EARTH PEOPLE TECHNOLOGY, Inc

CycloFlex DEVELOPMENT SYSTEM
User Manual

The CycloFlex is an FPGA development board that is designed for the intermediate user with some experience designing FPGA projects. This board provides a powerful platform for developing and debugging programmable logic code. It has been designed from the ground up to provide the functionality needed for the demanding projects from todays students and hobbyists. The board provides a convenient, user-friendly work flow 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.

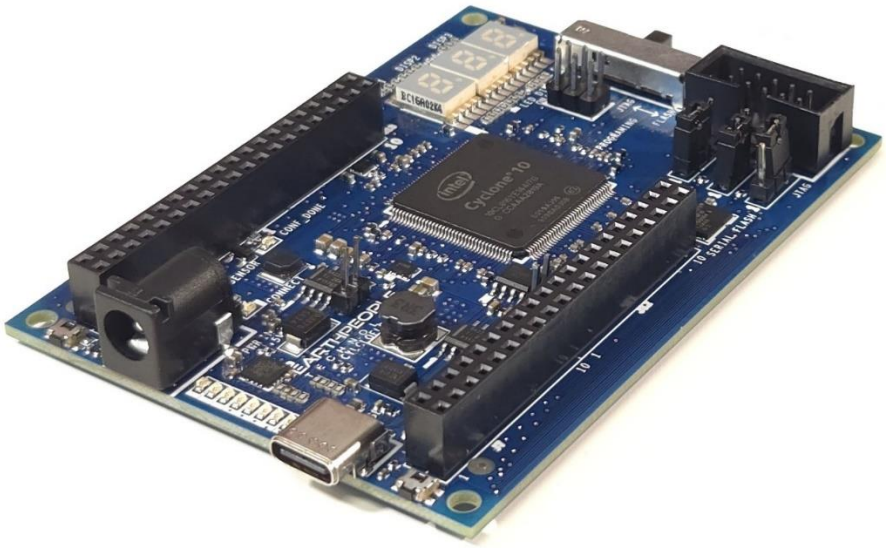
The core of the CycloFlex is the Altera Cyclone 10 FPGA. This powerful chip has 16,000 Logic Elements and 504 Kbits of Memory. The Cyclone 10 is easily scalable from the entry level college student to the most advanced projects like an audio sound meter with FFT.

CycloFlex Development System User Manual

http://www.earthpeopletechnology.com/

Contents

CycloFlex Development System User Manual



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

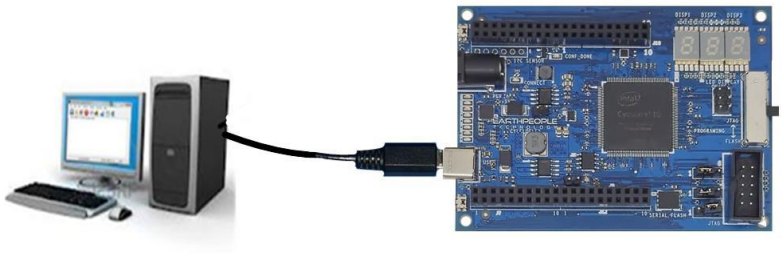
1 Introduction

The CycloFlex is an FPGA Development Board based on the Cyclone 10 FPGA. Core of the board is the Cyclone 10 chip. This chip is a powerful mid-range FPGA from Altera. It provides a high density sea of programmable gates, on-board resources and general purpose I/O's. The CycloFlex includes high speed USB to SPI communications which allow data transfers up to 4MegaBits per second. The board includes three Seven Segment LED Displays which gives the user more options for code debugging than simple LEDs. The CycloFlex also includes two pushbuttons, one RGB LED and Eight traditional Green LEDs and of course loads of General

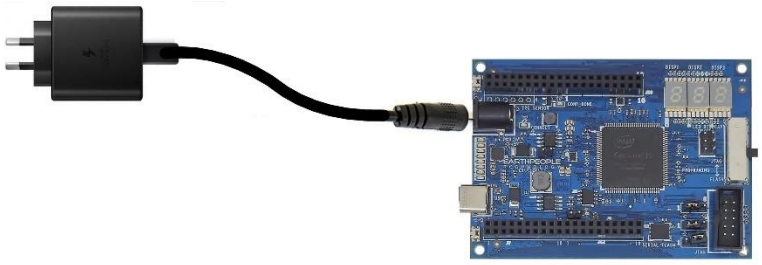
Purpose Inputs/Outputs. There is an on board 50MHz oscillator that feeds the internal PLLs of the Cyclone 10.

