

- Converter board common gateware [2]
- CONV-TTL-BLO User Guide [3]
- CONV-TTL-BLO schematics [4]
- CONV-TTL-BLO OHWR Wiki page [1]

2 Overview

A block diagram of the HDL is shown in Figure 1. This document will detail each of the blocks outside the converter common gateware block in the following sections. The contents of the common gateware block are detailed in the converter common gateware specification [5].

For a more general look at the pulse repetition logic tailored to the CONV-TTL-BLO, refer to the CONV-TTL-BLO User Guide [3].

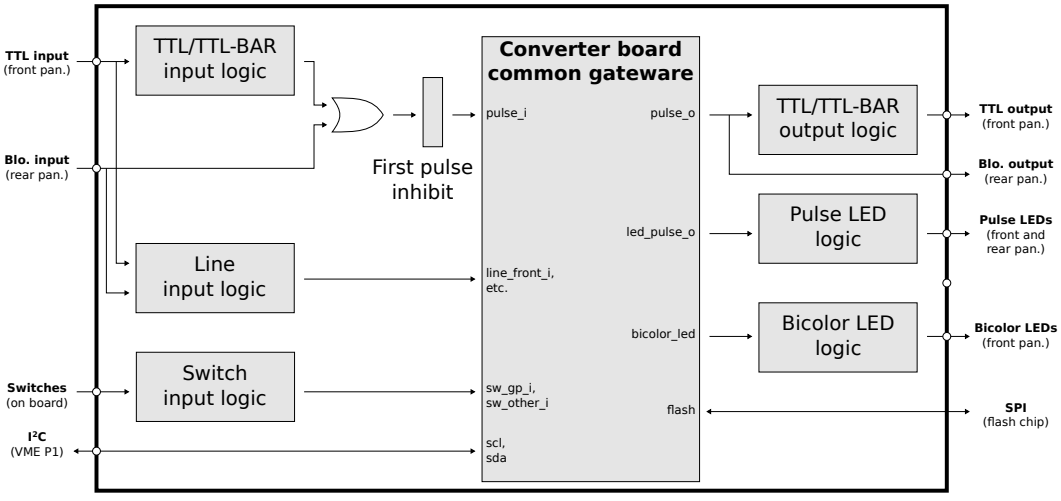


Figure 1: Block diagram of CONV-TTL-BLO gateware

3 Input logic

3.1 TTL/TTL-BAR input logic

The TTL/TTL-BAR input logic is shown in Figure 2. It assures an active-high pulse to the *pulse_i* input of the *conv_common_gw* component and adapts for TTL-BAR pulses that may be input when the TTL switch is on.

In addition, because in TTL-BAR mode a lack of signal on the line is high (due to the on-board Schmitt-trigger buffer), the *no signal detect* block (Figure 3) disables this line if it is high for 100 μs, to allow propagation of blocking pulses arriving on the rear panel while the channel has no cable plugged in while in TTL-BAR mode.

When in TTL-BAR mode, the FRONTFS bits of the lines status register (LSR – see Appendix A.1.4) contain the state of the no signal detect block for each channel and can be used to check if no cable is plugged into the channel. When in TTL mode, the FRONTFS bits are unused.

