BlueMax (10M16SA) Developer's Manual

Daniel Roggen

October 24, 2021

Abstract

This document is a quickstart to developing digital circuits on BlueMax, a miniature FPGA board based on the Intel/Altera 10M16SA (Max 10 family).

The design files are available on `https://github.com/droggen/BlueMax`.

1 Revision history

- 18 October 2021: Initial version.

2 Toolchain setup

2.1 Hardware

Connect the USB Blaster v2 as indicated in figure 1.

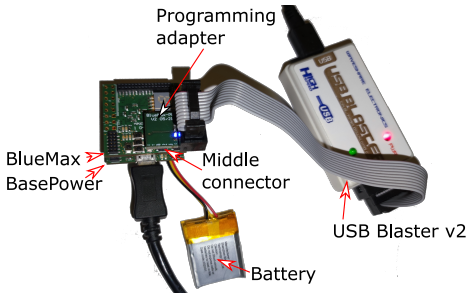

Figure 1: BlueMax is programmed with a "USB Blaster v2" JTAG programmer.

2.1.1 Hardware drivers

Verify that the driver for the hardware is correctly installed (see Fig. 2).

2.2 Software

Install Quartus Prime Lite edition (or the Standard or Pro version if available). At the time of writing the latest release is version 20.1.1.

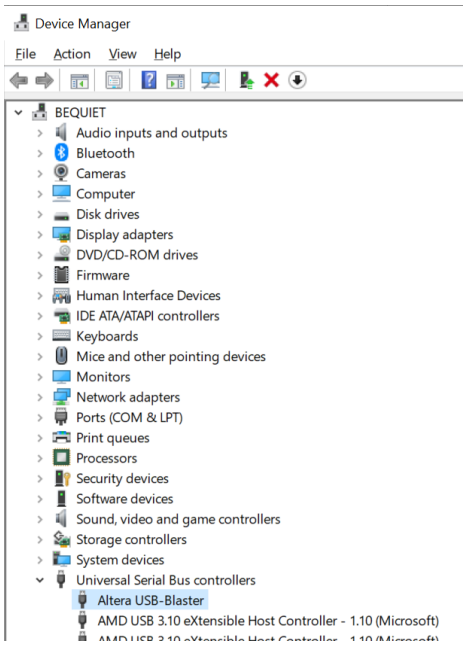

Figure 2: The presence of the driver appears in the Device Manager as "Altera USB-Blaster".

2.2.1 Device detection check

Check that Quartus can communicate with the programmer, and ultimately with the device. After launching Quartus, go to the Tools/Programmer menu, check that the USB-Blaster is listed and if necessary go to Hardware Setup to detect it, and select Auto Detect. The tool should 10M16SA (device with analog functions) or 10M16SC (device with compact functions), and select the 10M16SA, which is the FPGA used by BlueMax. At this stage, the JTAG chain should appear in the programming window, indicating successful communication between the computer and the programmer, and the programmer and the FPGA.

3 First circuit

Create a new project from scratch (not using a template) as follows (see fig 4:

1. File/New Project Wizard
2. Select the location of the project, the project name, and the project top level. For instance, use `bluemax_top` to indicate this is the top-level.
3. Select Empty Project
4. Do not add any file and select Next
5. Select the appropriate device: Family is Max 10; Device is Max 10 SA; Pin count is 169; Core speed is 8; finally select 10M08SAU169C8G in the available devices.

