MAXPROLOGIC DEVELOPMENT SYSTEM
User Manual

The MaxProLogic is an FPGA development board that is designed to be user friendly and a great introduction into digital design for Electrical Engineering students and hobbyists. This board provides innovative methods of developing and debugging programmable logic code. It has been designed from the ground up to provide the functionality needed for the demanding projects from today's students and hobbyists. The board provides a convenient, user-friendly workflow by connecting seamlessly with Altera's Quartus Prime Lite software. The user will develop the code in the Quartus environment on a Windows Personal Computer. The programmable logic code is loaded into the FPGA using only the Quartus Programmer tool.

The core of the MaxProLogic is the Altera MAX10 FPGA. This powerful chip has 4,000 Logic Elements and 200 Kbits of Memory. The MAX10 is easily scalable from the entry level college student to the most advanced projects like an audio sound meter with FFT. The MAX10 is in Altera's line up of low cost, multi-function FPGA's.

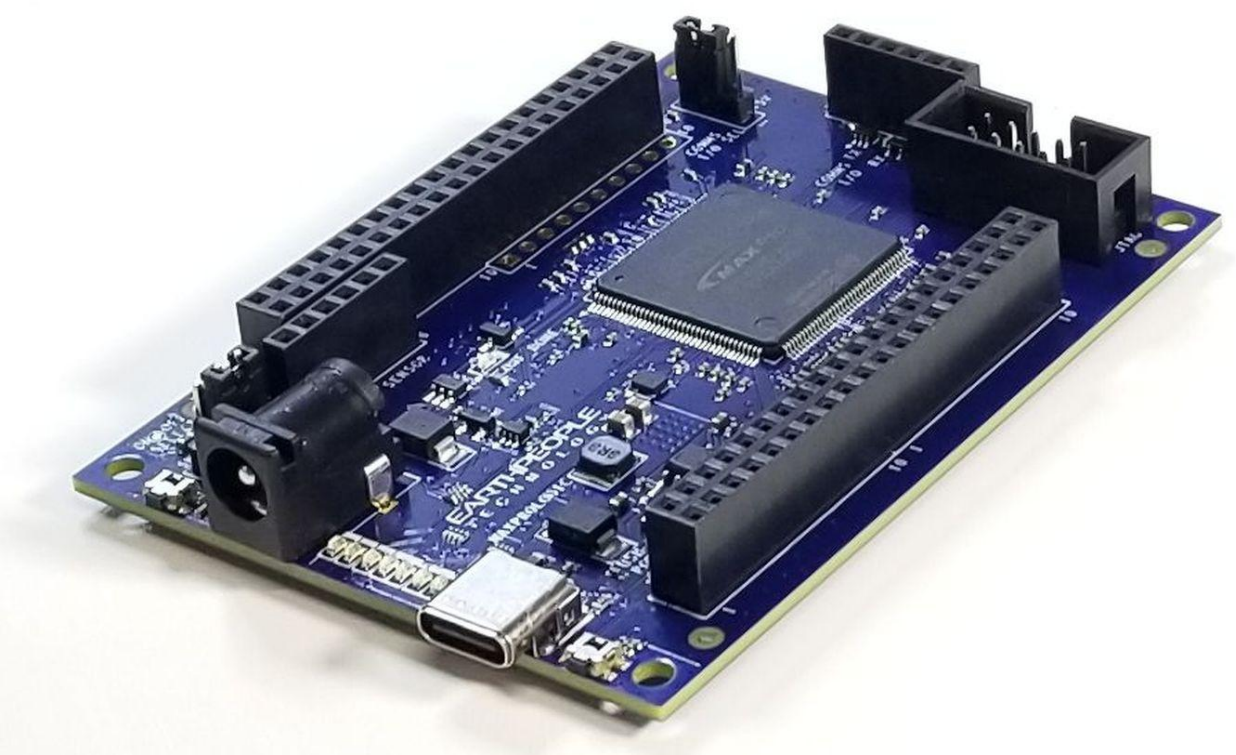
http://www.earthpeopletechnology.com/

EARTHPEOPLE Technology

MaxProLogic Development System User Manual

Contents

MaxProLogic Development System User Manual



- **Please Note: The MaxProLogic Does Not Contain On Board Programmer. JTAG Programming and Configuration Flash Programming must be performed by external Programmer Purchased Separately.**

1 Introduction

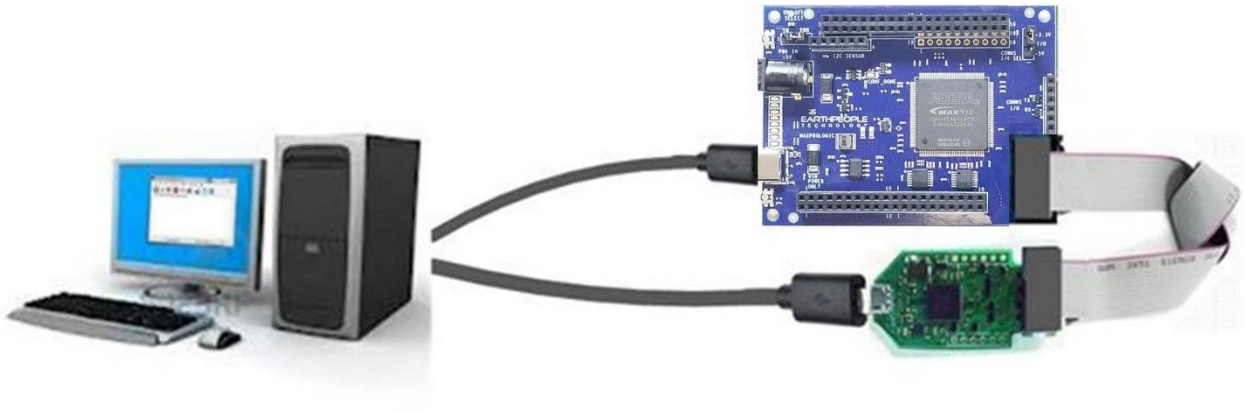
The MaxProLogic is an FPGA development board. It requires an external programmer to load user code into the flash of the FPGA. The core of the board is the MAX10 chip. It has a built in Flash for configuration and incorporates 8 channels of Analog to Digital Conversion. The board has two power options

- Standard USB-C connector
- 5.5mm Barrel connector

The MaxProLogic can be powered from a laptop with 2.5W of power. Or it can be run it from the +5V @ 2A wall USB chargers for 10W of power. The barrel connector can handle up to +5.5V @ 3 A for 16.5W of power. The MaxProLogic has a MicroSD connector on the bottom of the board. The board has an optional On/Off pushbutton switch that allows the user to turn the system on and off. The clocking comes from a 50MHz oscillator.

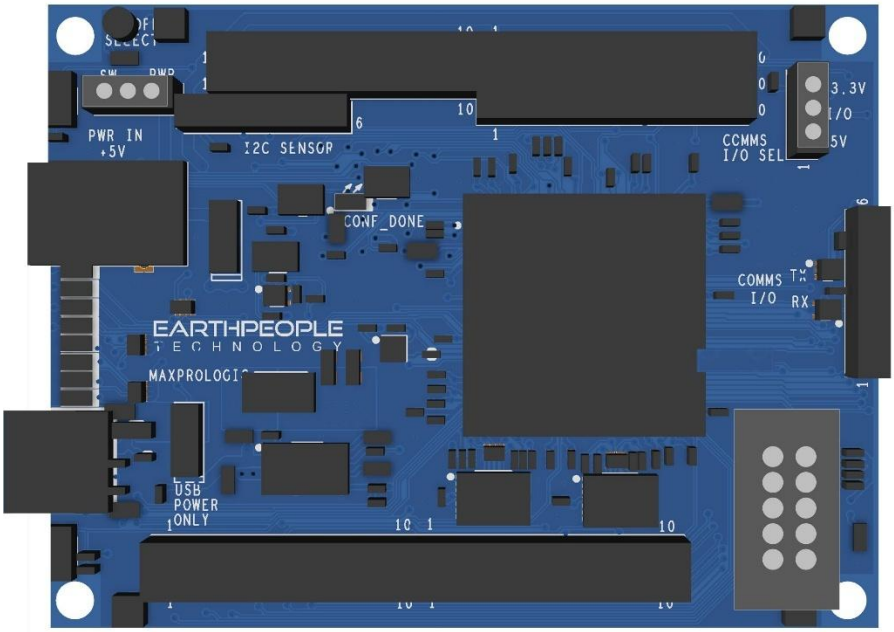
2 User Setup

The MaxProLogic is ready to go straight out of the box. Just connect power from either a PC/Laptop, +5 VDC Universal Wall Charger, +5 to 5.5 VDC through barrel connector. Install the On/Off jumper to select default mode or On/Off switch. See the Power Section for details on powering the board. Next, connect a JTAG Blaster or Altera USB Blaster to J13 for programming the MAX10 chip. Then, write some code, synthesize, and program the chip. See the Programming the MaxProLogic Section for details about programming the MaxProLogic board. EPT provides a project DVD with source code, compiled code, user manual, data sheet and schematics available for download on the EPT website.
https://earthpeopletechnology.com/products-page-2/modules/maxprologic

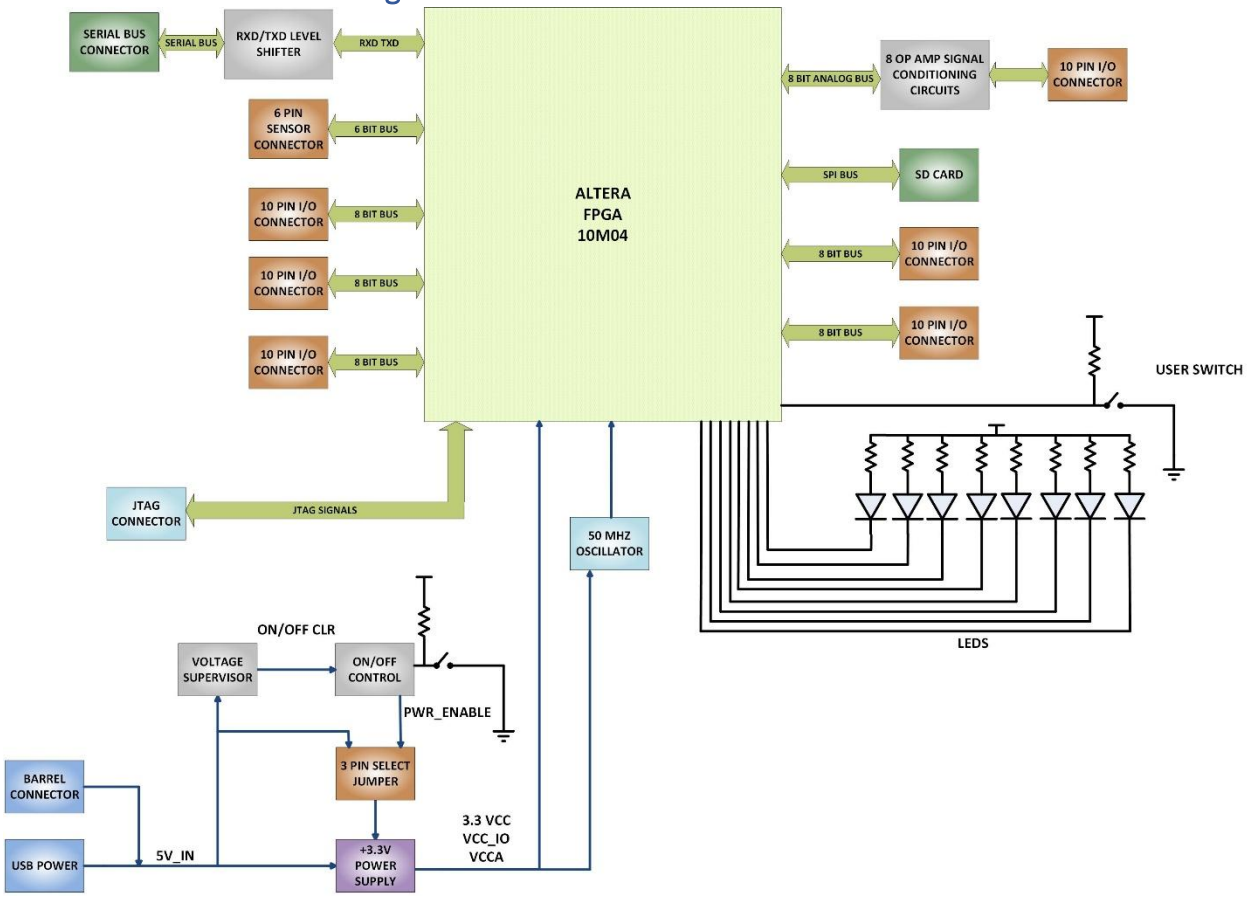


3 MaxProLogic Description

The MaxProLogic is designed to give the user lots of options to communicate with and test the functionality of the MAX10 FPGA. Most of the FPGA Inputs/Outputs are available at board connectors. All components are contained on one board. It operates with power either from the USB connector or the Barrel Connector.


PWR IN
+5V
I2C SENSOR
CONF_DONE
COMMS
I/O SEL
3.3V
I/O
5V
COMMS
I/O
TX
RX
EARTHPEOPLE
TECHNOLOGY
USB
POWER
ONLY

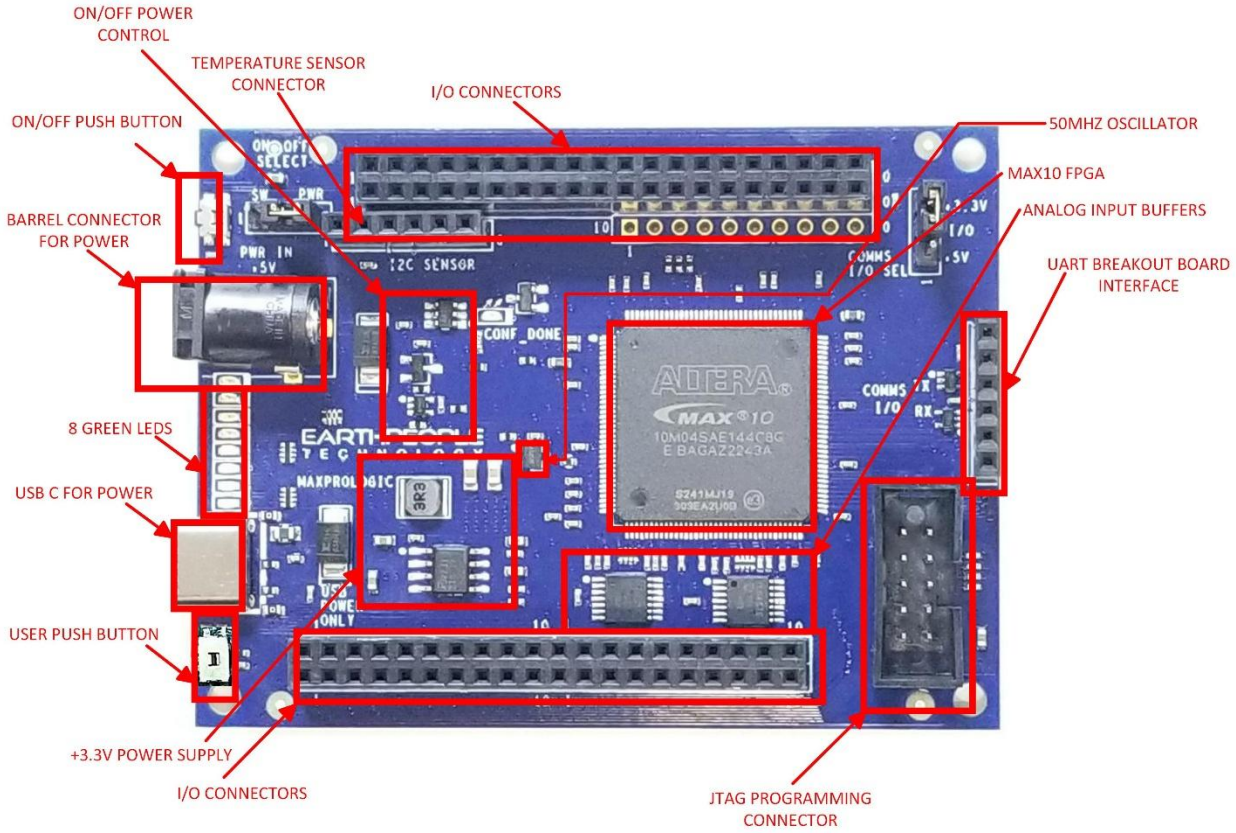
3.1 Functional Block Diagram



3.2 MAXPROLOGIC SPECIFICATIONS

- MAX 10 10M04SA FPGA From Altera 4,000 Logic Elements;
- 2.2 Mbit On chip Flash; 189 Kbit On Chip SRAM 8 Analog Input
- Channels; 12 bit; 1MSamples/Second 65 Available I/O's at connectors
- 65 Inputs/Outputs available at connectors on board
- 8 Green User configurable LEDs 1 Power Pushbutton Switch; 1 User Configurable Pushbutton Switch
- On Board MicroSD Card Slot
- Two Power options: Standard USB (+5V @ 2Amp) Using USB-C connector; 5mm Barrel Connector Accepts +5.5V @ 3Amp
- Switching Power Supply, Provides stable output under high load stress

- On/Off controller uses push button, optional bypass mode to allow board power up without On/Off control
- 50MHz Oscillator
- On board interface to Standard UART Serial Adapters
- Standard Programming Connector fits any Altera USB Blaster



3.3 FPGA

The MaxProLogic includes the Altera 10M04SAE144C8G FPGA. It is an EQFP 144 pin package. This FPGA incorporates both the configuration flash and the ADC on the chip. This is different from conventional FPGAs as these two items are usually off chip. Access to the configuration flash is transparent to the user. The user does have access to the User Flash. This access is provided through the Altera MegaFunctions at the project level. Access to the ADC is also provided through the Altera MegaFunctions at the project level.

Parameter	**10M04SA**
LEs (Logic Elements)	4,000

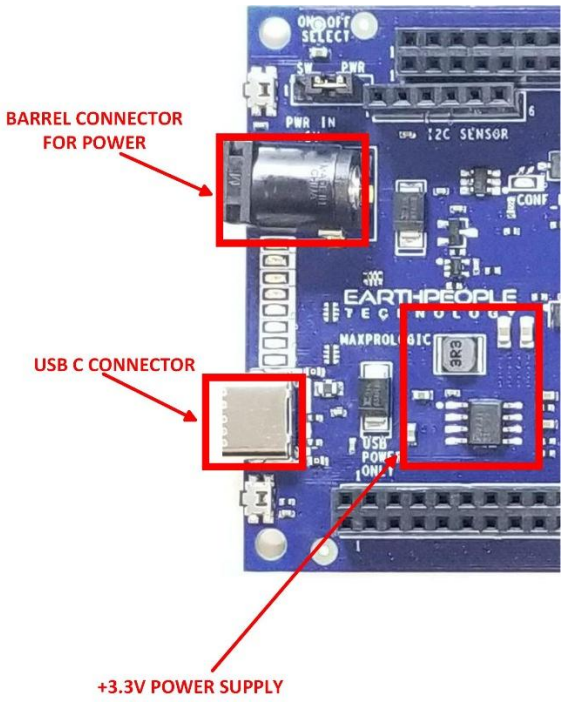
Block memory (Kb)	189
User flash memory1 (KB)	16 – 156
18 x 18 multipliers	20
Phase-locked loops (PLLs)	2
Internal configuration	Dual
Analog-to-digital converter (ADC)	8 Channels, 1MSa/Sec
External memory interface (EMIF)	Yes
Inputs/Outputs	101

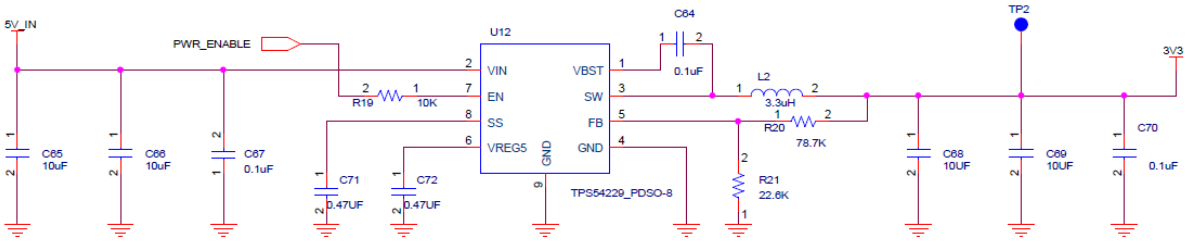
3.4 Power Supply

The MaxProLogic is designed to be operated from one of the following power sources:

- standard USB cable from Laptop/PC.
- +5 VDC through USB-C cable.
- +4.5 to +5.5 VDC supplied through the Barrel Connector.

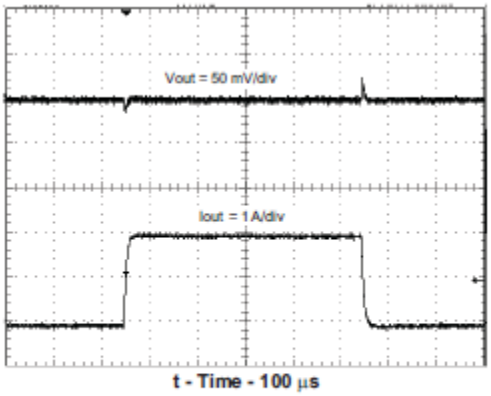
This input provides power for a high-efficiency switching regulator on-board to provide +3.3 VDC. The DC power supply provides reliable power under dynamic loads and high frequency switching internal to the FPGA. The core of the switching supply is the TI TPS54229 chip.





Buck Switching Regulator

The TPS54229 is a very stable synchronous buck converter. This allows it to provide a fast transient response. You can view this response here:



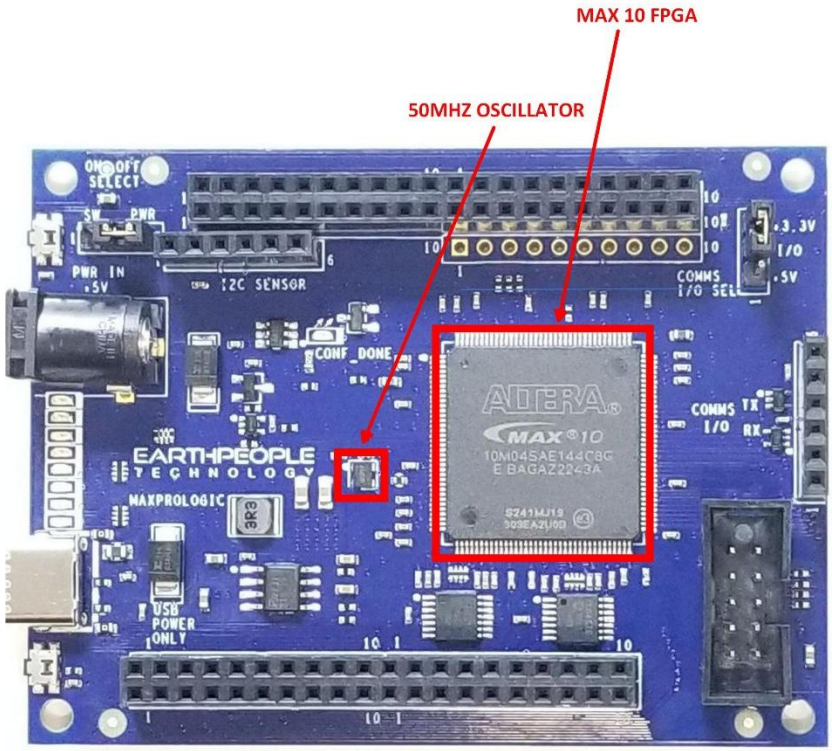
1.05-V, 50-mA to 2-A LOAD TRANSIENT RESPONSE

This switching supply can supply +3.3V up to 2.2 Amps. The current is limited by the 3.3uH inductor. This is a part from Bourns, P/N: SRN4018-3R3M. The saturation current is 2.3 Amps with a 2.2 Amp I_{RMS}. So, the supply has plenty of power for the FPGA and can supply power for off board projects. Please consult the "Powering the MaxProLogic" section for determining available output power per the input power. For example, running the input power from a standard USB 2.0 limits the power to 2.5 Watts.

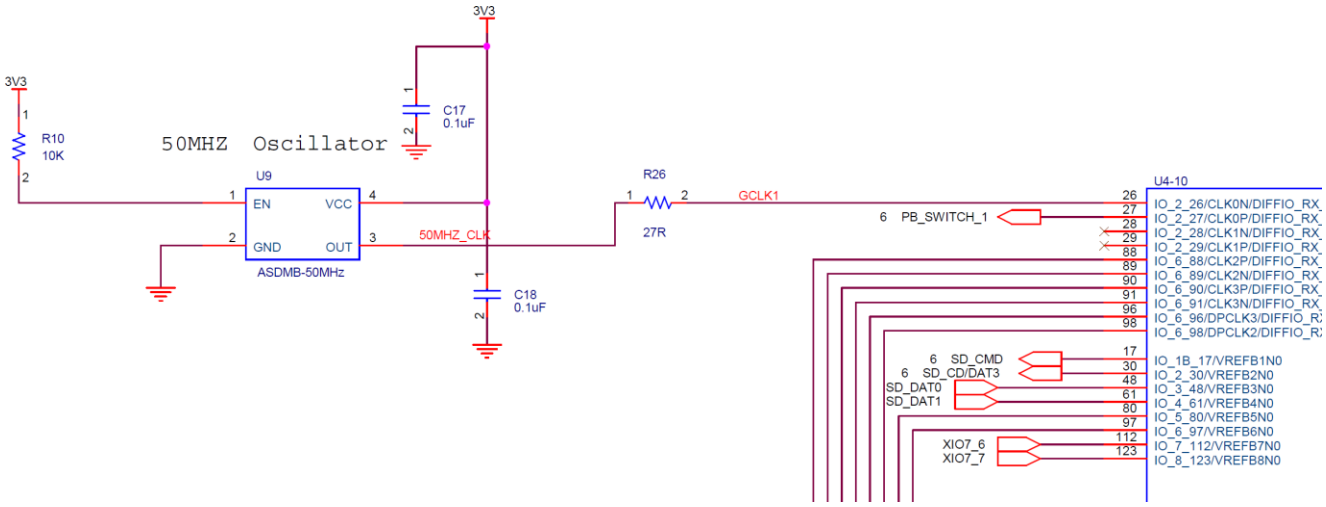
The TPS54229 has an Enable signal that allows the power supply to turn off. This signal is the net PWR_ENABLE. It is controlled by the On/Off controller MAX16054 or default. The operation of the On/Off controller is explained in section x. The default level for the PWR_ENABLE is high. The default level allows the power supply to turn on when power is applied to the board.

3.5 Clock Domain

The MaxProLogic provides an external clock domain to the MAX10 FPGA, 50 MHz. The 50



MHz oscillator is Part Number: ASDMB-50.000MHZ-LC-T and is a +3.3VDC device that provides a high speed clock to the FPGA. It is a CMOS device that provides a stable 50 MHz at ±50 ppm. This clock can be used directly in the user code or use it as an input to one of the PLL's internal to the FPGA. It is intended that this clock will drive the logic of the user code. If a different clock frequency is required in the user code, use the PLL scale up/down to produce the desired clocking.

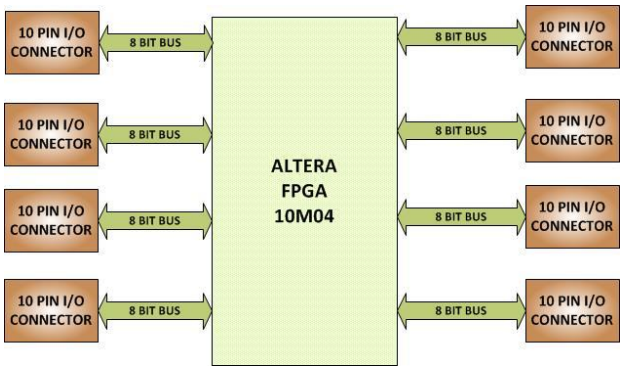


The 50MHz oscillator output is applied to pin 26 of U4 (MAX 10 FPGA) via a series resistor, R26.

Parameter	Min	Max	Units
Overall Frequency Stability:	-50	+50	ppm
Operating Temperature:	0	+70	°C
Output Load:		25	pF
Supply Current	9	16	mA

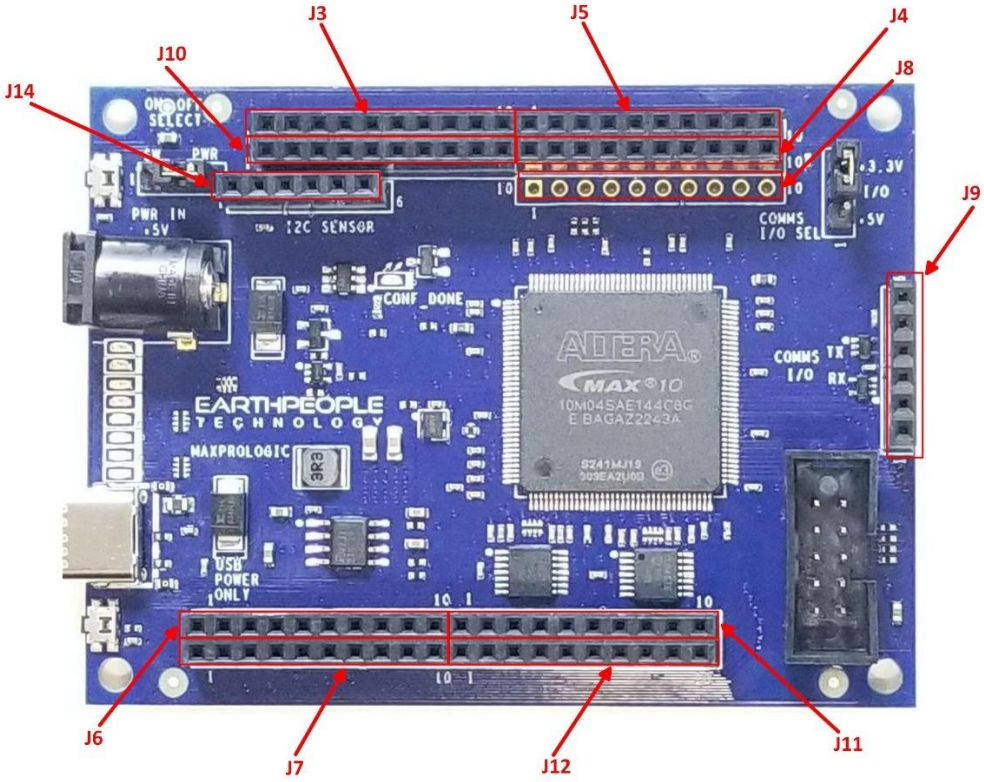
3.6 Digital I/Os

The MaxProLogic has eight 10 pin headers that provide 64 digital Inputs and Outputs. All of the I/O's are +3.3 VDC only. All I/O's connect directly from the FPGA to one of the ten pin headers.



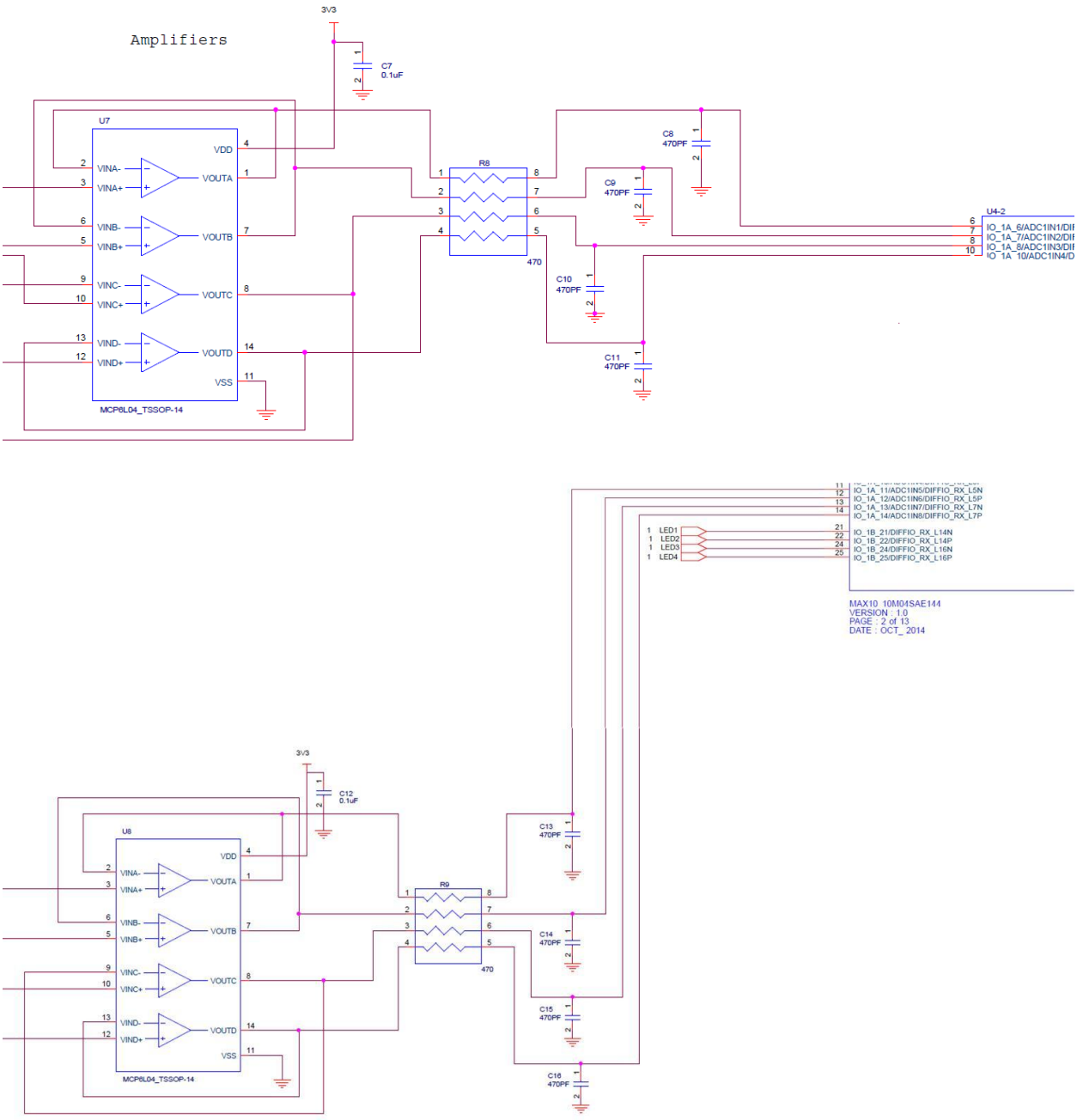
All I/O's are organized into separate banks of the FPGA. There are eight banks. These different banks provide different output speed technologies. Programmable Open Drain The optional open-drain output for each I/O pin is equivalent to an open collector output. If it is configured as an open drain, the logic value of the output is either high-Z or logic low. Use an external resistor to pull the signal to a logic high. Programmable Bus Hold Each I/O pin provides an optional bus-hold feature that is active only after configuration. When the device enters user mode, the bus-hold circuit captures the value that is present on the pin by the end of the configuration. The bus-hold circuitry holds this pin state until the next input signal is present. Because of this, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated. For each I/O pin, you can individually specify that the bus-hold circuitry pulls non-driven pins away from the input threshold voltage—where noise can cause unintended high-frequency switching. To prevent over-driving signals, the bus-hold circuitry drives the voltage level of the I/O pin lower than the VCCIO level. If you enable the bus-hold feature, you cannot use the programmable pull-up option. To configure the I/O pin for differential signals, disable the bus-hold feature. Programmable Pull-Up Resistor Each I/O pin provides an optional programmable pull-up resistor during user mode. The pull-up resistor weakly holds the I/O to the VCCIO level. If you enable the weak pull-up resistor, you cannot use the bus-hold feature. Programmable Current Strength You can use the programmable current strength to mitigate the effects of high signal attenuation that is caused by a long transmission line or a legacy backplane.

To provide maximum flexibility to system designers, all FPGA I/O are intrinsically bidirectional. Based on the I/O direction specified in the configuration file, each I/O can be configured as input-only, output-only, or bidirectional. While the output buffer of a bidirectional I/O has always included a dynamic output enable signal, MAX 10 FPGAs also include an input buffer enable/disable capability. When an input buffer is dynamically disabled, the buffer presents its last known state to the FPGA core fabric, so that no inputs inside the FPGA are left floating.



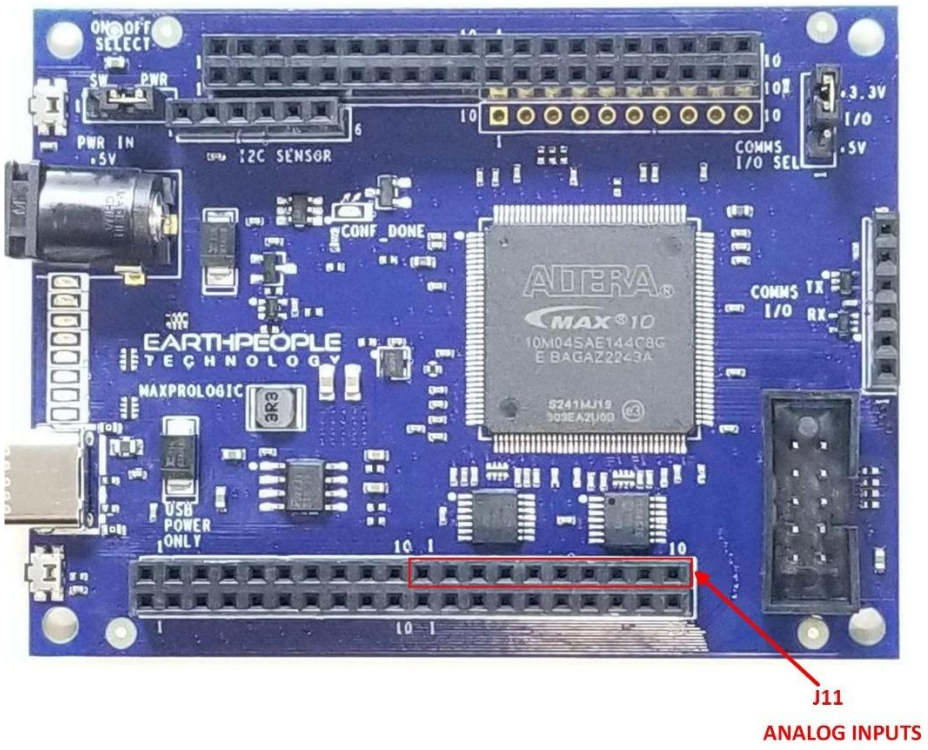
3.7 Analog Input

The MaxProLogic features an analog input of eight buffered channels. Each channel has its own 1MHz Unity Gain Amplifier to provide isolation and filtering. The Unity Gain Amplifier provides the best isolation between channels when using a Sample and Hold ADC and a high speed multiplexor.


Amplifiers
3V3
C7
0.1uF
U7
VDD
VINA-
VINA+
VOUTA
VINB-
VINB+
VOUTB
VINC-
VINC+
VOUTC
VIND-
VIND+
VOUTD
VSS
MCP6L04_TSSOP-14
R8
470
C8
470PF
C9
470PF
C10
470PF
C11
470PF
U4-2
IO_1A_6/ADC1IN1/DI
IO_1A_7/ADC1IN2/DI
IO_1A_8/ADC1IN3/DI
IO_1A_10/ADC1IN4/D
IO_1A_11/ADC1IN5/DIFFIO_RX_L5N
IO_1A_12/ADC1IN6/DIFFIO_RX_L5P
IO_1A_13/ADC1IN7/DIFFIO_RX_L7N
IO_1A_14/ADC1IN8/DIFFIO_RX_L7P
LED1
LED2
LED3
LED4
IO_1B_21/DIFFIO_RX_L14N
IO_1B_22/DIFFIO_RX_L14P
IO_1B_24/DIFFIO_RX_L16N
IO_1B_25/DIFFIO_RX_L16P
MAX10_10M04SAE144
VERSION : 1.0
PAGE : 2 of 13
DATE : OCT_ 2014
3V3
C12
0.1uF
U8
VDD
VINA-
VINA+
VOUTA
VINB-
VINB+
VOUTB
VINC-
VINC+
VOUTC
VIND-
VIND+
VOUTD
VSS
MCP6L04_TSSOP-14
R9
470
C13
470PF
C14
470PF
C15
470PF
C16
470PF

The Analog Inputs connect up to the MAX 10 FPGA using the following pins:

Connector RefDes-Pin Number	MaxProLogic Schematic Signal	FPGA Signal Name	MAX 10 Pin Number
J11-1	+5V		NC
J11-2	ANALOG_IN_1	USER_AIN[0]	6
J11-3	ANALOG_IN_2	USER_AIN[1]	7
J11-4	ANALOG_IN_3	USER_AIN[2]	8
J11-5	ANALOG_IN_4	USER_AIN[3]	10
J11-6	ANALOG_IN_5	USER_AIN[4]	11
J11-7	ANALOG_IN_6	USER_AIN[5]	12
J11-8	ANALOG_IN_7	USER_AIN[6]	13
J11-9	ANALOG_IN_8	USER_AIN[7]	14
J11-10	GND		NC

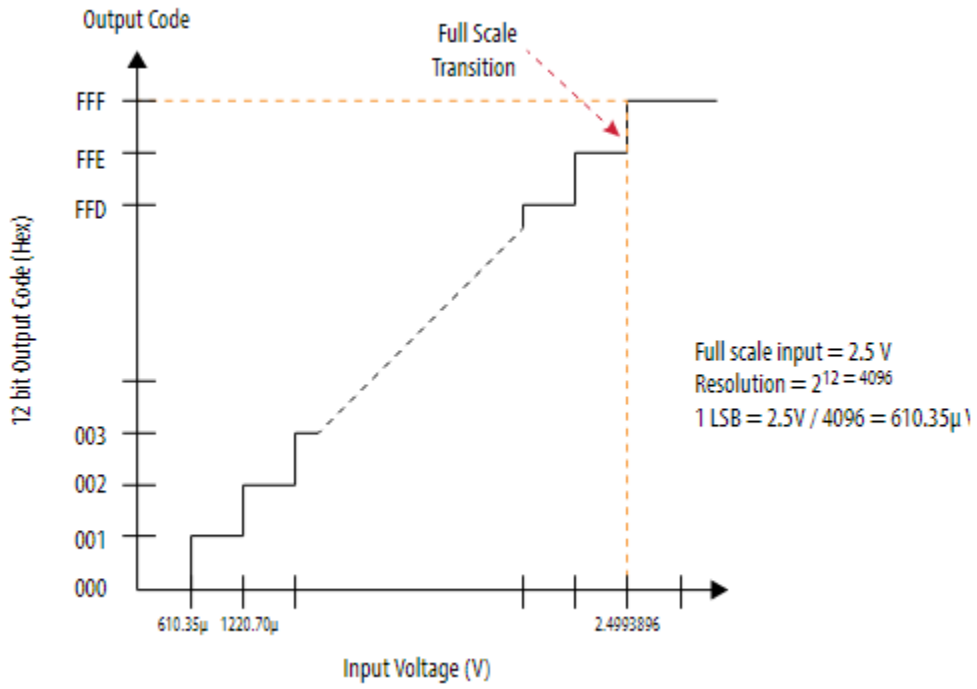


The ADC provide the MAX 10 devices with built-in capability for on-die temperature monitoring and external analog signal conversion. The ADC solution consists of hard IP blocks in the MAX 10 device periphery and soft logic through the Altera Modular ADC IP core. The ADC solution provides you with built-in capability to translate analog quantities to digital data for information processing, computing, data transmission, and control systems. The basic function is to provide a 12 bit digital representation of the analog signal being observed. The ADC monitors up to 8 single-ended external inputs with a cumulative sampling rate of one megasymbols per second (Msps).