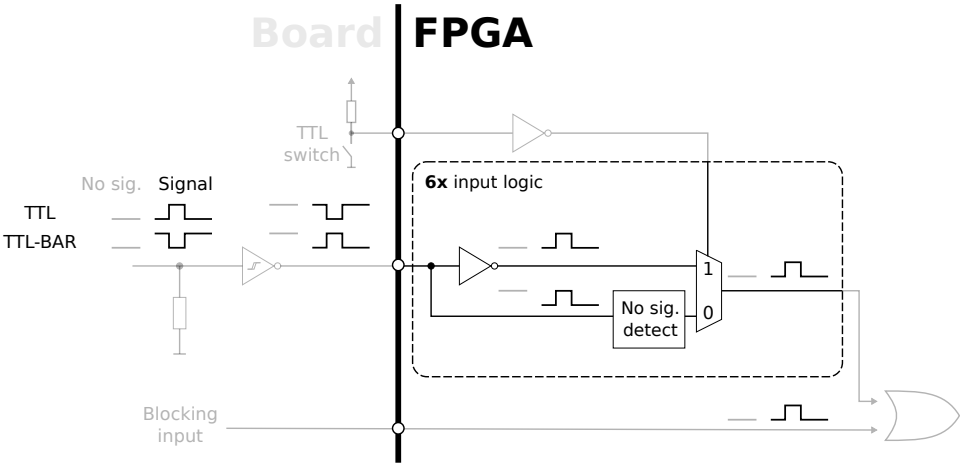


Figure 2: TTL/TTL-BAR input logic

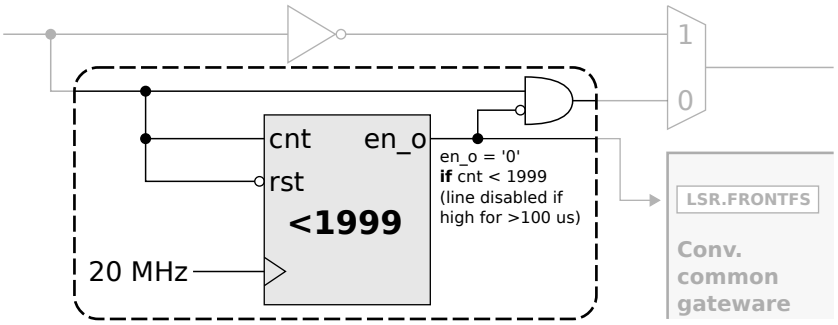


Figure 3: No signal detect block

3.2 First pulse inhibit mechanism

The first pulse inhibit mechanism (Figure 4) is implemented in the form of a counter which waits for 100 μs after reset prior to enabling the line. It is implemented because in TTL-BAR mode, until an inactive line is disabled, the TTL line is high and this may lead to a pulse triggered on the channel, due to reset of modules within the *conv_common_gw* component.

By keeping the line disabled until the no signal detect block in the TTL input logic (Section 3.1) disables the line, no pulse is triggered on the channel. As seen in Figure 4, an extra clock cycle delay is needed before the channel is enabled, to make sure that all reset states inside the *conv_common_gw* block have been finished and no pulses are generated.

To keep the logic simple, the pulse inhibit logic disables the line even when the board is in TTL repetition mode. Since in practice the effect it has on the input to the *conv_common_gw* is extending the 100 ms reset by 100 μs, an extra 0.1% delay from reset to full pulse replication capabilities is deemed insignificant in comparison to logic simplification.

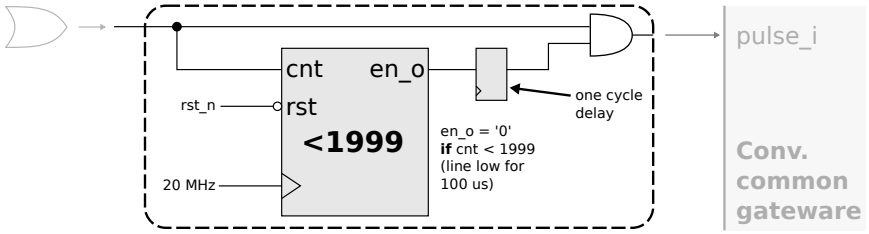


Figure 4: First pulse inhibit mechanism

3.3 Line input logic

The line input logic adapts the levels present at the FPGA inputs due to various on-board circuitry, so that the levels in the line status register (LSR) is active-high. As seen in Figure 5, only the TTL and INV-TTL lines need adaptation in the case of the CONV-TTL-BLO, since the blocking inputs are already adapted on-board for active-high logic.

When in TTL-BAR mode, the FRONTFS bits of the lines status register (LSR – see Appendix A.1.4) contain the state of the no signal detect block for each channel and can be used to check if no cable is plugged into the channel. When in TTL mode, the FRONTFS bits are unused.