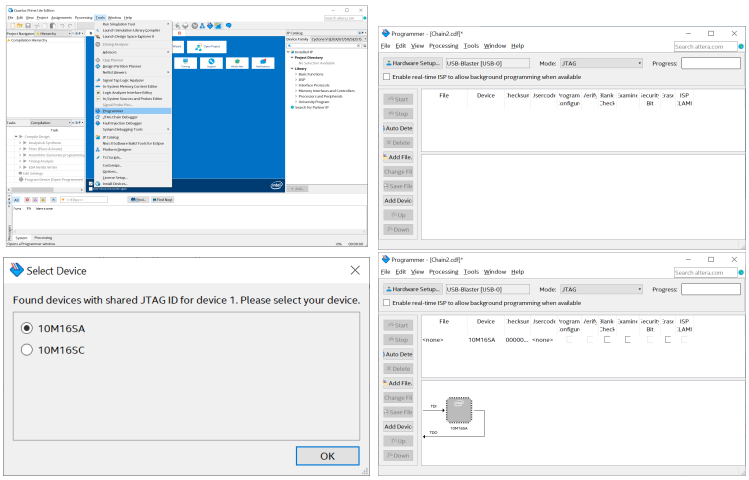

Figure 3: Check the communication with the programmer and the device. Top left: start the programmer tool. Top right: verify the presence of the USB-Blaster and if necessary atdd it via Hardware Setup. Bottom left: the programmer detects the FPGA; select 10M16SA which is the device used by BlueMax. Bottom right: the JTAG chain should appear.

6. Select the EDA tools. Unless other tools have been installed, only select the Simulation tool "ModelSim-Altera".
7. The tool presents a summary of the choices and creates an empty project after confirming.
8. Quartus returns to the main window with an empty project.

3.1 FPGA Configuration Parameters

Go to the *Assignment/Settings* menu, then choose *Device and Pin Options* and verify or adjust the following parameters:

- In *General*:
  - Select *Enable device-wide reset (DEV_CLRn).*
  - Leave other parameters unchanged
- In *Unused Pins*:
  - Select to reserve unused pins as *input tri-stated.* Note that a few unused pins are wired to ground, so weak pull-up would increase power consumption.
  - Leave other parameters unchanged
- In *Voltage*: Select *Default I/O Standard* as *3.0-V LVTTL.*

3.2 Pin mapping

The pin mapping can be performed manually but given the large number of pins it is more efficient to assign them by importing a text configuration file.

Perform the following steps:

- Open the project .qsf file.
- Add at the end of this file the content of bluemax_default.qsf.
- Regenerate the design.

3.2.1 bluemax_default.qsf

The file bluemax_default.qsf contains a default pin mapping. The name of the pins of the side Samtec connectors follows the naming used on the BlueSense expansion connector [1], for consistency reasons if BlueMax is used as an extension of BlueSense.

The file bluemax_default.qsf contains a default assignment where:

- All the on-board peripherals are mapped (LEDs, buttons, 7 segments), with the buttons configured with a pull-up.
- All the expansion connector pins are declared as input with weak pull-up. This must be adjusted according to the actual usage of the pins. For instance, if a pin is driven by a peripheral or used as an output the push-pull must be deactivated.
- The global clock signal is defined and its frequency set to 24MHz.
- All the unused pins which are floating are declared as input with a weak pull-up activated.
- All the unused pins which are unused and wired to ground are declared as inputs.

3.2.2 Pin mapping and usage as an extension to BlueSense or BaseBasePower

It is important that the input or output direction reflects the way BlueMax is wired to actual peripherals. In the example in 6 most of the IO ports are set as input, except the ones which are actually outputs (i.e. the LEDs and 7-segment displays).

The default pin mapping file bluemax_default.qsf uses the naming convention of the expansion connectors of BlueSense, and sets all expansion connectors with a pull-up. It is important that this is adjusted to reflect the way BlueMax is wired to actual peripherals. Notably: adjust the input or output direction in the top-level; and deactivate the pull-up in the qsf file if the pin is used as output or is driven by a push-pull peripheral.

In the example in 6 most of the IO ports are set as input, except the ones which are actually outputs (i.e. the LEDs and 7-segment displays).

3.2.3 Using BlueMax with BasePower

BasePower is a regulated power supply for BlueMax. It also includes a UART to USB transceiver, which allows BlueMax to exchange data over UART interface with a PC.