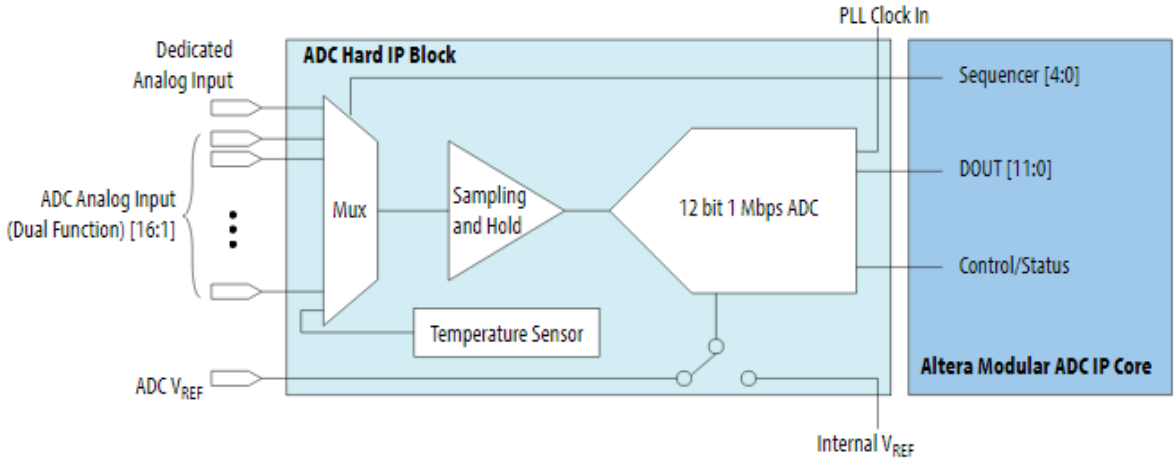
Altera MAX 10 ADC Conversion

The ADC in dual supply Altera® MAX® 10 devices can measure from 0 V to 2.5 V. In single supply Altera® MAX® 10 devices, it can measure up to 3.0 V or 3.3 V, depending on your power supply voltage.

In prescaler mode, the analog input can measure up to 3.0 V in dual supply Altera® MAX® 10 devices and up to 3.6 V in single supply Altera® MAX® 10 devices.
The analog input scale has full scale code from 000h to FFFh. However, the measurement can only display up to full scale – 1 LSB.
For the 12 bits corresponding value calculation, use unipolar straight binary coding scheme.



The Altera® MAX® 10 ADC is a 1 MHz successive approximation register (SAR) ADC. If you set up the PLL and Altera Modular ADC IP core correctly, the ADC operates at up to 1 MHz during normal sampling and 50 kHz during temperature sensing.



3.7.1 Voltage Representation Conversion

Use the following equations to convert the voltage between analog value and digital representation.

Conversion from Analog Value to Digital Code

$$\text{Digital Code}=(V_{IN}V_{REF})\times 212\text{Digital Code}=(VINVREF)\times 212$$

Conversion from Digital Code to Analog Value

$$\text{Analog Value}=\text{Digital Code}\times(V_{REF}212)\text{Analog Value}=\text{Digital Code}\times(VREF212)$$

3.7.2 Calculation Example for V_{REF} of 2.5 V

Analog voltage value to digital code (in decimal), where signal in is 2 V:

$$\text{Digital Code}=(22.5)\times 4096=3277\text{Digital Code}=(22.5)\times 4096=3277$$

Digital code to analog voltage value, approximation to 4 decimal points:

$$\text{Analog Value} = 3277\times(2.54096)=2.0000$$

3.7.3 ADC Analog Input Pins

The analog input pins support single-ended and unipolar measurements.
The ADC block in Altera® MAX® 10 devices contains two types of ADC analog input pins:

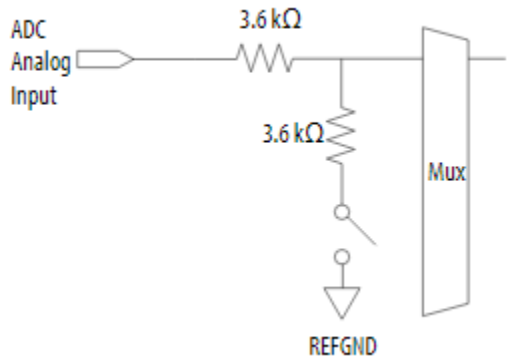
- Dedicated ADC analog input pin—pins with dedicated routing that ensures both dedicated analog input pins in a dual ADC device has the same trace length.
- Dual function ADC analog input pin—pins that share the pad with GPIO pins.

If you use bank 1A for ADC, you cannot use the bank for GPIO.

Each analog input pin in the ADC block is protected by electrostatic discharge (ESD) cell.

3.7.4 ADC Prescaler

The ADC block in Altera® MAX® 10 devices contains a prescaler function.
The prescaler function divides the analog input voltage by half. Using this function, you can measure analog input greater than 2.5 V. In prescaler mode, the analog input can handle up to 3 V input for the dual supply Altera® MAX® 10 devices and 3.6 V for the single supply Altera® MAX® 10 devices.



The prescaler feature is available on these channels in each ADC block:

- Single ADC device—channels 8 and 16 (if available)

3.7.5 ADC Clock Sources

The ADC block uses the device PLL as the clock source. The ADC clock path is a dedicated clock path. You cannot change this clock path.
For devices that support two PLLs, you can select which PLL to connect to the ADC. You can configure the ADC blocks with one of the following schemes:

- Both ADC blocks share the same clock source for synchronization.
- Both ADC blocks use different PLLs for redundancy.

If each ADC block in your design uses its own PLL, the Altera® Quartus® Prime Fitter automatically selects the clock source scheme based on the PLL clock input source:

- If each PLL that clocks its respective ADC block uses different PLL input clock source, the Altera® Quartus®Prime Fitter follows your design (two PLLs).
- If both PLLs that clock their respective ADC block uses the same PLL input clock source, the Altera®Quartus® Prime Fitter merges both PLLs as one.

In dual ADC mode, both ADC instance must share the same ADC clock setting.

Related information

PLL Locations, Altera® MAX® 10 Clocking and PLL User Guide

3.7.6 ADC Voltage Reference

Each ADC block in Altera® MAX® 10 devices can independently use an internal or external voltage reference. There is only one external `VREF` pin in each Altera® MAX® 10 device. Therefore, if you want to assign external voltage reference for both ADC blocks in dual ADC devices, share the same external voltage reference for both ADC blocks.
Altera recommends that you use a clean external voltage reference with a maximum resistance of 100 Ω for the ADC blocks. If the ADC block uses an internal voltage reference, the ADC block is tied to its analog voltage and the conversion result is ratiometric.

3.7.7 Creating Altera MAX 10 ADC Design

To create your ADC design, you must customize and generate the ALTPLL and Altera Modular ADC IP cores.
The ALTPLL IP core provides the clock for the Altera Modular ADC IP core.

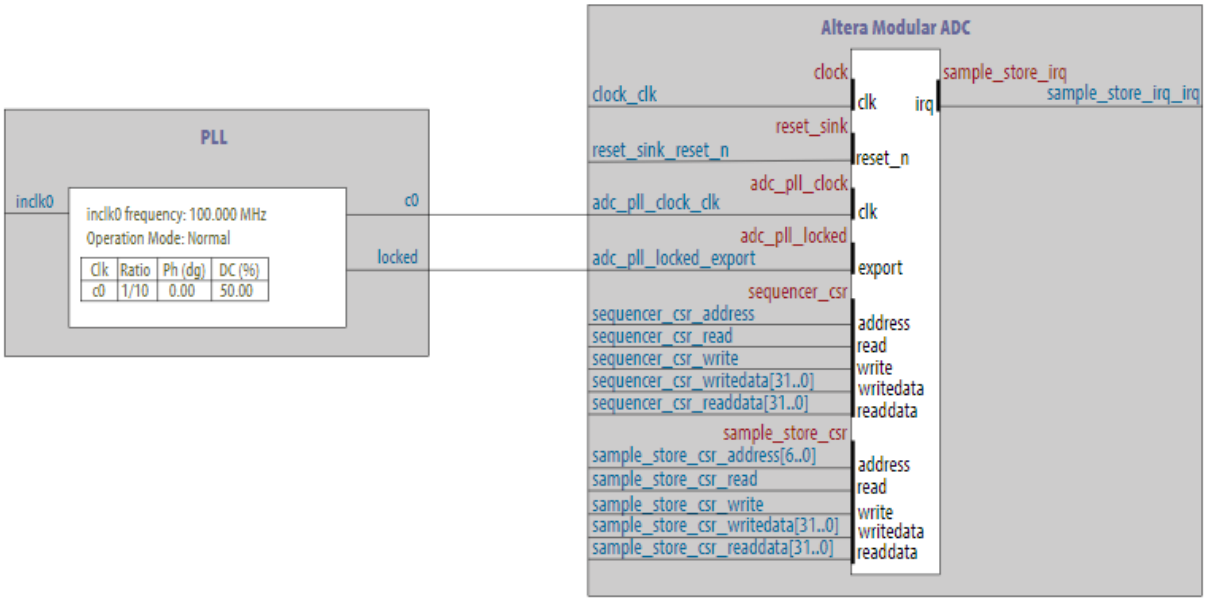
1. Customize and generate the ALTPLL IP core.
2. Customize and generate the Altera Modular ADC IP core.
3. Connect the ALTPLL IP core to the Altera Modular ADC IP core.
4. Create ADC Avalon slave interface to start the ADC.

3.7.8 Parameters Settings for Generating ALTPLL IP Core

Navigate through the ALTPLL IP core parameter editor and specify the settings required for your design. After you have specified all options as listed in the following table, you can generate the HDL files and the optional simulation files.
For more information about all ALTPLL parameters, refer to the related information.

3.7.9 Completing ADC Design

The ADC design requires that the ALTPLL IP core clocks the Altera Modular ADC IP core.
Generate the ALTPLL and Altera Modular ADC IP cores with the settings in the related information.



1. Create the design as shown in the preceding figure.

2. Connect the `c0` signal from the ALTPLL IP core to the `adc_pll_clock_clk` port of the Altera Modular ADCIP core.
3. Connect the `locked` signal from the ALTPLL IP core to the `adc_pll_locked_export` port of the Altera Modular ADC IP core.
4. Create the ADC Avalon slave interface to start the ADC.

Table 12. ALTPLL Parameters Settings.To generate the PLL for the ADC, use the following settings.

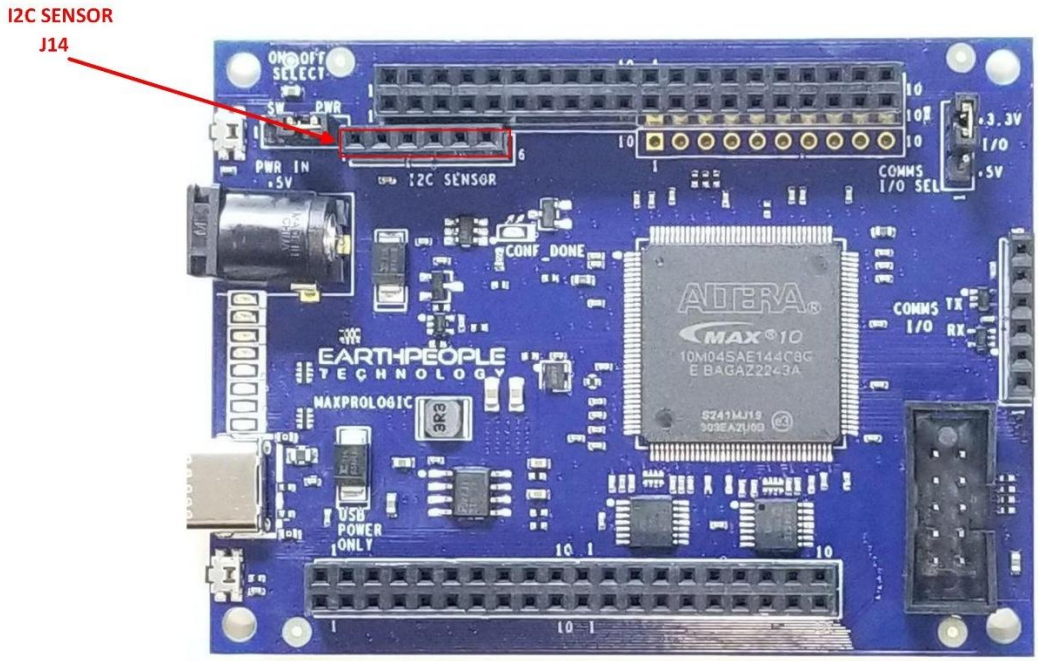
Tab	Parameter	Setting
Parameter Settings > General/Modes	What is the frequency of the inclk0 input?	Specify the input frequency to the PLL.
Parameter Settings > Inputs/Lock	Create an 'areset' input to asynchronously reset the PLL	Turn off this option.
	Create 'locked' output	Turn on this option. You need to connect this signal to the `adc_pll_locked` port of the Altera Modular ADC or Altera Modular Dual ADC IP core.
Output Clocks > clk c0	Use this clock	Turn on this option.
	Enter output clock frequency	Specify an output frequency of 2, 10, 20, 40, or 80 MHz. You can specify any of these frequencies. The ADC block runs at

Table 12. ALTPLL Parameters Settings.To generate the PLL for the ADC, use the following settings.

Tab	Parameter	Setting
		1 MHz internally but it contains a clock divider that can further divide the clock by a factor of 2, 10, 20, 40, and 80. Use this same frequency value in your Altera Modular ADC or Altera Modular Dual ADC IP core. You need to connect this signal to the `adc_pll_clock` port of the Altera Modular ADC or Altera Modular Dual ADC IP core. Different ADC sampling rates support different clock frequencies. For a valid sampling rate and clock frequency combination, refer to the related information.

3.8 I2C Sensor Connector

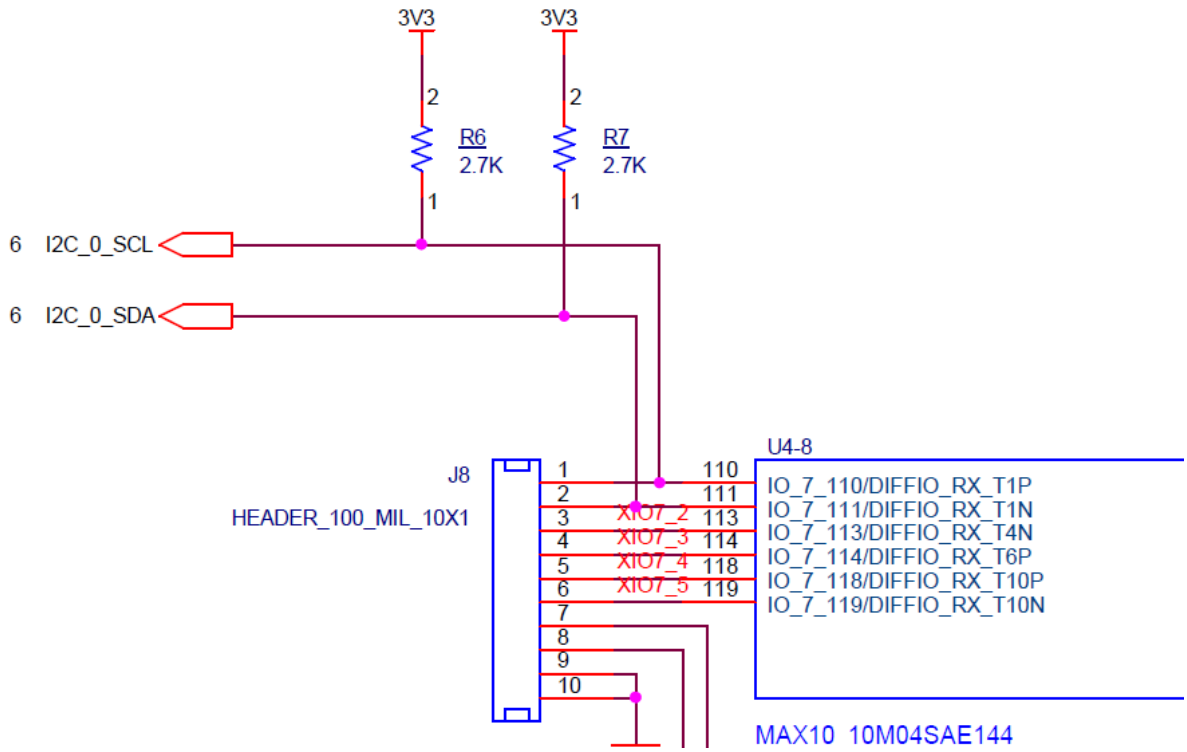
The MaxProLogic includes an I2C Sensor connector. This connector is designed to accept inline six pin sensor boards that are popular with bread board connections.



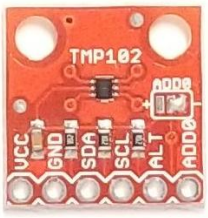
The sensor is accessed through an I2C bus. This bus is connected to the FPGA through the following pins:

Connector RefDes-Pin Number	MaxProLogic Schematic Signal	FPGA Signal Name	MAX 10 Pin Number
J14-1	+3.3V		NC
J14-2	GND		NC
J14-3	I2C_0_SDA	I2C_SDA	111
J14-4	I2C_0_SCL	I2C_SCL	110
J14-5	NC		NC
J14-6	ND		NC

The bus uses 2.7K pullups on both the SDA and SCK signals.



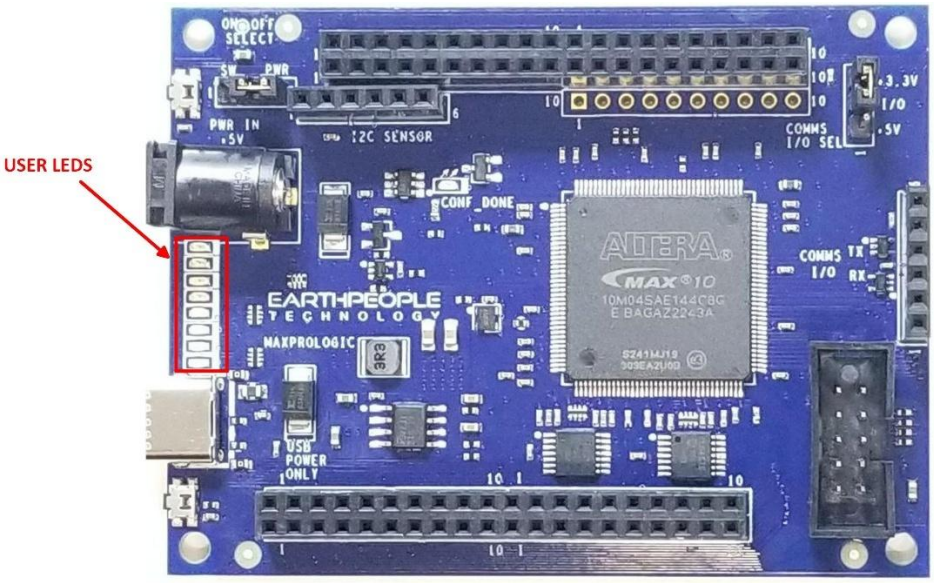
These I2C signals can connect to any I2C device or Master. The user can implement code in the FPGA will perform either Master or Device functionality. The MaxProLogic Project DVD includes sample code to get the user started writing an I2C driver. This sample code is designed to connect to the TMP102 Temperature Sensor Breakout Board.

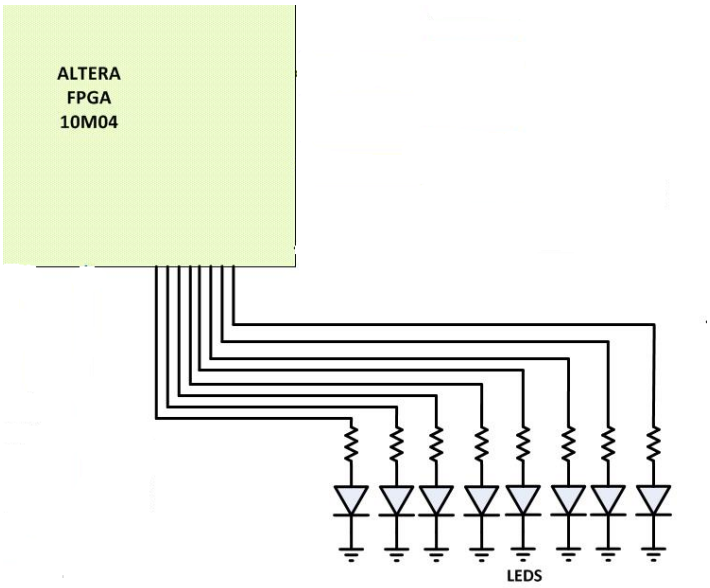


This little board will connect up with J14 and is pin compatible.

3.9 User LEDs

The MaxProLogic includes eight user LEDs. The LEDs are directly driven from the FPGA. User code can implement any pattern on the LEDs by asserting a Ground to light up the diode and a high to turn it off. The LEDs are located at the front of the board between the USB-C and Barrel connectors.





They use the +3.3V I/O's along with a 220 Ohm series resistor for each LED. This provides the following current through the LEDS

$$I_{LED} = \frac{V_O - V_F}{R}$$

$$I_{LED} = \frac{3.3V\ - 2.0V}{220}$$

$$I_{LED} = 5.9mA$$

The code to drive the LEDs is either zero (1'b0) or floating (1'bz). First, declare the LED as an output. In the example below, the vector LED is set to 'reg' because it is driven in an always block.

```
module EPT_10M04_AF_S2_Top (


    input  wire                 CLK_10MHZ,
    input  wire                 CLK_32KHZ,
    input  wire                 RST,

    output wire                 CLK_10MHZ_ENABLE,
    output wire                 CLK_32KHZ_ENABLE,

    output wire   [7:0]         XIO_1,        //
    input  wire   [7:0]         XIO_2,        //
    input  wire   [7:0]         XIO_3,        //
    output wire   [8:0]         XIO_4,        //

    input wire                  PWR_ENABLE,


    output reg     [7:0]          LED
    );
```

To turn the selected LED on, set the signal equal to 1'b0. This will apply a ground to the cathode side of the LED and allow current to flow through the circuit turning the LED on. To turn the selected LED off, set the signal equal to 1'bz. This will float the cathode side of the LED and no current will flow through the LED.

```
//-----------------------------------------------
// Set the LED outputs
//-----------------------------------------------
always @(posedge CLK_10MHZ or negedge RST)
begin
  if(!RST)
      LED <= 8'hz;
  else
  begin
    if(state[LOAD_LEDS])
    begin
      if ( led_reg[0] )
          LED[0] = 1'b0;
      else
          LED[0] = 1'bz;

      if ( led_reg[1] )
          LED[1] = 1'b0;
      else
          LED[1] = 1'bz;

      if ( led_reg[2] )
          LED[2] = 1'b0;
      else
          LED[2] = 1'bz;
```

The User LEDs are available at the following pins:

LED Number	MaxProLogic Schematic Signal	FPGA Signal Name	MAX 10 Pin Number
D2	LED1	USER_LEDS[0]	21
D3	LED2	USER_LEDS[1]	22
D4	LED3	USER_LEDS[2]	24
D5	LED4	USER_LEDS[3]	25
D10	LED5	USER_LEDS[4]	32
D8	LED6	USER_LEDS[5]	33
D9	LED7	USER_LEDS[6]	38

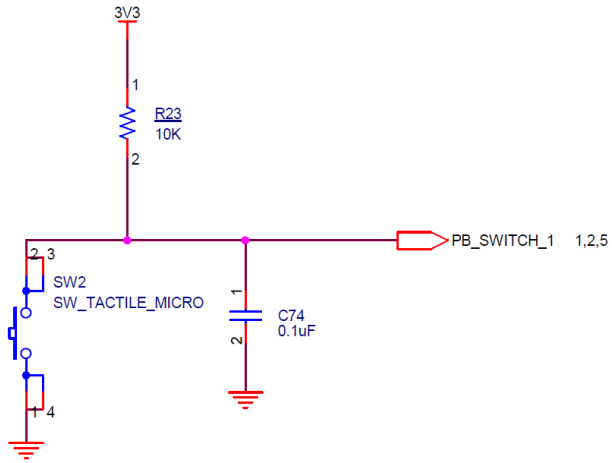
D7	LED8	USER_LEDS[7]	39

3.10 PushButton Switches

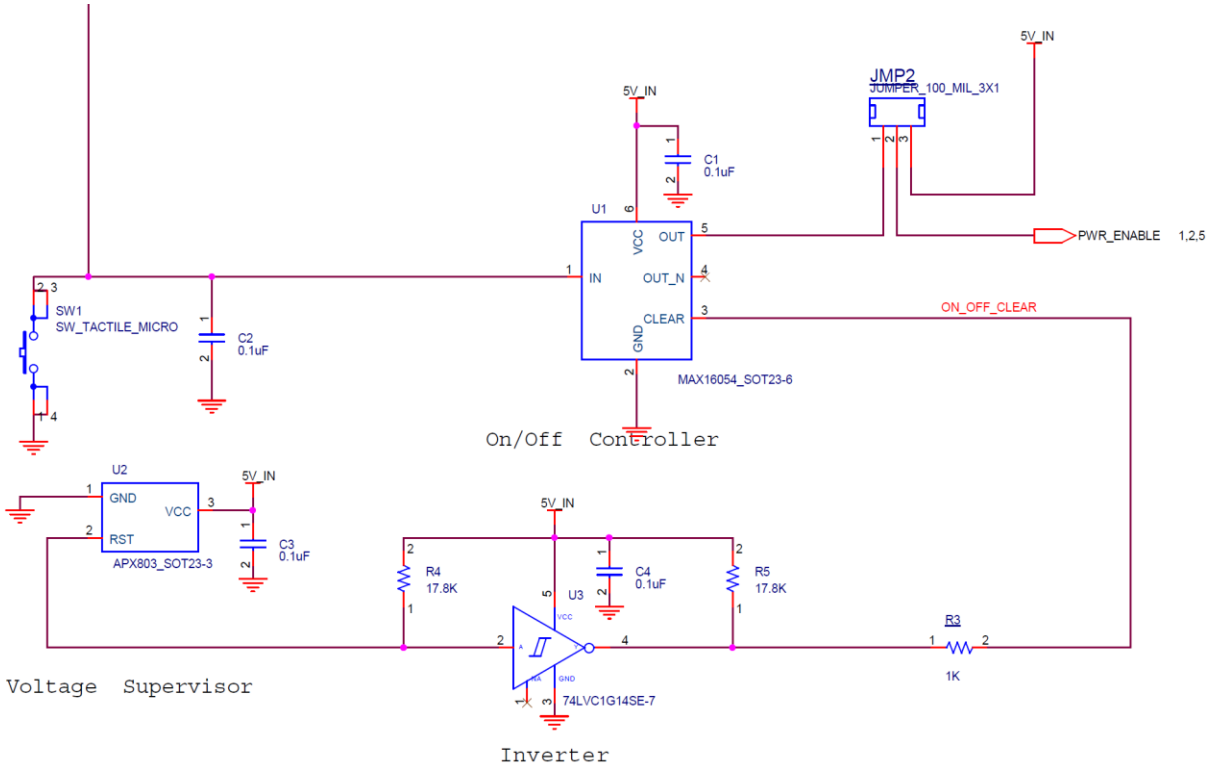
There are two pushbutton switches on the MaxProLogic. Both are momentary contact switches. They include a 0.1uF cap to ground to debounce both switches. The User Push Button Switch is configurable as input only to the MAX10. The user code can read the state of the switch or use it as an event to trigger some action in the code.

Component	Net Name	Pin Number on MAX10	Signal in EPT Project Pinout
SW1	ON_OFF_CONTROL	NA	NA
SW2	PB_SWITCH_1	27	PB_SWITCH_1

SW2 is the User Configurable switch. It is assert low with a constant pullup to +3.3V. So, the switch will read high until the button is pressed in which it will read low.



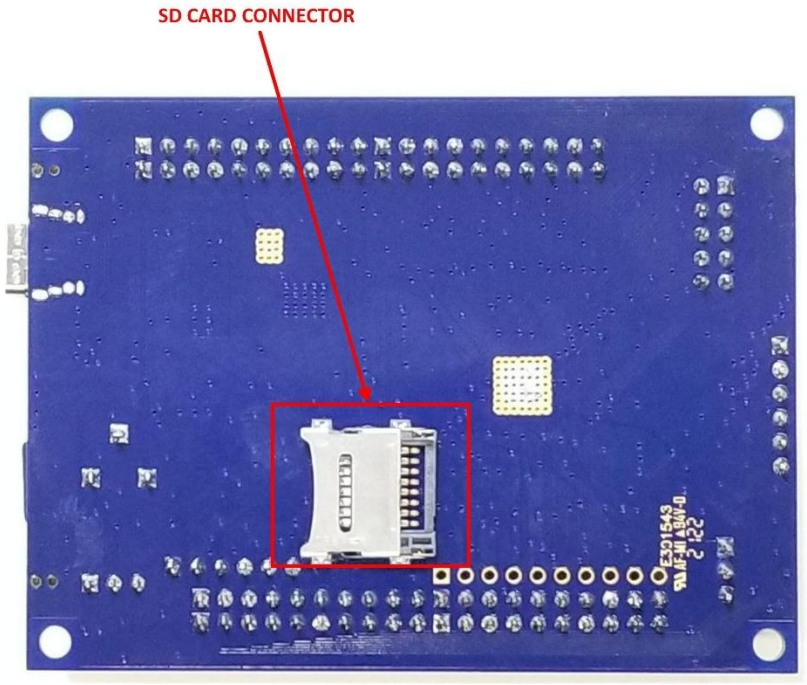
SW1 is configured as the Power Switch for the MaxProLogic. JMP2 selects whether the On/Off Controller U1 is connected to the power system of the board. If the On/Off Controller (MAX16054) is not connected to the circuit, the MaxProLogic will power on when power is applied at the USB-C connector or barrel connector. If the On/Off Controller (MAX16054) is connected to the power circuit, the power button

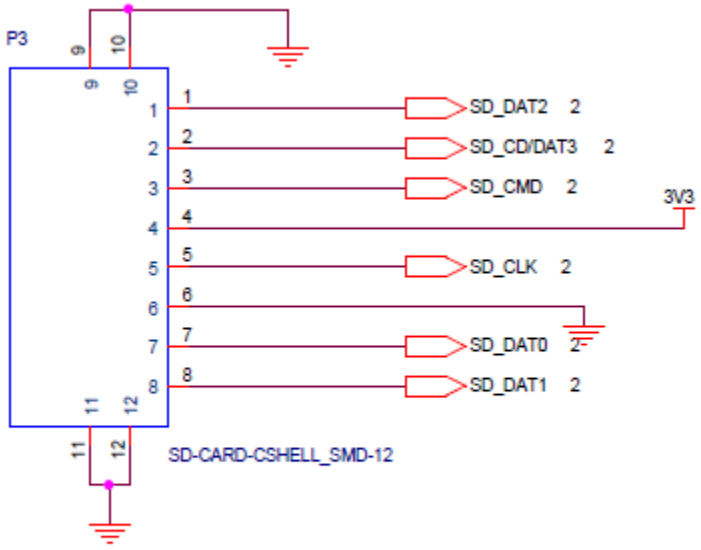


3.11 SD Card Interface

The SD Connector uses a standard cell phone SD Card Connector. This connector has a mechanism that opens out away from the board on a hinge and allows the card to be inserted into

the hinged door. Close the hinged door and pull it towards the front of the MaxProLogic to secure it.





SD Card Connector

The SD Card signals are available at the following pins:

SD Connector Pin Number	MaxProLogic Schematic Signal	FPGA Signal Name	MAX 10 Pin Number
P3-1	SD_DAT2	SD_DATA[2]	60
P3-2	SD_CD/DAT3	SD_DATA[3]	30
P3-3	SD_CMD	SD_CMD	17
P3-4	+3.3V	NC	--
P3-5	SD_CLK	SD_CLK	15
P3-6	GND	NC	--
P3-7	SD_DAT0	SD_DATA[0]	48
P3-8	SD_DAT1	SD_DATA[1]	61

3.11.1 SD Card Protocol

There's a ton of information out there on using the MMC/SD SPI protocol to access SD cards but not much on the native protocol. This page hopes to rectify that with information helpful to those implementing a SD host or trying to understand what they're seeing on an oscilloscope.

3.11.2 References

- The full MMC Specification is available as JESD84-A44 from the JEDEC website
- The SD Simplified Specification from the SD Card Association covers most of the protocol
- The full SD specification is available after joining the SD Card Association

3.11.3 Compatibility

MMC, SD, and SDHC cards are broadly compatible at the electrical and framing level. A properly designed controller should be able to handle them all. Some differences are:

- MMC cards are available in both High Voltage (2.7 - 3.6 V) and Dual Voltage (2.7 - 3.6 and 1.70 - 1.95 V)
- MMC is designed to support multiple cards on the same bus
- During initialisation MMC cards are clocked at 400 kHz or less
- MMC, SD, and SDHC all have different initialisation sequences

3.11.4 Protocol

The protocol is a strict master/slave arrangement where data is clocked synchronously from the host to the card or from the card to the host over digital lines. Commands are sent from the host to the card and all commands have either no response, a 48 bit response, or a 136 bit response. Some commands may also start a data transfer to or from the card.

There are three types of signal:

- CLK, carrying the clock signal from the host
- CMD, carrying commands from the host and responses from the card
- DAT, carrying data from the host or data from the card

There may be 1, 4, or 8 DAT lines. SD Cards can run at 0 - 25 MHz in Default Mode or 0 to 50 MHz in High-Speed Mode. MMC cards come in different grades that can run at up to 20, 26, or 52 MHz. No matter what all cards start up in 3.3 V, single DAT, and low speed mode with any other features negotiated during the initialisation.

3.11.5 Physical layer

All communication are at 3.3 V logic levels with 3.3 V being a high and 0 V being a low. CLK comes from the host and idles low. CMD and DAT are bidirectional and idle high. All are driven in a push/pull mode for speed.

Data is clocked into the host or card on the rising edge of CLK and changes on the falling edge. This is equivalent to the SPI (0, 0) mode.

3.11.6 Framing

The framing is a bit unusual. It feels like it was written by a embedded software engineer instead of a hardware or protocol engineer as the framing and use of CRCs is unusual and inconsistent. The advantage is that the framing maps through to a software only implementation pretty well.

All transfers start with a zero start bit and finish with a one stop bit. A card may signal that it is still working on the response by keeping the CMD line high until the response is ready.

All commands are 48 bits (6 bytes) long and all responses are either 48 bits (6 bytes) or 136 bits (17 bytes) long. The 48 bit transfers can be thought of as an 8 bit message ID, 32 bit argument, and 8 bit checksum.

Bytes are transferred most significant bit first. Words are transferred most significant byte first.

3.11.7 Commands and Responses

A command or response has the following format:

Bit	#	Value	Name
47	1	0	Start bit
46	1	1 for commands, 0 for responses	Transmitter bit
45-40	6		Command ID
39-8	32		Argument
7-1	7		CRC

0	1	1	Stop bit

The CRC is a 7 bit CRC with polynomial $x^7 + x^3 + 1$. A table driven form can be found in the Linux kernel under lib/crc7.c. Bitwise forms may be generated using pycrc with the parameters

```
--width=7 --poly=9 --reflect-in=0 --reflect-out=0 --xor-out=0 --xor-in=0 --generate c
--algorithm=bit-by-bit-fast
```

such as

```
for (int b = 0; b < 8; b++)
{
    uint bit = crc & 0x40;

    if ((data & 0x80UL) != 0)
    {
        bit ^= 0x40;
    }

    data <<= 1;
    crc <<= 1;

    if (bit != 0)
    {
        crc ^= 0x09;
    }
}
```

Note that the final CRC must be ANDed with 0x7F.

The CRC seed is zero and is calculated over the start, transmitter, command ID, and argument fields. The resulting CRC is compared for equality with the CRC from the message.

Some examples are:

An APP_CMD (55) command that prefixes a SD specific command
Bytes: `0x770000000065`
Bits: `0 1 110111 00000000000000000000000000000000 0110010 1`
Fields:

- Start bit = 0
- Transmitter = 1
- Command = 55 (decimal)
- Argument = 00000000
- CRC = 0x32
- Stop bit = 1

The CRC can be generated by feeding 0x77, 0x00, 0x00, 0x00, 0x00 into the CRC function above.

The response to the APP_CMD
Bytes: `0x370000012083`
Bits: `0 0 110111 00000000000000000000000100100000 1000001 1`
Fields:

- Start bit = 0
- Transmitter = 0 (this is a response)
- Response = 55 (decimal)
- Argument = 00000120
- CRC = 0x41
- Stop bit = 1

The response to CMD3 SEND_RELATIVE_ADDR
Bytes: `0x03B368050019`
Bits: `0 0 000011 10110011011010000000010100000000 0001100 1`
Fields:

- Start bit = 0
- Transmitter = 0 (this is a response)
- Response = 3 (decimal)
- Argument = 0xB3680500
- CRC = 0x0C
- Stop bit = 1

Note that this cards Relative Card Address (RCA) is 0xB368

3.11.8 Data

Data has the following format:

Bit	#	Value	Name
4113	1	0	Start bit
4112-17	512*8		Data bits
16-1	16		CRC
0	1	1	Stop bit

Note that this is for a typical transfer of a block of 512 bytes. The host knows from the command that was sent how many bytes to expect back. There is no other way of knowing the message length.

The CRC is is the ITU-T V.41 16 bit CRC with polynomial 0x1021. A table driven version can be found in the Linux kernel under lib/crc-itu-t.c. Note that there is no such thing as 'the' CRC16 so make sure you get the right one.
Unlike the commands or responses the CRC is calculated over all of the data bytes and does not include the start bit. The calculated CRC is checked for equality with the received CRC.

PENDING: Add an example data message with CRC.

3.11.9 Handover

The CMD and DAT lines are bidirectional. Handover occurs at the end of a command where both the host and the card switch to input mode for two clocks before the card starts driving in push/pull mode. Some commands must be responded to in a fixed number of clocks but most allow an arbitrary time before the response must start.

3.11.10 Initialization

To initialise a SD or SDHC card, send the following:

- Write 74 clocks with CMD and DAT high
- Write `CMD0 GO_IDLE_STATE`. This will reset the card.
- Write `CMD8 SEND_IF_COND` for 3.3 V parts. If any SDHC cards are

present then you will get a wired-OR response with 0x3F as the command and 0xFF as the CRC and stop bit. Note that this must be sent or SDHC cards will not respond to the following steps

- Write `CMD55 APP_CMD`
- Receive a 55 response
- Write `ACMD41 SD_SEND_OP_COND`
- Expect a wired-OR response with 0x3F as the command and 0xFF as the CRC and stop bit
- Check the ready bit in the previous response. If the card is not

ready then repeat the CMD55/ACMD41 until it is

- Write `CMD2 ALL_SEND_CID`
- Expect a wired-OR response with 0x3F as the command and 0xFF as the CRC and stop bit
- Write `CMD3 SEND_RELATIVE_ADDR`
- Expect a 3 response. The upper two bytes of the argument is the

Relative Card Address (RCA) which is used in the next step

- Write `CMD7 SELECT_CARD` with the RCA
- Expect a 7 response

The card is now selected and ready to transfer data. See Figure 4-1 'SD Memory Card State Diagram' in the simplified spec for more information.

See section 4.7.4 'Detailed Command Description' in the simplified spec for more information on the commands and responses.

MMC cards are initialised using a similar but different method.

An example flow captured from Linux on a SC2440 is:

Phase	Command	Response	Notes

CMD0	4000000000 95	None	
CMD55	7700000000 65	370000012083	
ACMD41 SEND_OP_COND	6900100000 5F	3F00FF8000FF	Card is busy
CMD55	7700000000 65	370000012083	
ACMD41	6900100000 5F	3F00FF8000FF	Card is still busy
CMD55	7700000000 65	370000012083	
ACMD41 SEND_OP_COND	6900100000 5F	3F80FF8000FF	Card is ready
CMD2 ALL_SEND_CID	4200000000 4D	3F1D4144534420202010A0400BC1008 8ADFF	
CMD3 SEND_RELATIVE_A DDR	4300010000 7F	03B368050019	RCA of 0xB36 8

Note the missing CMD8 as this controller does not support SDHC. I didn't capture the final CMD7.

3.11.11 Reading

Once initialised reading from a card is straight forward.

To read a single page:

- Send `CMD17 READ_SINGLE_BLOCK` with the offset to read from as the argument
- Receive on the DAT lines
- Send `CMD12 STOP_TRANSMISSION` once the block has been received

To read consecutive pages:

- Send `CMD18 READ_MULTIPLE_BLOCK` with the starting offset as the argument
- Receive as many blocks as you want on the DAT lines
- Send `CMD12 STOP_TRANSMISSION` once the done

Note that there will be a response to these commands and the response may be interleaved with the data. See Figure 3-3: (Multiple) Block Read Operation for more information.