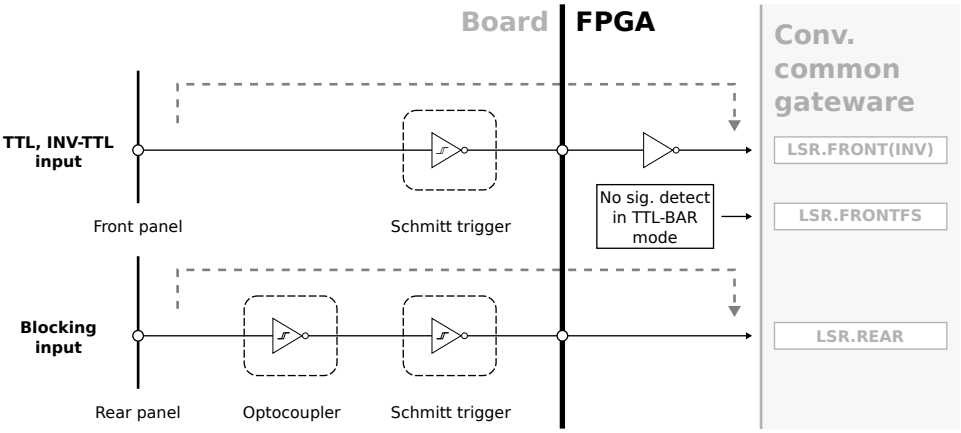


Figure 5: Line input logic

3.4 Switch input logic

Similar to the line input logic (Section 3.3), the general-purpose switch lines must be negated for their active-high reflection in the SR, as shown in Figure 6.

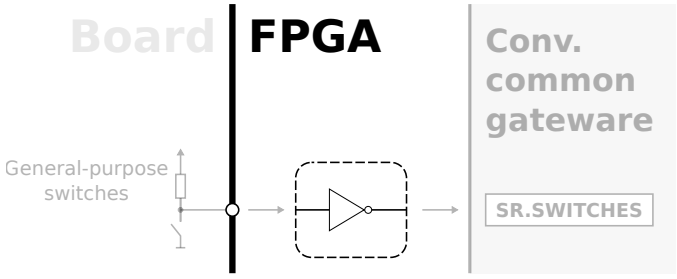


Figure 6: Switch input logic

4 Output logic

4.1 TTL/TTL-BAR output logic

The TTL/TTL-BAR output logic (Figure 7) ensures that TTL pulses are propagated from the *pulse_o* output of *conv_common_o* to the FPGA output when the TTL switch is ON, or that TTL-BAR pulses are propagated when it is OFF.

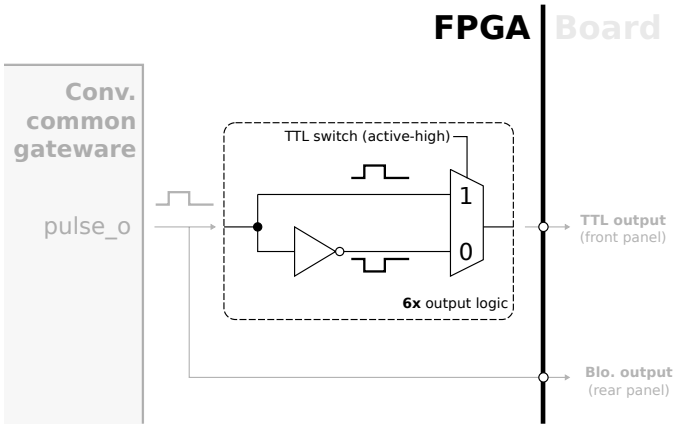


Figure 7: TTL output logic

4.2 Pulse LED output logic

Since in the CONV-TTL-BLO schematics the pulse LEDs are driven from inverting Schmitt triggers to ground, the active-high pulse LED output from *conv_common_gw* must be inverted prior to driving the Schmitt trigger. This is done via the pulse LED logic (Figure 8).