The UART interface of BasePower is as follows:

- X_1_ADC1: TXD; data coming out of the USB transceiver (input to FPGA)
- X_0_ADC0: RXD; data coming into the USB transceiver (output of the FPGA)

If BlueMax is used as an extension of BasePower the pull-ups on X_0_ADC0 and on X_1_ADC1 must be deactivated, and on X_0_ADC0 must be set as an output in the top-level.

3.3 Create the top-level entity

Create the top-level entity following these steps:

1. Select File/New and VHDL File (or alternatively select another HDL)
2. Create a top-level entity. Figure 6 shows a minimalistic top level file. Note that all the ports are declared, even the unused ones.
3. Save the file, e.g. as bluemax_top.vhd.

Note that all the device ports must be declared in the top-level, even if they are unused. If ports are not declared they revert to the device-wide default I/O setting. This is not suitable as some of the pins must be configured with pull-ups and others without, therefore the device-wide default settings should be avoided.

3.4 Device programming

In order to program the device (Fig. 1):

1. Select Program Device.
2. Select/add the appropriate programming file. The programming file can be a SOF or POF file: SOF configures the SRAM of the FPGA (changes are lost after a power cycle); POF configures the flash of the FPGA (the circuit is stored and reloaded after a power cycle). SOF programming is much faster and does not wear the flash and therefore is more suitable for rapid development cycles.
3. In the Programmer window, select Start to program the FPGA.

4 Clock for timing analysis

In order to perform timing analysis the project sdc file must be modified to specify the 24MHz clock by adding .

```
create_clock -name {CLK_MAIN} -period 41.700 -waveform { 0.000 20.850 } [get_ports {CLK_MAIN}]
```

5 Open points

BlueMax is a project under development in a very preliminary stage. This documentation should be considered an overall guidance, but there may be errors or suboptimal advice.

Some of the open issues are:

- Clarify whether unused pins should be input tri-stated or input tri-stated with bus-hold circuitry. Likely input tri-stated with bus-hold would be more suitable as it both prevents inputs from toggling and allows for pins which are unconnected as well as connected to VCC or GND.

6 Example circuits

The project repository contains a number of example designs.

6.1 Basic

This circuit displays a number 8 on one of the 7-segment displays, and the push-buttons control the LEDs.

6.2 Basic with self-test

This design separates the top-level from the "core" functionality which is in bluemax_core.vhd. This allows the core to be slightly cleaner: the unused FPGA pins which are in the port declaration of the top-level are not mapped to the core. In addition, the design contains a self-test function in bluemax_selftest.vhd.

Upon power-up or reset, the state of the center button is checked. If pressed, the self-test circuit is active. If released, then core circuit is active.

Implementation-wise, both self-test and core execute concurrently, but the display peripherals (7 segment and LEDs) are multiplexed either to the self-test circuit or to the core circuit.

When modifyin the I/O direction of the FPGA ports both the top-level and the core entity must be changed accordingly.

6.3 UART with self-test

This design is similar to that described in Sec. 6.2 with an additional UART. The design assumes BlueMax is plugged onto BasePower, which has a UART-USB transceiver connected to pins X_0_ADC0 (pin A9, FPGA UART TX, i.e. FPGA output) and X_2_ADC1 (pin A7; FPGA UART RX, i.e. FPGA input).

This design uses a third party MIT-licensed UART implementation.

A PC-side terminal must be configured to exchange data at 9600 bps.

The design continuously sends a stream of data. Received data is displayed on the 7-segment displays.