2 User Setup

The CycloFlex is ready to go straight out of the box. Just connect power from either a USP Port (from PC/Laptop), USB-C Charger (10Watt to 45Watt Charger allowed) or +5 to 5.5 VDC through barrel connector.

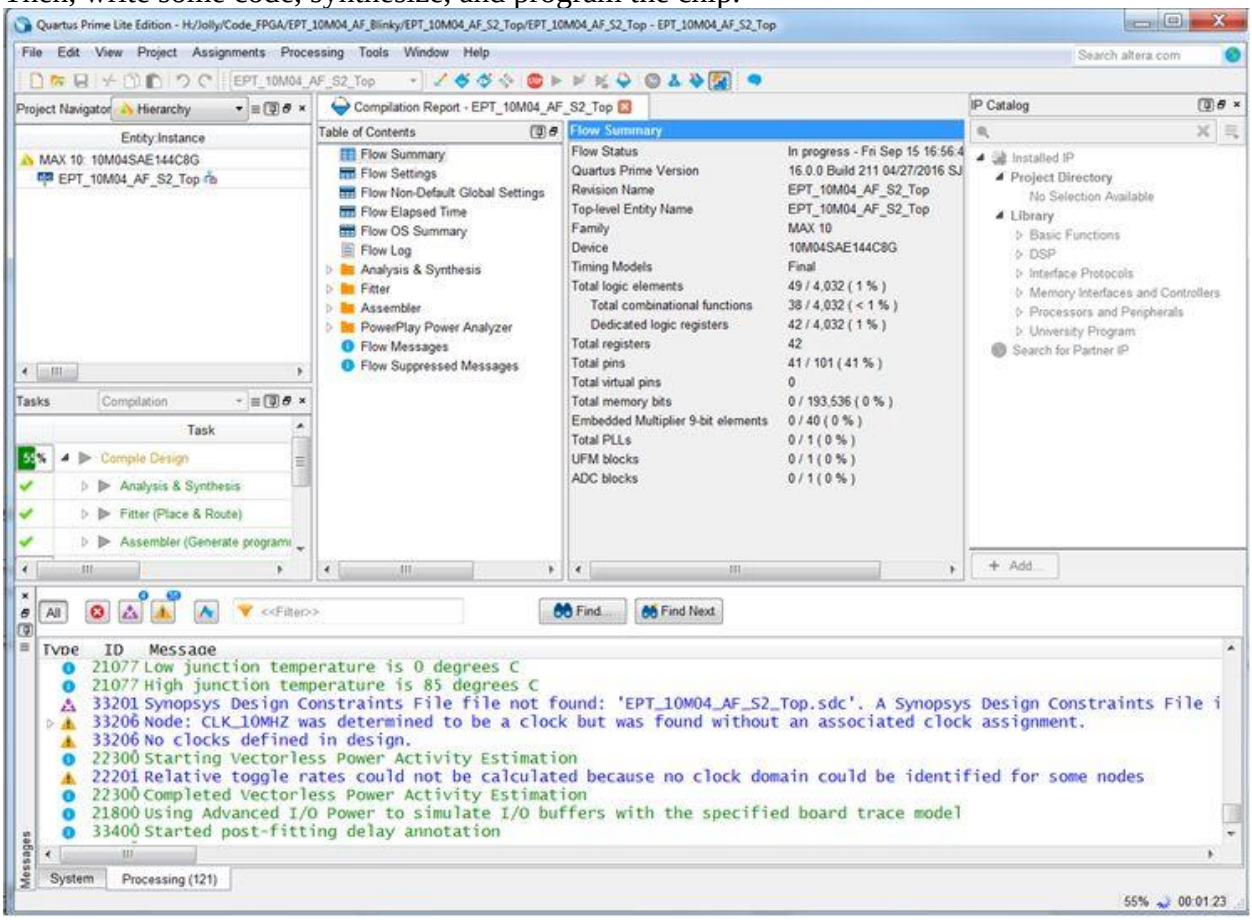


OR



Next, connect a JTAG Blaster or Altera USB Blaster to J2 for programming the Cyclone 10 chip.