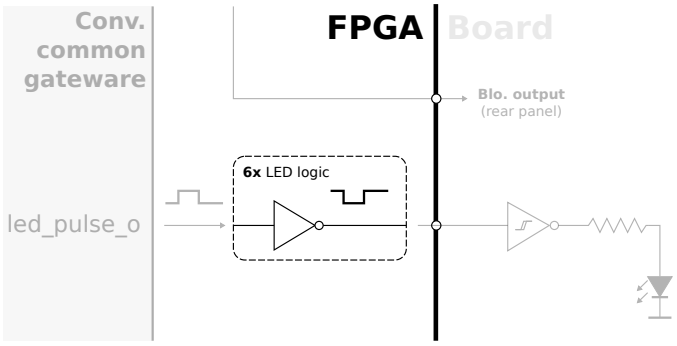


Figure 8: Pulse LED logic

4.3 Bicolor LED output logic

The bicolor LED logic external to the *conv_common_gw* takes the bicolor LED outputs as well as specific control pins (such as, for example, the I^2C LED drive pin, flashing four times on an I^2C transfer) and connects them to the bicolor LEDs, adding multiplexer logic where needed to control the lighting and color of the LED.

The way in which each LED is turned on is described in the CONV-TTL-BLO User Guide [3].

Appendices

A Memory map

Table 1 shows the complete memory map of the gateware. The following sections list the memory map of each peripheral.

In order to convert address values to register index values for SNMP access, the following formula should be used:

$$reg.index = \frac{addr}{4} + 1$$

Table 1: *conv_common_gw* memory map

Peripheral	Address range		Description
Board registers	0x000	0x0ff	Coverter board registers
MultiBoot	0x100	0x11f	MultiBoot module
SDB descriptor	0xf00	0xfff	SDB descriptor (see [6])

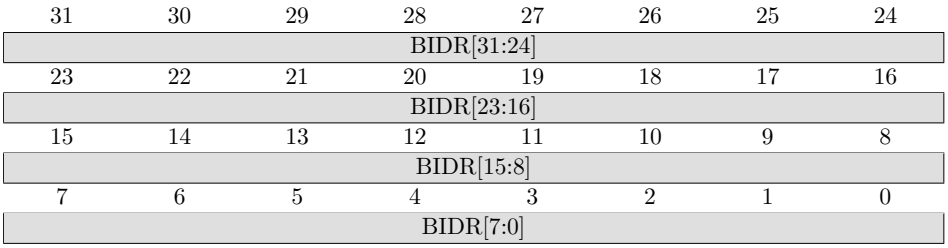
A.1 Converter board registers

Base address: 0x000

Offset	Reset	Name	Description
0x0	0x54424c4f	BIDR	Board ID Register
0x4	(1)	SR	Status Register
0x8	0x00000000	CR	Control Register
0x88	(2)	LSR	Line Status Register

Note (1): The reset value of the SR cannot be specified, since it is based on the gateware version, the state of the on-board switches and whether an RTM is plugged in or not.
Note (2): The reset value of the LSR cannot be specified, since it depends on whether a cable is plugged into the channel or not.

A.1.1 BIDR – Board ID Register



- **BIDR** [*read-only*]: ID register bits
  Reset value: 0x54424c4f

- **Unimplemented bits**: write as '0', read undefined

A.1.2 SR – Status Register

31	30	29	28	27	26	25	24
-	PMISSE[5:0]						I2C_ERR
23	22	21	20	19	18	17	16
WRPRES	I2C_WDTO	RTM[5:0]					
15	14	13	12	11	10	9	8
SWITCHES[7:0]							
7	6	5	4	3	2	1	0
GWVERS[7:0]							

- **GWVERS** [*read-only*]: Gateware version
  Leftmost nibble hex value is major release decimal value
  Rightmost nibble hex value is minor release decimal value
  e.g.
  0x11 – v1.1
  0x2e – v2.14

- **SWITCHES** [*read-only*]: Status of on-board general-purpose switches
  1 – switch is ON
  0 – switch is OFF