References

[1] Roggen, D.: ARM cortex M4-based extensible multimodal wearable platform for sensor research and context sensing from motion & sound. In:

Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers. (2020) 284–289

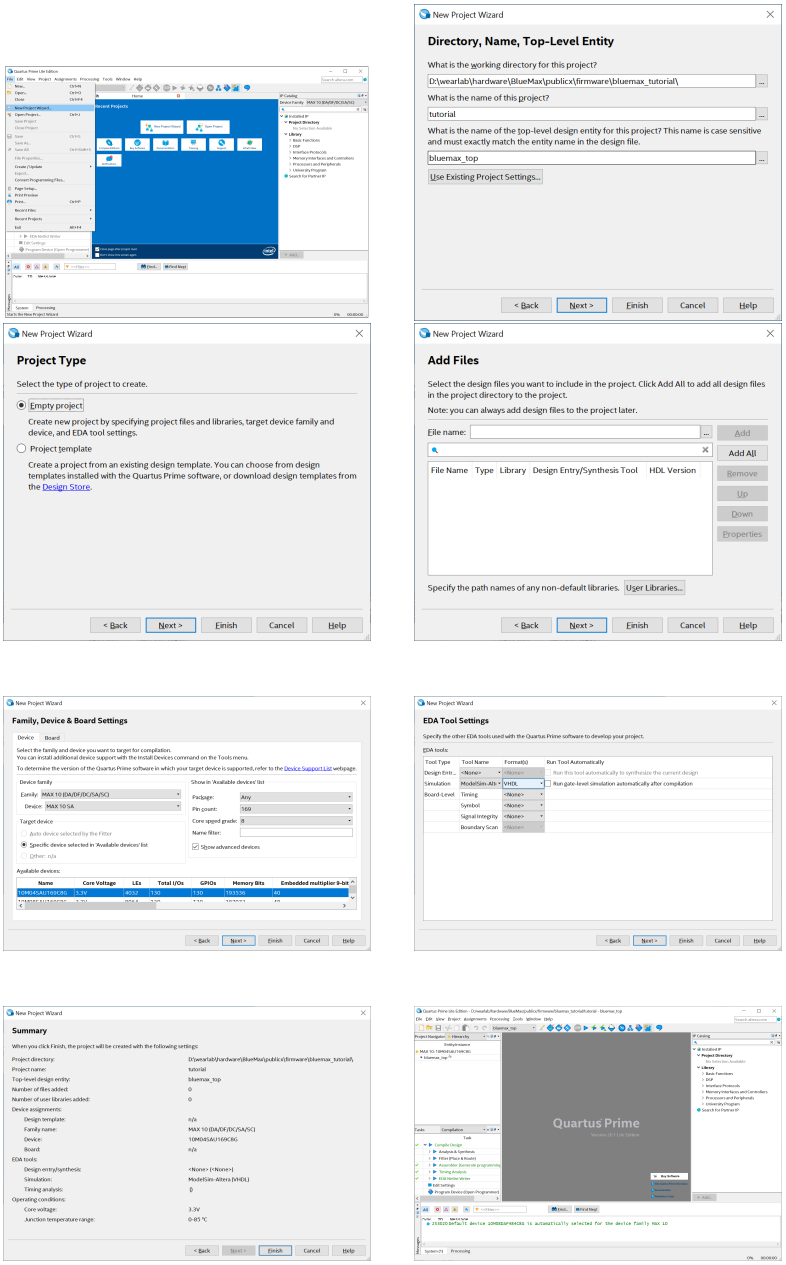

Figure 4: Steps to create a new project from scratch, without using a template.