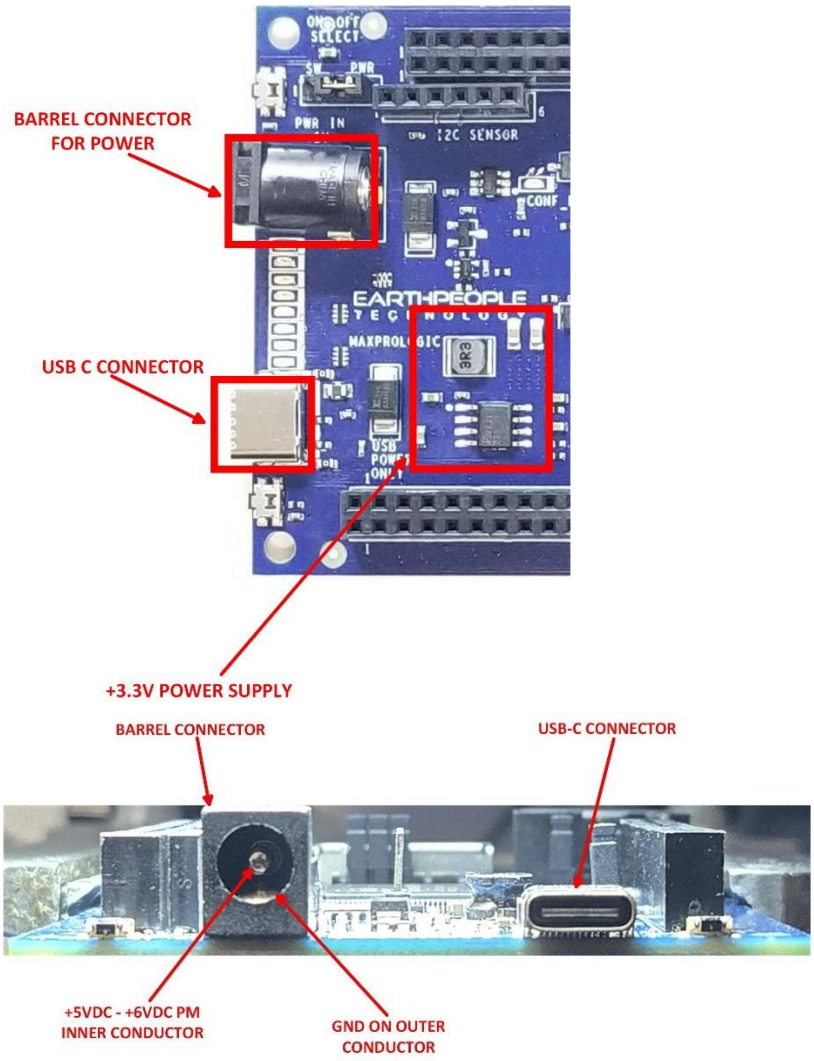
On SDHC cards the offset is in terms of 512 byte blocks. On SD cards the offset is in bytes and the number of bytes received depends on`CMD16 SET_BLOCKLEN`

3.12 Power Input

The MaxProLogic is designed to be operated from one of four different power sources:

- Standard USB cable from Laptop/PC.
- +5 VDC wall charger (phone charger) through USB cable.
- +4.5 to +5.5 VDC supplied through the DC power jack.

This provides power for a high-efficiency switching regulator on-board to provide +3.3 VDC. The power supply provides reliable power under dynamic loads and high frequency switching internal to the FPGA.

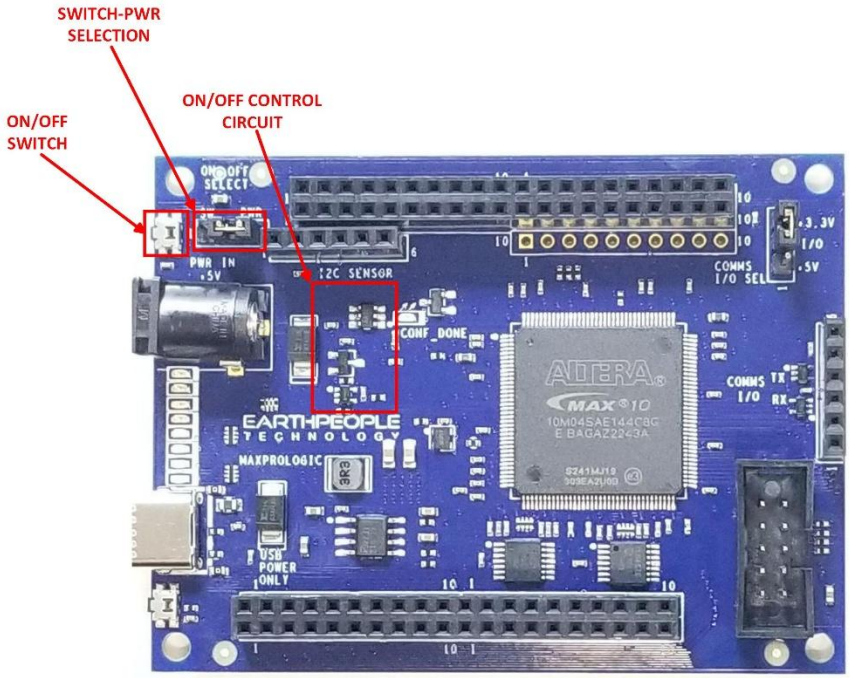


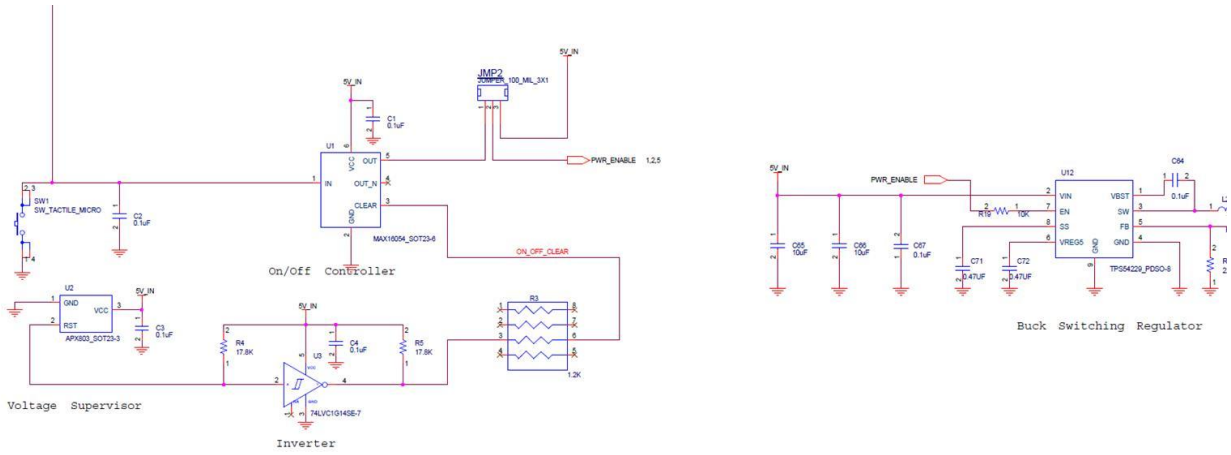
3.13 On/Off Control

The MaxProLogic is equipped with an On/Off circuit. This circuit consists of a push button switch that is momentary contact and a MAX16054 controller chip. The controller senses the change in state of the momentary switch and drives the output “PWR_ENABLE” signal to the

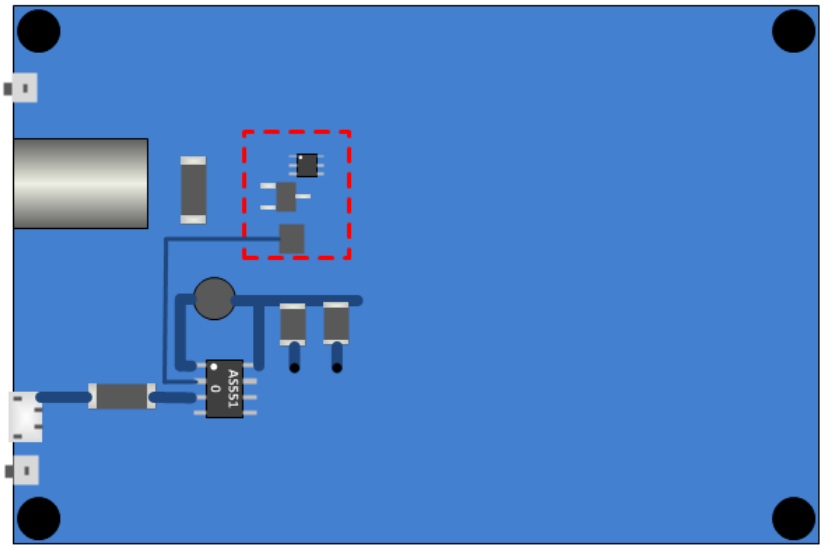
+3.3 VDC power supply. The enable pin of the TPS54229 has the following truth table:

Enable Pin	Output of TPS54229
High	+3.3 V
Low	0V

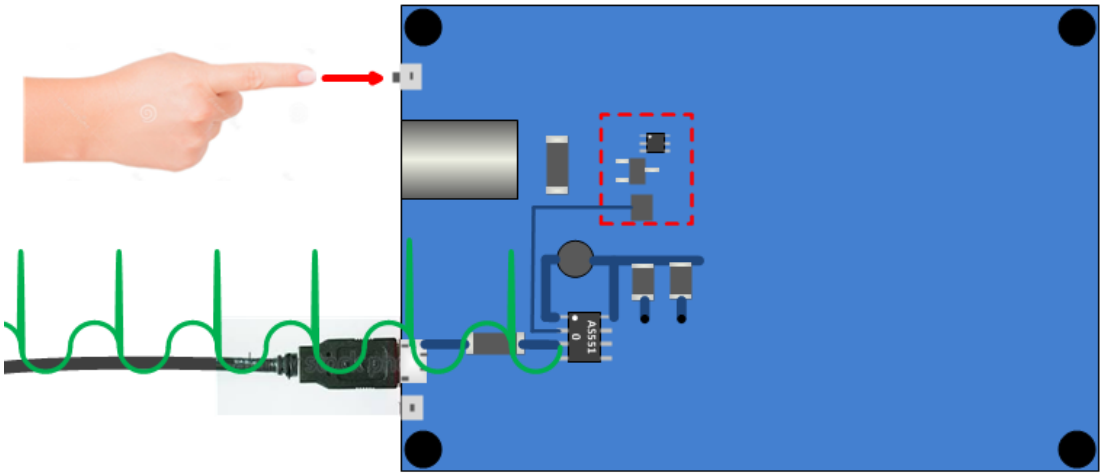


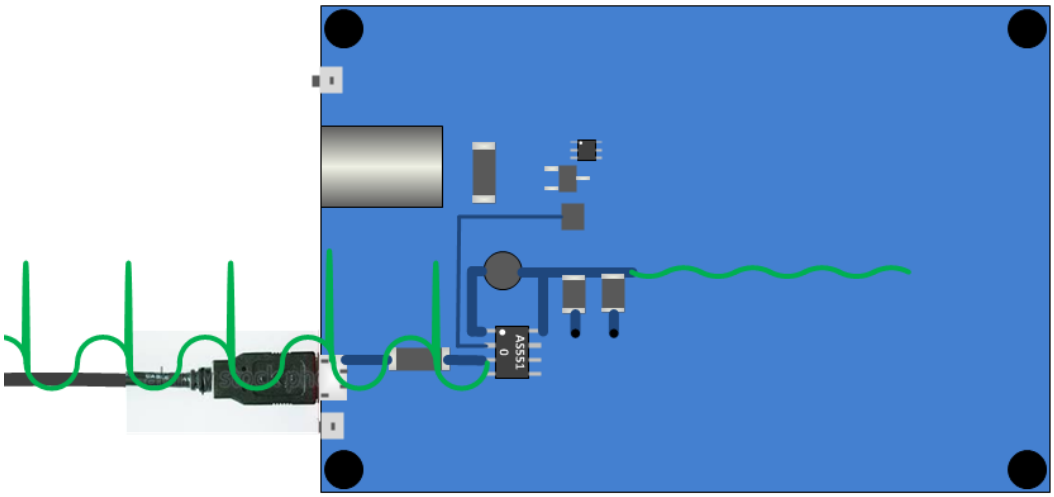


Use the Pushbutton SW1 to turn the power on to the MaxProLogic.



The TPS54229 Synchronous Buck Regulator will take any noisy input power and provide a smooth stable output. It has a fast transient response with a large inductor on the output.



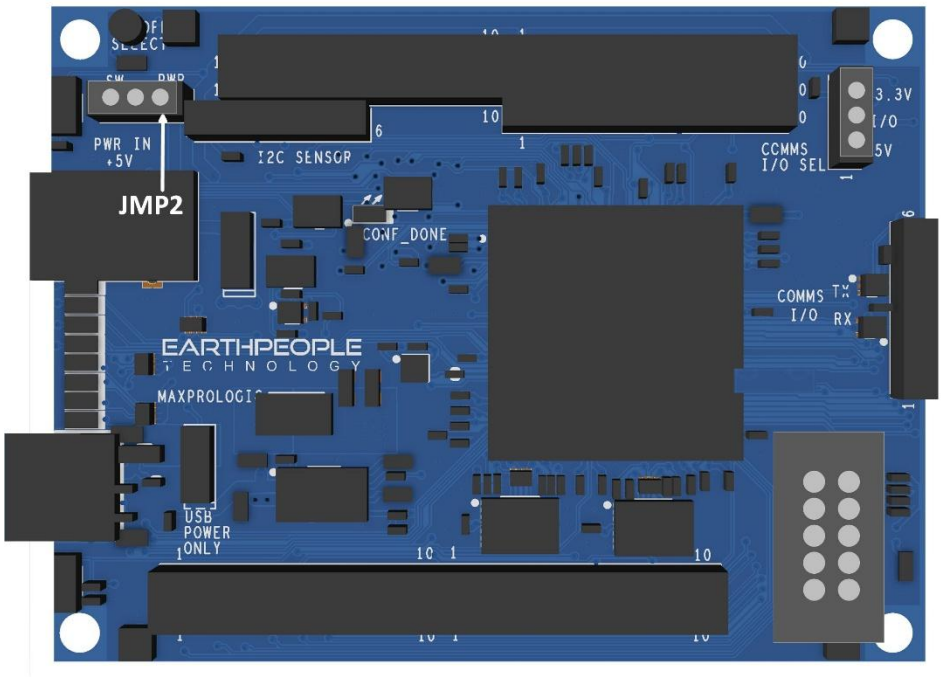


The MAX16054 Controller chip also has a ‘CLEAR’ input. This input allows secondary device to cause a shutdown. The CLEAR input has the following truth table:

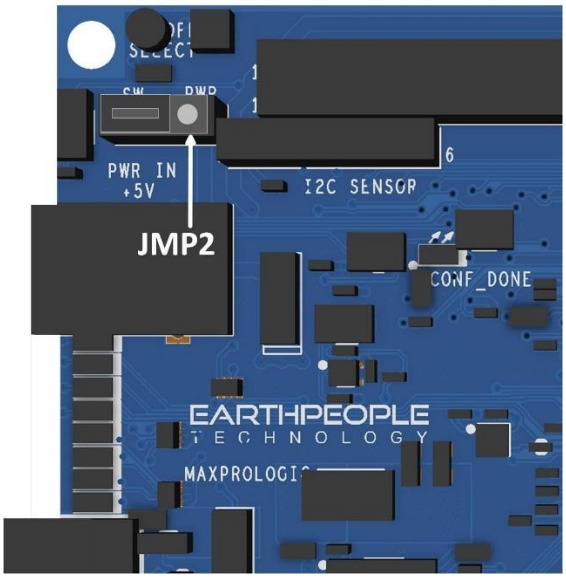
Clear Pin	Output of TPS54229
High	+3.3 V

Low	0V

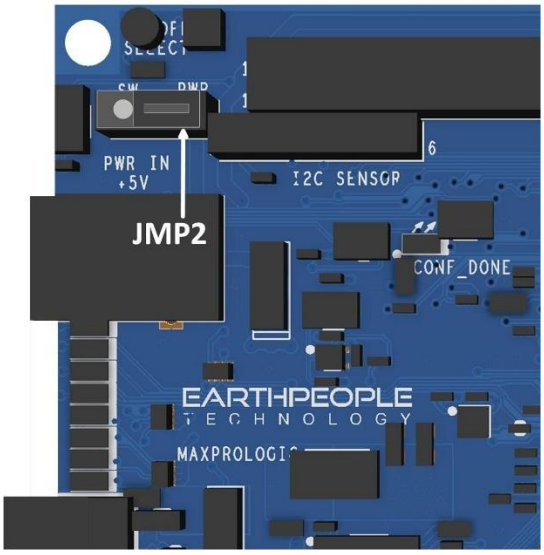
The On/Off controller can be bypassed and allow the MaxProLogic to power up whenever power is applied. This selection of the “SW” or “PWR” is made using the JMP2 jumper.



The JMP2 provides the bypass. Set the jumper to the ‘1’ position to use the On/Off pushbutton switch.



Uses the ‘2’ position to allow the MaxProLogic to power up with board power.

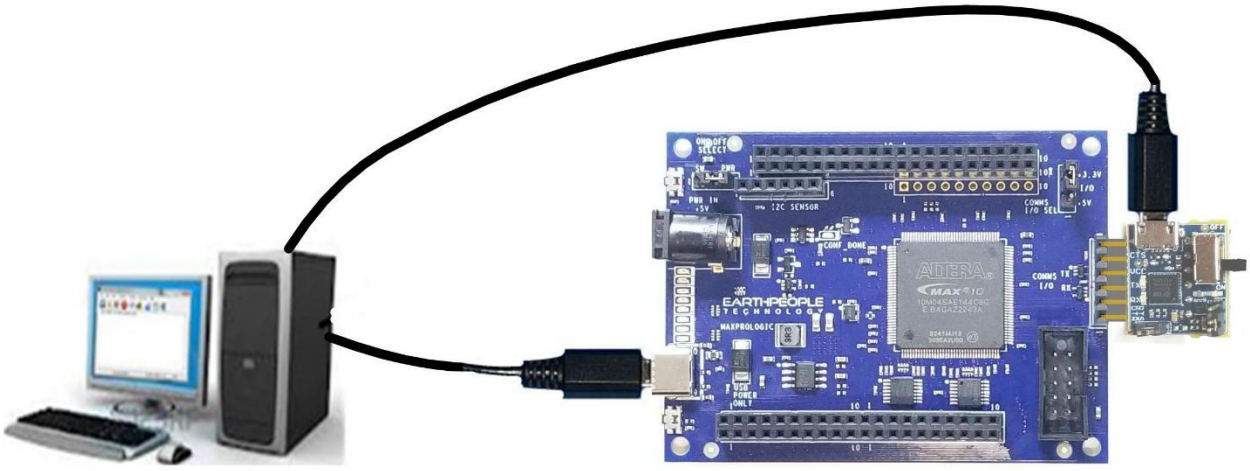


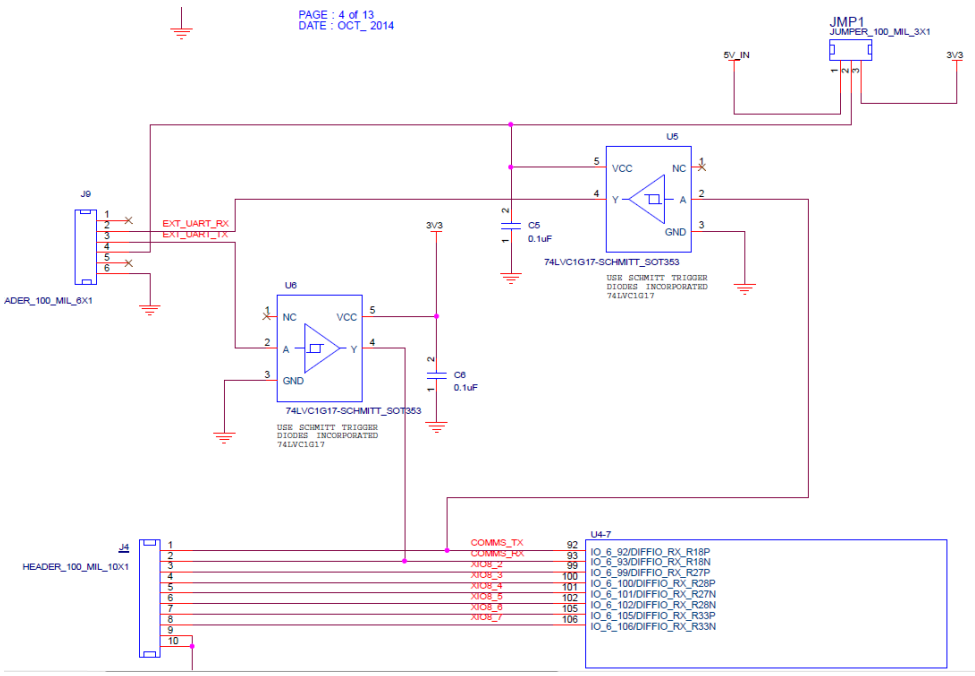
In this JMP2 setting, the MaxProLogic will power up when power is applied either through the USB-C or Barrel Connector.

3.14 Communications Interface

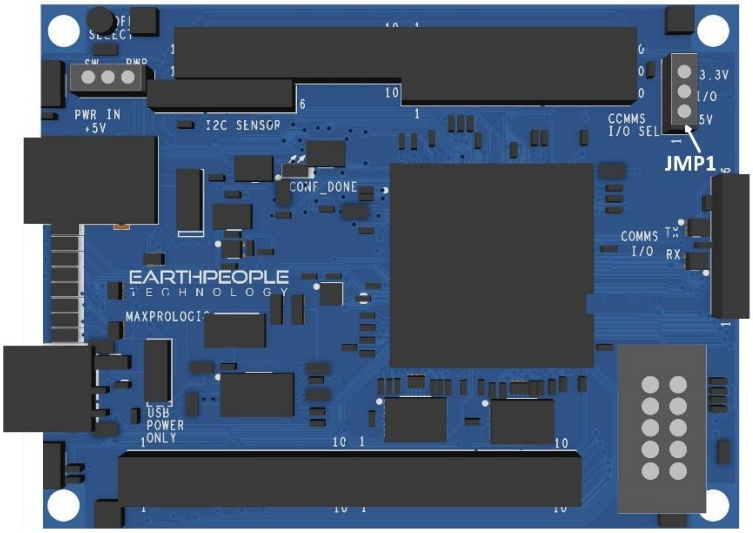
The MaxProLogic is equipped with a communications port that is compatible with FTDI Breakout Boards. It is a 6 pin female header that connects directly with most Breakout boards. The Communications Connector provides a path between the FTDI Breakout Board and the MAX10 FPGA. It brings both the RX and TX signals into the FPGA at the following pins.

Communications Connector-Pin Number	MaxProLogic Schematic Signal	FPGA Signal	MAX 10 Pin Number
J9-1	NC	--	
J9-2	RX	COMMS_UART_RX	92
J9-3	TX	COMMS_UART_TX	93
J9-4	VCC	--	
J9-5	NX	--	
J9-6	GND	--	

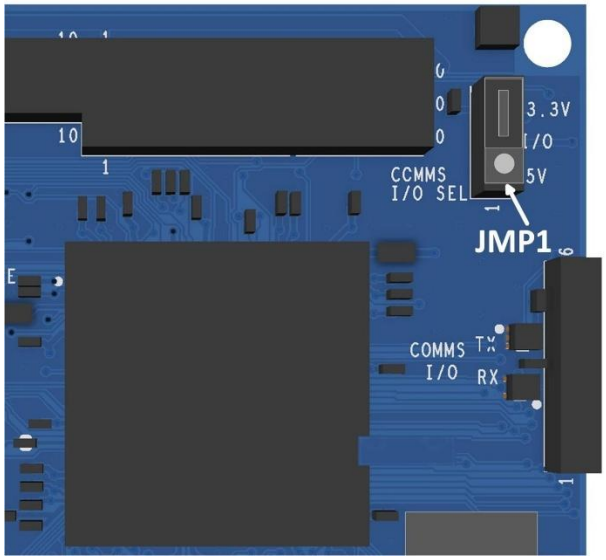




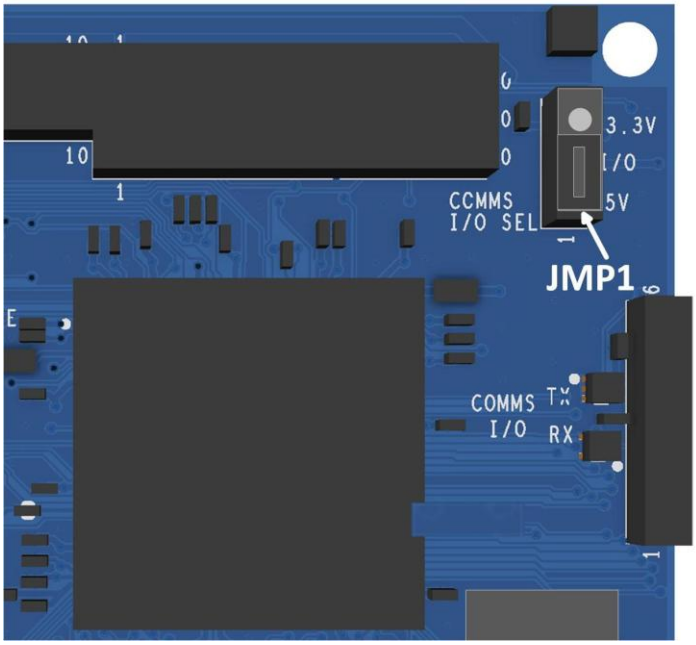
The MaxProLogic Communications Interface is compatible with both +5V and +3.3V Breakout Boards. The JMP1 provides a user selection jumper to select the +5V and +3.3V interface. There are two 74LVC1G17 Schmitt Trigger chips to provide voltage level compatibility with the FPGA.



Add a two pin jumper to pins 1 and 2 of JMP1 to select +3.3V Input/Output.

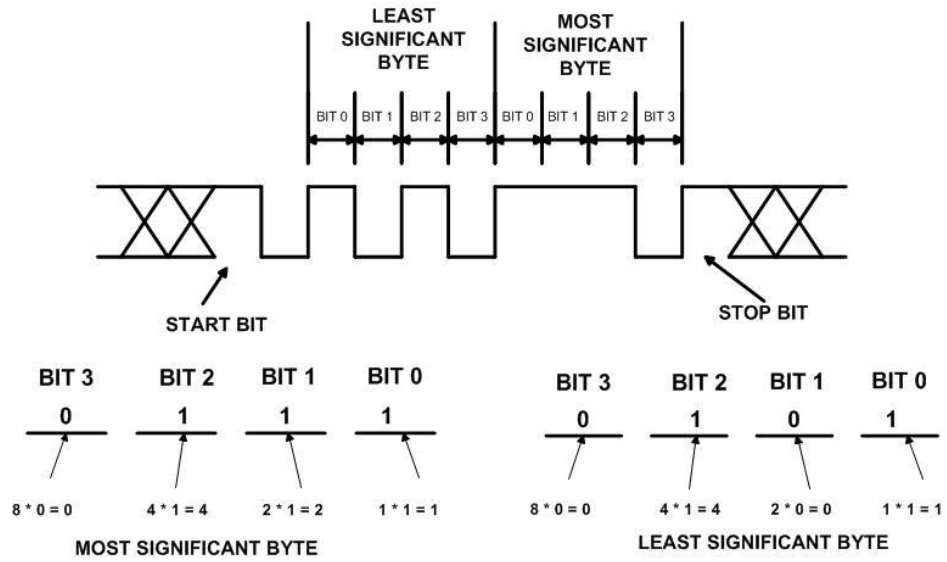


Add a two pin jumper to pins 2 and 3 of JMP1 to select +5V Input/Output.



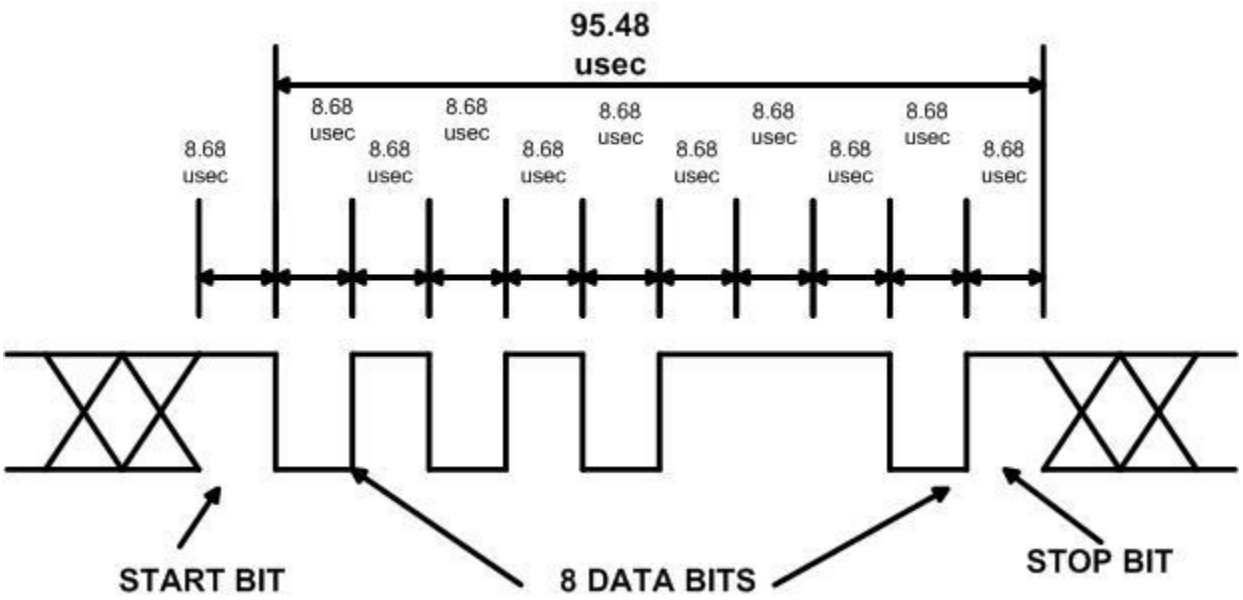
3.14.1 UART Communication Protocol

The FPGA must run Verilog code to perform the full duplex UART communications. The code must decode in the incoming RX signal and produce the outgoing TX signals with an accuracy of greater than 99%. This means up to 1% error is acceptable. The UART signals are based on a protocol. Where the UART protocol indicates one start bit, eight data bits and one stop bit.

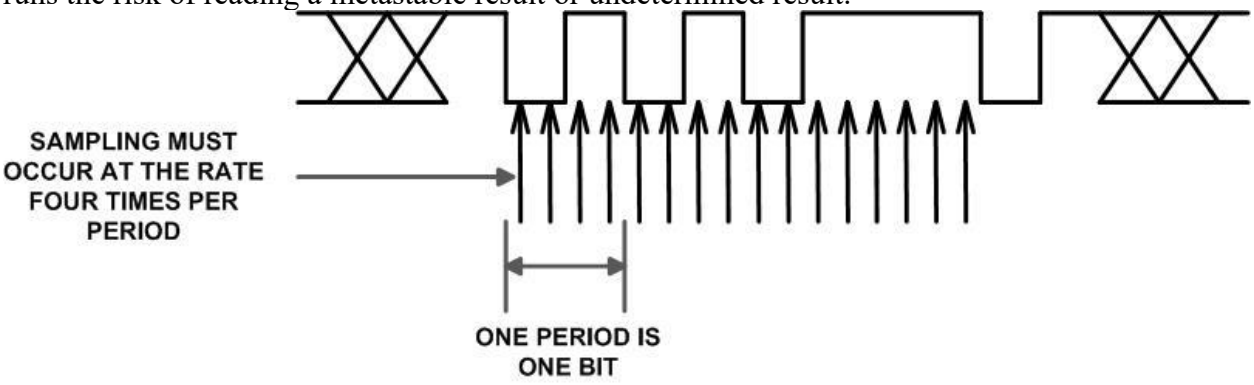


Each bit of the protocol must assert in the allocated time slot for each bit. What this means is that the clock used to time each bit of the protocol is embedded in the data. Both the sending device and receiving device must use the same clock rate. This embedding of the clock into data rate is called the baud rate.

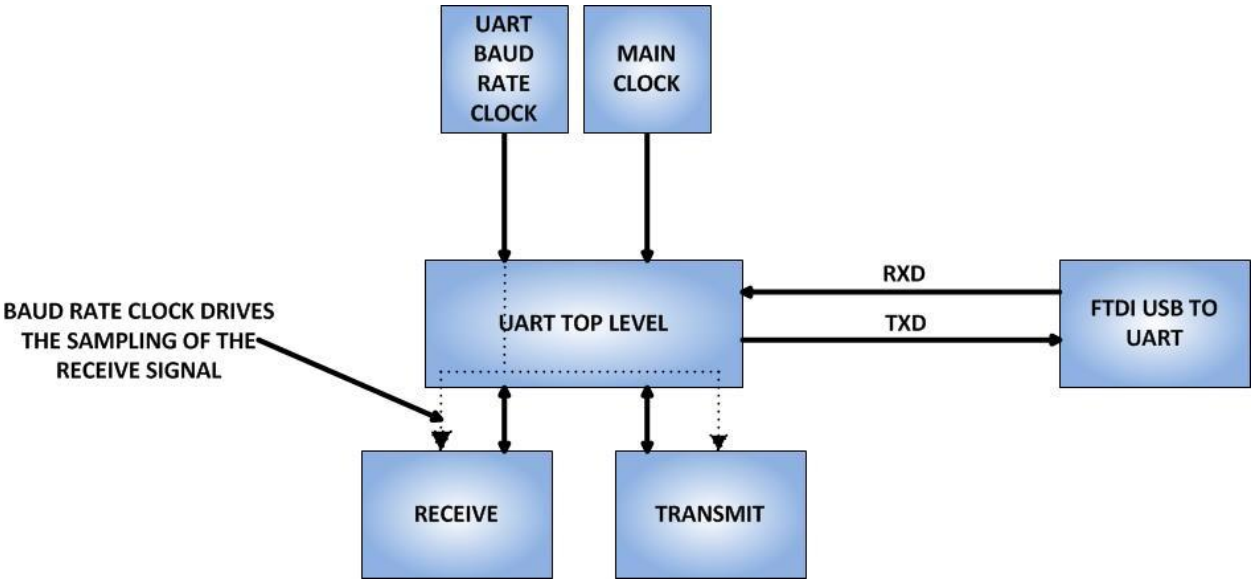
The baud rate describes the data rate in bits per second. The timing diagram for a baud rate of 115,200 is 8.68 microseconds per bit.



For the incoming RX signal, the FPGA must sample each bit of the protocol at four times per bit. This sampling rate is critical as the sending device clock and the receiving device clock are not synchronized. If the sampling rate is too low, a transition may be missed. If the sampling rate is only two times greater than the period of the receive device clock, samples could fall in the rising of falling edge of the received signal. Any time a sample occurs on the rising or falling edge, it runs the risk of reading a metastable result or undetermined result.



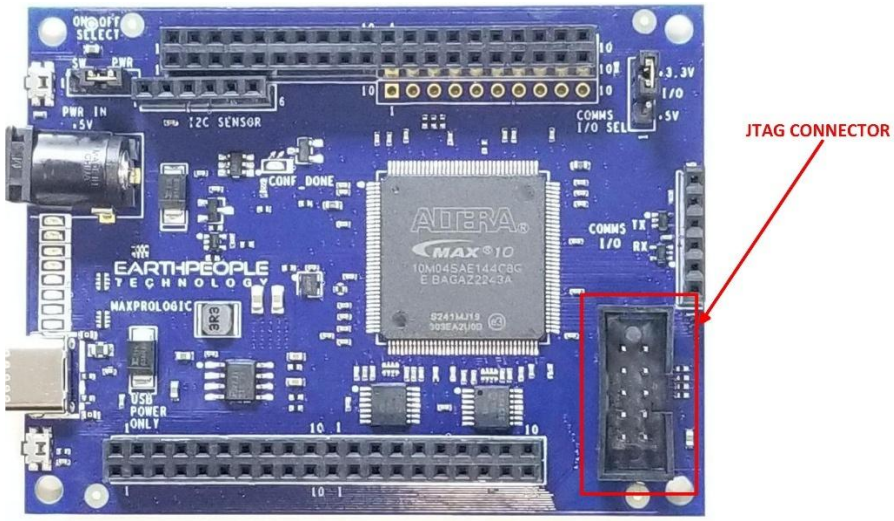
The Verilog code is written so that a special Baud Rate Clock is used to drive the UART Receive Block.



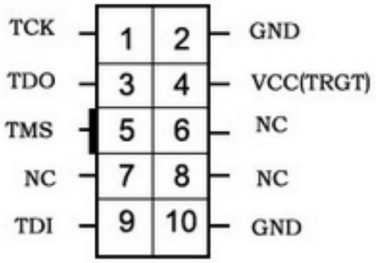
This code will use the Baud Rate Clock to drive a conditional if statement. The if statement will branch when a transition occurs and records the state of the signal (1 or 0). The Baud Rate Clock must be set to four times the incoming signal bit rate.

3.15 JTAG Interface

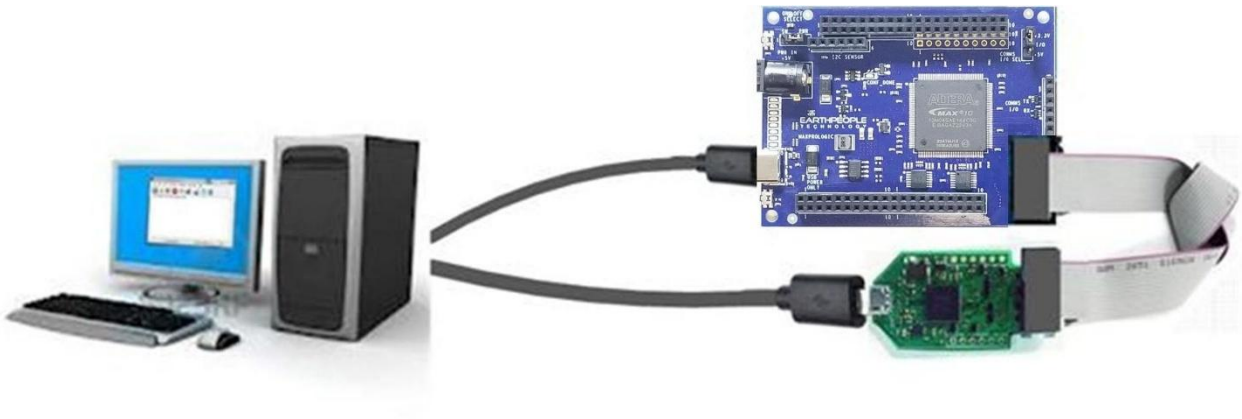
The MaxProLogic has a 5x2 header for use in programming the MAX10 FPGA via JTAG. The connector is located in the bottom right corner of the MaxProLogic. It is shrouded and keyed to allow easier insertion.



This connector uses the standard Altera Blaster connector pinout.



The VCC(TRGT) is set to +3.3V on the MaxProLogic. There are no jumper settings to make in order to program the MAX10 FPGA. Just connect a compatible Blaster to the connector and the PC, then use the Quartus software to program the FPGA.

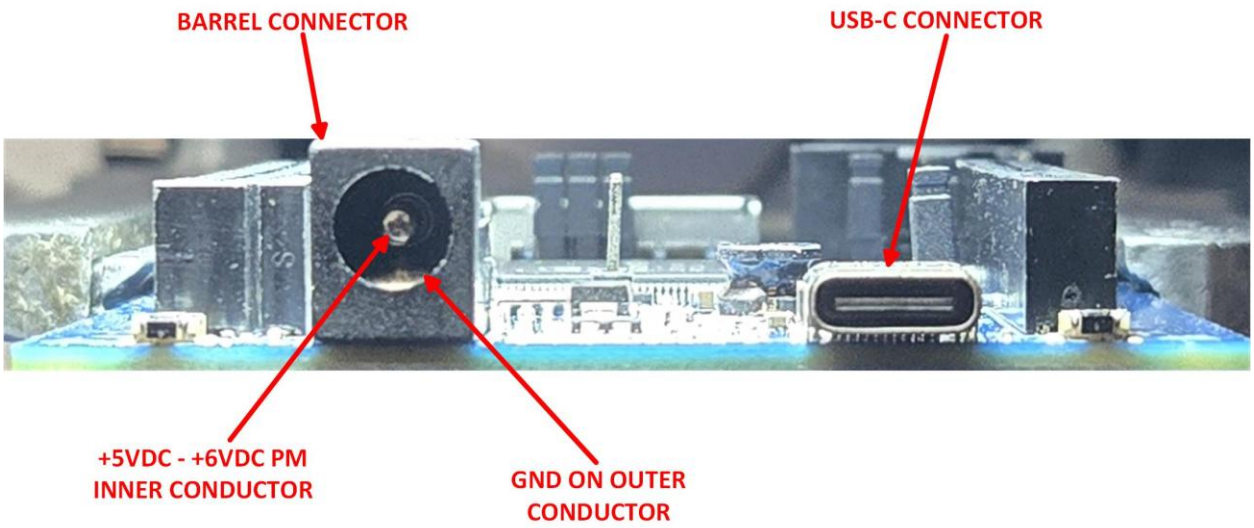


Please see the "Programming the FPGA" section for a full description of the JTAG programming process.

4 Powering the MaxProLogic

You can run the MaxProLogic from a laptop with 2.5W of power. Or you can run it from the +5V @ 2A wall USB chargers for 10W of power. The barrel connector can handle up to +5.5V @ 3 A for 15W of power.

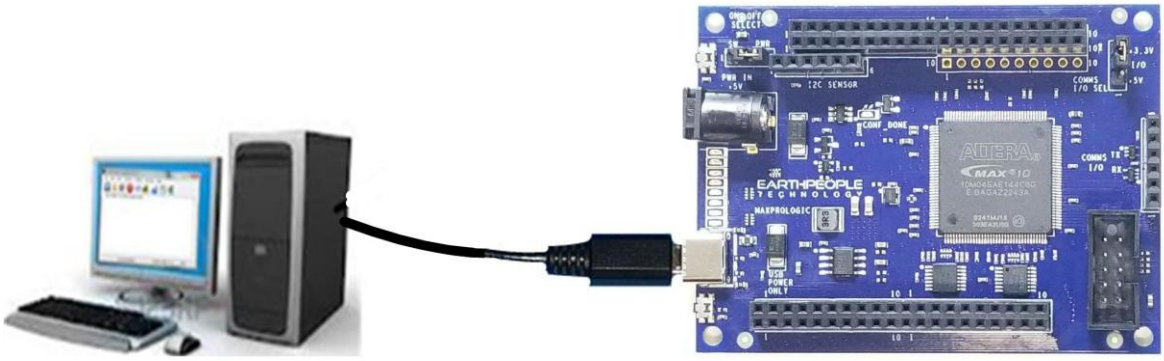
- Standard USB cable from Laptop/PC.
- +5 VDC wall charger (phone charger) through USB cable.
- +4.5 to +5.5 VDC supplied through the DC power jack.