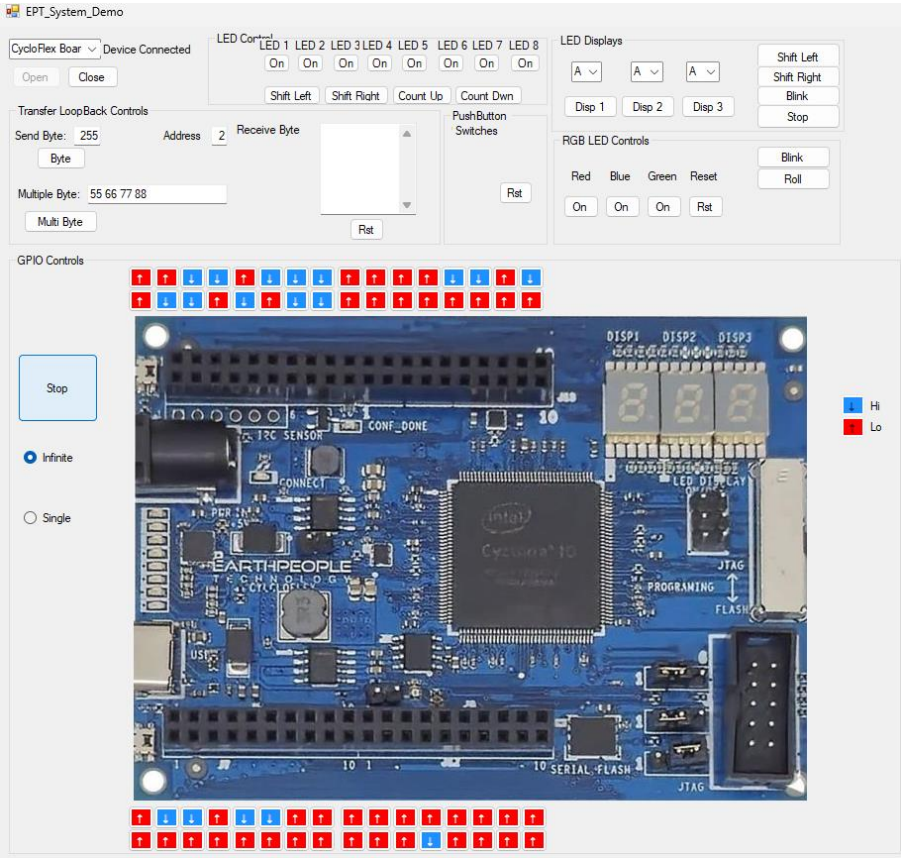
Then, write some code, synthesize, and program the chip.



See Programming and Configuration Section for details about programming the CycloFlex board.

2.1 Getting Started with the CycloFlex

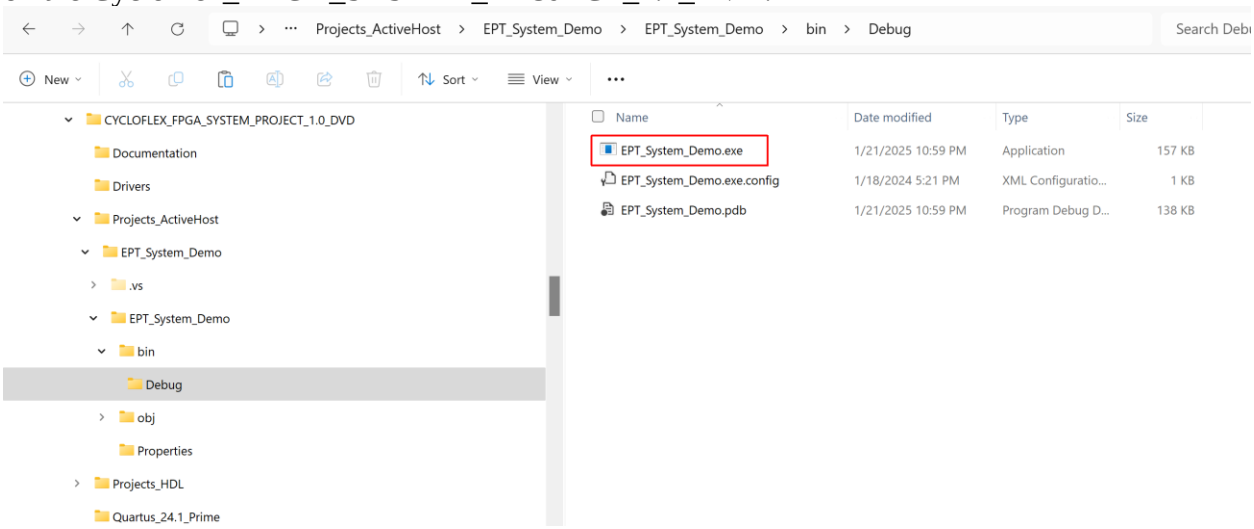
The CycloFlex board comes pre-loaded with the EPT_System_Demo HDL project in the FPGA. This project allows the user to test out the functions of the Active Host API and the board hardware.