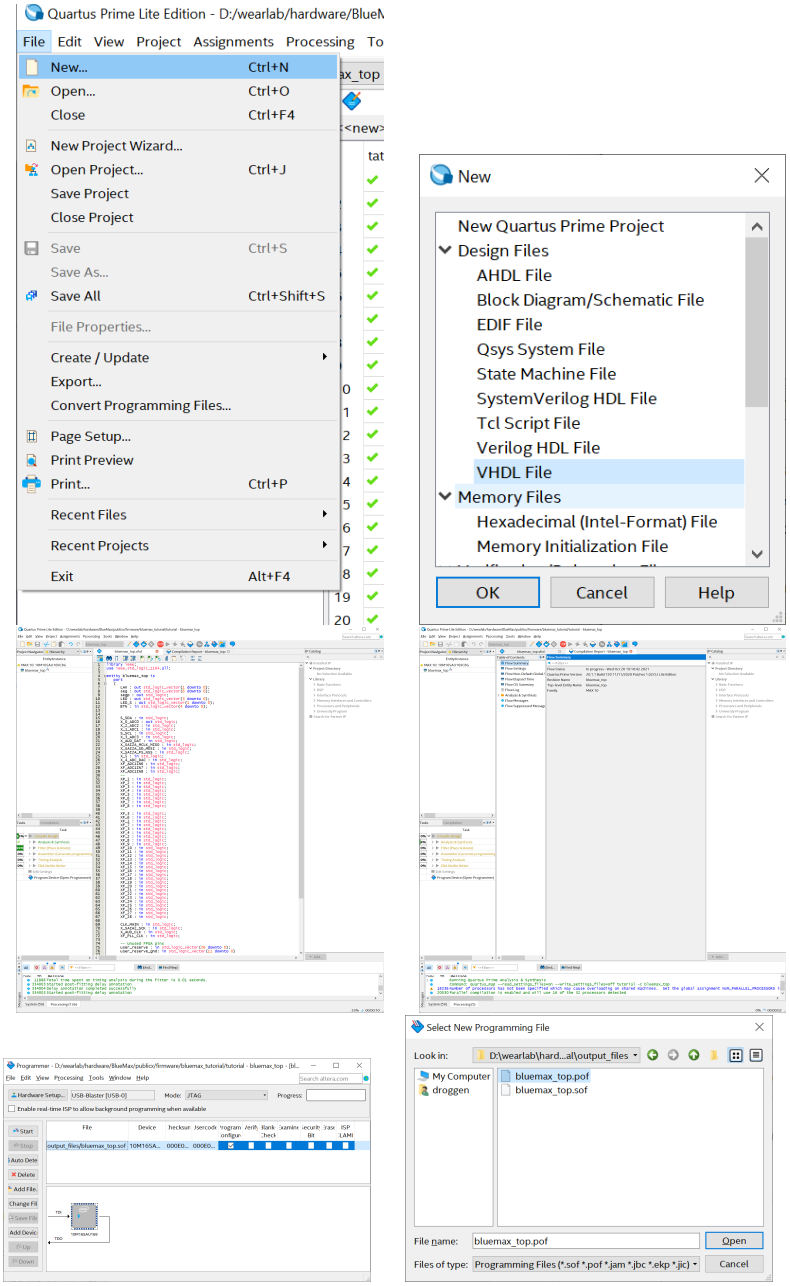

Figure 5: Creation of a top-level entity.

```
library ieee;
use ieee.std_logic_1164.all;

entity bluemax_top is
    port
    (
        -- On-board peripherals
        LAn : out std_logic_vector(1 downto 0)
        ;seg : out std_logic_vector(6 downto 0)
        ;segp : out std_logic
        ;LED : out std_logic_vector(3 downto 0)
        ;LED_S : out std_logic_vector(1 downto 0)
        ;BTN : in std_logic_vector(4 downto 0)
        ;CLK_MAIN : in std_logic

        -- Samtec lateral expansion connectors (for plugging on top of BlueSense or BasePower)
        -- Left connector
        ;X_SAIA2_SCK : in std_logic
        ;X_SAI2A_MCLK_MISO : in std_logic
        ;X_SAI2A_SD_MOSI : in std_logic
        ;X_AUD_DAT : in std_logic
        ;X_AUD_CLK : in std_logic

        -- Right connector
        ;X_0_ADC0 : in std_logic    -- On BasePower: USB TX (output of FPGA).
        ;X_1_ADC1 : in std_logic    -- On BasePower: USB RX (input to FPGA).
        -- Other pins set as input
        ;X_2_ADC2 : in std_logic
        ;X_3_ADC3 : in std_logic
        ;X_4_ADC_DAC : in std_logic
        ;X_5 : in std_logic
        ;X_SAI2A_FS_NSS : in std_logic
        ;S_SDA : in std_logic
        ;S_SCL : in std_logic

        -- XP connector
        ;XP_1 : in std_logic
        ;XP_2 : in std_logic
        ;XP_3 : in std_logic
        ;XP_4 : in std_logic
        ;XP_5 : in std_logic
        ;XP_6 : in std_logic
        ;XP_7 : in std_logic