The barrel connector is the typical size used on many popular DIY boards such as the Arduino series. It has the following mechanical specs:

- 2.0mm Inner Diameter
- 5.5mm Outer Diameter

The barrel connector does not include a diode protection to prevent reverse polarity connection. So, care must be exercised when connecting up your cable to the barrel connector. Please ensure the correct polarity connections are made before connecting to the MaxProLogic. Also, there is no discrete protection to the power input. The power supply does include a high current protection circuit. The current limit is around 4.7Amps. But, the MaxProLogic is only designed to handle 2Amps of current. So, damage may occur to the MaxProLogic if the user does not exercise care in design and use of the Inputs/Outputs.



Power the MaxProLogic directly from the PC. +5V@0.5A

Power the MaxProLogic directly from the wall charger. +5V@2A

5 Installing Quartus

You must install Quartus Prime to configure the CycloFlex. Altera Quartus Prime must be downloaded from the Altera website.

Download the Quartus Prime by following the directions in the Section Downloading Quartus.

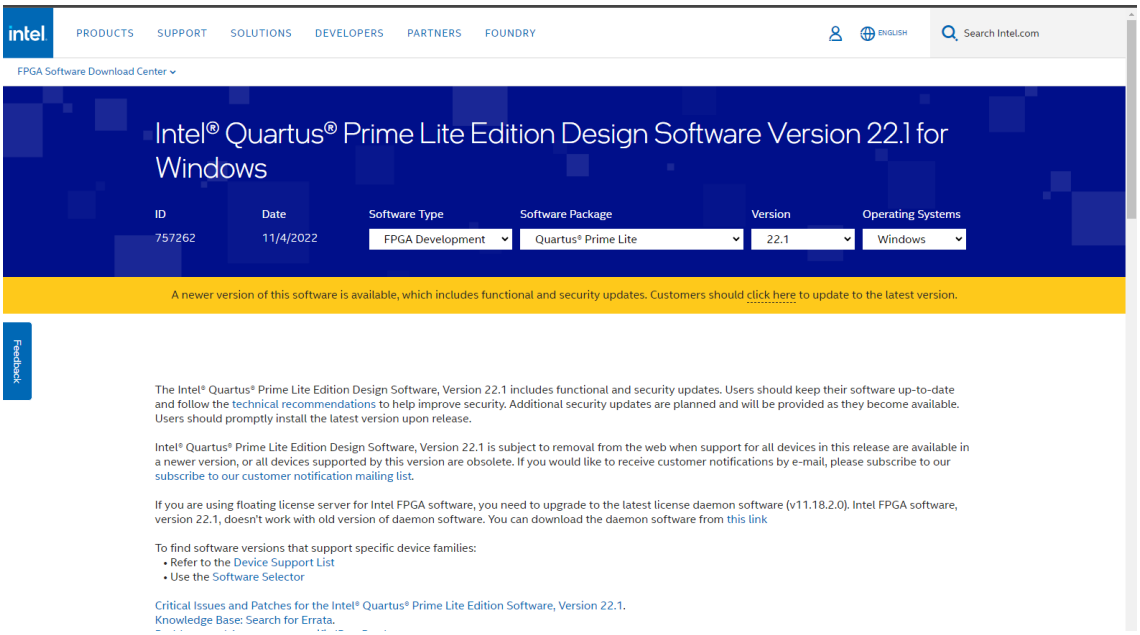
5.1.1 Downloading Quartus

The first thing to do in order build a project in Quartus is to download and install the application. You can find the latest version of Quartus at:

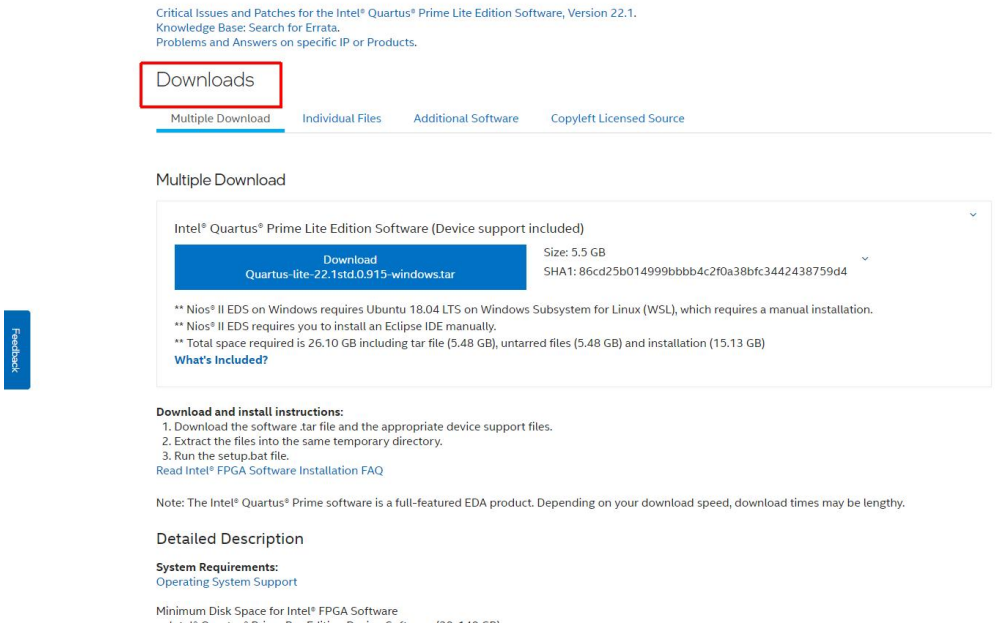
Intel FPGA Quartus Prime Lite

****Please be advised, Intel is prone to changing their website graphics every few months. The guide below is several months out of date. Please follow the latest instructions on the Quartus Prime Lite Download website. The instructions here are only meant as a guide to getting the software downloaded.****

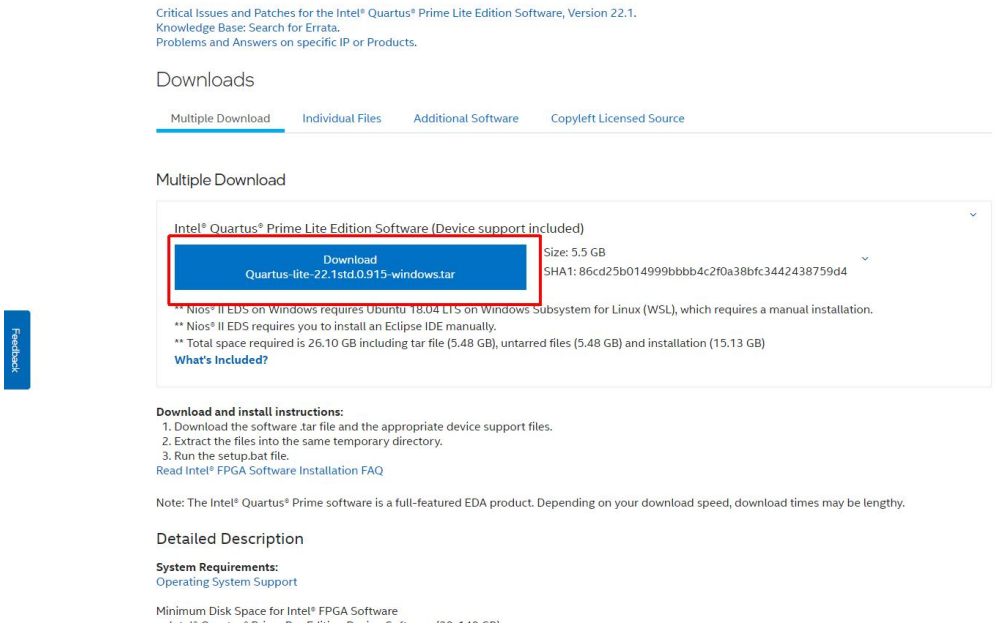
You will first need to apply for an account with Intel. Then use your login and password to access the download site. Click on the Download Windows Version.



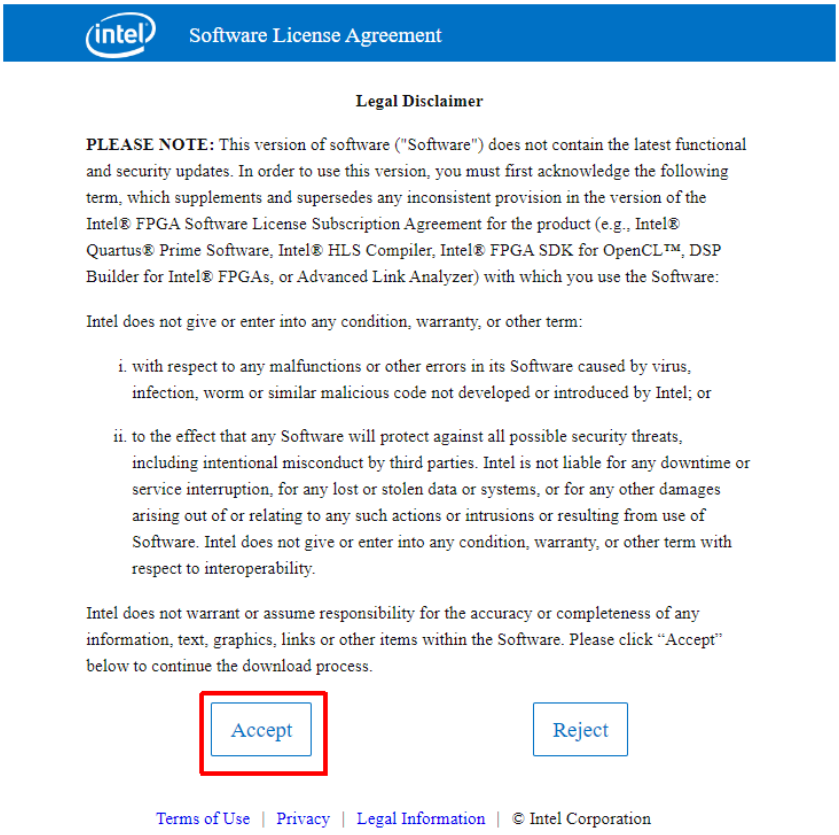
Scroll down the page to the “Downloads” section.



Click on the “Download Quartus-lite-22.1 xxxxxx Windows Tar.



Click through the Legal Stuff.



Software License Agreement

Legal Disclaimer

PLEASE NOTE: This version of software ("Software") does not contain the latest functional and security updates. In order to use this version, you must first acknowledge the following term, which supplements and supersedes any inconsistent provision in the version of the Intel® FPGA Software License Subscription Agreement for the product (e.g., Intel® Quartus® Prime Software, Intel® HLS Compiler, Intel® FPGA SDK for OpenCL™, DSP Builder for Intel® FPGAs, or Advanced Link Analyzer) with which you use the Software:

Intel does not give or enter into any condition, warranty, or other term:

i. with respect to any malfunctions or other errors in its Software caused by virus, infection, worm or similar malicious code not developed or introduced by Intel; or

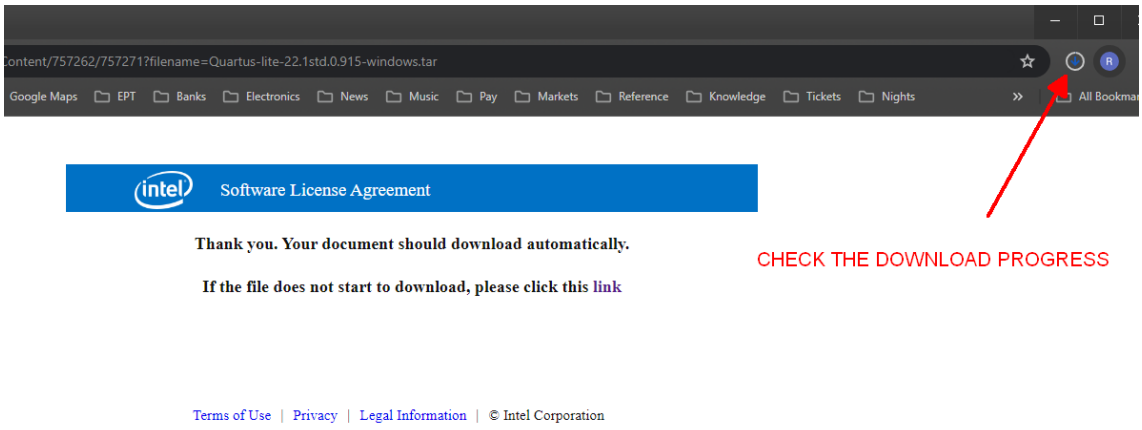
ii. to the effect that any Software will protect against all possible security threats, including intentional misconduct by third parties. Intel is not liable for any downtime or service interruption, for any lost or stolen data or systems, or for any other damages arising out of or relating to any such actions or intrusions or resulting from use of Software. Intel does not give or enter into any condition, warranty, or other term with respect to interoperability.

Intel does not warrant or assume responsibility for the accuracy or completeness of any information, text, graphics, links or other items within the Software. Please click "Accept" below to continue the download process.

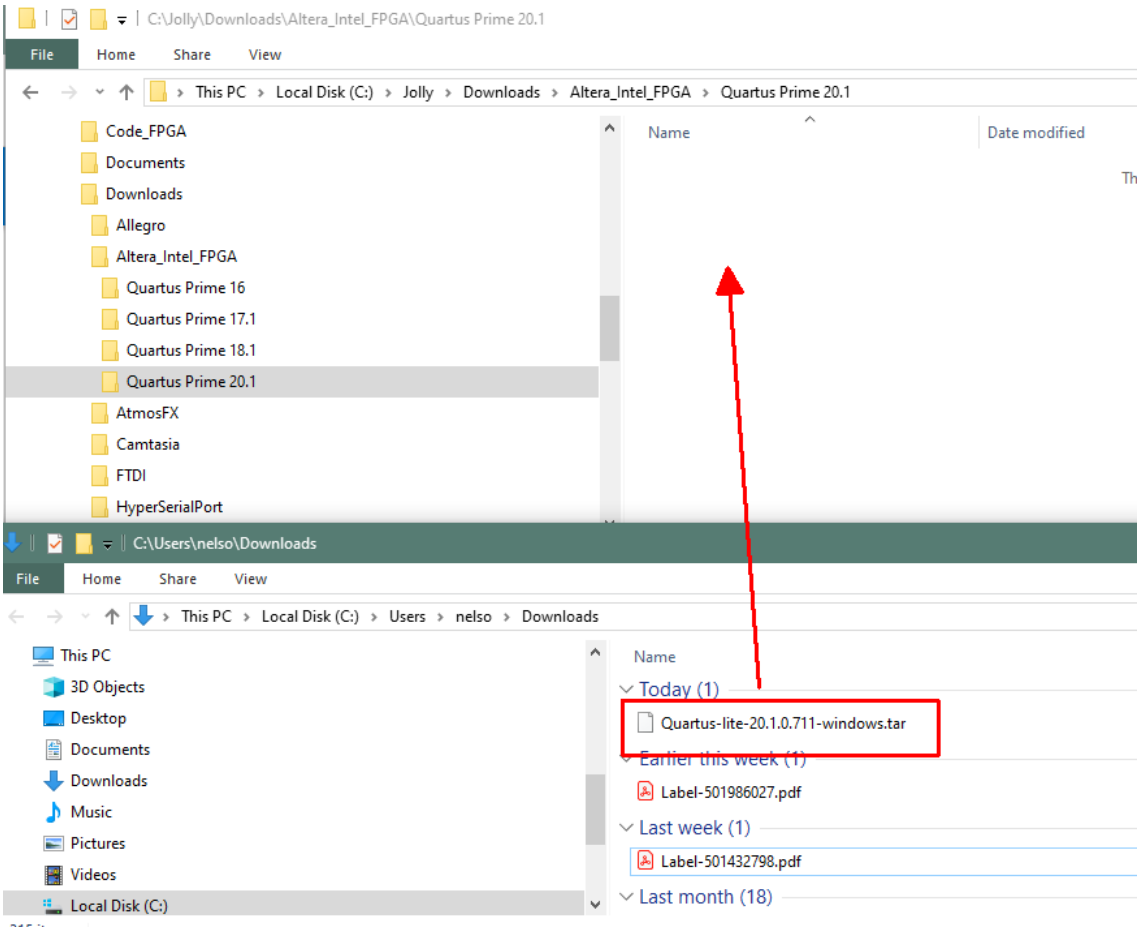
Accept Reject

Terms of Use | Privacy | Legal Information | © Intel Corporation

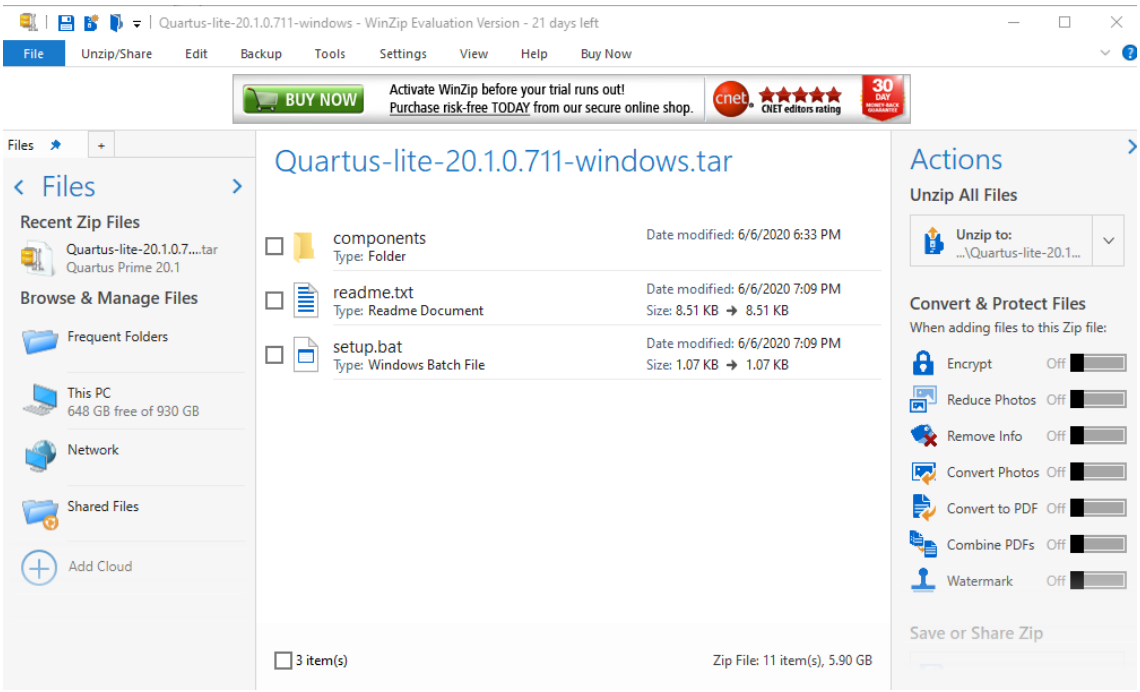
This will start the download.



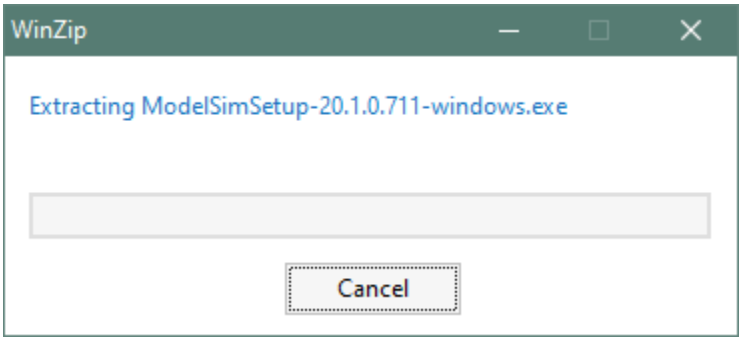
The file is 13.9 GB (or greater), so this could take a couple of hours depending on your internet connection. When download is complete, store the *.tar file in a directory on your PC.



Use a tool such as WinZip to Extract the *.tar file.

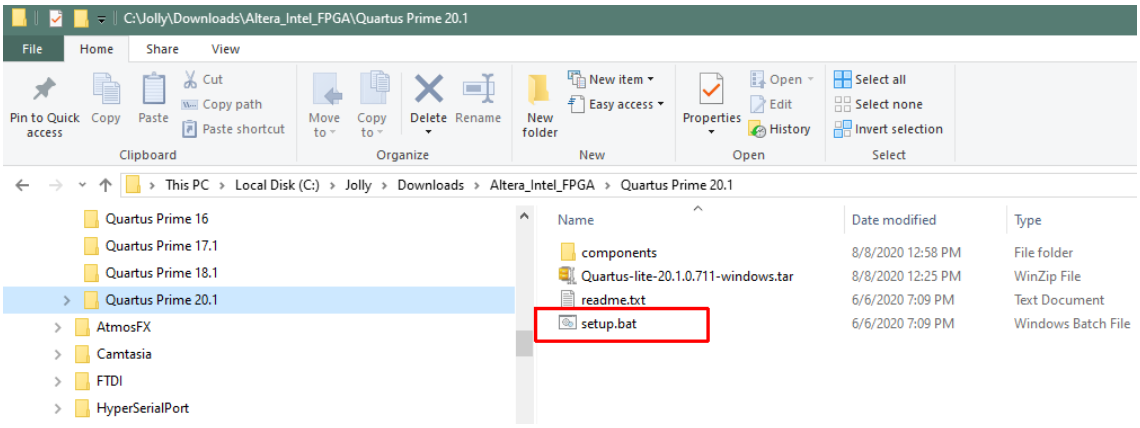


The tool will unpack all files.

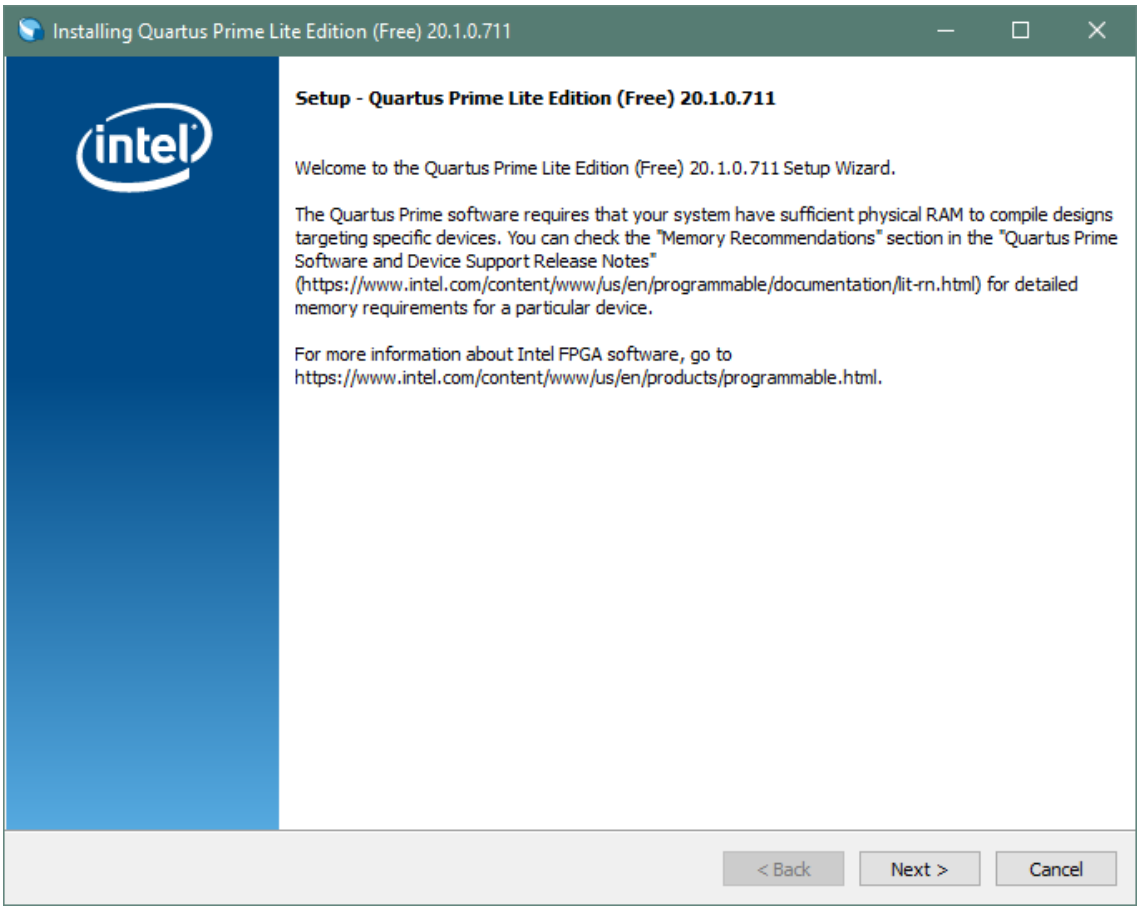


5.1.2 Quartus Installer

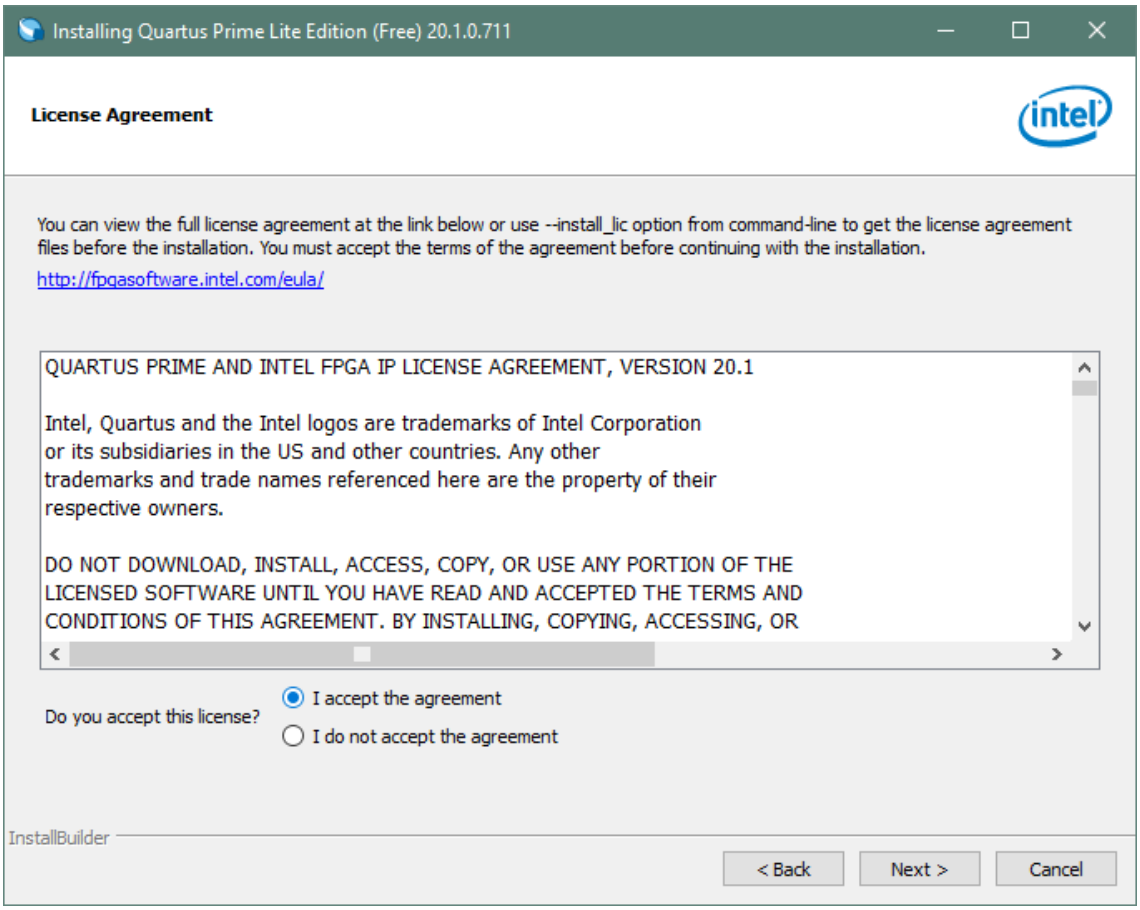
When the unpacking finishes from the previous section, double click the setup.bat file in the download folder.



Click “Next” on the Introduction Window.

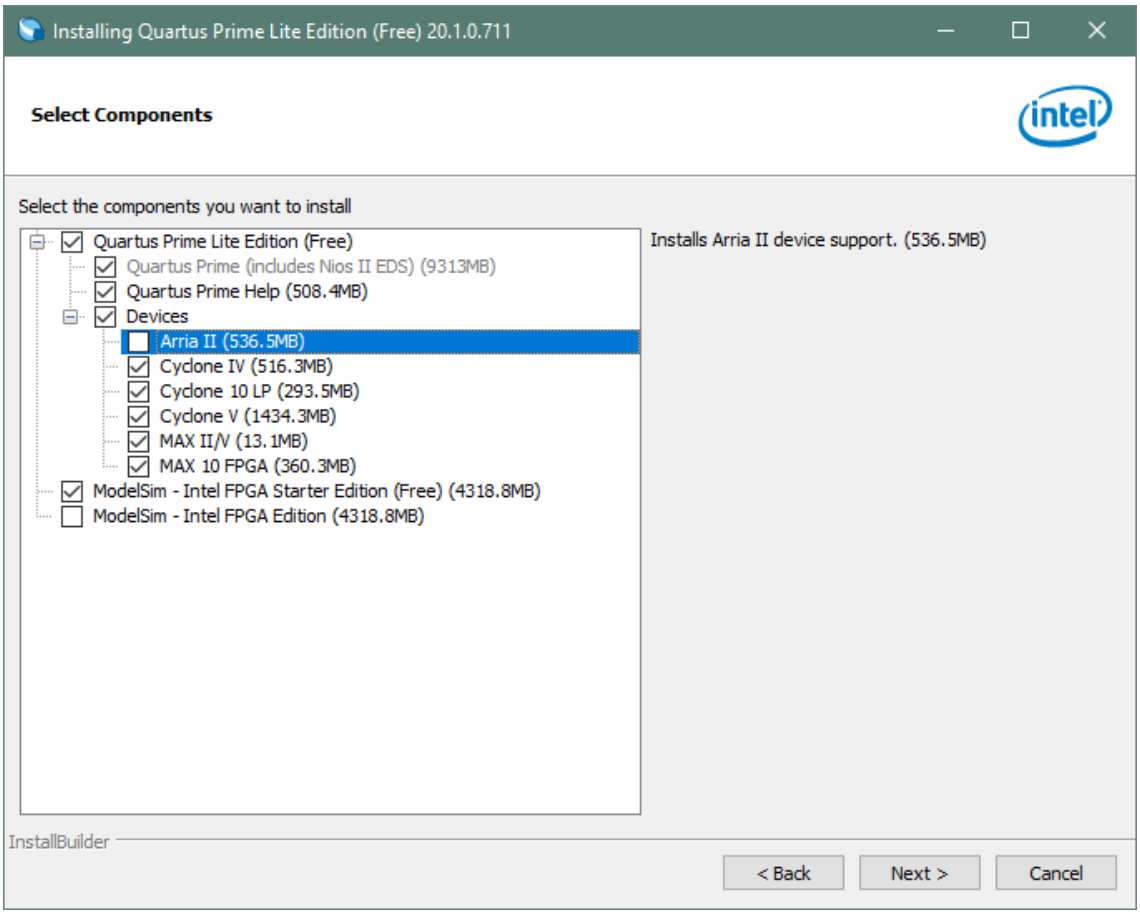


Click the checkbox to agree to the license terms. Then click “Next”.

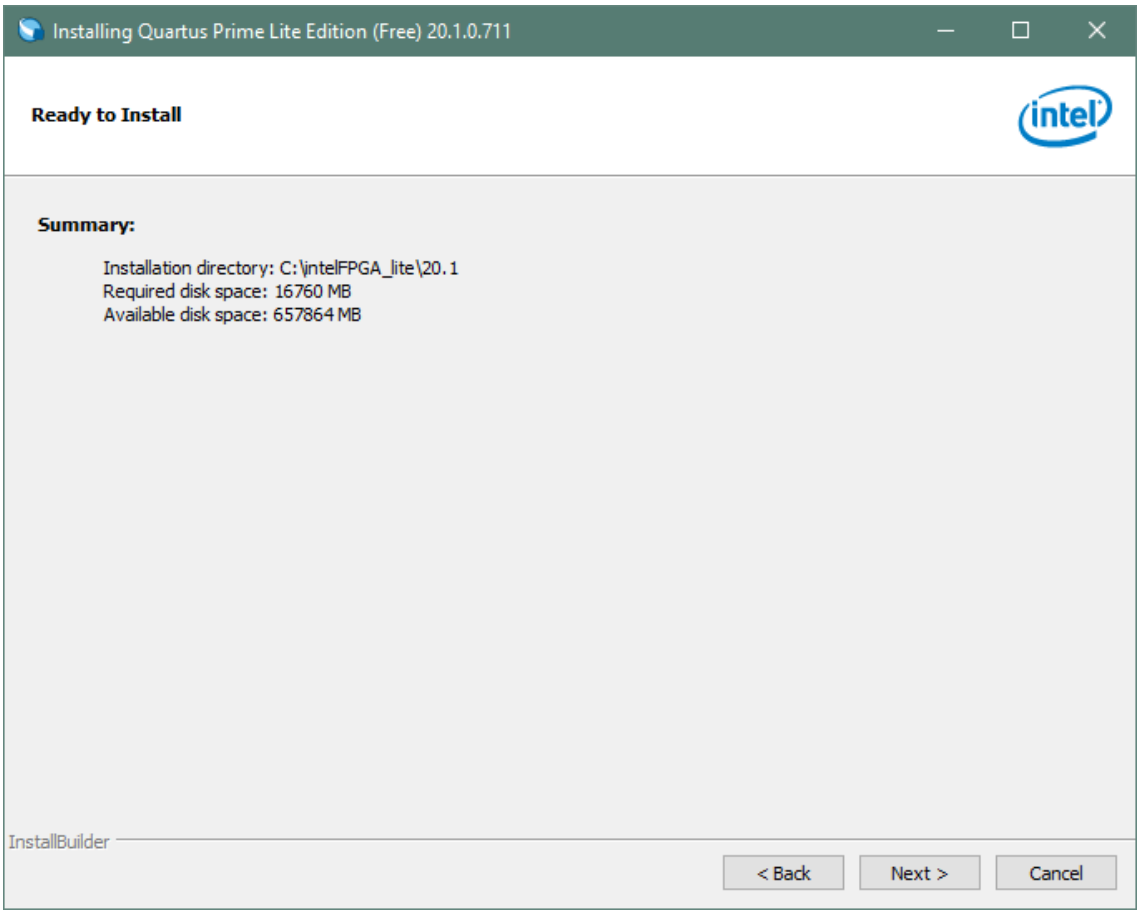


Click “Next” and accept the defaults.

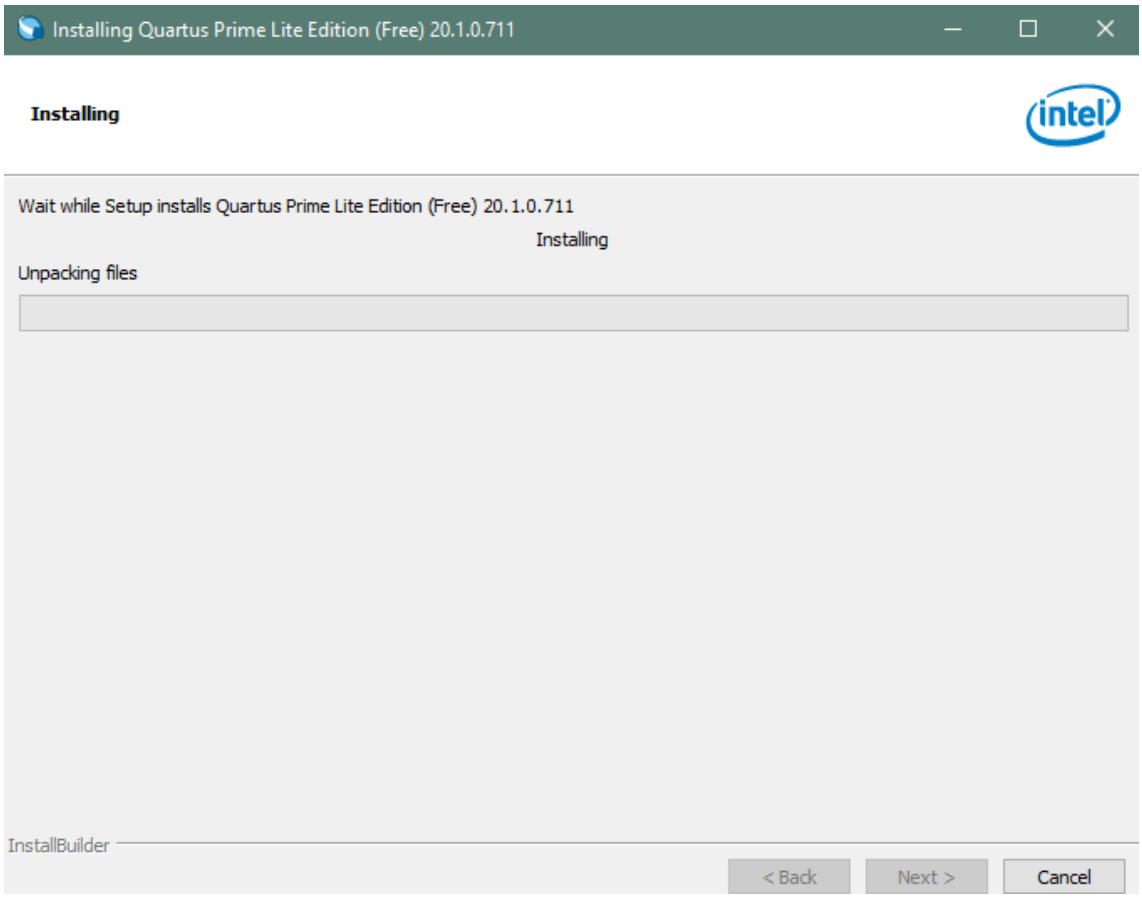
At the Select Products Window, de-select the Quartus Prime Supbscription Edition by clicking on its check box so that the box is not checked. Then click on the check box by the Quartus Prime Web Edition (Free).



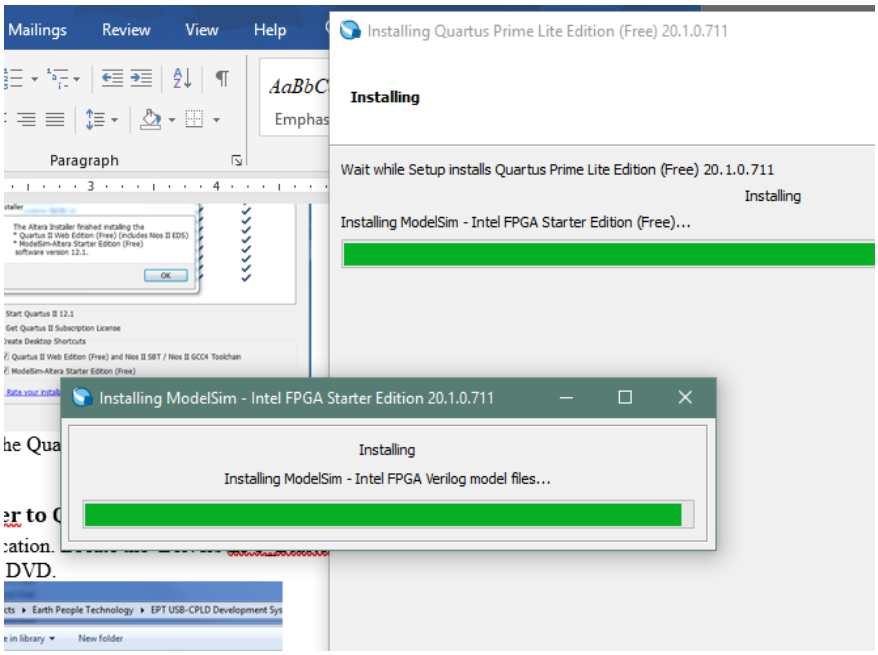
Click “Next” to accept the defaults

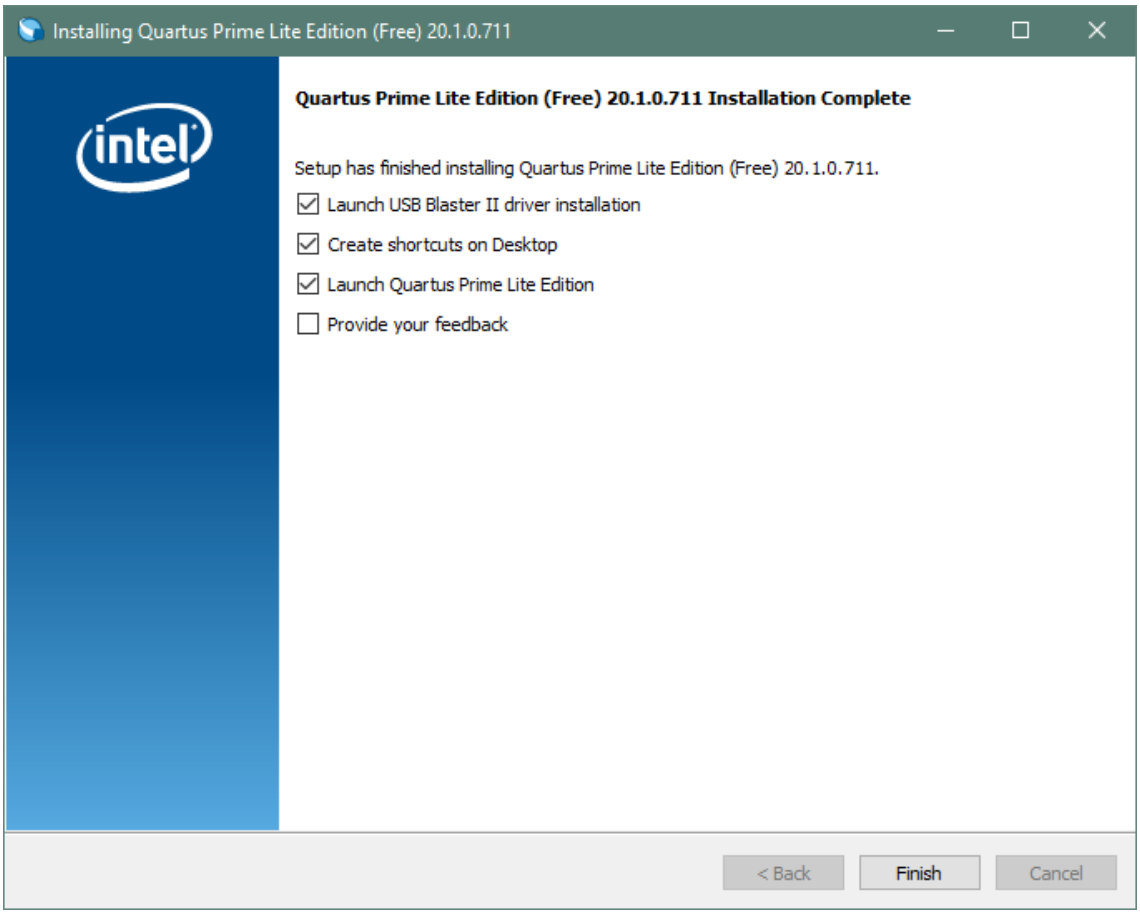


Click “Next” to accept the defaults

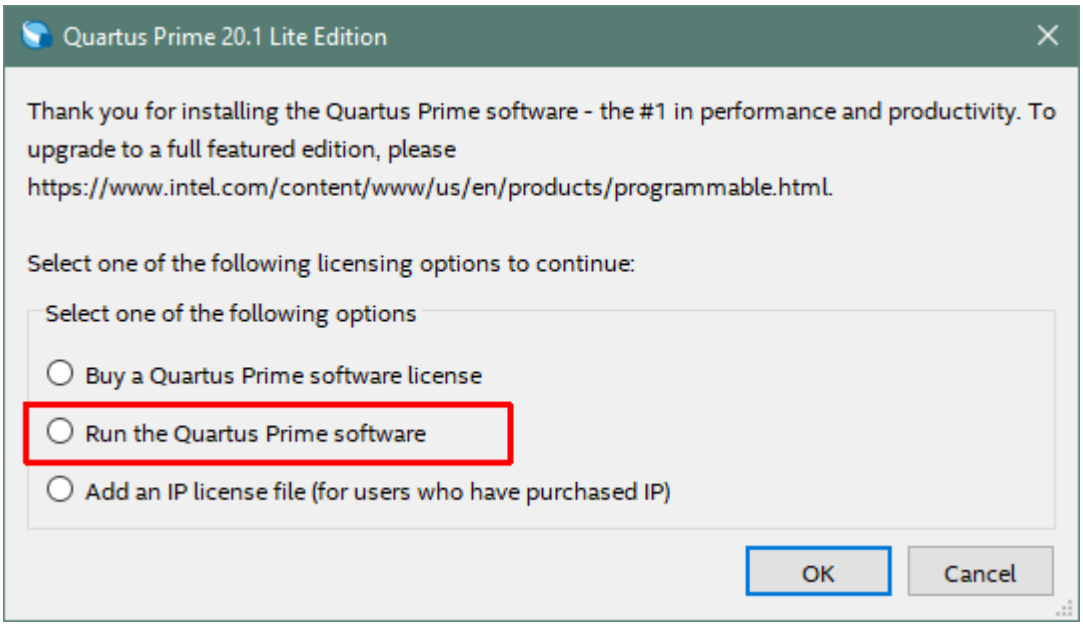


Wait for the installation to complete.