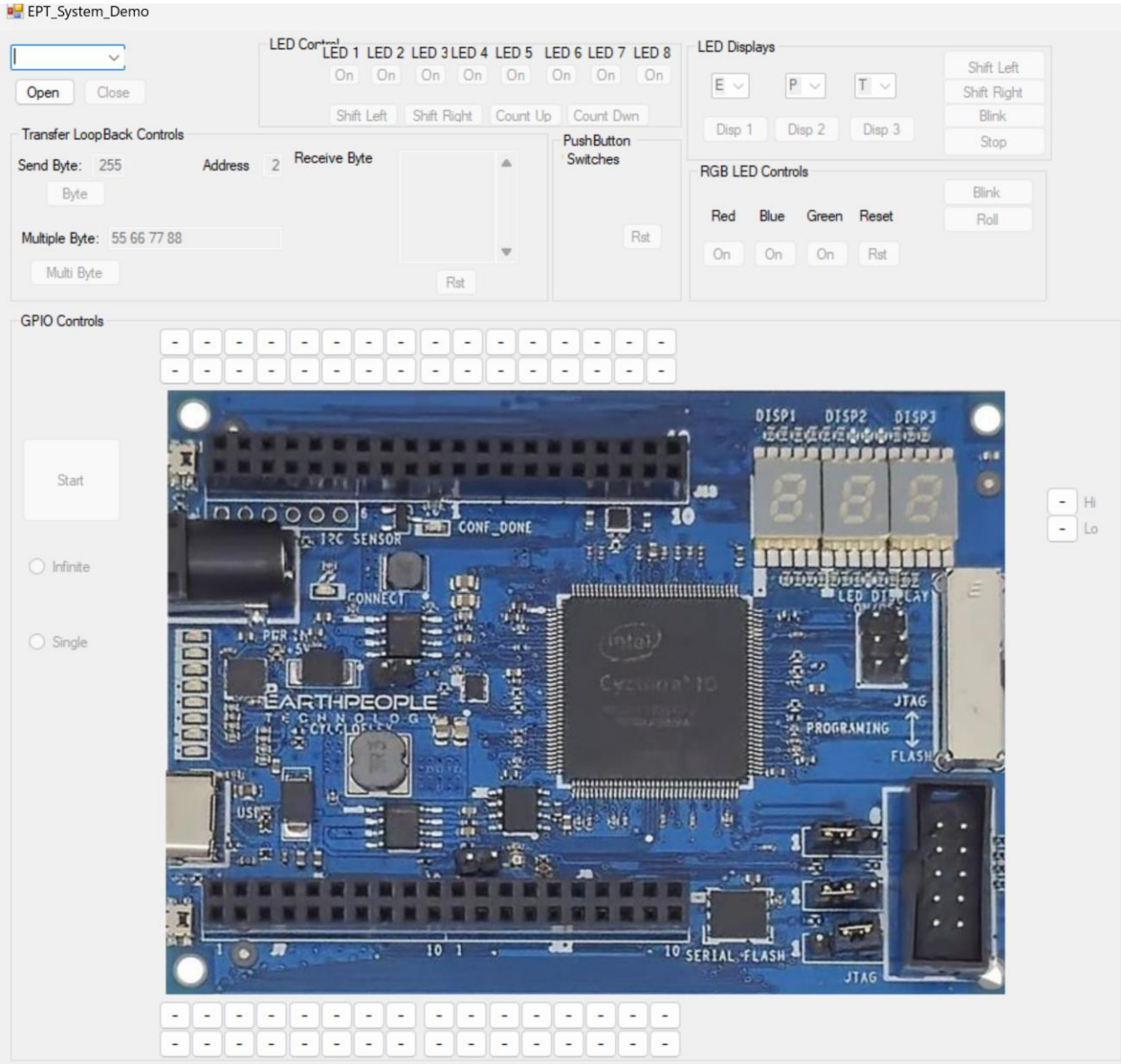
To test drive the application, connect the CycloFlex to the Windows PC using a USB-C cable. Load the driver for the board. See the section “EPT Drivers” for instructions on loading the CycloFlex driver. If the USB driver fails to load, the Windows OS will indicate that no driver was loaded for the device. In the case of the failed USB driver, try rebooting the PC and following the steps in the EPT Drivers section of this User Manual.