        ;XP_8 : in std_logic

        -- XF connector
        ;XF_5 : in std_logic
        ;XF_6 : in std_logic
        ;XF_1 : in std_logic
        ;XF_7 : in std_logic
        ;XF_3 : in std_logic
        ;XF_4 : in std_logic
        ;XF_2 : in std_logic
        ;XF_8 : in std_logic
        ;XF_9 : in std_logic
        ;XF_10 : in std_logic
        ;XF_11 : in std_logic
        ;XF_12 : in std_logic
        ;XF_13 : in std_logic
        ;XF_14 : in std_logic
        ;XF_15 : in std_logic
        ;XF_16 : in std_logic
        ;XF_17 : in std_logic
        ;XF_18 : in std_logic
        ;XF_19 : in std_logic
        ;XF_20 : in std_logic
        ;XF_21 : in std_logic
        ;XF_22 : in std_logic
        ;XF_23 : in std_logic
        ;XF_24 : in std_logic
        ;XF_25 : in std_logic
        ;XF_26 : in std_logic
        ;XF_27 : in std_logic
        ;XF_28 : in std_logic
        ;XF_ADC1IN6 : in std_logic
        ;XF_ADC1IN7 : in std_logic
        ;XF_ADC1IN8 : in std_logic
        ;XF_PLL_CLK : in std_logic

        -- Unused FPGA pins
        ;user_reserve : in std_logic_vector(26 downto 0)
        ;user_reserve_gnd: in std_logic_vector(11 downto 0)
    );
end bluemax_top;

architecture dan of bluemax_top is
begin
    LAn <= "10";
    seg <= "0000000";

    segp <= '0';
    LED <= BTN(3 downto 0);
    LED_S <= BTN(4)&BTN(4);
end;
```

Figure 6: Minimalistic top-level file.

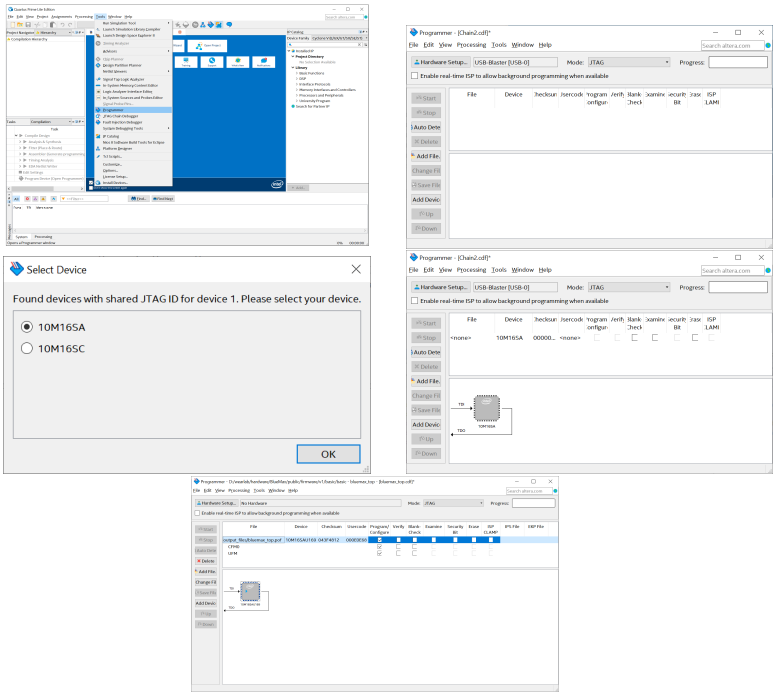

Figure 7: Creation of a top-level entity.