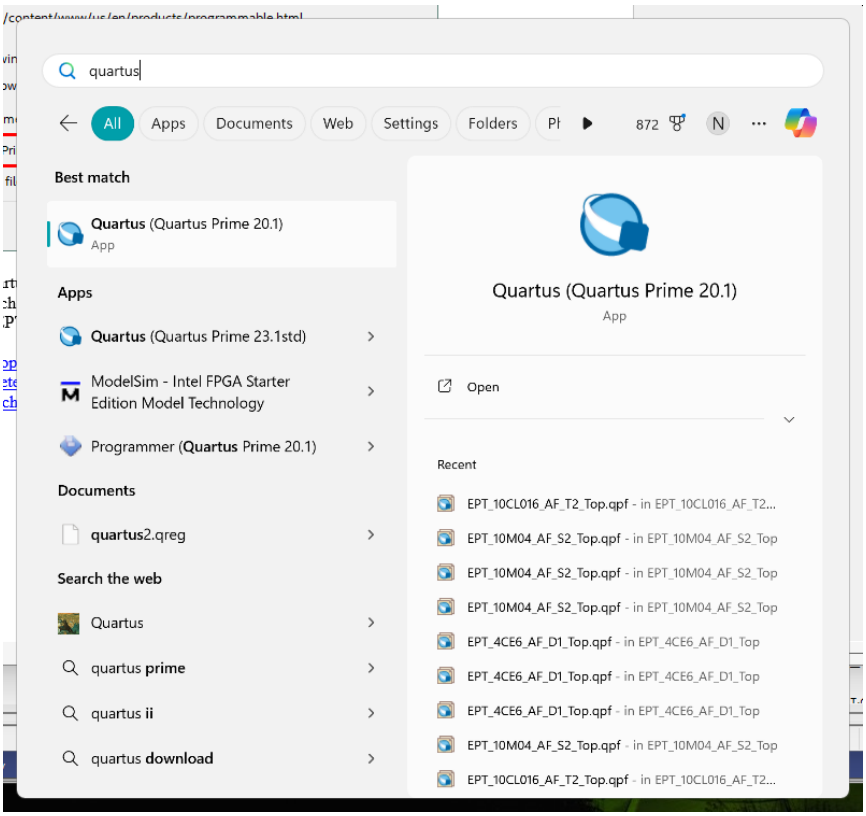

Mailings
Review
View
Help
Paragraph
Installing Quartus Prime Lite Edition (Free) 20.1.0.711
Installing
Wait while Setup installs Quartus Prime Lite Edition (Free) 20.1.0.711
Installing
Installing ModelSim - Intel FPGA Starter Edition (Free)...
Installing ModelSim - Intel FPGA Starter Edition 20.1.0.711
Installing
Installing ModelSim - Intel FPGA Verilog model files...
DVD.



Click “Ok”, then click “Finish”. The Quartus Prime is now installed and ready to be used.



At this point the Quartus Prime Lite software is installed. Go to the Windows button and type in "quartus" at the search prompt.



The Windows Search should locate the installed version of Quartus. Click on the icon and the software should load properly. If this does not occur, contact Earth People Technology for support. There are three methods to contact EPT for support:

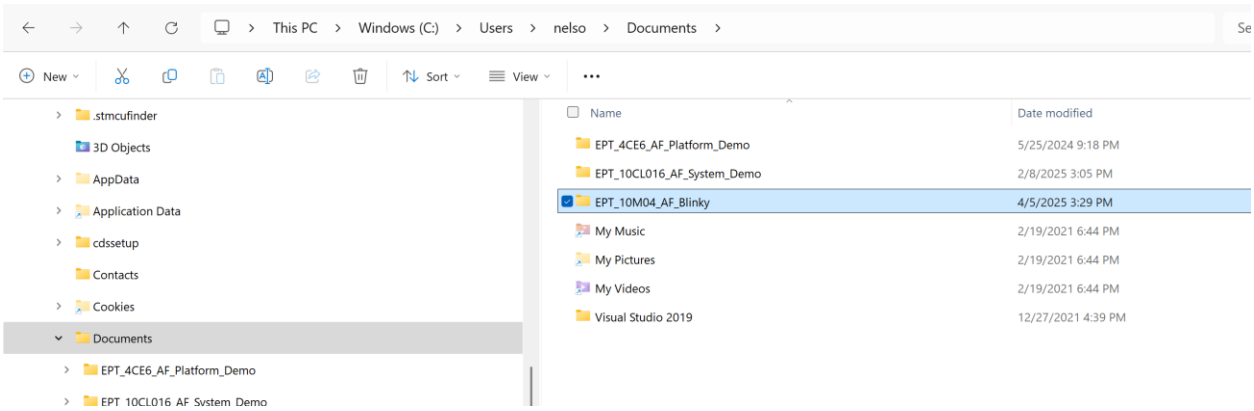
https://www.earthpeopletechnology.com->Forums
support@earthpeopletechnology.com
sales@earthpeopletechnology.com

6 Compiling, Synthesizing, and Programming the FPGA

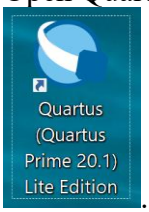
The FPGA on the EPT-10M04-AF-S2can be programmed with the Active Transfer Library and custom HDL code created by the user. Programming the FPGA requires the use of the Quartus Prime Lite software and a standard USB cable. There are no extra parts to buy, just plug in the USB cable. Once the user HDL code is written according to the syntax rules of the language (Verilog and VHDL) it can be compiled and synthesized using the Quartus Prime Lite software. This manual will not focus on HDL coding or proper coding techniques, instead it will use the example code to compile, synthesize and program the FPGA.

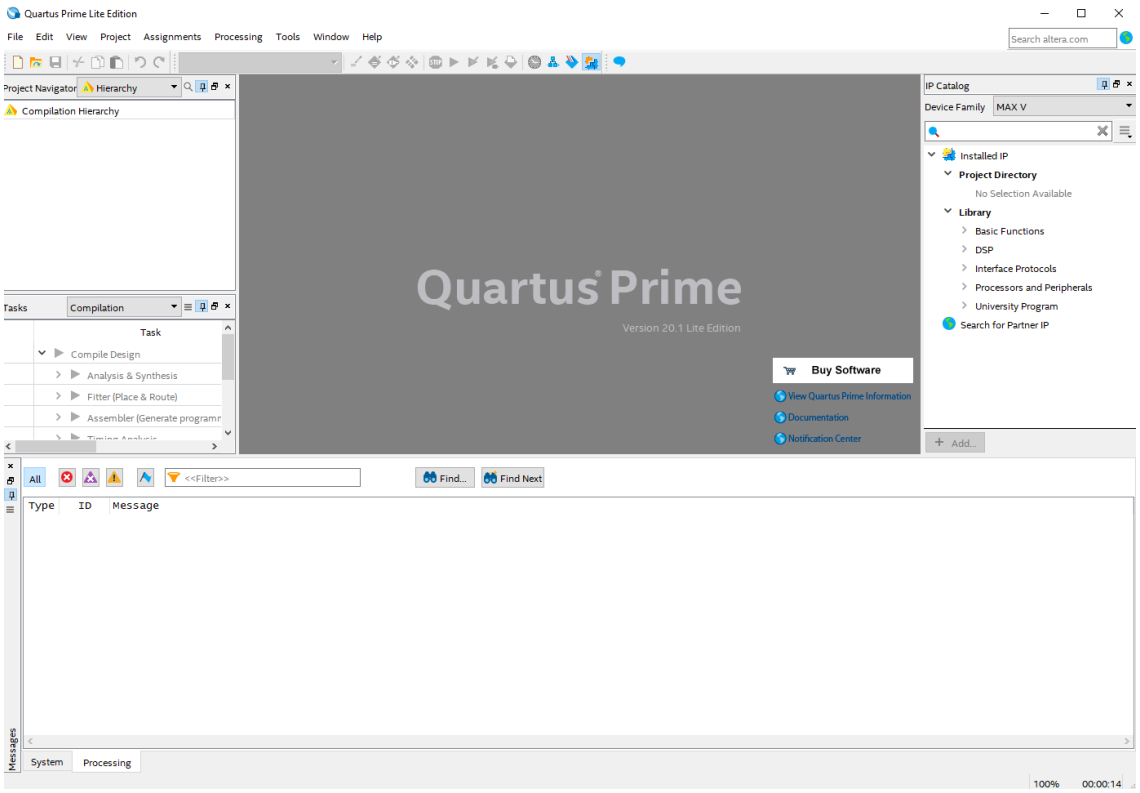
6.1 Setting up the Project and Compiling

Once the HDL code (Verilog or VHDL) is written and verified using a simulator, a project can be created using Quartus Prime Lite. Writing the HDL code and simulating it will be covered in later sections. Bring up Quartus Prime Lite, then use Windows Explorer to browse to C:\Users\<Your PC>\Documents create a new directory called: “EPT_10M04_AF_Blinky”.

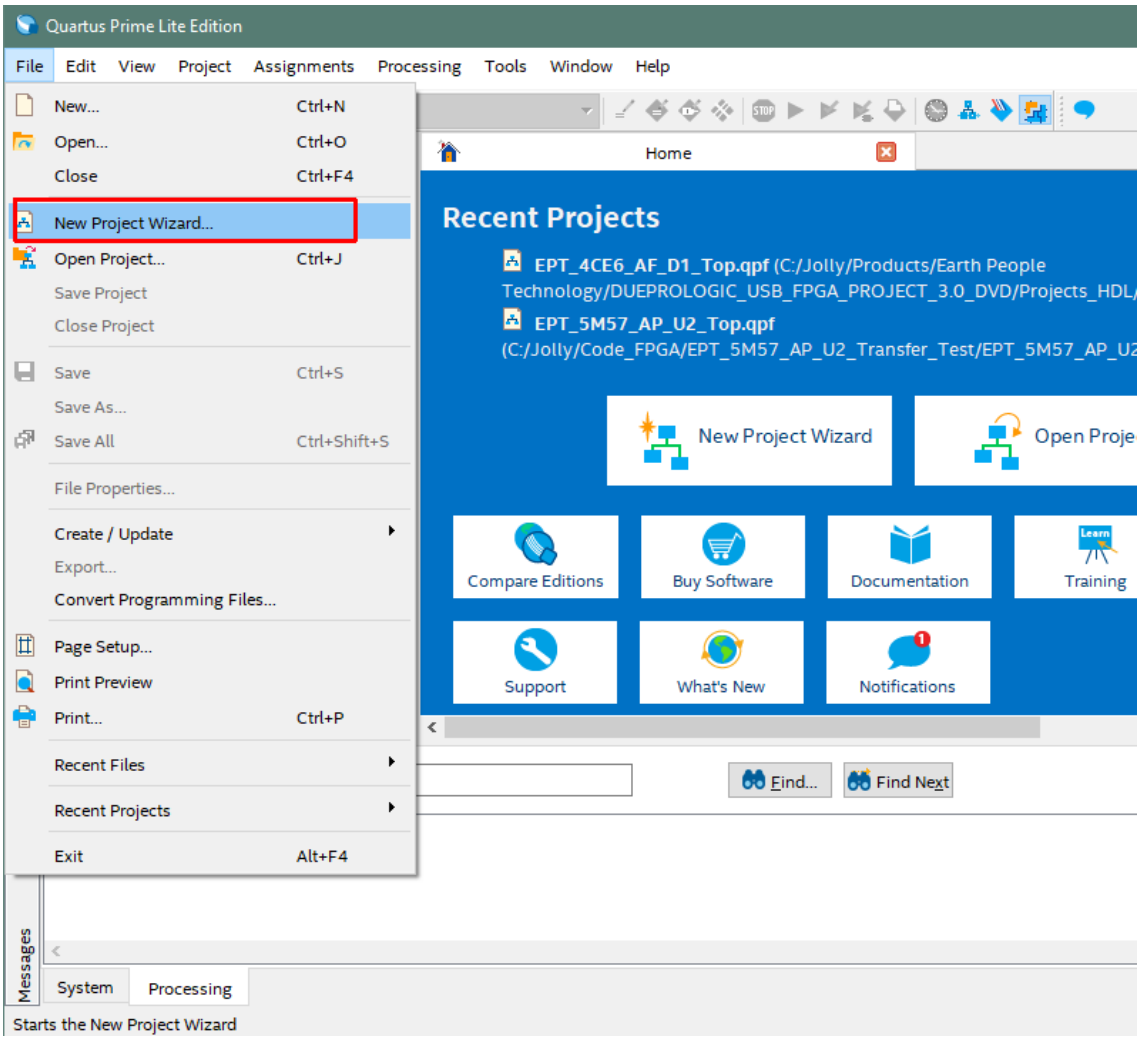


Open Quartus Prime Lite by clicking on the icon

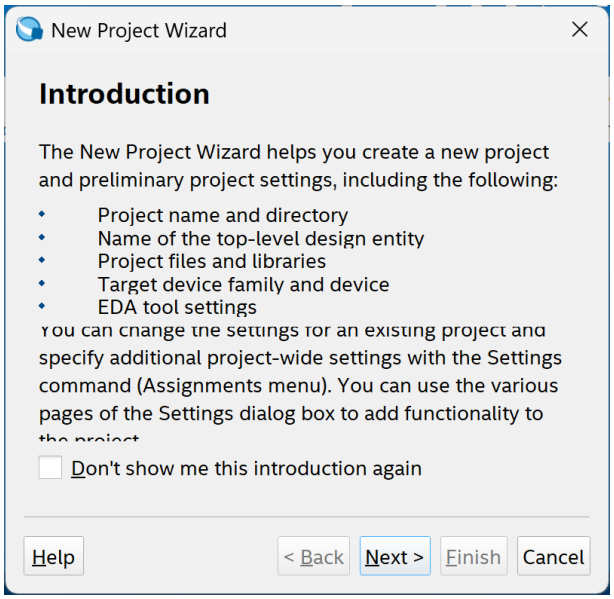




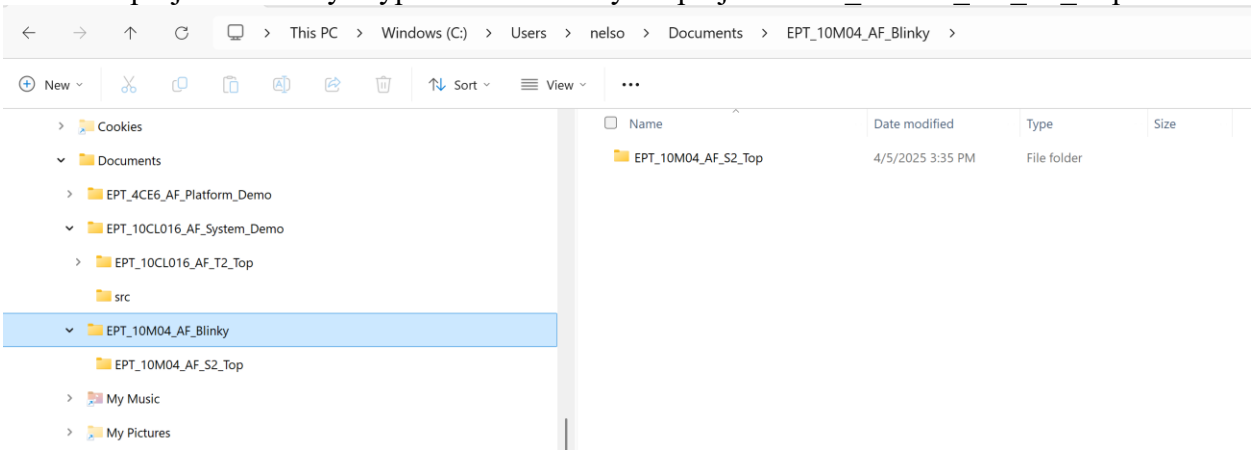
Under Quartus, Select File->New Project Wizard. The Wizard will walk you through setting up files and directories for your project.



The Introduction Page will open.

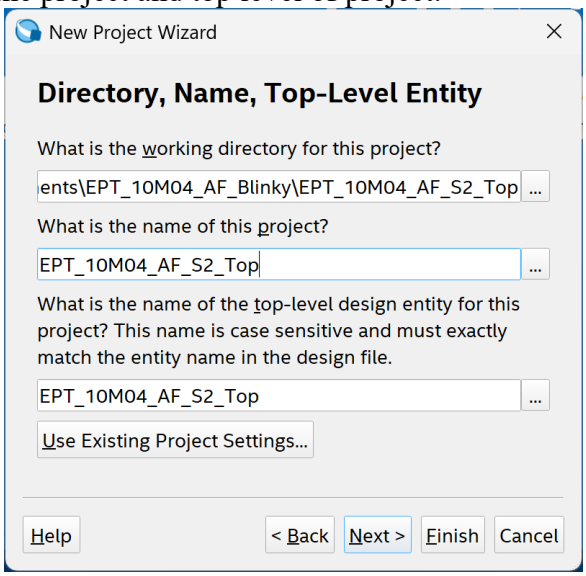


Go to your project saved at C:\Users\<Your PC>\Documents directory and add a new directory under the project directory. Type in a name for your project “EPT_10M04_AF_S2_Top”.

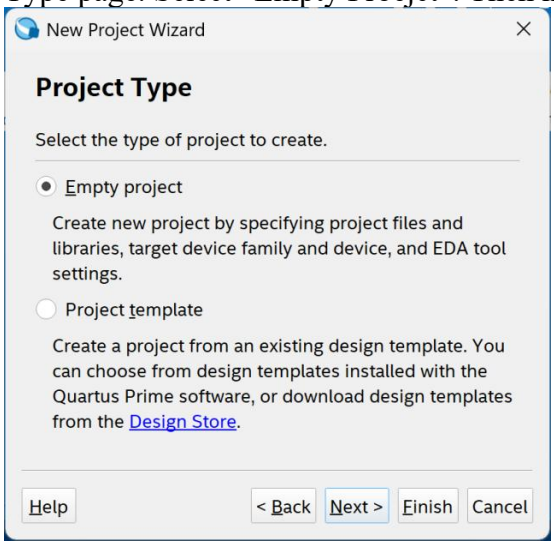


Hit the “Next” key on the Intro page.

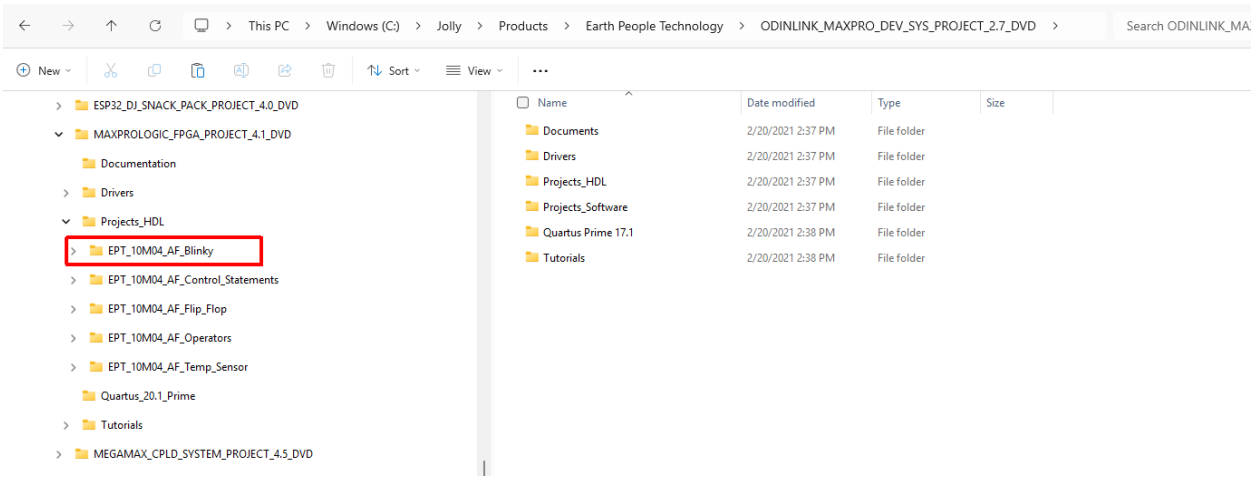
At the Top-Level Entity page, browse to the C:\Users\<Your PC>\Documents directory to store your project. Select the directory under the project name: EPT_10M04_AF_S2_Top. Then add this name to the name of the project and top level of project.



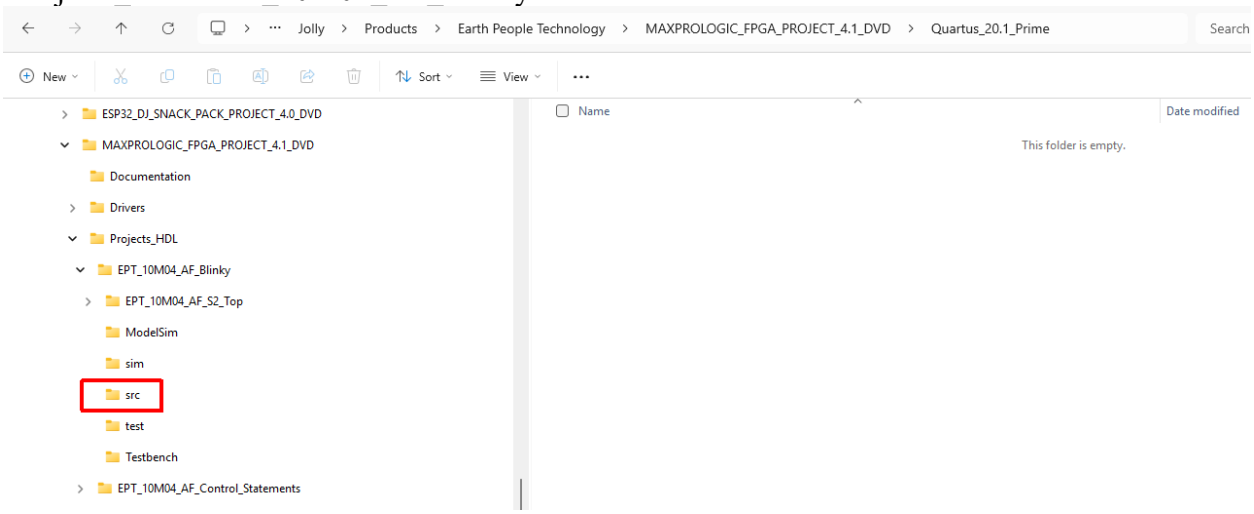
Select Next. At the Project Type page. Select “Empty Proejct”. Then hit Next button.



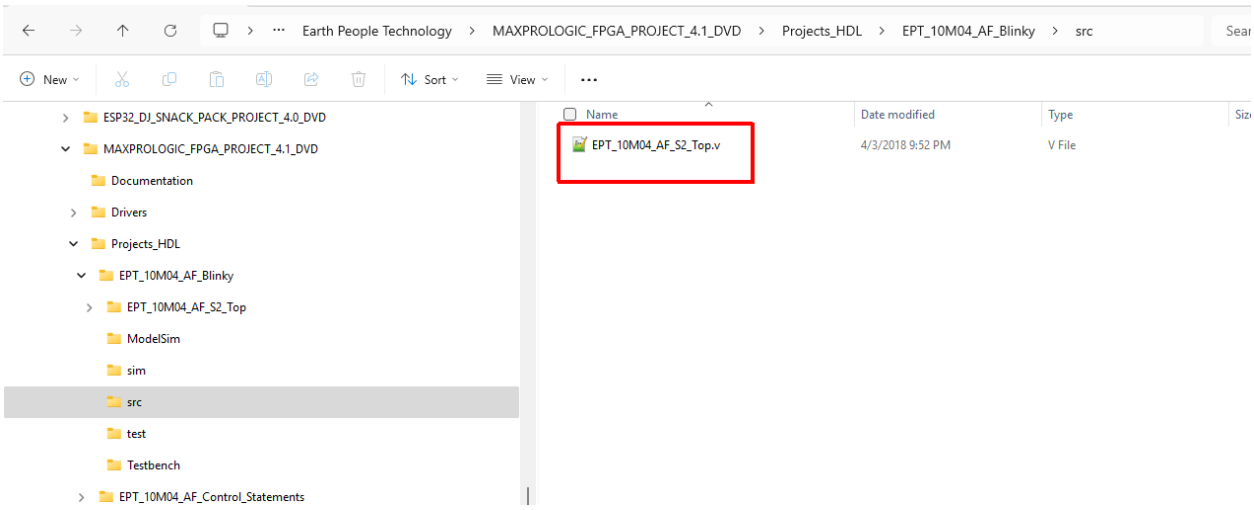
Next, navigate to the MaxProLogic Project DVD. Locate the $/MAXPROLOGIC_FPGA_PROJECT_4.x_DVD /Projects_HDL/EPT_10M04_AF_Blinky folder.



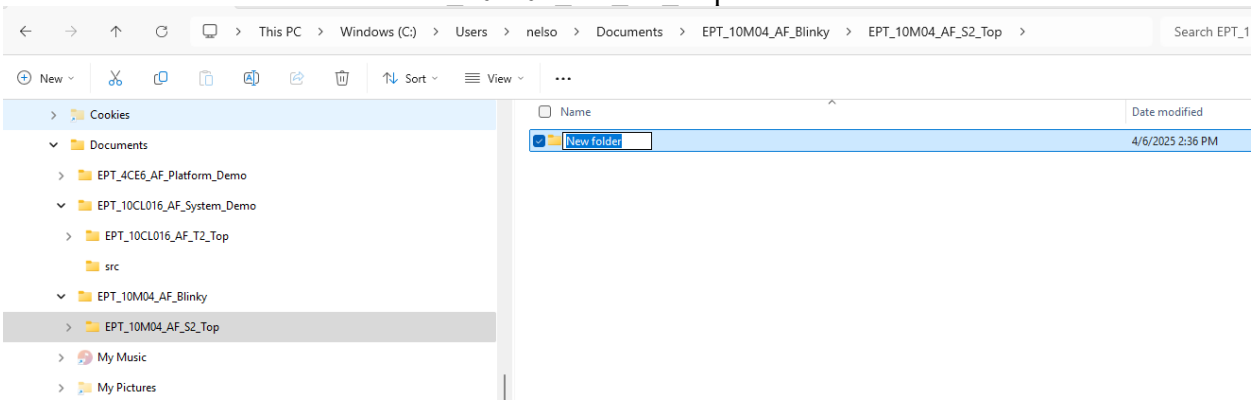
Navigate to the $/MAXPROLOGIC_FPGA_PROJECT_4.x_DVD /Projects_HDL/EPT_10M04_AF_Blinky/src source folder.

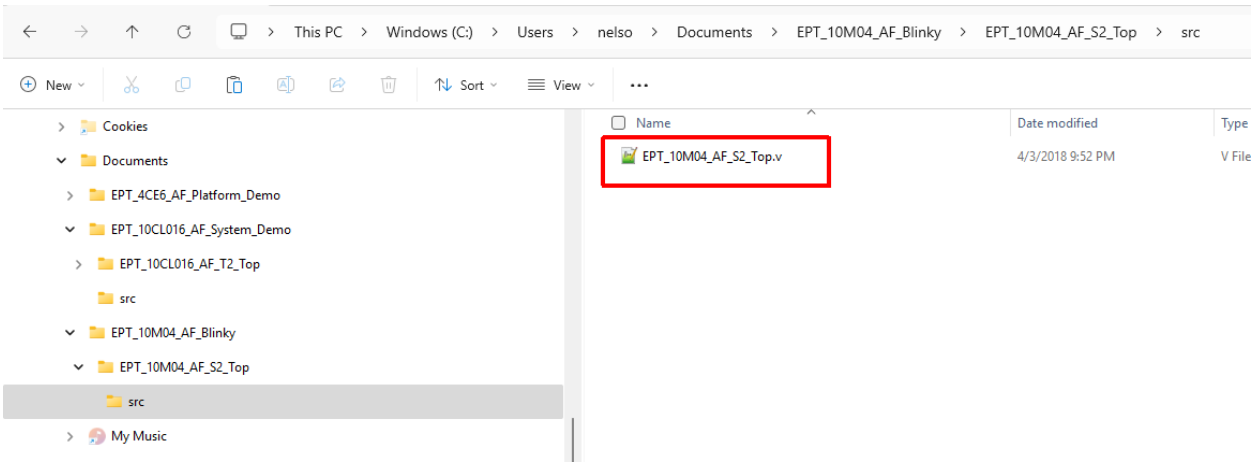


Select the “EPT_10M04_AF_S2_Top.v” file, right-click and select copy.

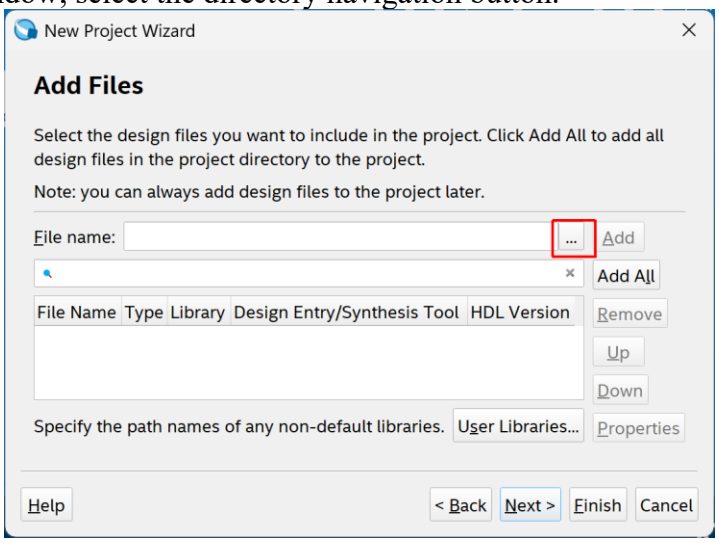


Then navigate to your local proejct window. C:\Users\<Your PC>\Documents. Create a new folder called “src”. Paste the “EPT_10M04_AF_S2_Top.v” file into the new “src” folder.





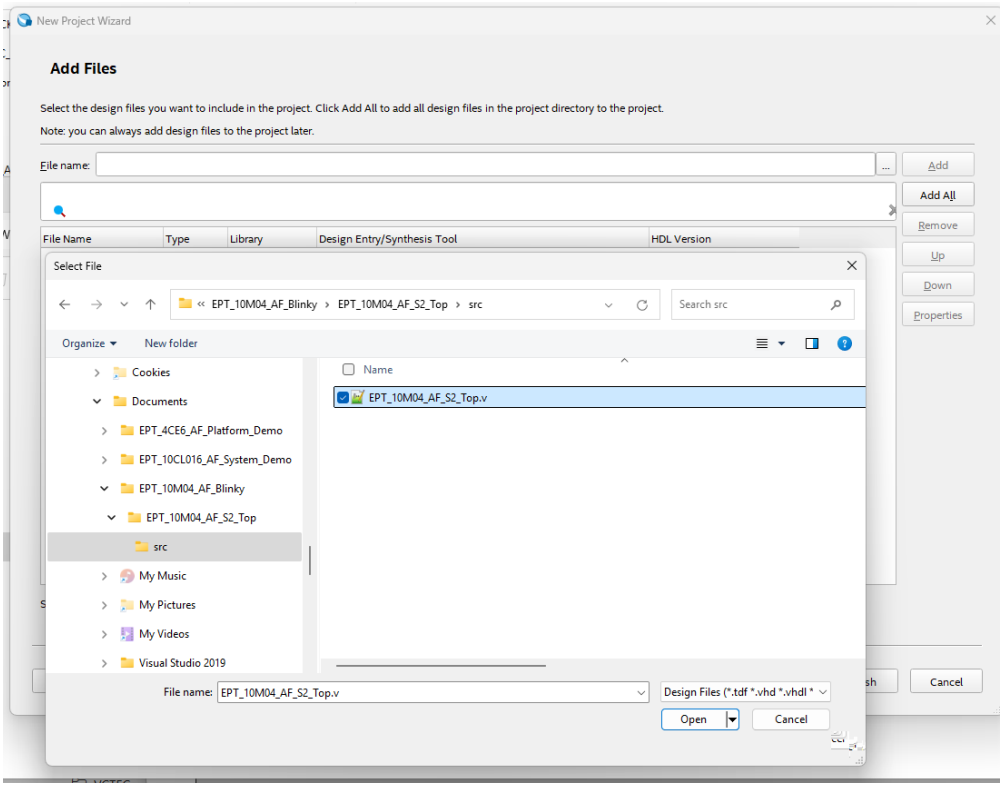
At the Add Files window, select the directory navigation button.

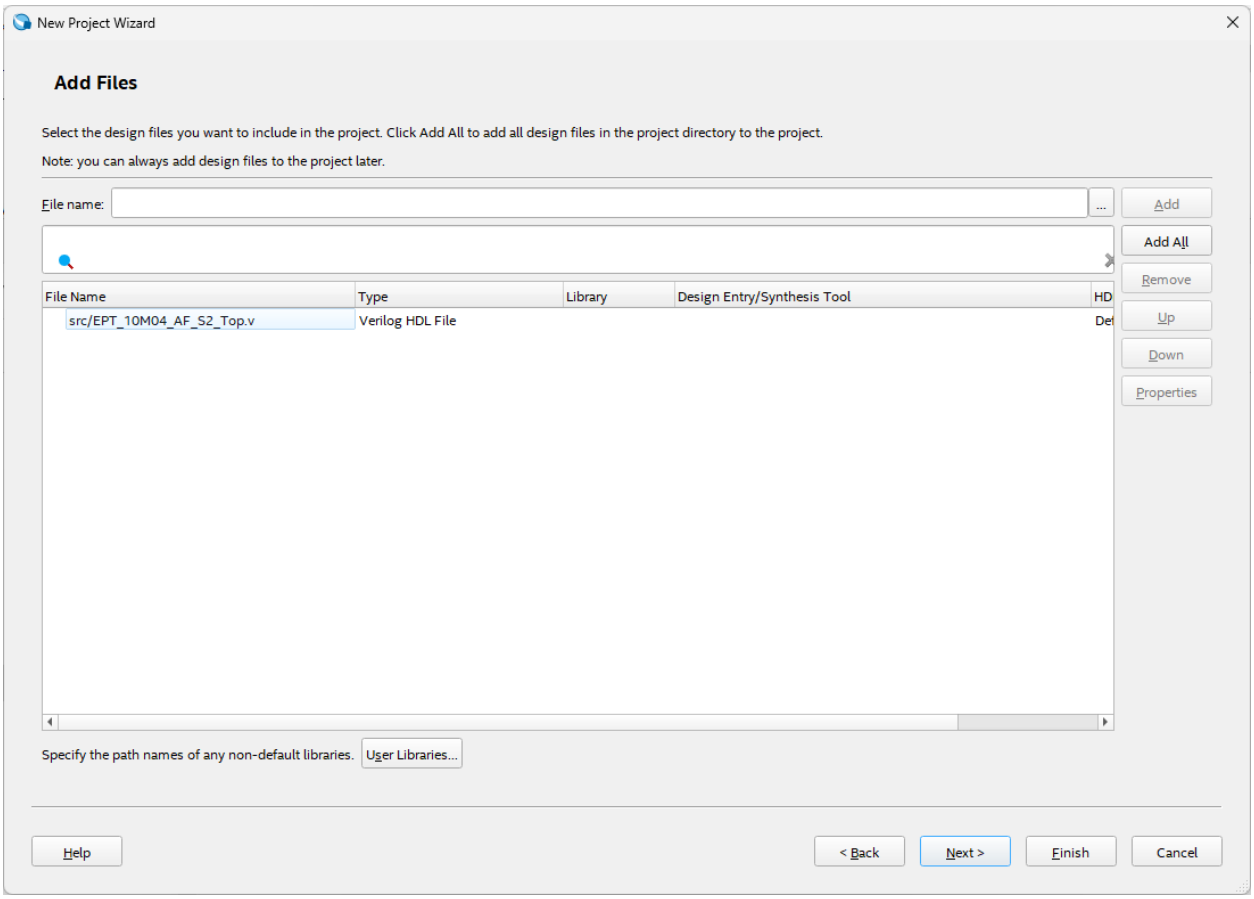


At the Add Files window: Browse to the $\MAXPROLOGIC_FPGA_PROJECT_4.x_DVD \Projects_HDL\EPT_10M04_AF_Blinky \src folder of the EPT USB-FPGA Development System CD. Copy the files from the \src directory.