- **RTM** [*read-only*]: RTM detection lines [7]
  1 – line active
  0 – line inactive

- **I2C_WDTO** [*read/write*]: I2C communication watchdog timeout error
  1 – timeout occured
  0 – no timeout
  This bit can be cleared by writing a '1' to it

- **WRPRES** [*read-only*]: White Rabbit present
  1 – White Rabbit present
  0 – White Rabbit not present

- **I2C_ERR** [*read/write*]: I2C communication error
  1 – attempted to address non-existing address
  0 – idle
  This bit can be cleared by writing a '1' to it

- **PMISSE** [*read/write*]: Pulse missed error
  1 – pulse arrived during pulse rejection phase
  0 – idle
  Bit 0 – channel 1
  Bit 1 – channel 2
  etc.
  Each bit can be cleared by writing a '1' to it

- **Unimplemented bits**: write as '0', read undefined

A.1.3 CR – Control Register

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	RST	RST_UNLOCK

- **RST_UNLOCK** [*read/write*]: Reset unlock bit
  1 – Reset bit unlocked
  0 – Reset bit locked

- **RST** [*read/write*]: Reset bit
  1 – initiate logic reset
  0 – no reset

A.1.4 LSR – Line Status Register

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	FRONTFS[5:0]					
15	14	13	12	11	10	9	8
REAR[5:0]						FRONTINV[3:2]	
7	6	5	4	3	2	1	0
FRONTINV[1:0]		FRONT[5:0]					

- **FRONT** [*read-only*]: Front panel channel input state
  Line state at board input
  Bit 0 – channel 1
  Bit 1 – channel 2
  etc.

- **FRONTINV** [*read-only*]: Front panel INV-TTL input state
  Line state at board input
  Bit 0 – channel A
  Bit 1 – channel B
  Bit 2 – channel C
  Bit 3 – channel D

- **REAR** [*read-only*]: Rear panel input state
  Line state at board input
  Bit 0 – channel 1
  Bit 1 – channel 2
  etc.

- **FRONTFS** [*read-only*]: TTL-BAR no signal detect state
  High if no cable is plugged in while in TTL-BAR mode
  Unused in TTL mode
  Bit 0 – channel 1
  Bit 1 – channel 2
  etc.

- **Unimplemented bits**: write as '0', read undefined

A.2 MultiBoot controller

Base address: 0x100

Offset	Reset	Name	Description
0x0	0x00000000	CR	Control Register
0x4	0x00000000	SR	Status Register
0x8	0x00000000	GBBAR	Golden Bitstream Base Address Register
0xc	0x00000000	MBBAR	MultiBoot Bitstream Base Address Register
0x10	0x10000000	FAR	Flash Access Register

A.2.1 CR – Control Register

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	IPROG	IPROG_UNLOCK
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	RDCFGREG	CFGREGADR[5:0]					

- **CFGREGADR** [*read/write*]: Configuration register address
  Address of FPGA configuration register to read.
- **RDCFGREG** [*write-only*]: Read FPGA configuration register
  1 – Start FPGA configuration register sequence.
  0 – No effect.
- **IPROG_UNLOCK** [*read/write*]: Unlock bit for the IPROG command
  1 – Unlock IPROG bit.
  0 – No effect.
- **IPROG** [*read/write*]: Start IPROG sequence
  1 – Start IPROG configuration sequence
  0 – No effect
  This bit needs to be unlocked by writing the IPROG_UNLOCK bit first.
  A write to this bit with IPROG_UNLOCK cleared has no effect.
- **Unimplemented bits**: write as '0', read undefined

A.2.2 SR – Status Register

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	WDTO	IMGVALID
15	14	13	12	11	10	9	8
CFGREGIMG[15:8]							
7	6	5	4	3	2	1	0
CFGREGIMG[7:0]							

- **CFGREGIMG** [*read-only*]: Configuration register image
  Image of the FPGA configuration register at address CFGREGADR (see Configuration Registers section in Xilinx UG380 [8]); validated by IMGVALID bit