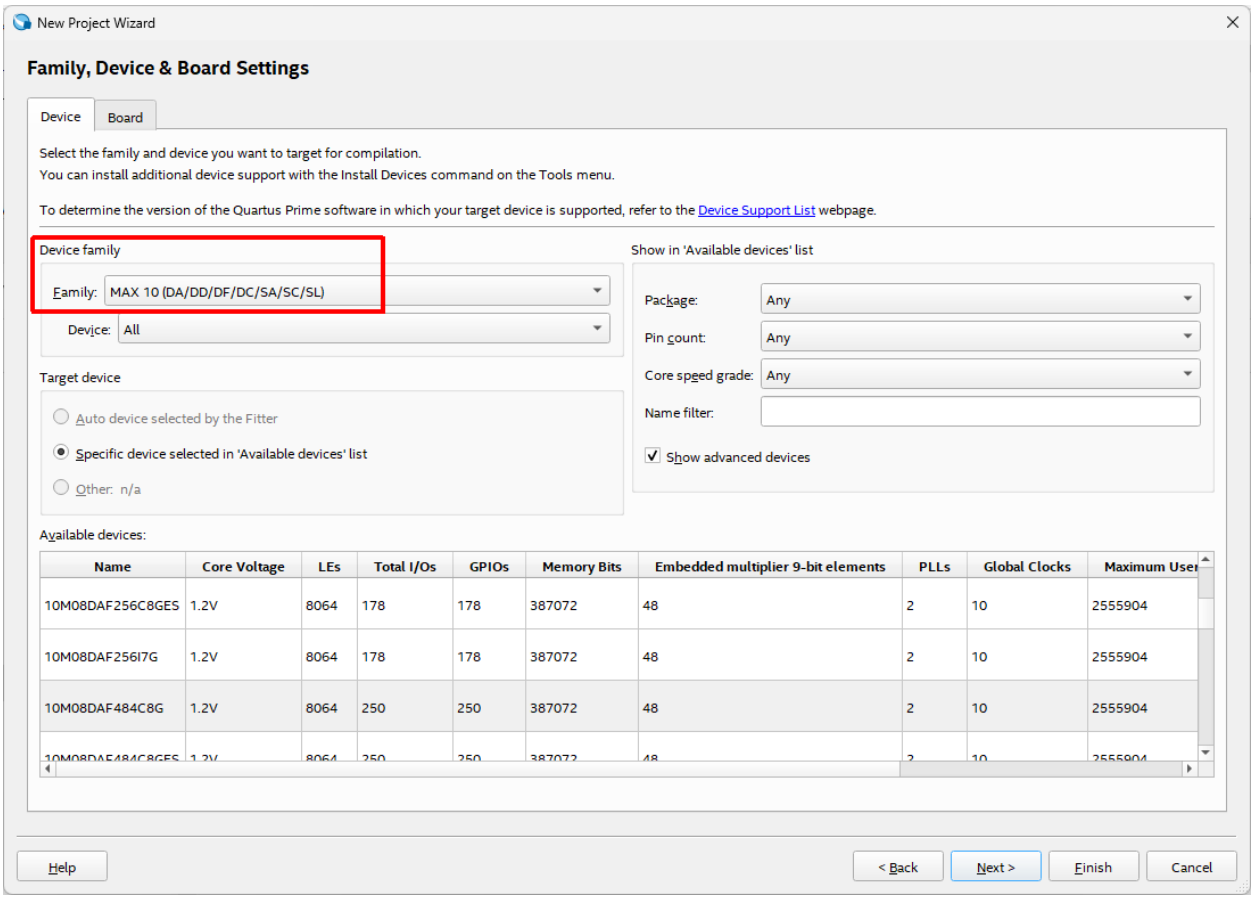
- EPT_10M04_AF_S2_Top.v

MaxProLogic Development System User Manual

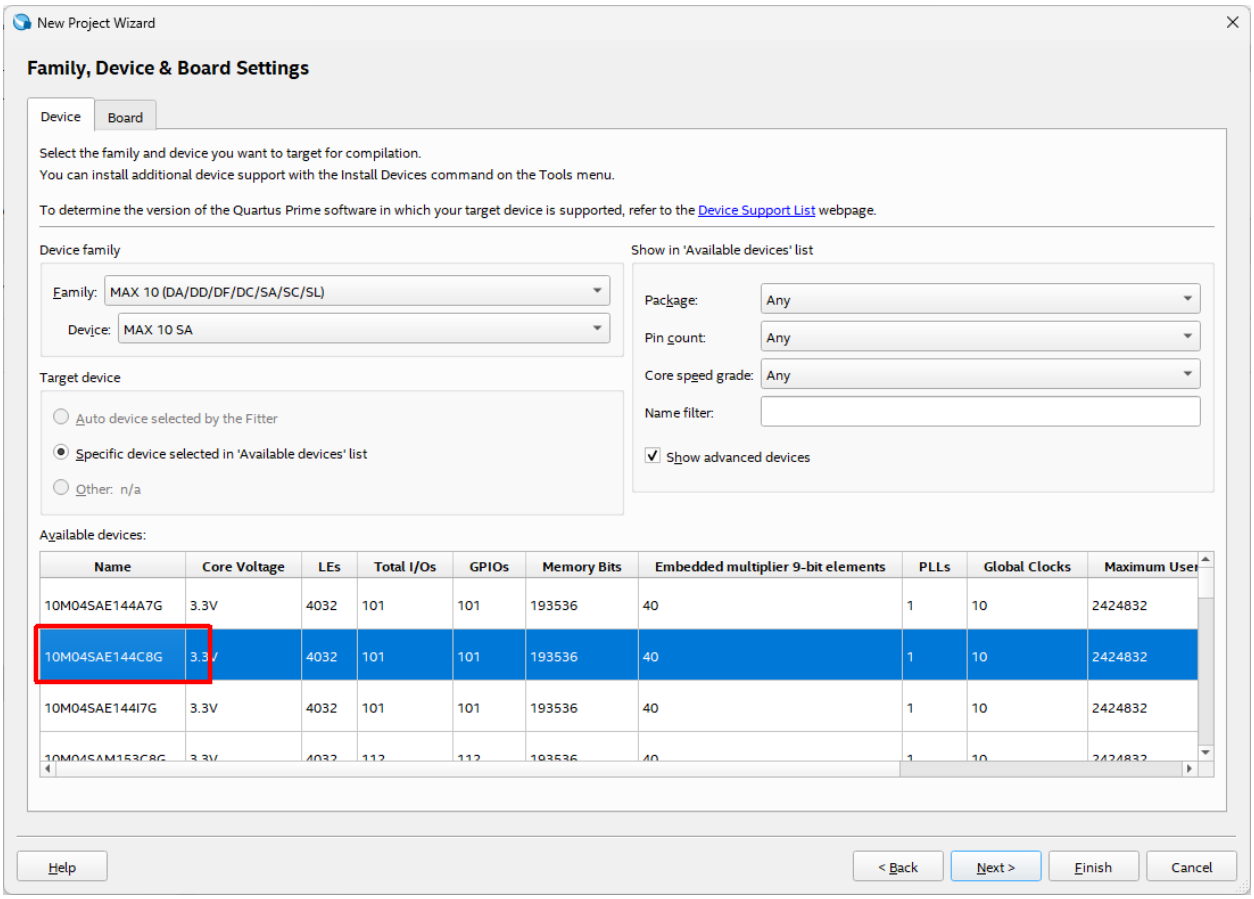




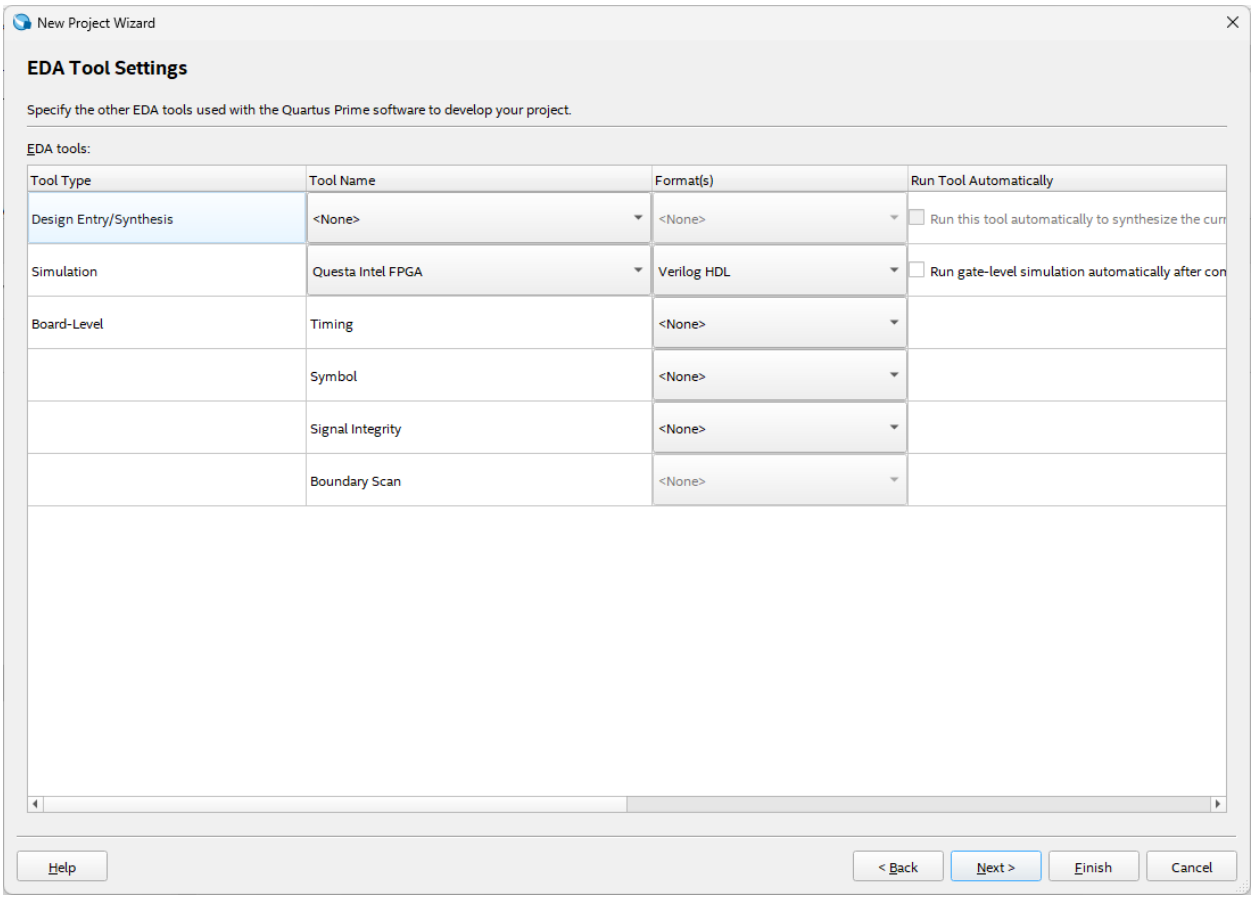
Select Next, at the Device Family group, select MAX 10 for Family.



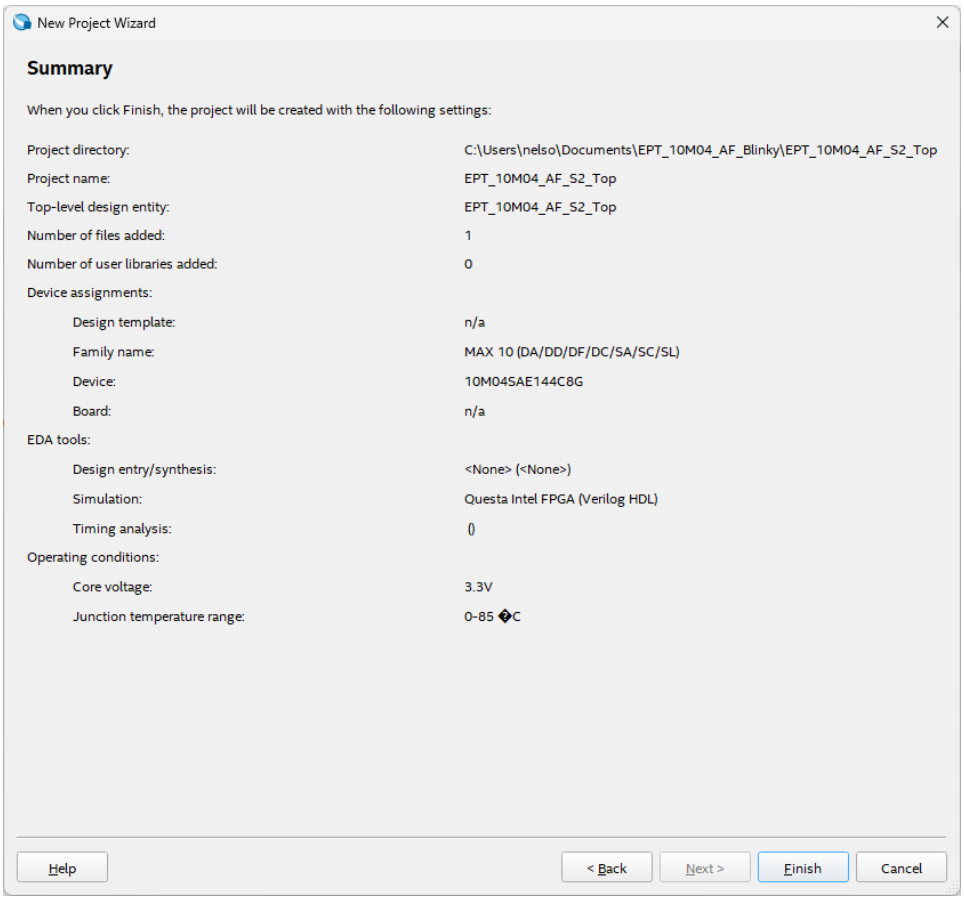
In the Available Devices group, browse down to 10M04SAE144C8 for Name.



Click the “Next” button.



Select Next, leave defaults for the EDA Tool Settings.



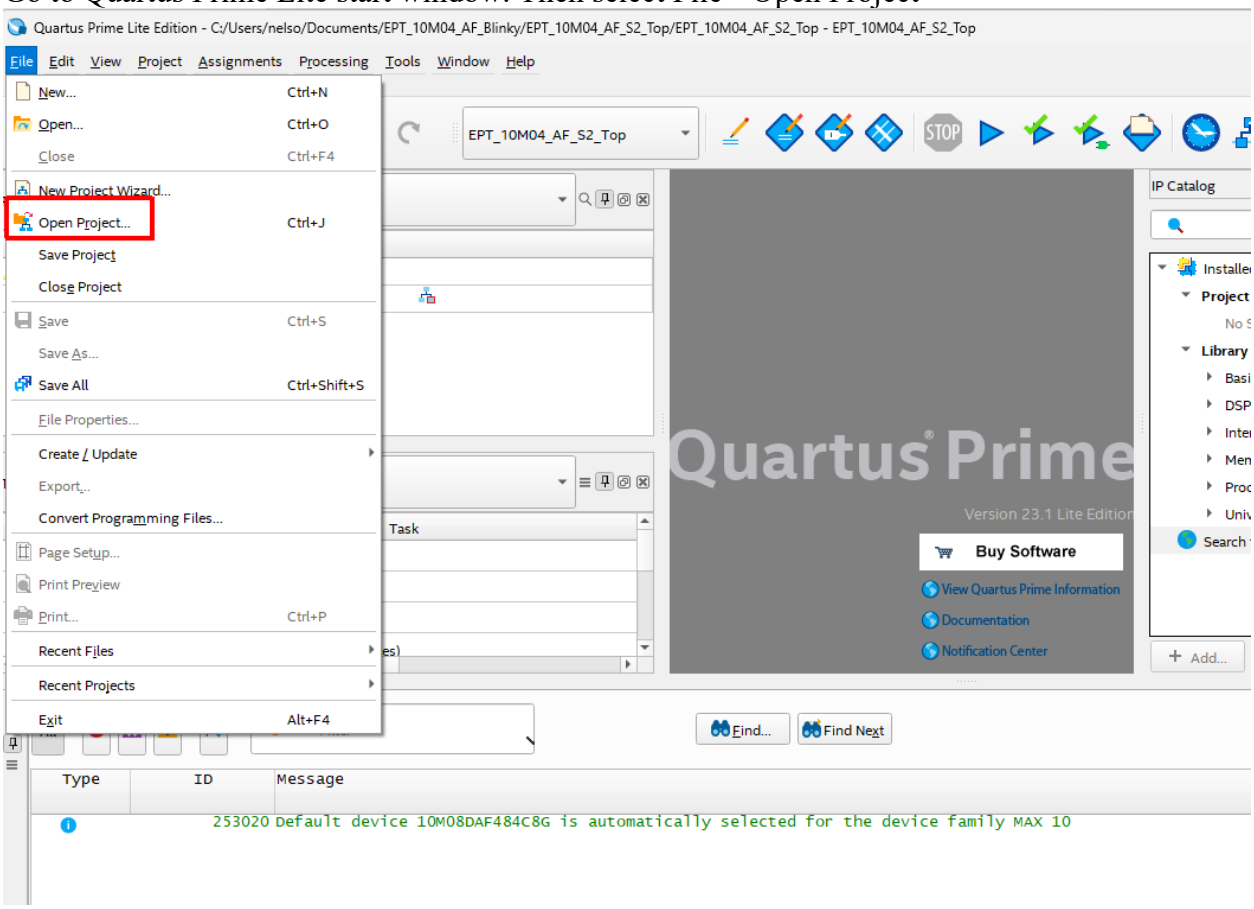
Select Next, then select Finish. You are done with the project level selections.

Next, we will select the pins and synthesize the project.

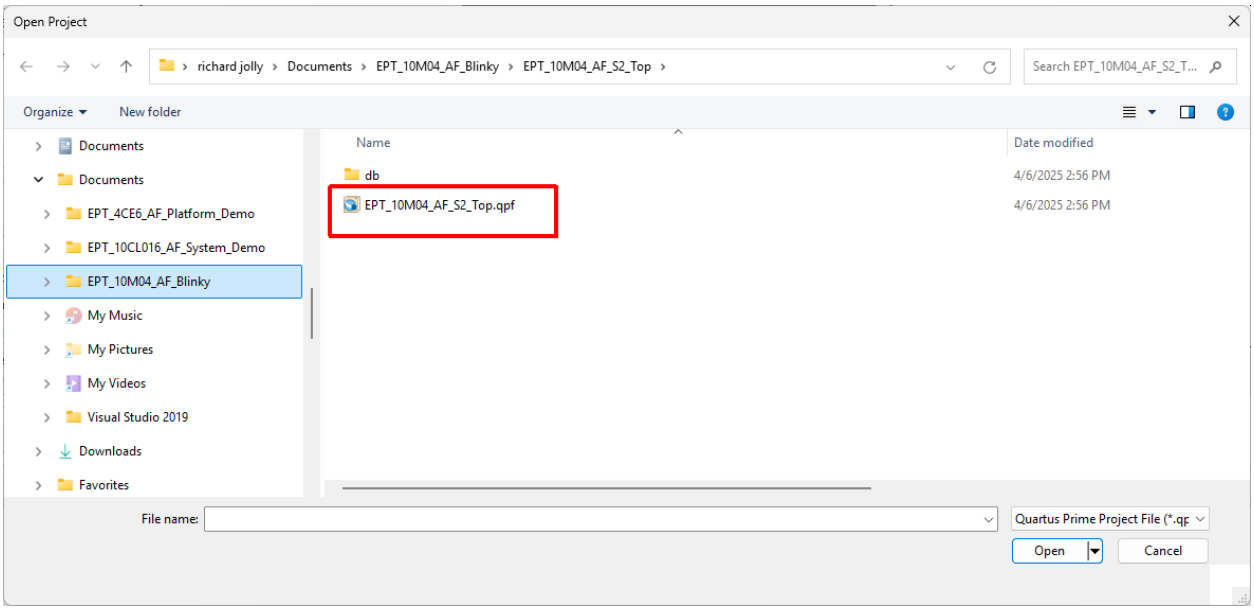
6.1.1 Selecting Pins and Synthesizing

With the project created, we need to assign pins to the project. The signals defined in the top level file (in this case: EPT_10M04_AF_S2_Top.v) will connect directly to pins on the FPGA.

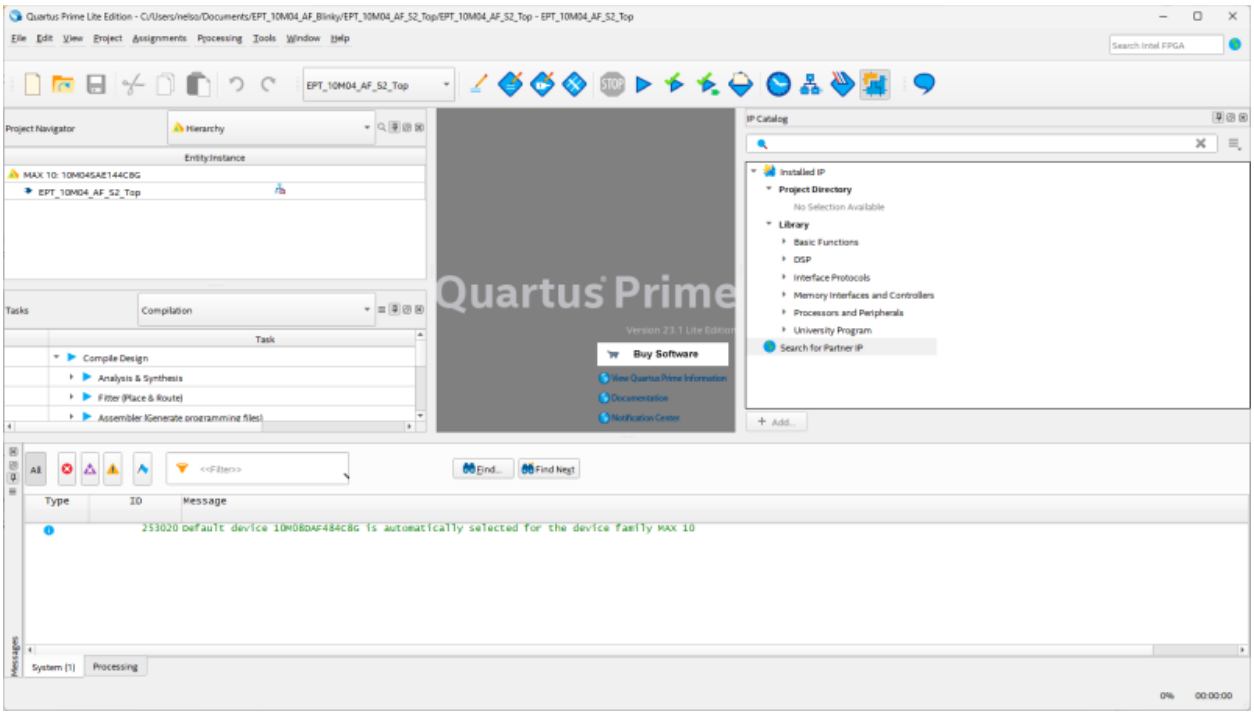
The Pin Planner Tool from Quartus Prime Lite will add the pins and check to verify that our pin selections do not violate any restrictions of the device. In the case of this example we will import pin assignments that created at an earlier time.
Go to Quartus Prime Lite start window. Then select File->Open Project



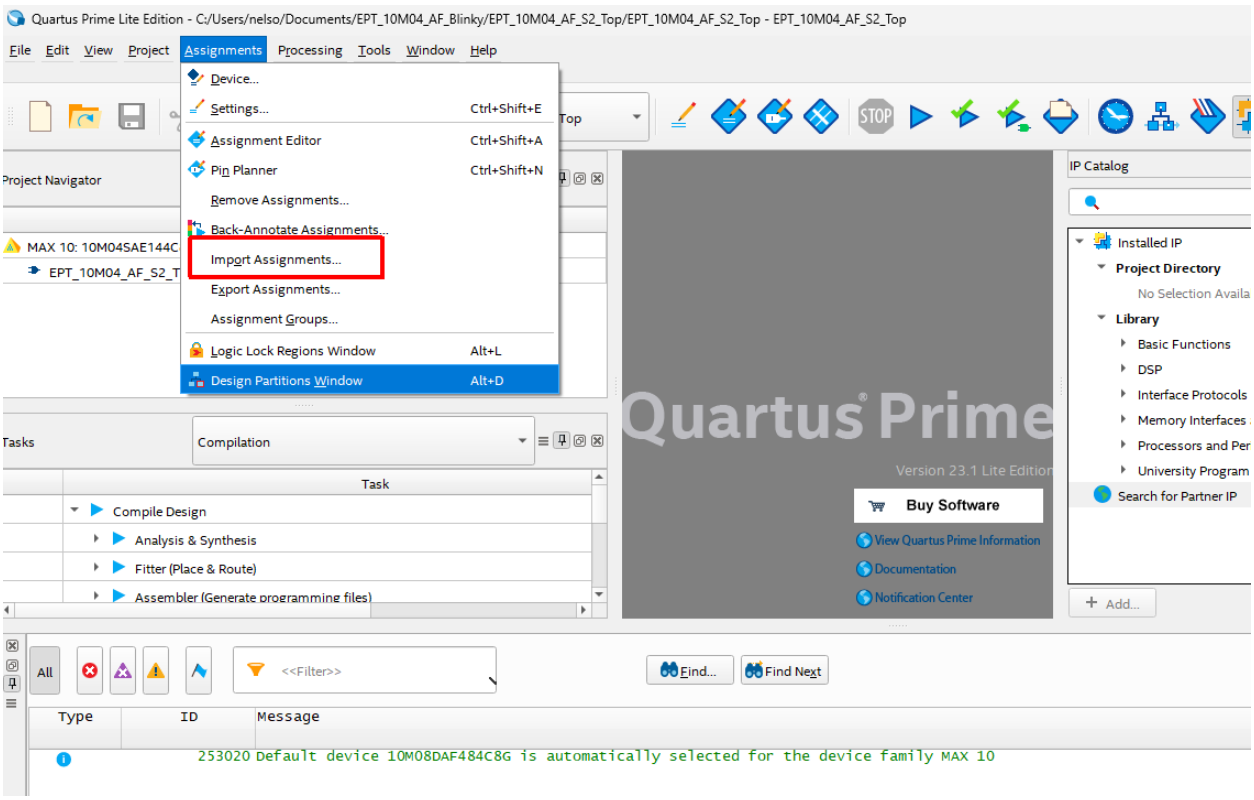
Navigate to the project created in previous section.



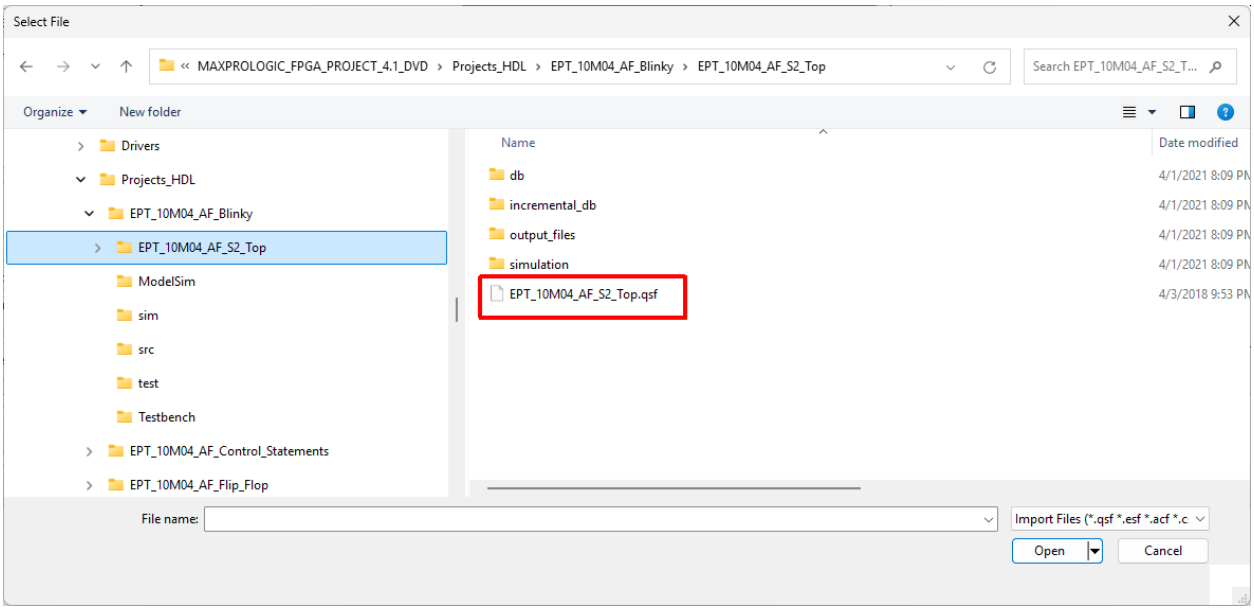
Select “Open”. The Quartus Window will be set up with the selected project.



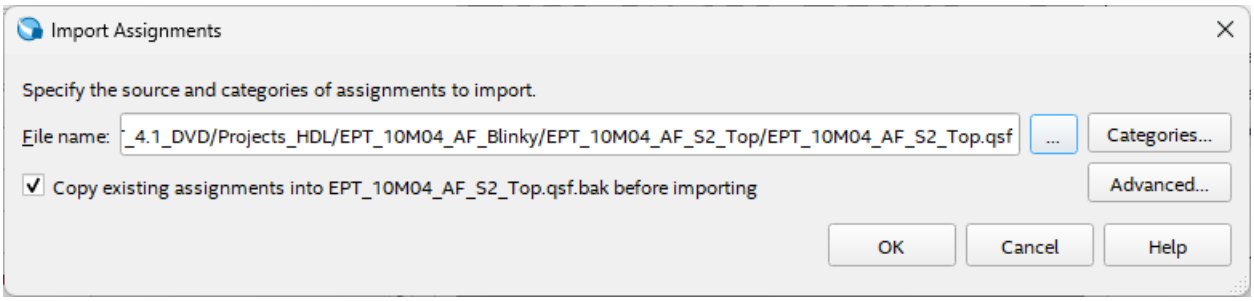
Under Assignments, Select Import Assignments.



At the Import Assignment dialog box, Browse to the $\MAXPROLOGIC_FPGA_PROJECT_4.x_DVD \Projects_HDL\EPT_10M04_AF_Blinky \ EPT_10M04_AF_S2_Top folder of the EPT FPGA Development System DVD. Select the "EPT_10M04_AF_S2_Top.qsf" file.

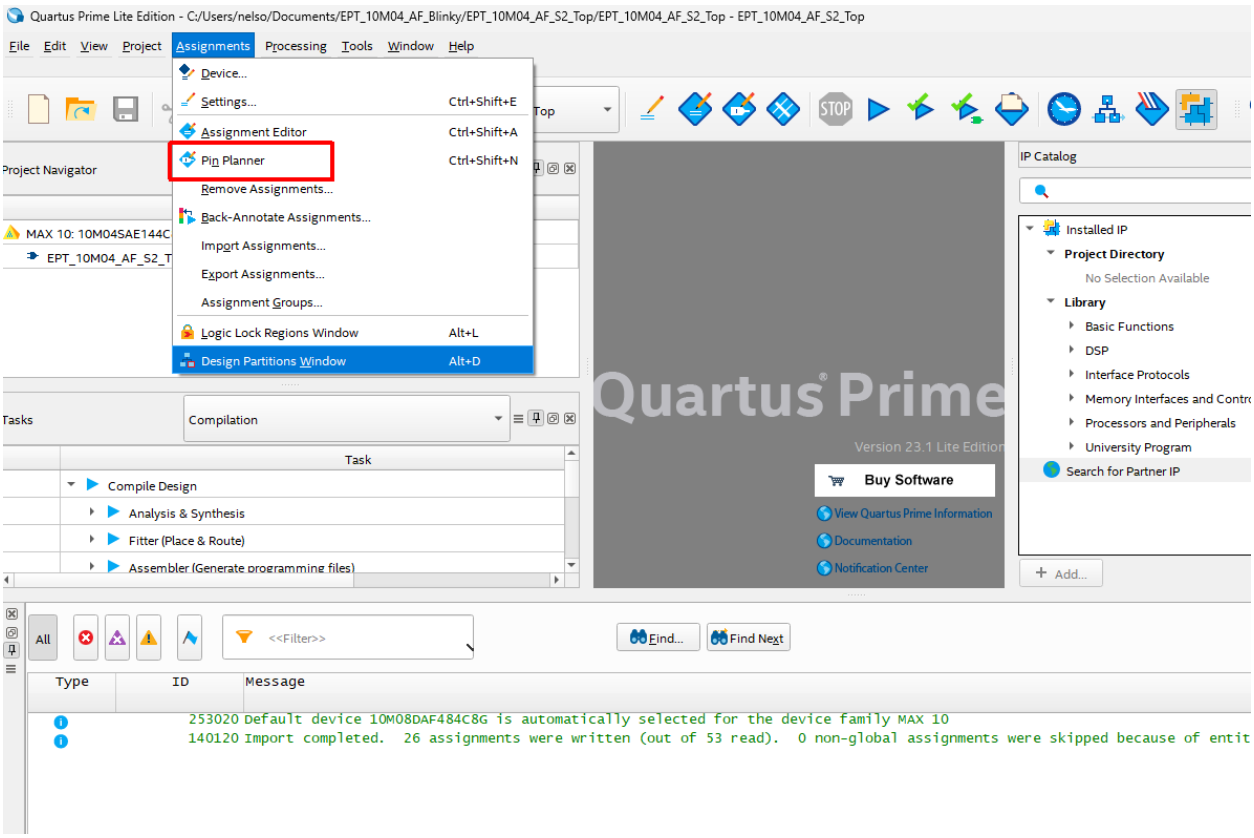


Select “Open”.

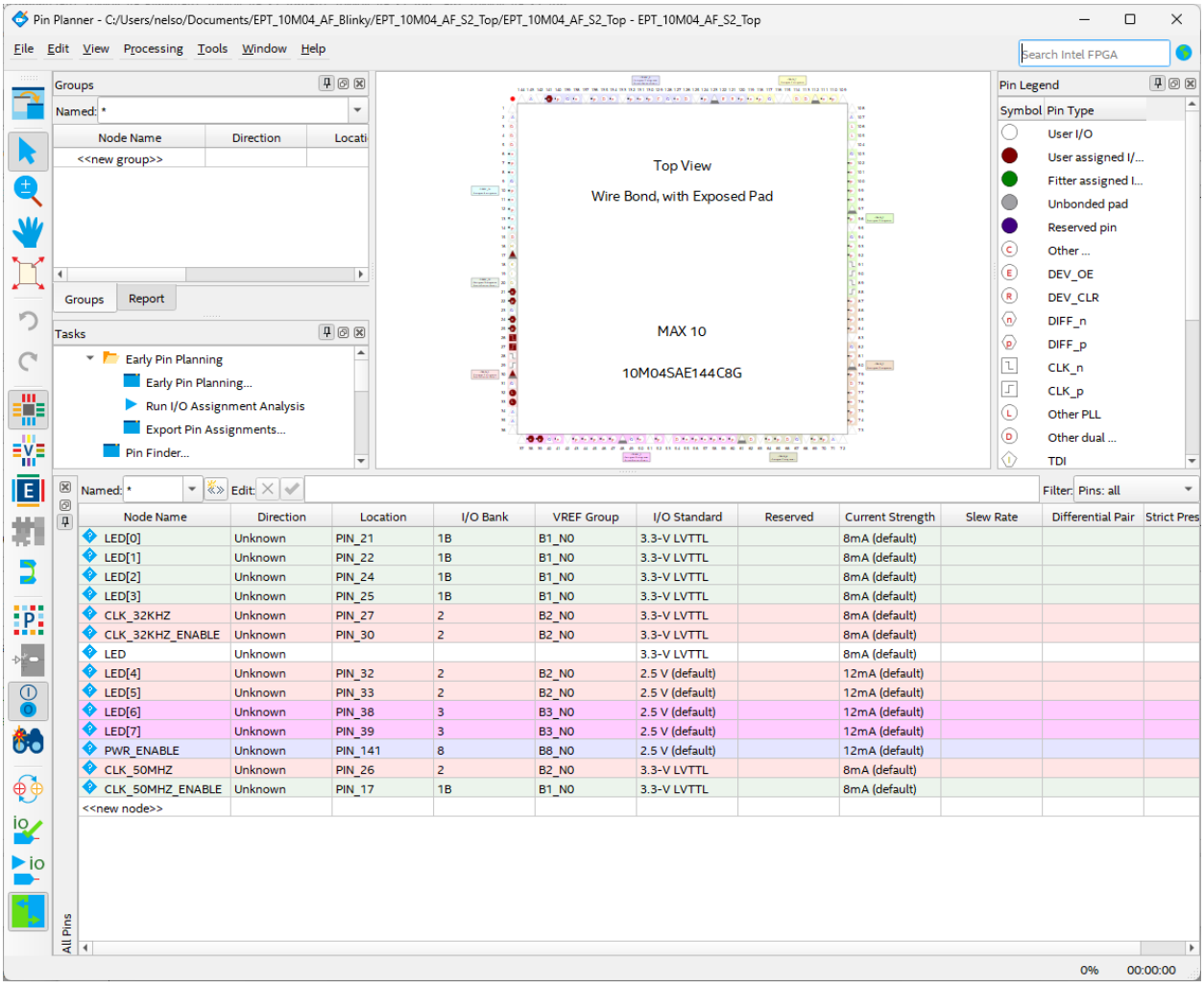


Click Ok. Under Assignments, Select Pin Planner. Verify the pins have been imported correctly.

MaxProLogic Development System User Manual



The pin locations should not need to be changed for EPT FPGA Development System. However, if you need to change any pin location, just click on the “location” column for the particular node you wish to change. Then, select the new pin location from the drop down box.

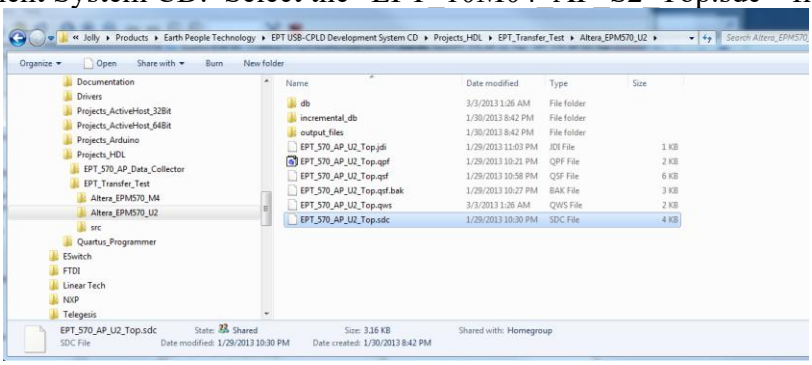


Exit the Pin Planner. Next, we need to add the Synopsys Design Constraint file. This file contains timing constraints which forces the built in tool called TimeQuest Timing Analyzer to analyze the path of the synthesized HDL code with setup and hold times of the internal registers.

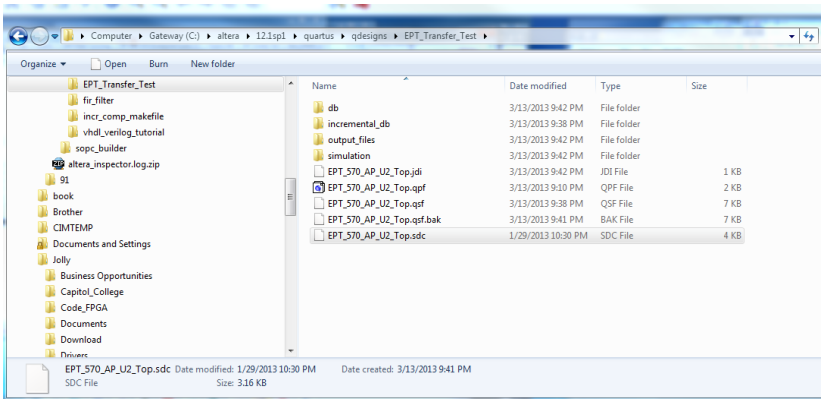
It takes note of any path that may be too long to appropriately meet the timing qualifications. For more information on TimeQuest Timing Analyzer, see

http://www.altera.com/literature/hb/qts/qts_qii53018.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=TimeQuest Timing Analyzer

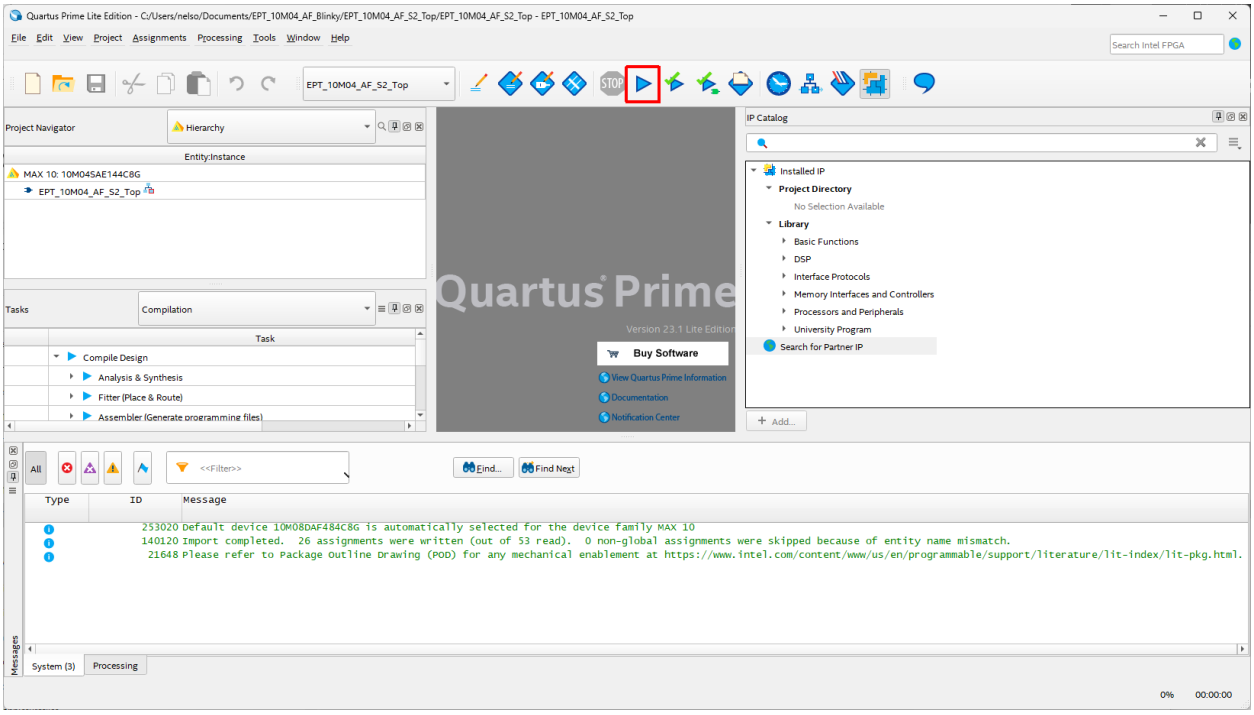
Browse to the $\MAXPROLOGIC_FPGA_PROJECT_4.x_DVD \Projects_HDL\EPT_10M04_AF_Blinky \ EPT_10M04_AF_S2_Top folder of the EPT USB-FPGA Development System CD. Select the "EPT_10M04_AF_S2_Top.sdc" file.



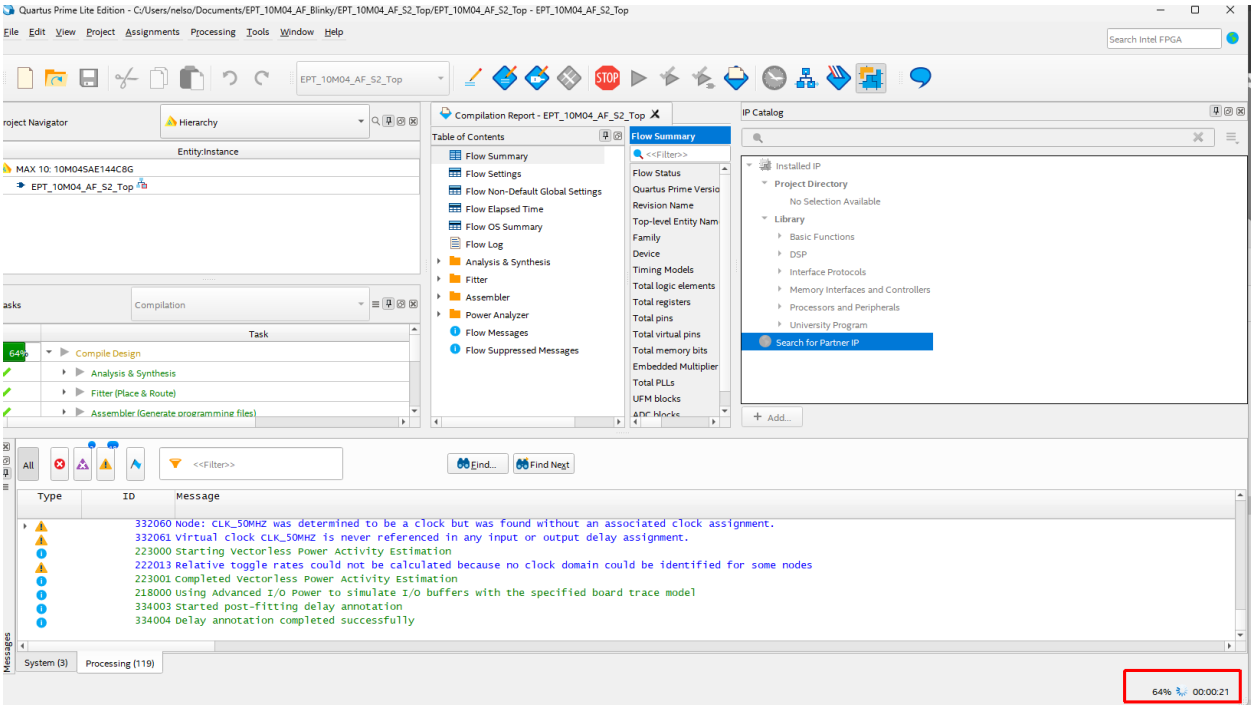
Copy the file and browse to c:\altera\xxx\quartus\qdesigns\EPT_10M04_AF_Blinky directory. Paste the file.



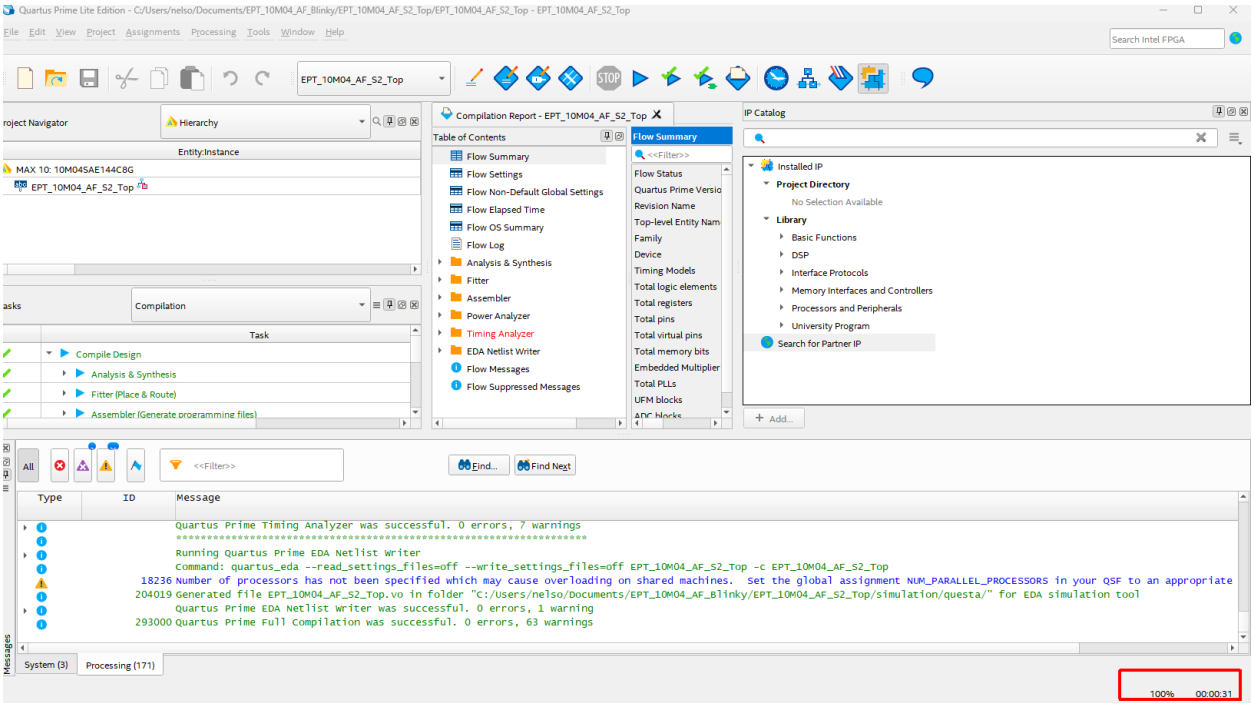
Select the Start Compilation button.



During compilation, the window will show this screen:



After successful completion, the screen should look like the following:



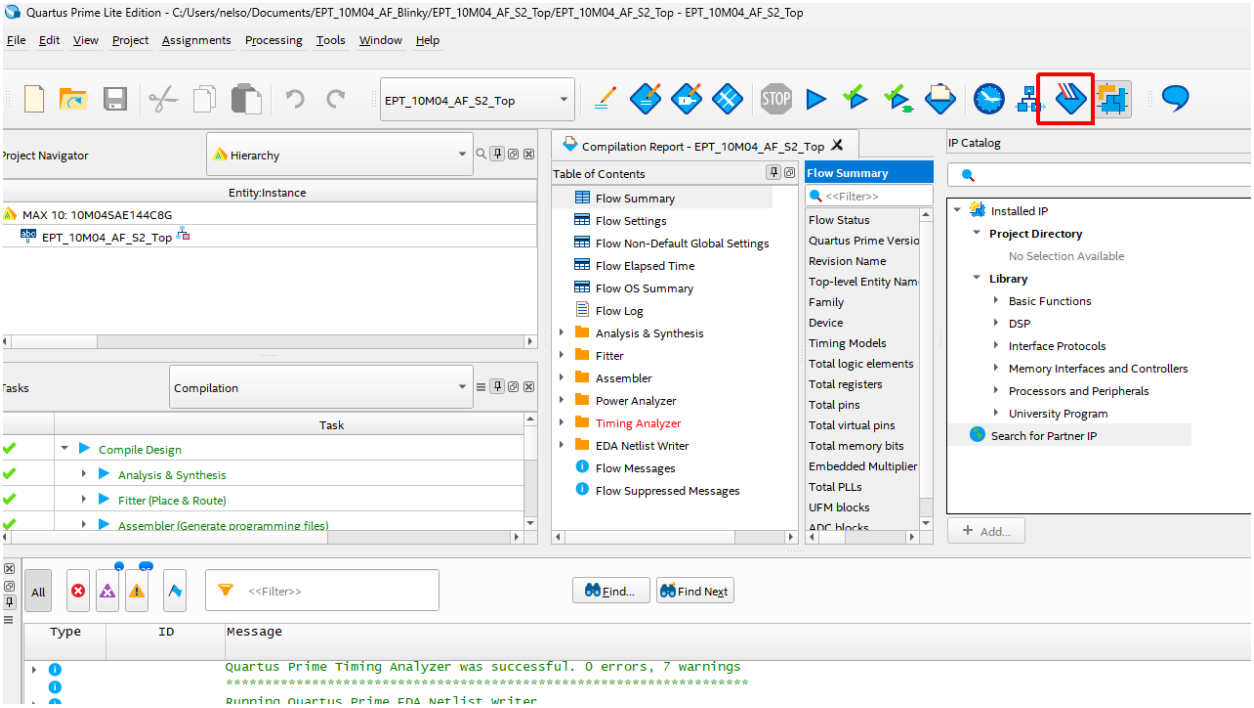
At this point the project has been successfully compiled, synthesized and a programming file has been produce. See the next section on how to program the FPGA.

7 Programming the MaxProLogic

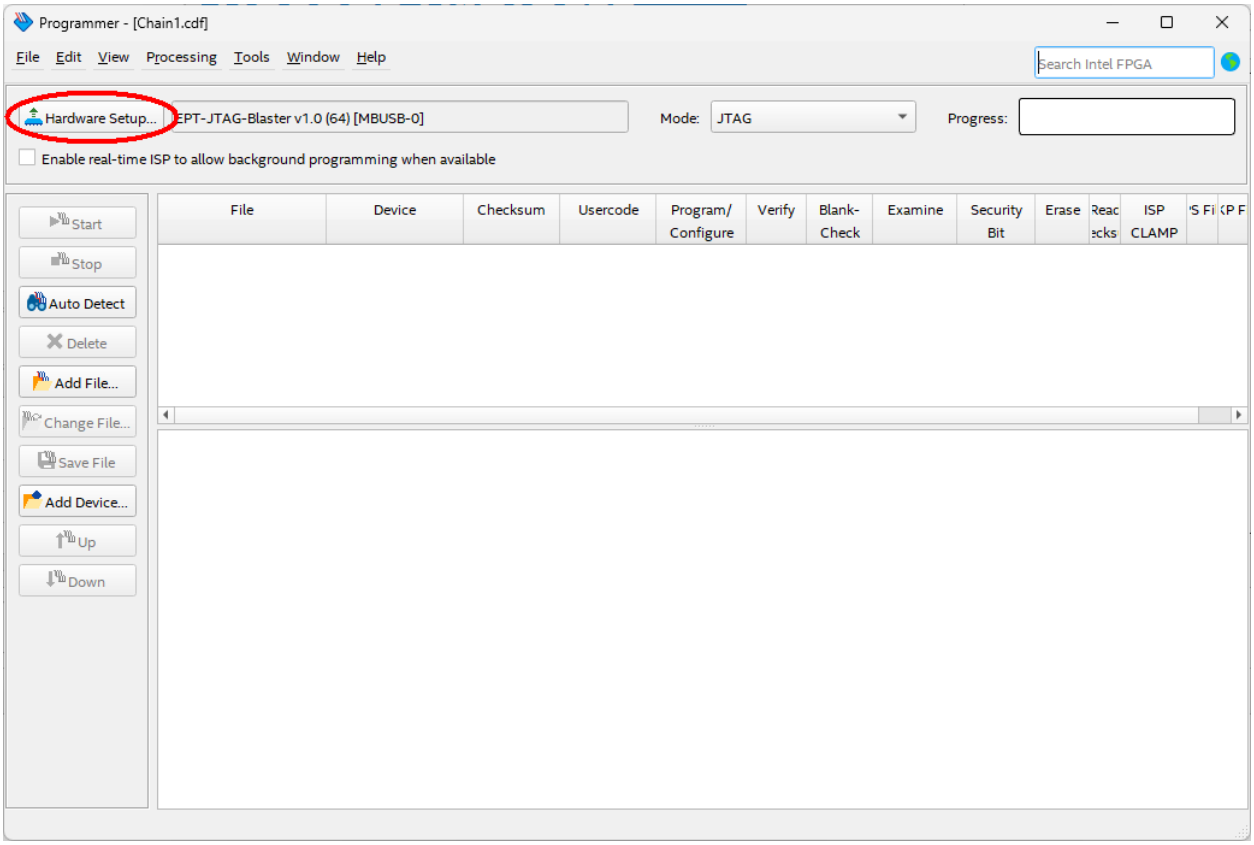
- **Please Note: The MaxProLogic Does Not Contain On Board Programmer. JTAG Programming and Configuration Flash Programming must be performed by external Programmer Purchased Separately.**

Configuring the FPGA is quick and easy. All that is required is a standard USB-C cable and a compatible Blaster Programmer. Connect the MaxProLogic to the PC, open up Quartus Prime Lite, open the programmer tool, and click the Start button. To program the MAX10 Configuration Flash, connect the Blaster and ensure the JTAG Driver is loaded for Quartus Prime Lite.

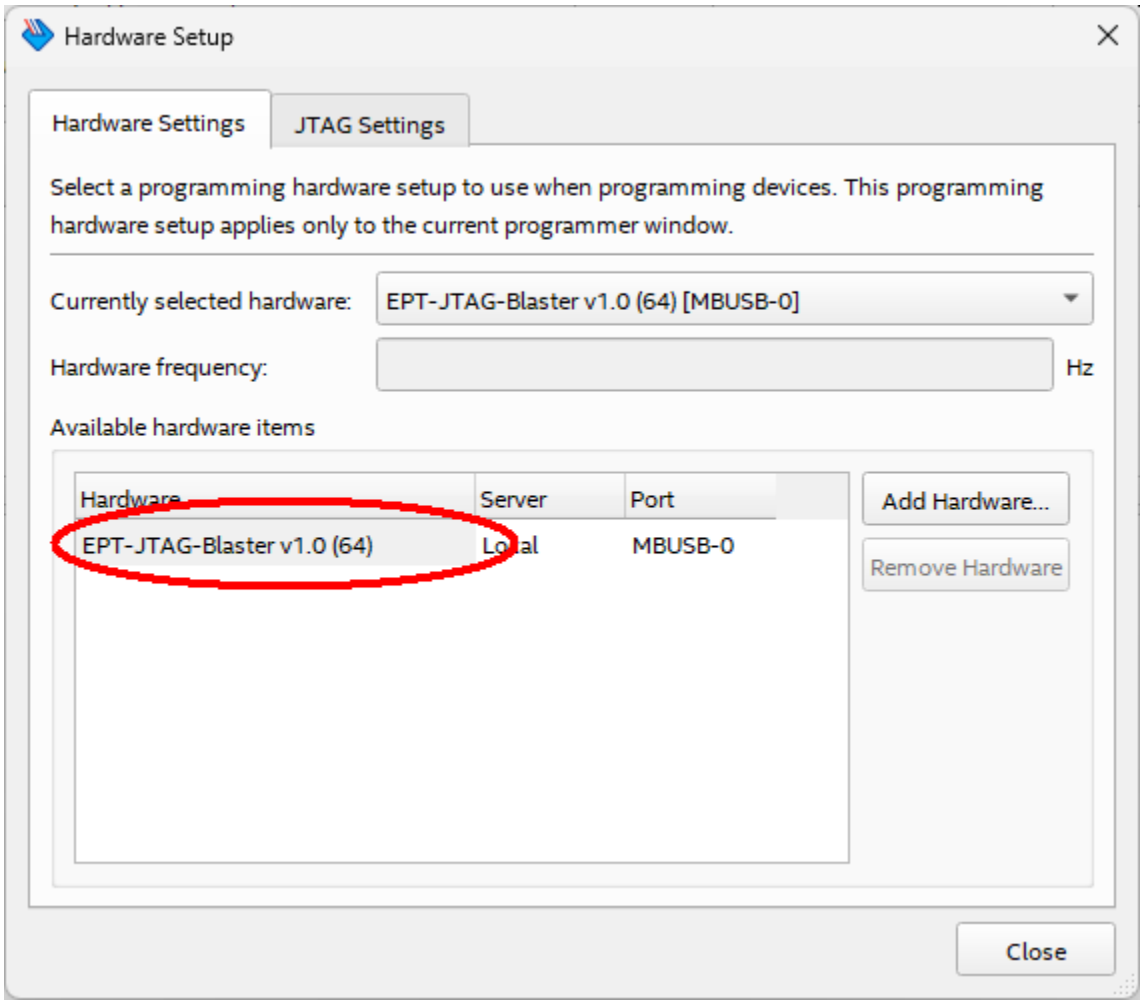
If the project created in the previous sections is not open, open it. Click on the Programmer button.



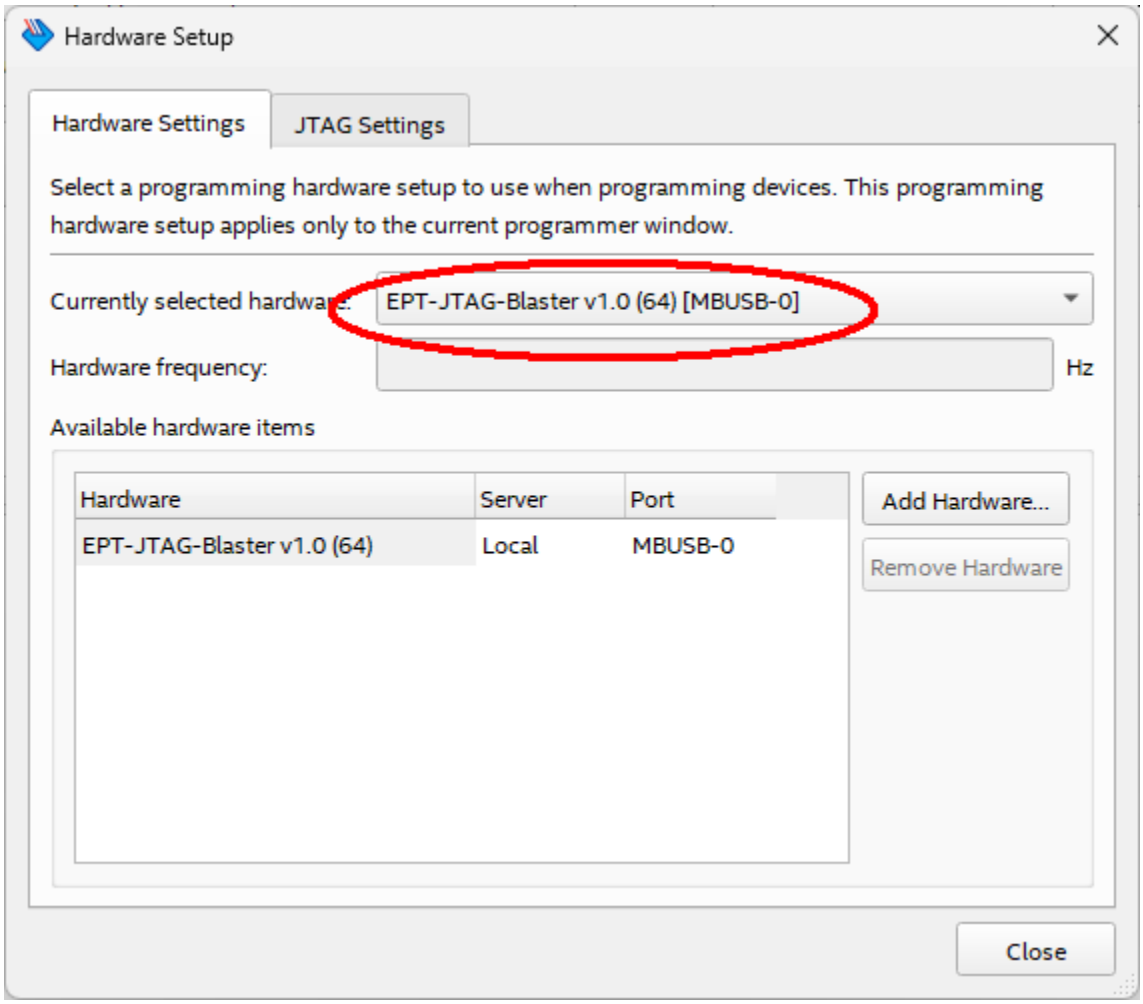
The Programmer Window will open up with the programming file selected. Click on the Hardware Setup button in the upper left corner.



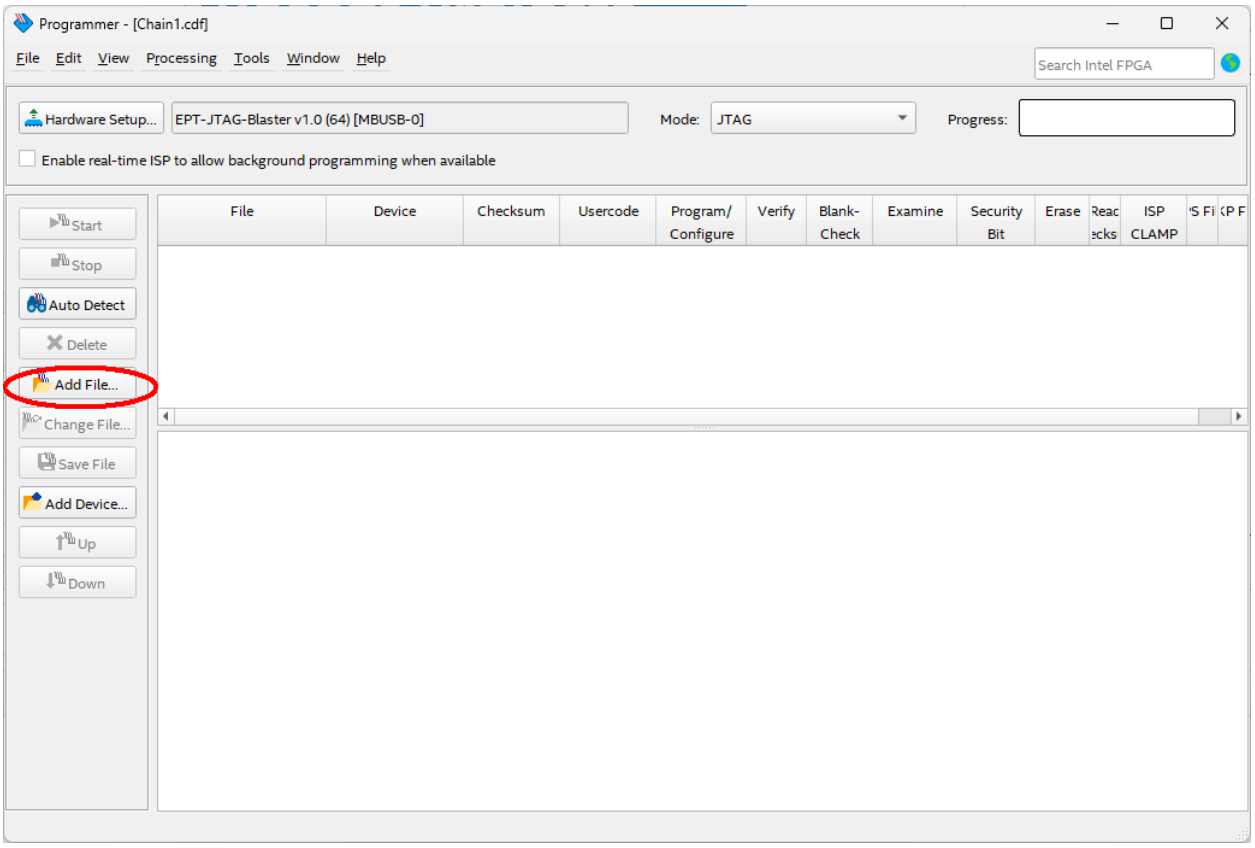
The Hardware Setup Window will open. In the “Available hardware items”, double click on “EPT-JTAG-Blaster v1.0b (64)”.

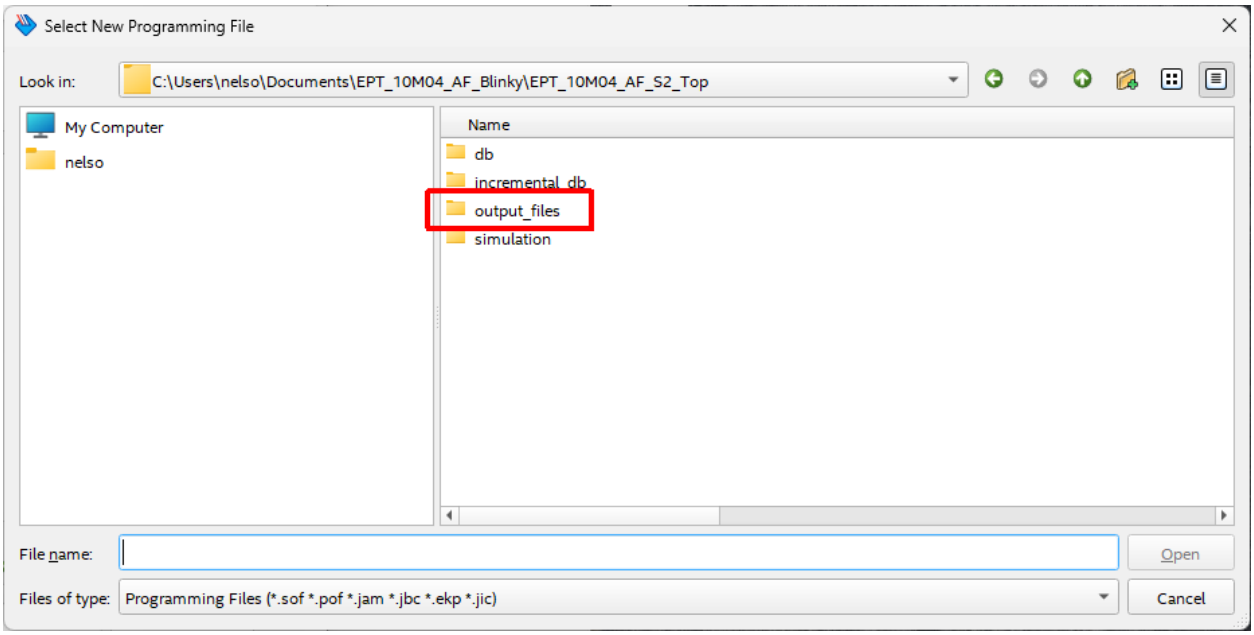


If you successfully double clicked, the “Currently selected hardware:” dropdown box will show the “EPT-JTAG-Blaster v1.0b (64)”.

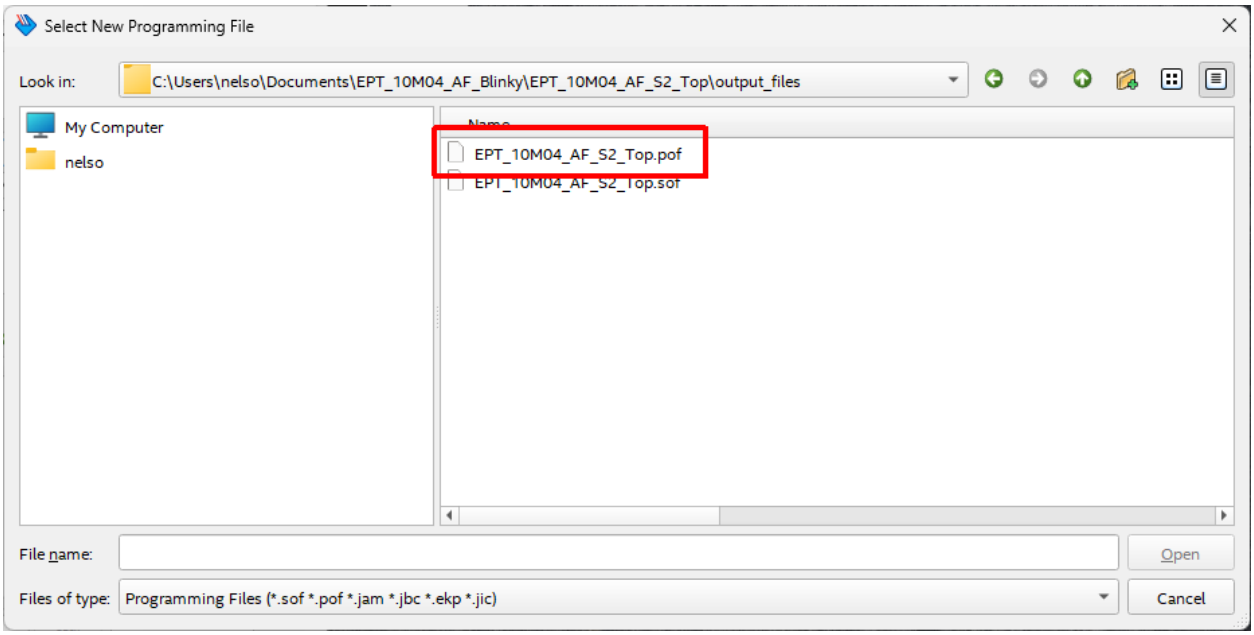


Click on the “Add File” button

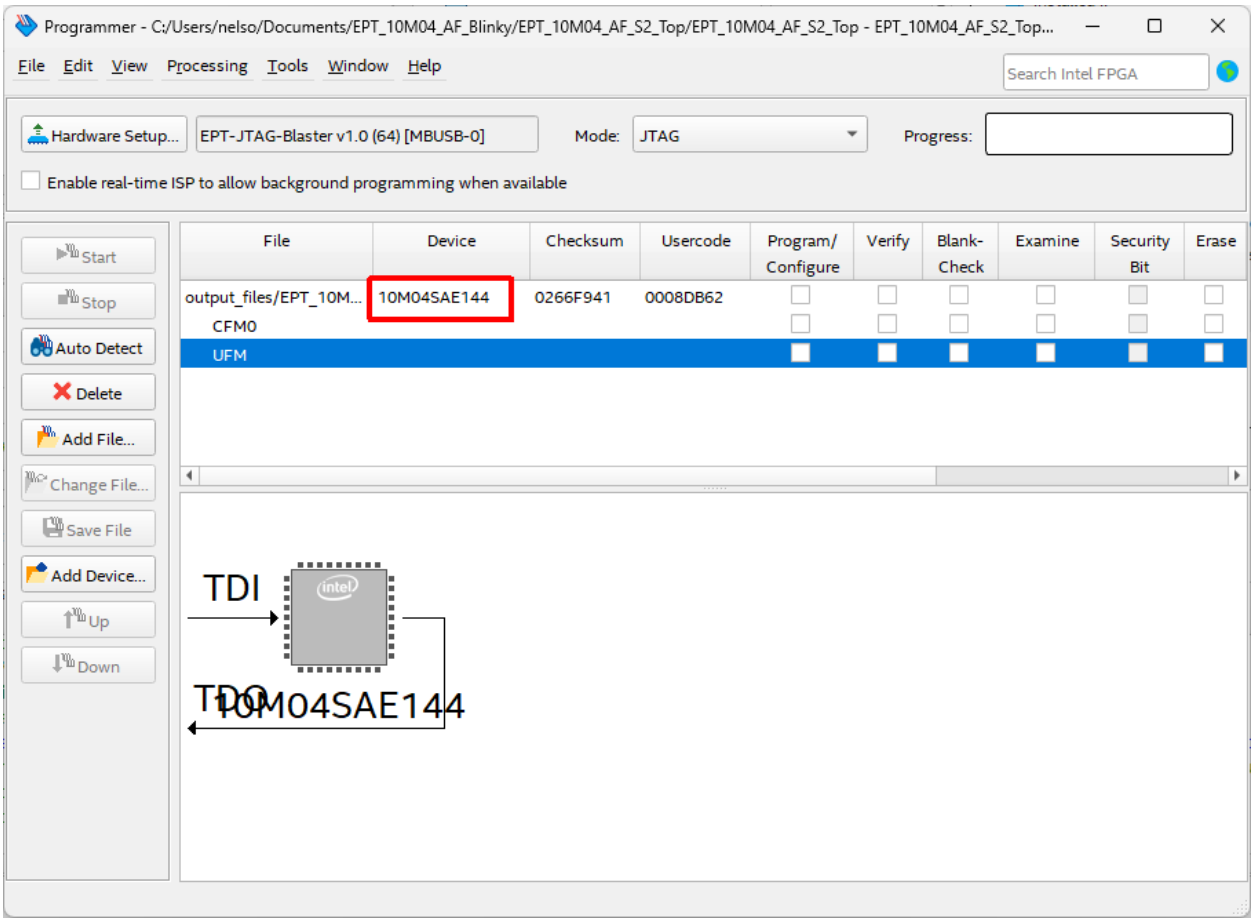

Programmer - [Chain1.cdf]
File Edit View Processing Tools Window Help
Search Intel FPGA
Hardware Setup...
EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]
Mode: JTAG
Progress:
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File
Device
Checksum
Usercode
Program/ Configure
Verify
Blank-Check
Examine
Security Bit
Erase
ISP CLAMP



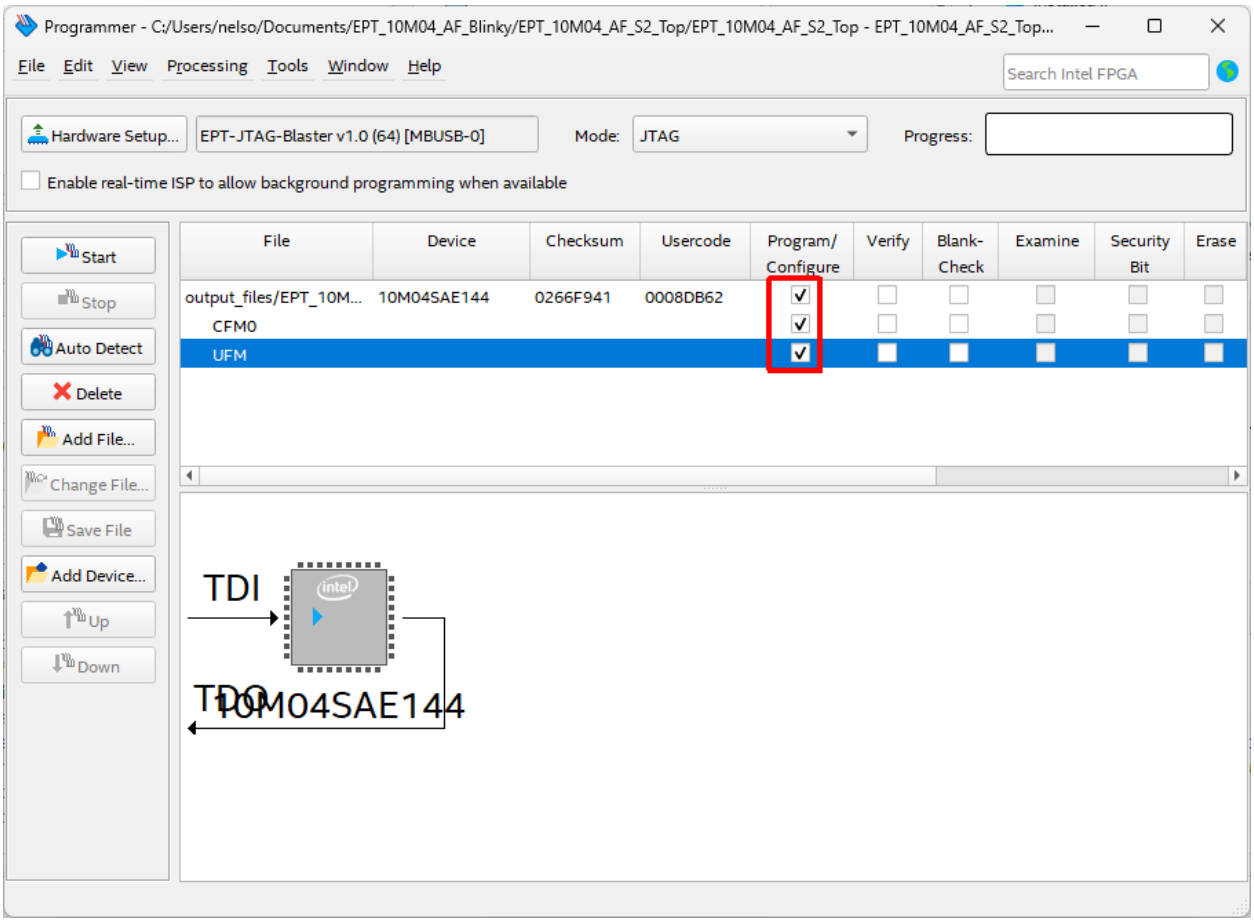
At the Browse window, double click on the output files folder.



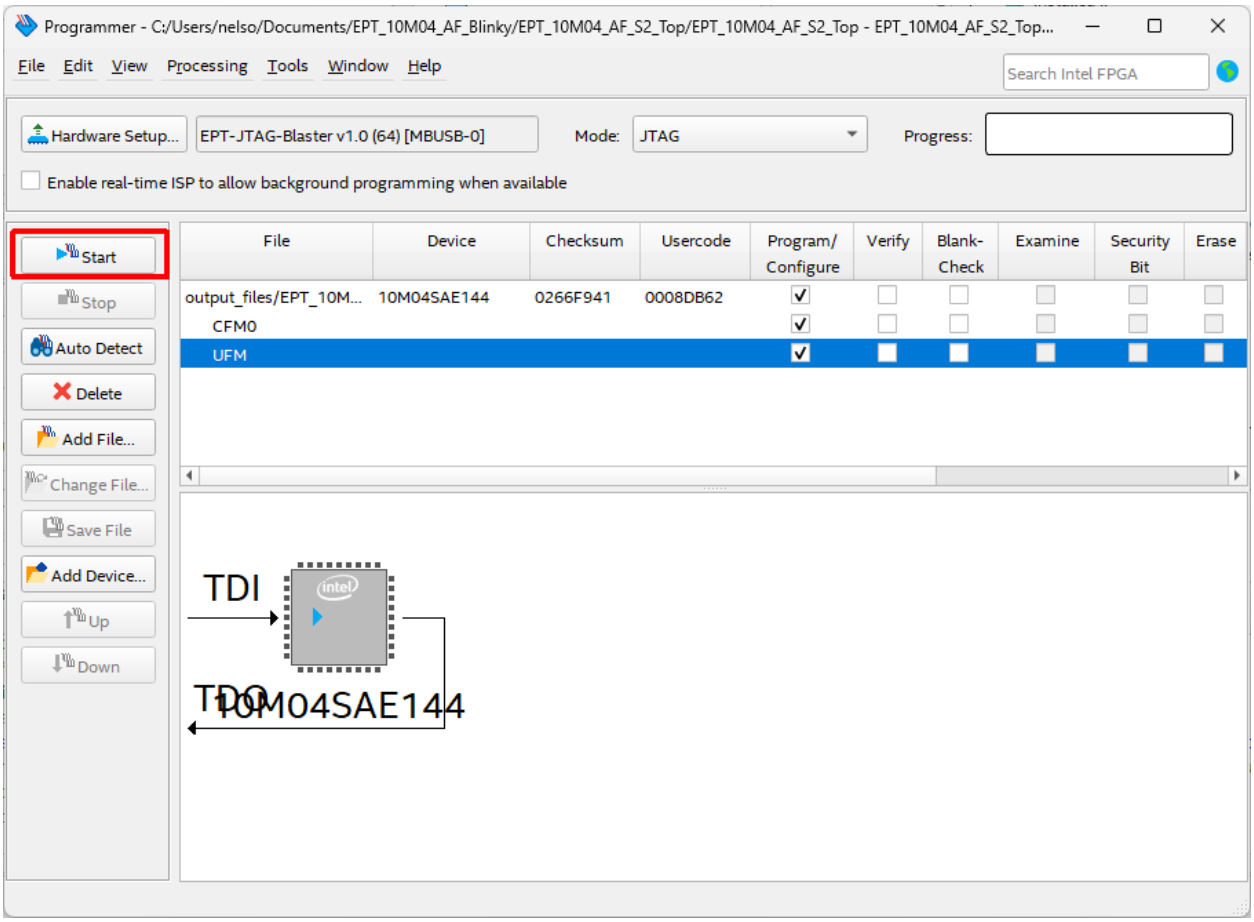
Double click on the “EPT_10M04_AF_S2_Top.pof” file. Click the Open button in the lower right corner.



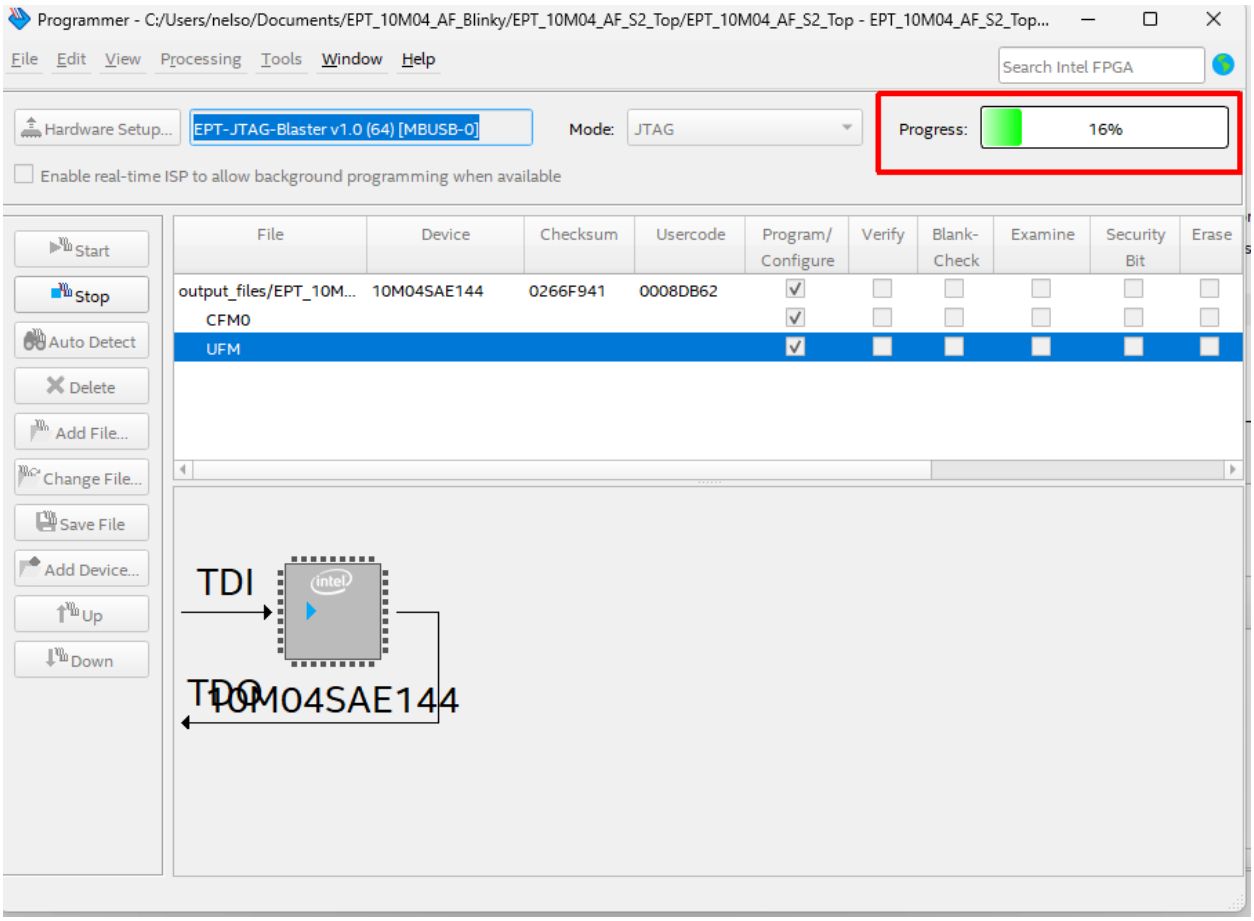
Next, selet the checkbox under the “Program/Configure” of the Programmer Tool.