- **IMGVALID** [*read-only*]: Configuration register image valid
  1 – CFGREGIMG valid
  0 – CFGREGIMG not valid;

- **WDTO** [*read/write*]: MultiBoot FSM stalled at one point and was reset by FSM watchdog
  1 – FSM watchdog fired
  0 – FSM watchdog has not fired

- **Unimplemented bits**: write as '0', read undefined

A.2.3 GBBAR – Golden Bitstream Base Address Register

31	30	29	28	27	26	25	24
BITS[31:24]							
23	22	21	20	19	18	17	16
BITS[23:16]							
15	14	13	12	11	10	9	8
BITS[15:8]							
7	6	5	4	3	2	1	0
BITS[7:0]							

- **BITS** [*read/write*]: Bits of GBBAR register
  31..24 – Read or fast-read OPCODE of the flash chip (obtain it from the flash chip datasheet)
  23..0 – Golden bitstream address in flash

- **Unimplemented bits**: write as '0', read undefined

A.2.4 MBBAR – MultiBoot Bitstream Base Address Register

31	30	29	28	27	26	25	24
BITS[31:24]							
23	22	21	20	19	18	17	16
BITS[23:16]							
15	14	13	12	11	10	9	8
BITS[15:8]							
7	6	5	4	3	2	1	0
BITS[7:0]							

- **BITS** [*read/write*]: Bits of MBBAR register
  31..24 – Read or fast-read OPCODE of the flash chip (obtain it from the flash chip datasheet)
  23..0 – MultiBoot bitstream start address in flash

- **Unimplemented bits**: write as '0', read undefined

A.2.5 FAR – Flash Access Register

31	30	29	28	27	26	25	24
-	-	-	READY	CS	XFER	NBYTES[1:0]	
23	22	21	20	19	18	17	16
DATA[23:16]							
15	14	13	12	11	10	9	8
DATA[15:8]							
7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA** [*read/write*]: Flash data field
  23..16 – DATA[2]; after an SPI transfer, this register contains the value of data byte 2 read from the flash
  15..8 – DATA[1]; after an SPI transfer, this register contains the value of data byte 1 read from the flash
  7..0 – DATA[0]; after an SPI transfer, this register contains the value of data byte 0 read from the flash

- **NBYTES** [*read/write*]: Number of DATA fields to send and receive in one transfer:
  0x0 – Send 1 byte (DATA[0])
  0x1 – Send 2 bytes (DATA[0], DATA[1])
  0x2 – Send 3 bytes (DATA[0], DATA[1], DATA[2])

- **XFER** [*write-only*]: Start transfer to and from flash
  1 – Start transfer
  0 – Idle

- **CS** [*read/write*]: Chip select bit
  1 - Flash chip selected (CS pin low)
  0 - Flash chip not selected (CS pin is high)
- **READY** [*read-only*]: Flash access ready
  1 - Flash access completed
  0 - Flash access in progress
- **Unimplemented bits**: write as '0', read undefined

References

[1] "CONV-TTL-BLO Project Page on OHWR." http://www.ohwr.org/projects/conv-ttl-blo.

[2] "Converter board common gateware project page on OHWR." http://www.ohwr.org/projects/conv-common-gw.

[3] T.-A. Stana, "CONV-TTL-BLO User Guide." http://www.ohwr.org/documents/263.

[4] "CONV-TTL-BLO on CERN EDMS." https://edms.cern.ch/nav/EDA-02446.

[5] "Converter board common gateware specification on OHWR." http://www.ohwr.org/documents/352.

[6] "SDB specification v1.1." http://www.ohwr.org/documents/256.

[7] "Rear Transition Module detection." http://www.ohwr.org/projects/conv-ttl-blo/wiki/RTM_board_detection.

[8] Xilinx, "UG380 - Spartan-6 Configuration Guide." http://www.xilinx.com/support/documentation/user_guides/ug380.pdf, Jan. 2013. v2.5.