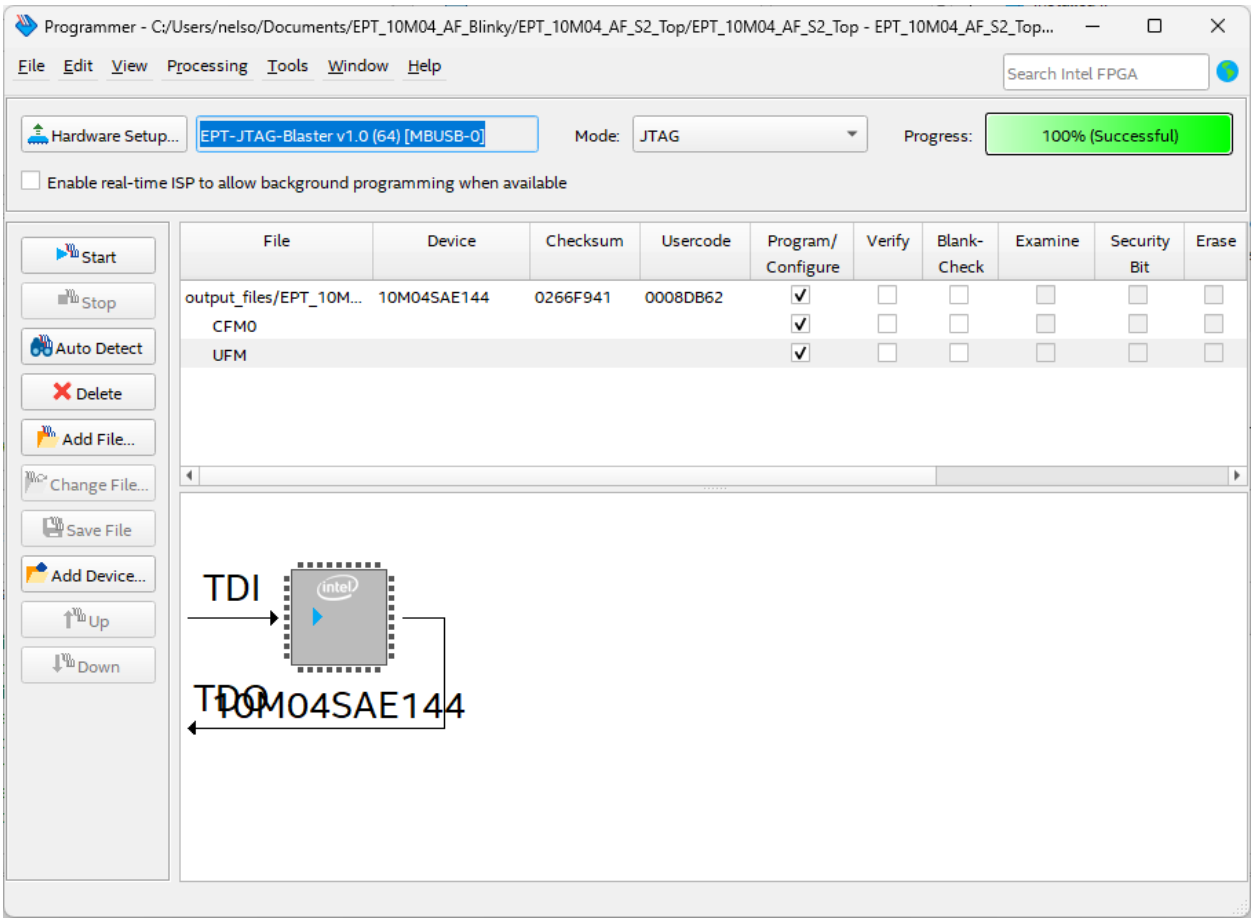
Click on the Start button to to start programming the FPGA.



The Progress bar will indicate the progress of programming.

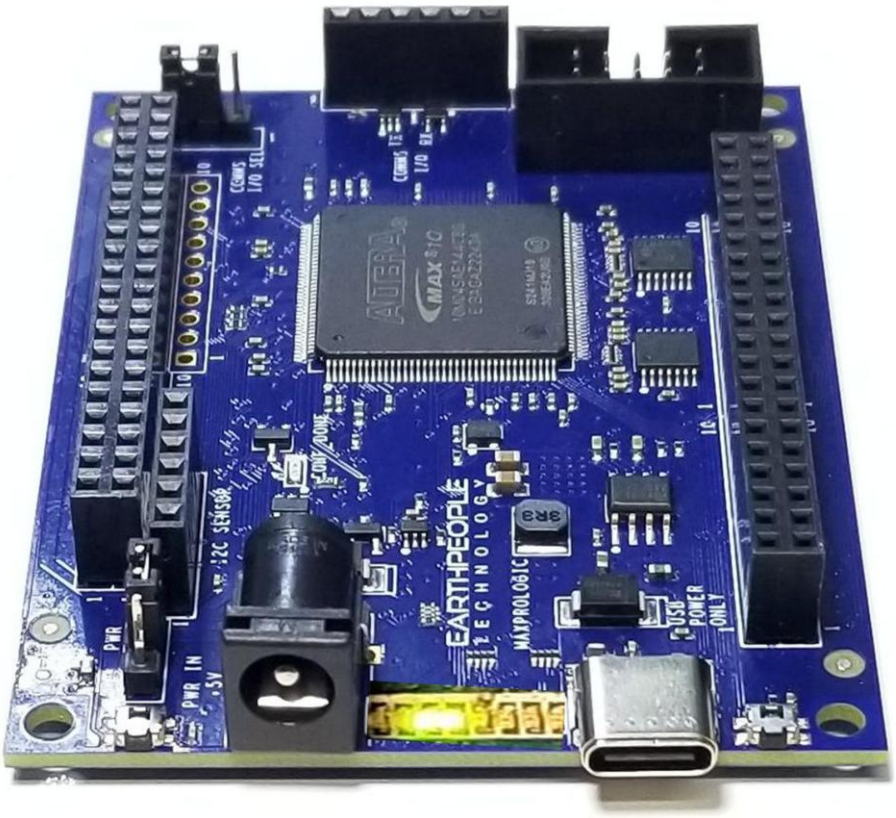


When the programming is complete, the Progress bar will indicate success.



At this point, the MaxProLogic is programmed and ready for use.

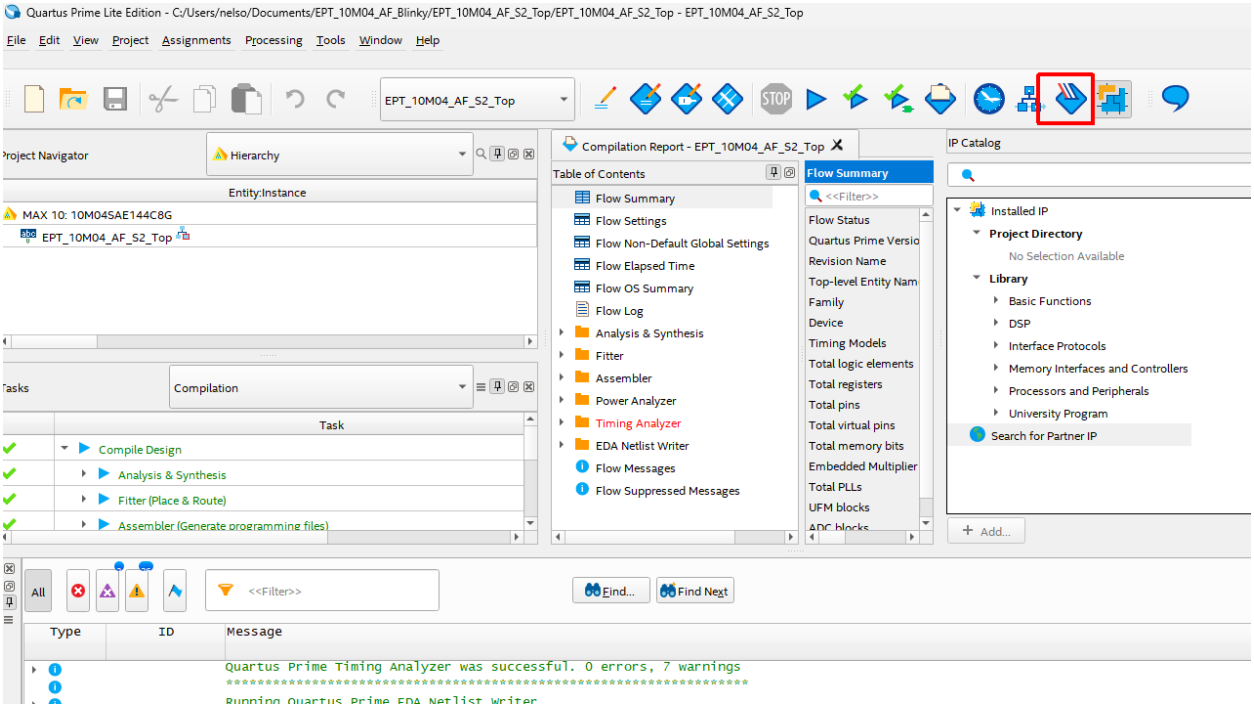
The eight LEDs on the front of the MAXPROLOGIC will blink one after the other.



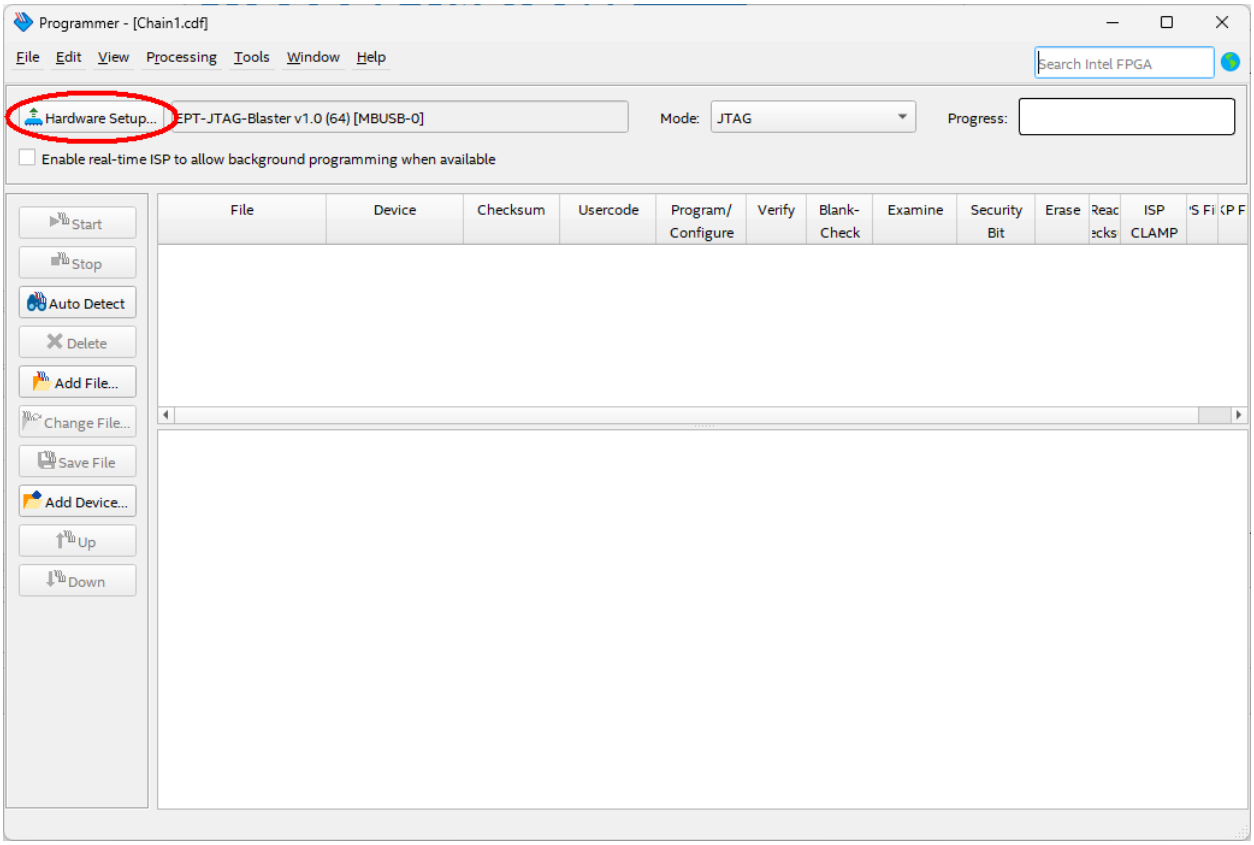
Alternatively, the user can program the MAX10 FPGA using the *.sof programming file. This method programs the FPGA using the JTAG connections. However, the programming file is NOT loaded into the on chip Flash. Instead, the programming file is loaded directly into the FPGA's SRAM, thereby configuring it. SOF files are used for direct, volatile configuration, meaning the FPGA configuration is lost when power is removed.

To program the MAX10 using SOF file, connect the Blaster and ensure the JTAG Driver is loaded for Quartus Prime Lite.

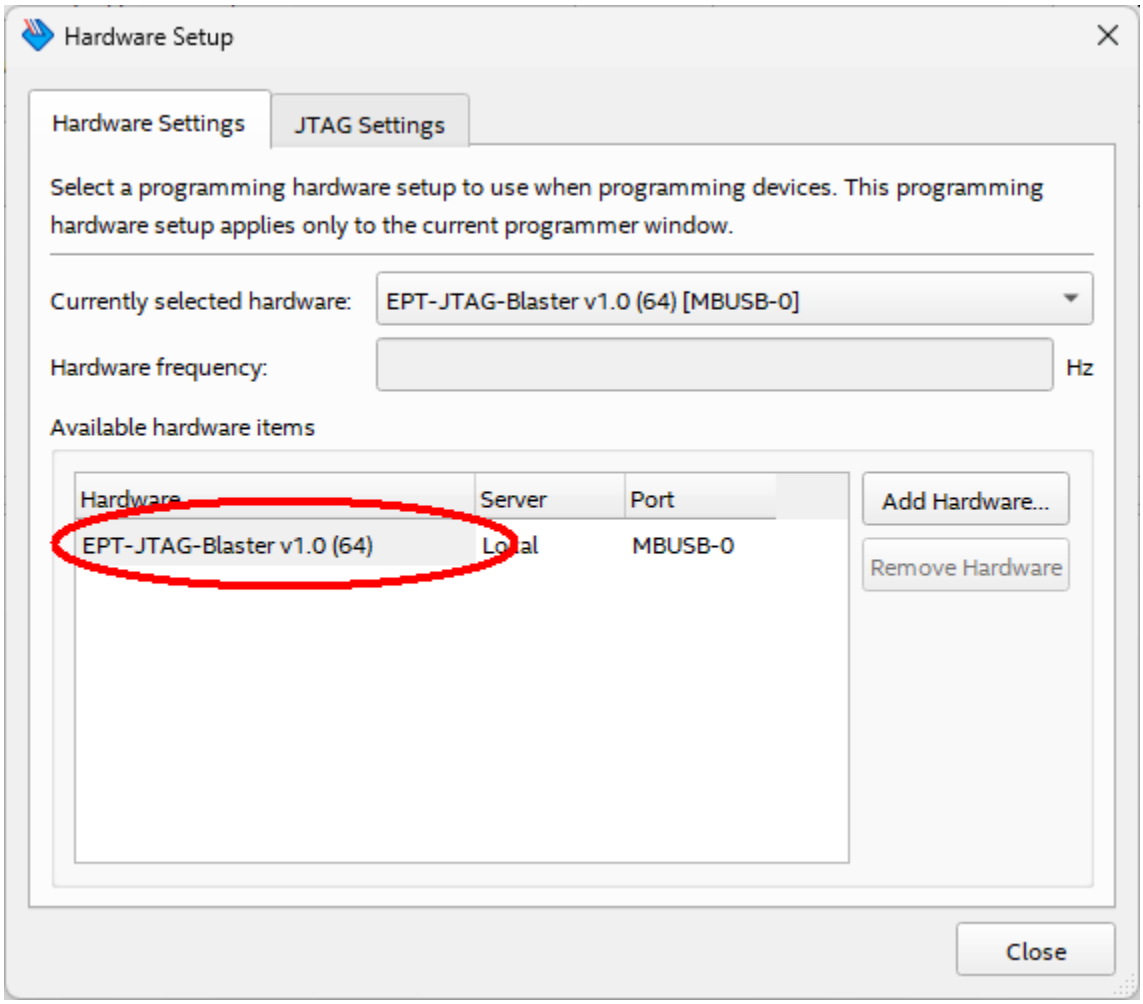
If the project created in the previous sections is not open, open it. Click on the Programmer button.



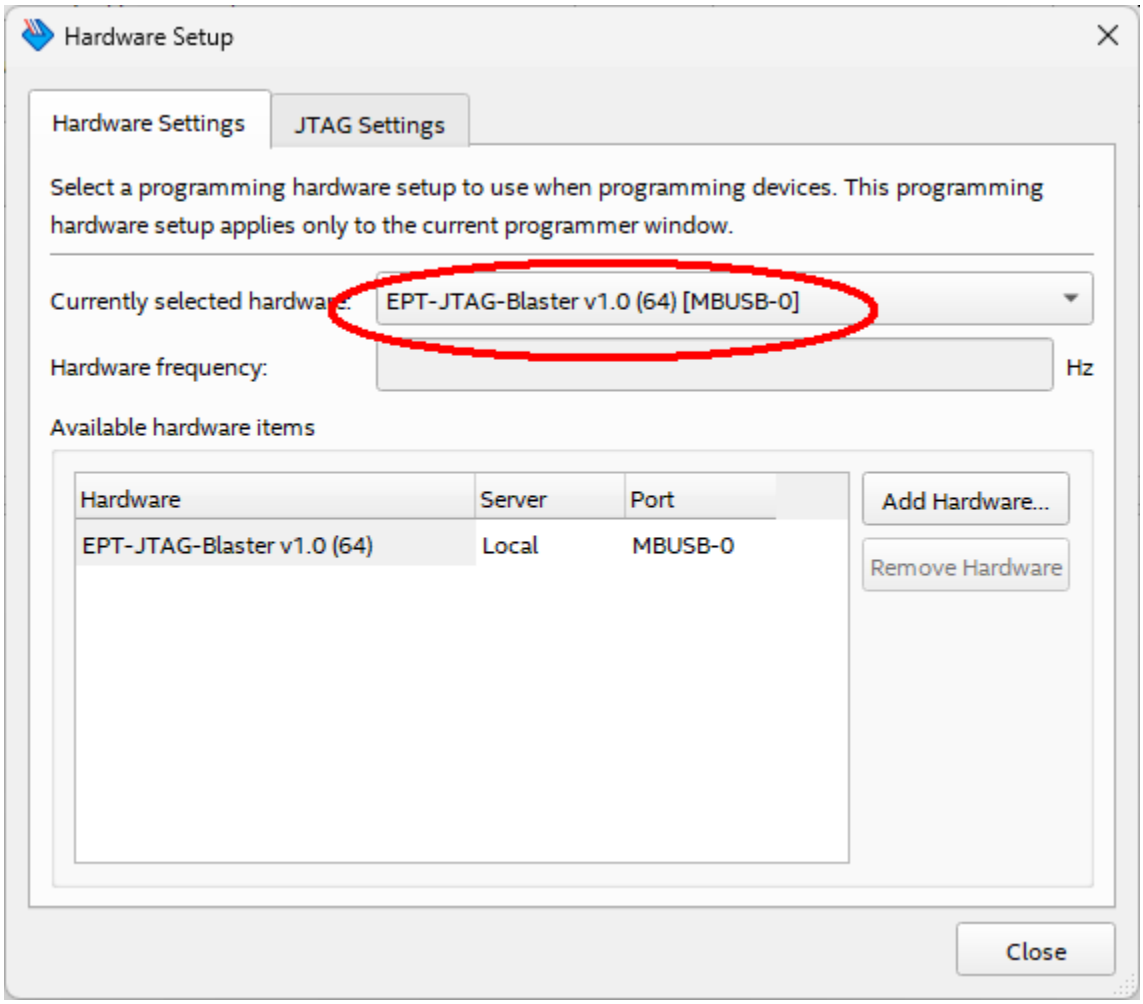
The Programmer Window will open up with the programming file selected. Click on the Hardware Setup button in the upper left corner.



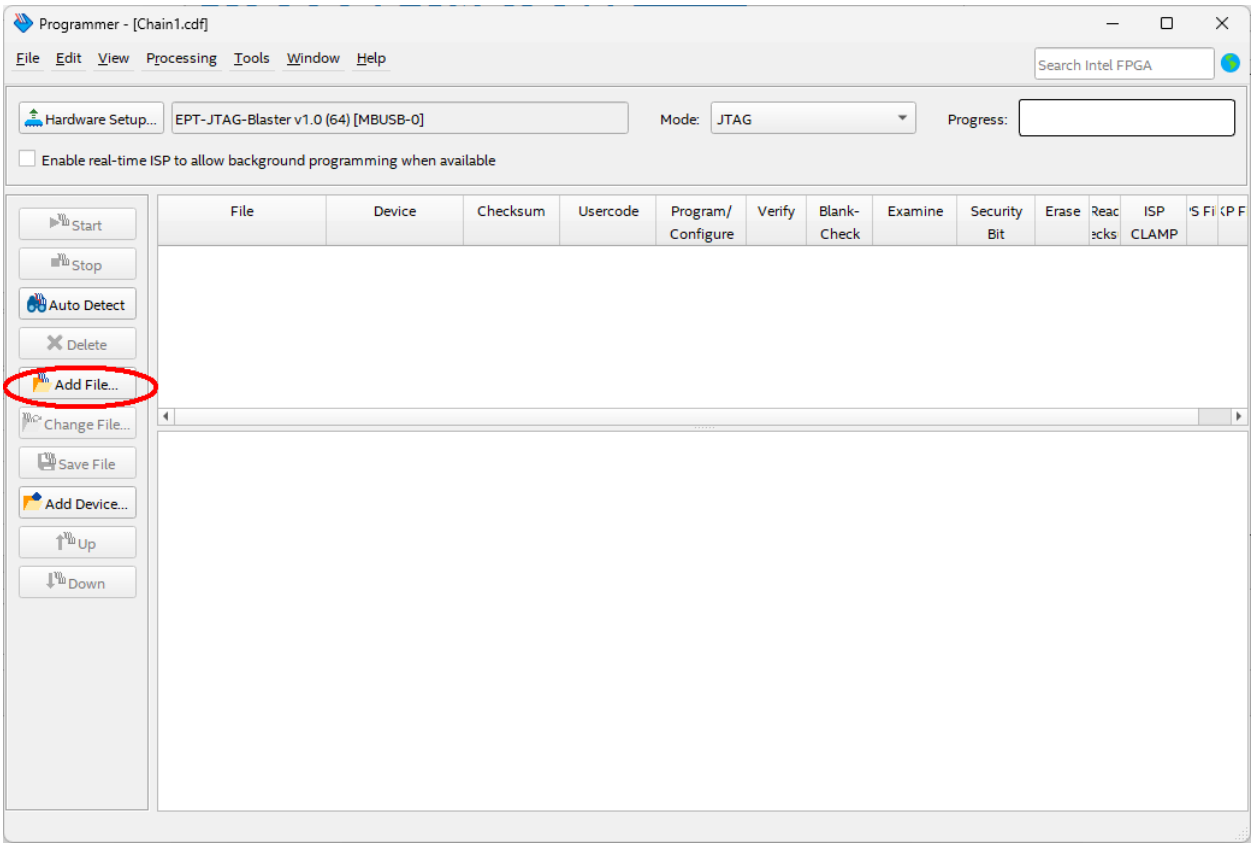
The Hardware Setup Window will open. In the “Available hardware items”, double click on “EPT-JTAG-Blaster v1.0b (64)”.

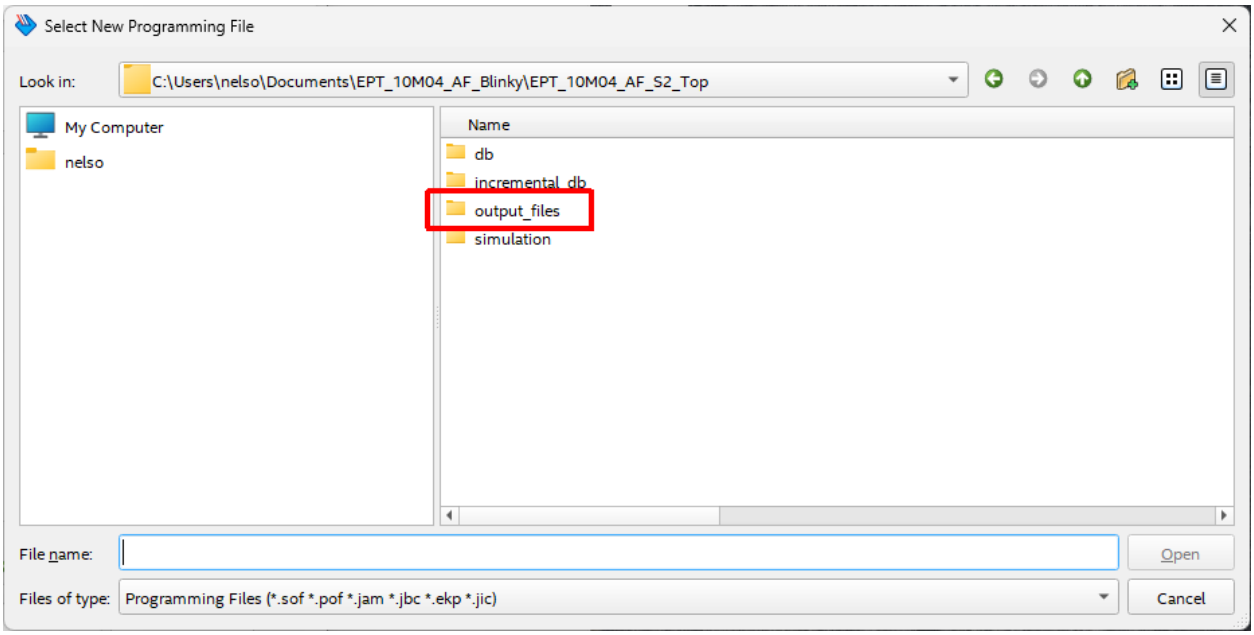


If you successfully double clicked, the “Currently selected hardware:” dropdown box will show the “EPT-JTAG-Blaster v1.0b (64)”.

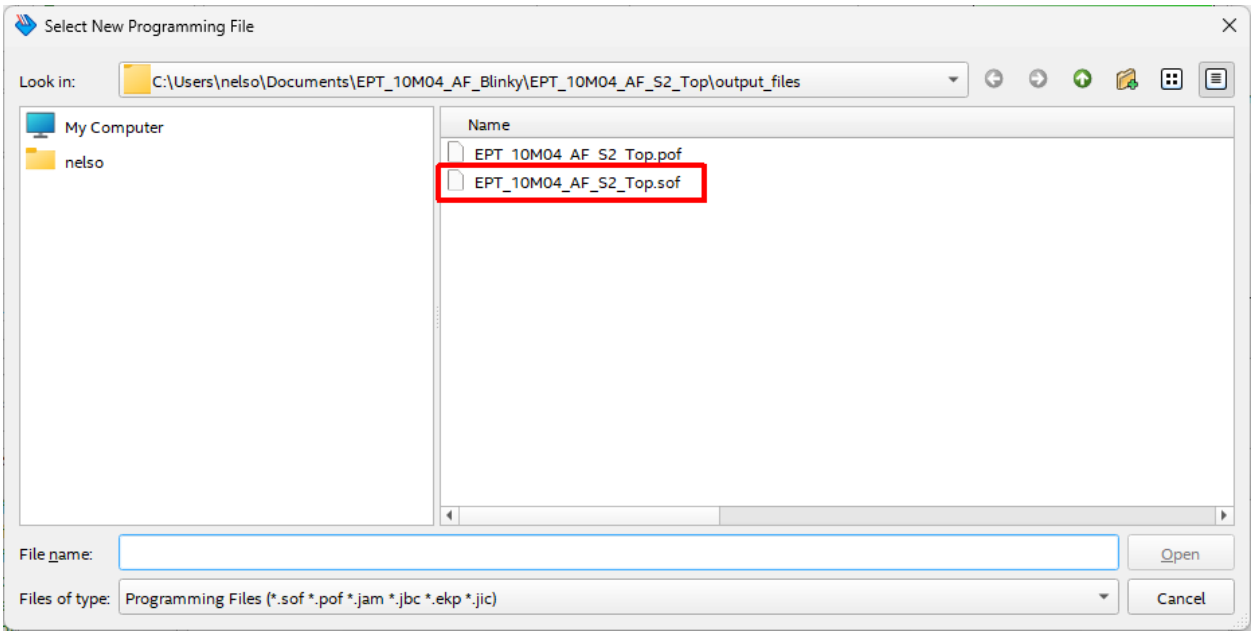


Click on the “Add File” button

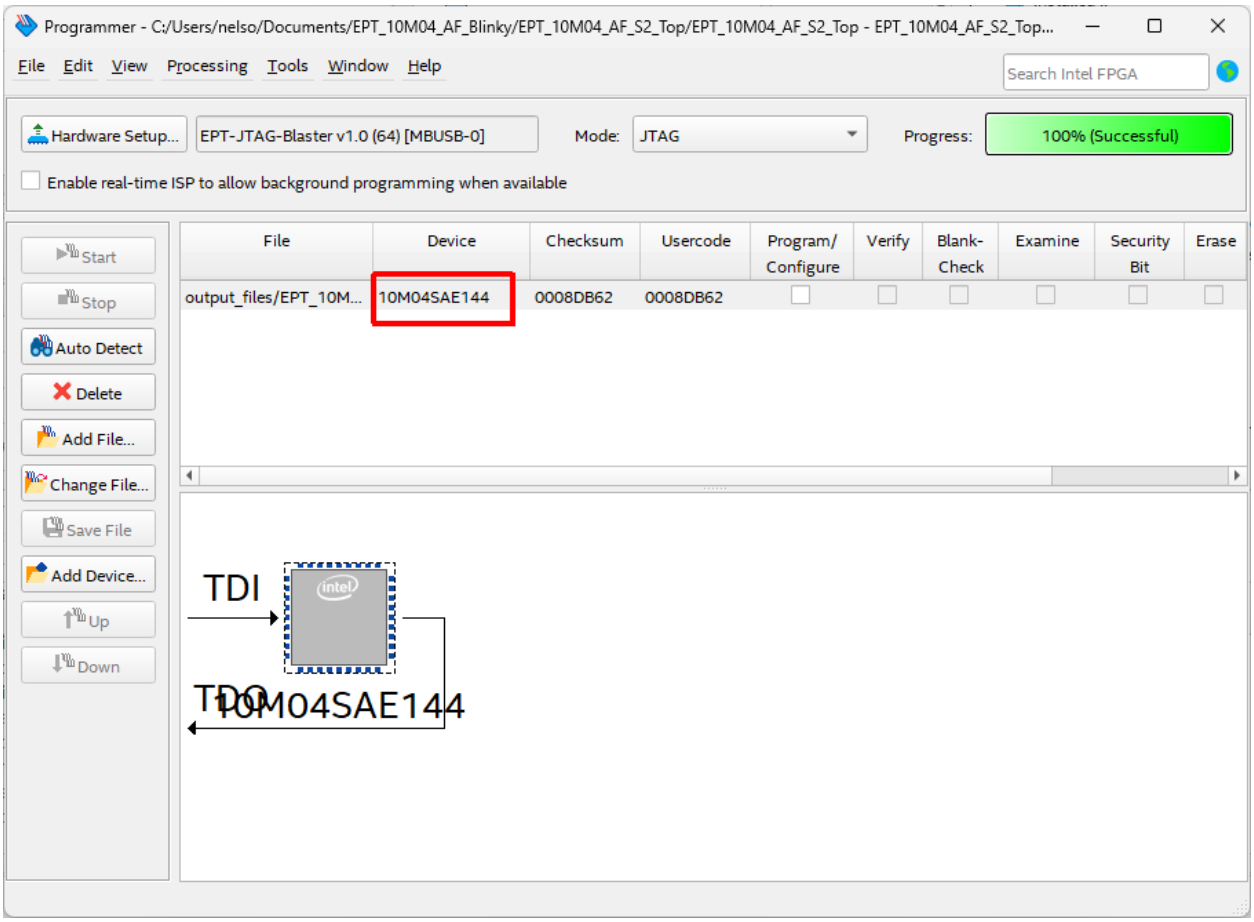

Programmer - [Chain1.cdf]
File Edit View Processing Tools Window Help
Search Intel FPGA
Hardware Setup...
EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]
Mode: JTAG
Progress:
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File
Device
Checksum
Usercode
Program/ Configure
Verify
Blank-Check
Examine
Security Bit
Erase
ISP CLAMP



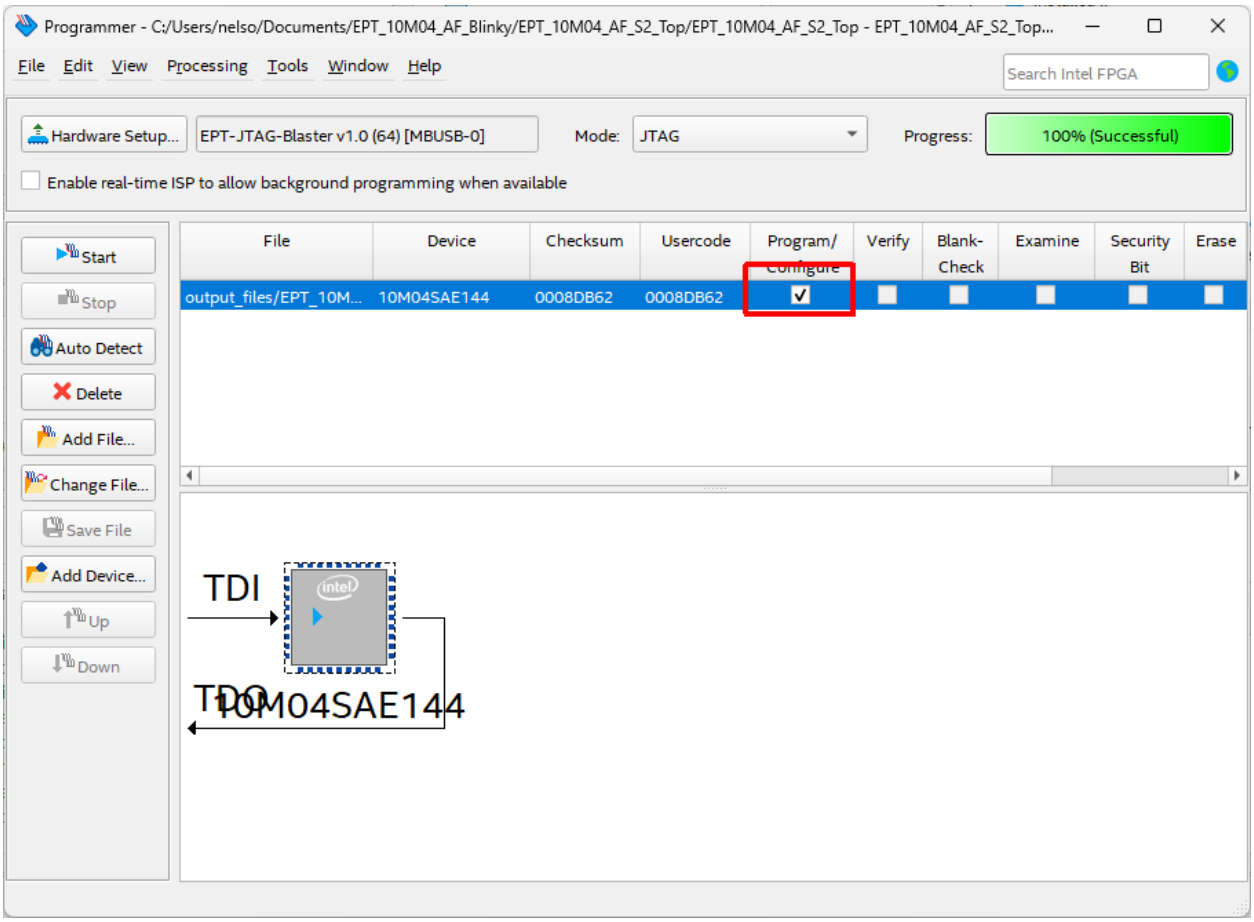
At the Browse window, double click on the output files folder.



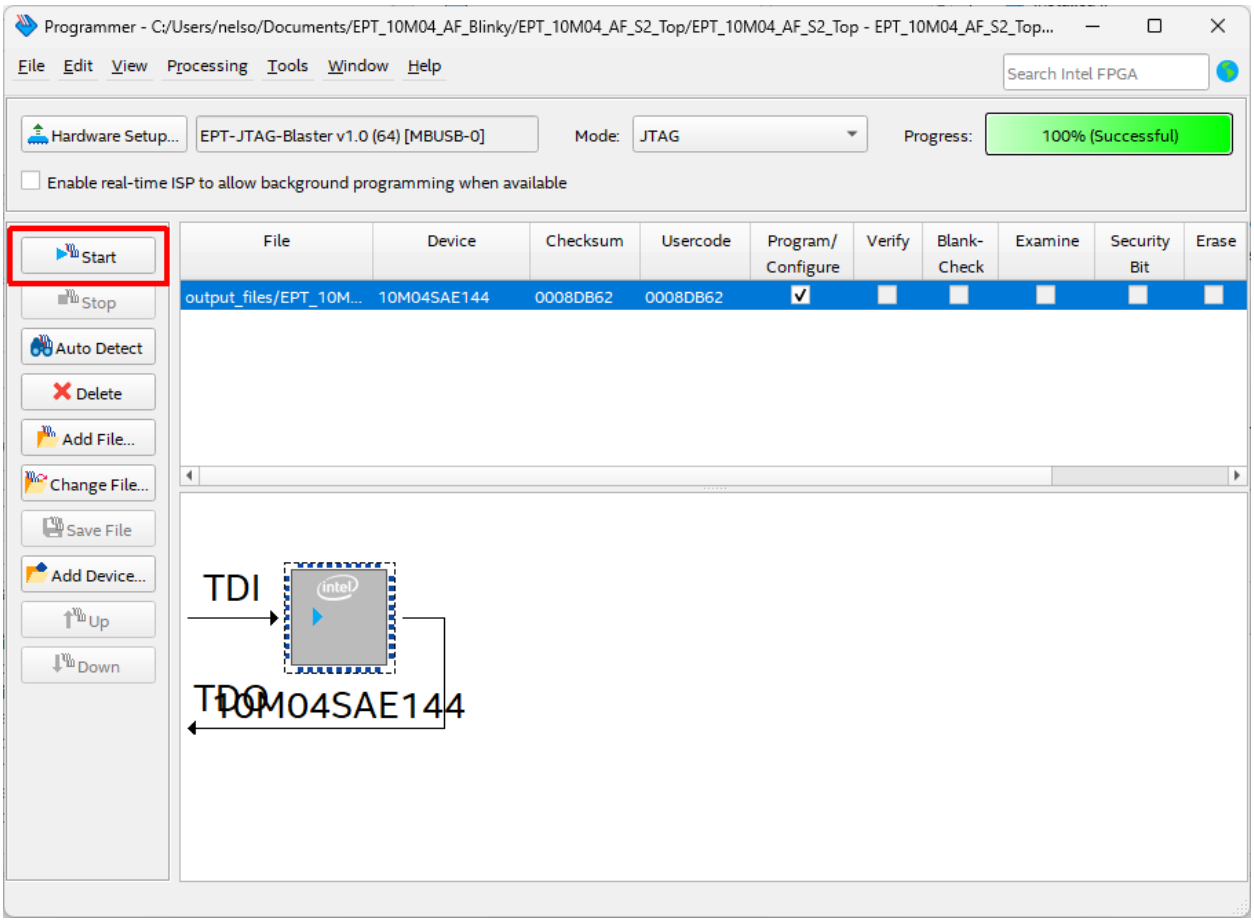
Double click on the “EPT_10M04_AF_S2_Top.sof” file. Click the Open button in the lower right corner.



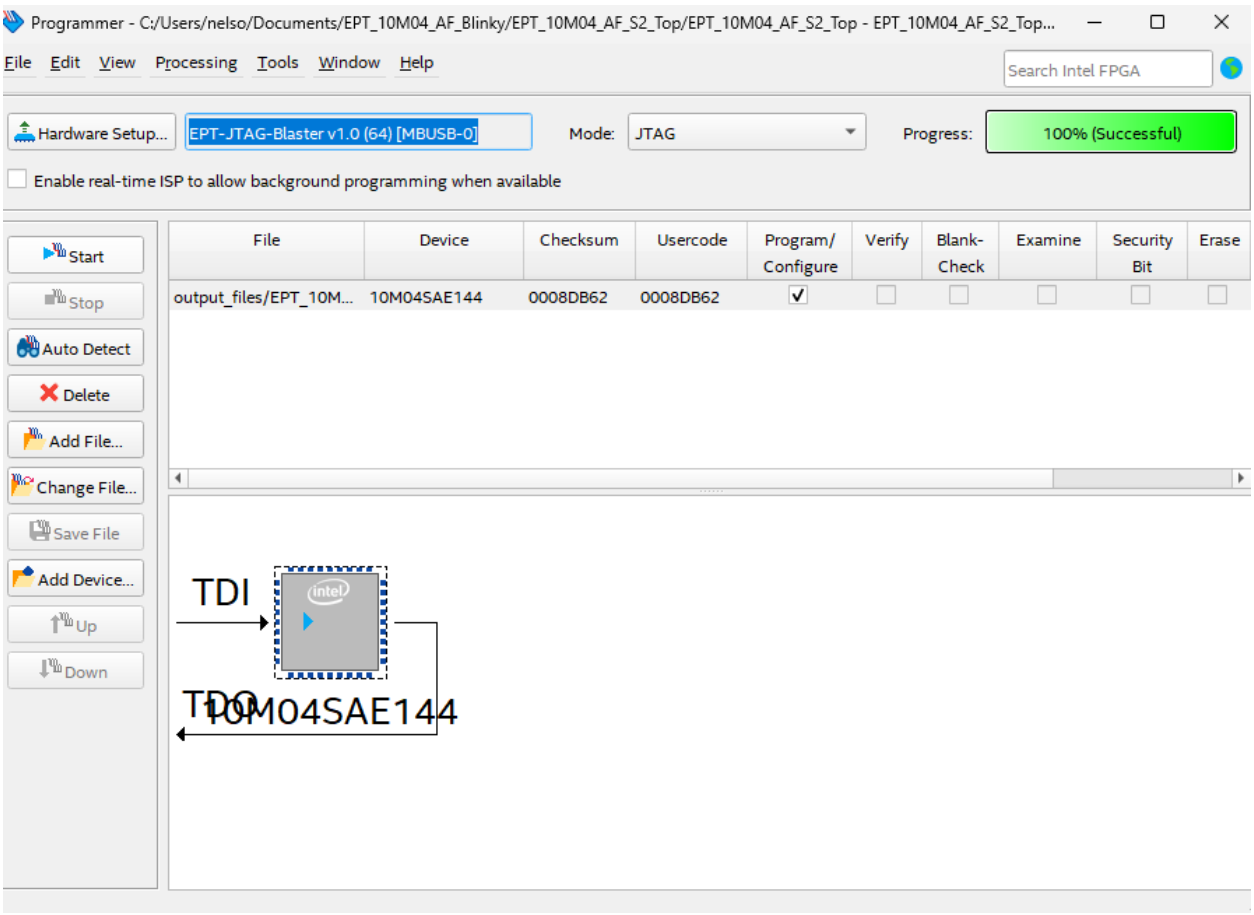
Next, selet the checkbox under the “Program/Configure” of the Programmer Tool.



Click on the Start button to to start programming the FPGA.



When the programming is complete, the Progress bar will indicate success.



At this point, the MaxProLogic is programmed and ready for use.

The eight LEDs on the front of the MAXPROLOGIC will blink one after the other.

MaxProLogic Development System User Manual