Next, open a Windows Explorer browser.

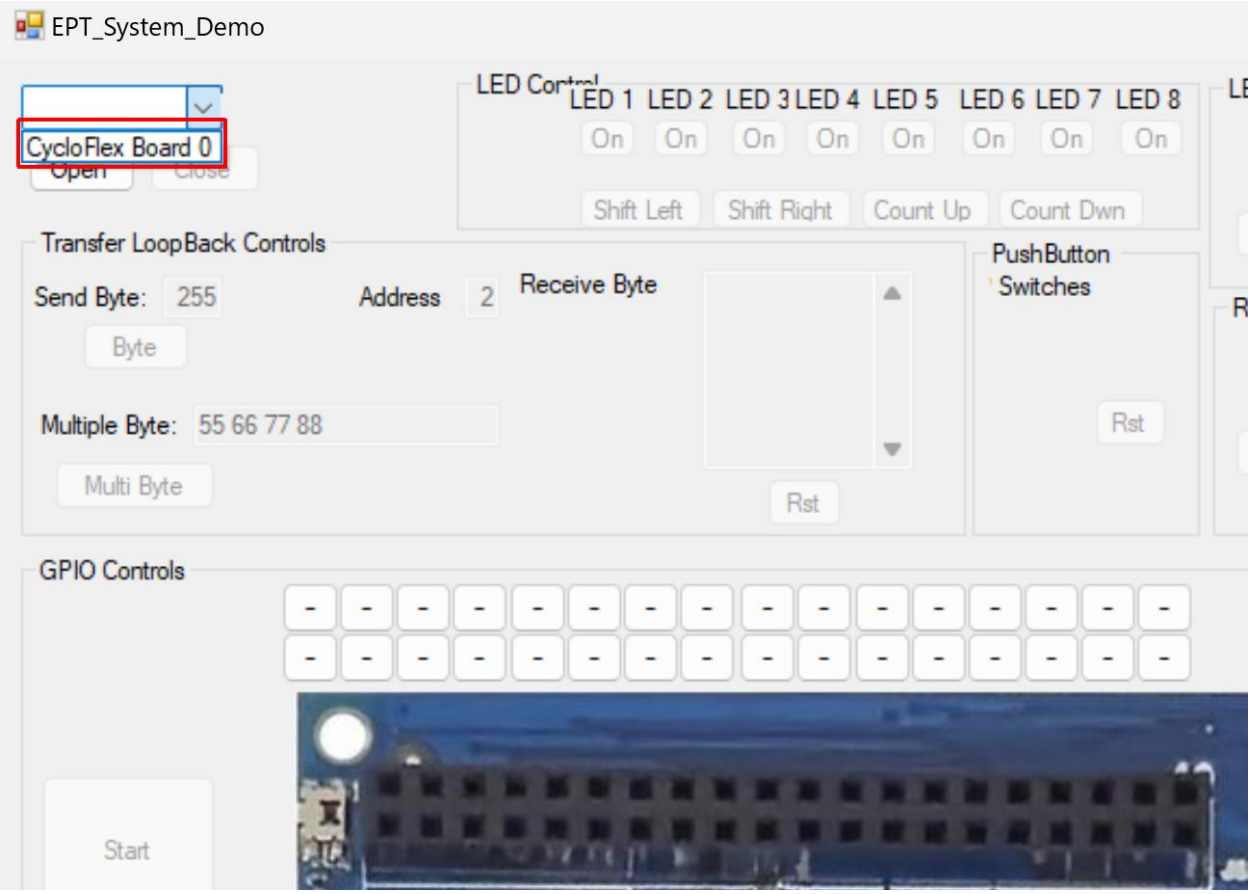
Browse to the Projects_ActiveHost\EPT_System_Demo\EPT_System_Demo\bin\Debug\ folder on the CycloFlex_ FPGA_SYSTEM_PROJECT_x.x_DVD.



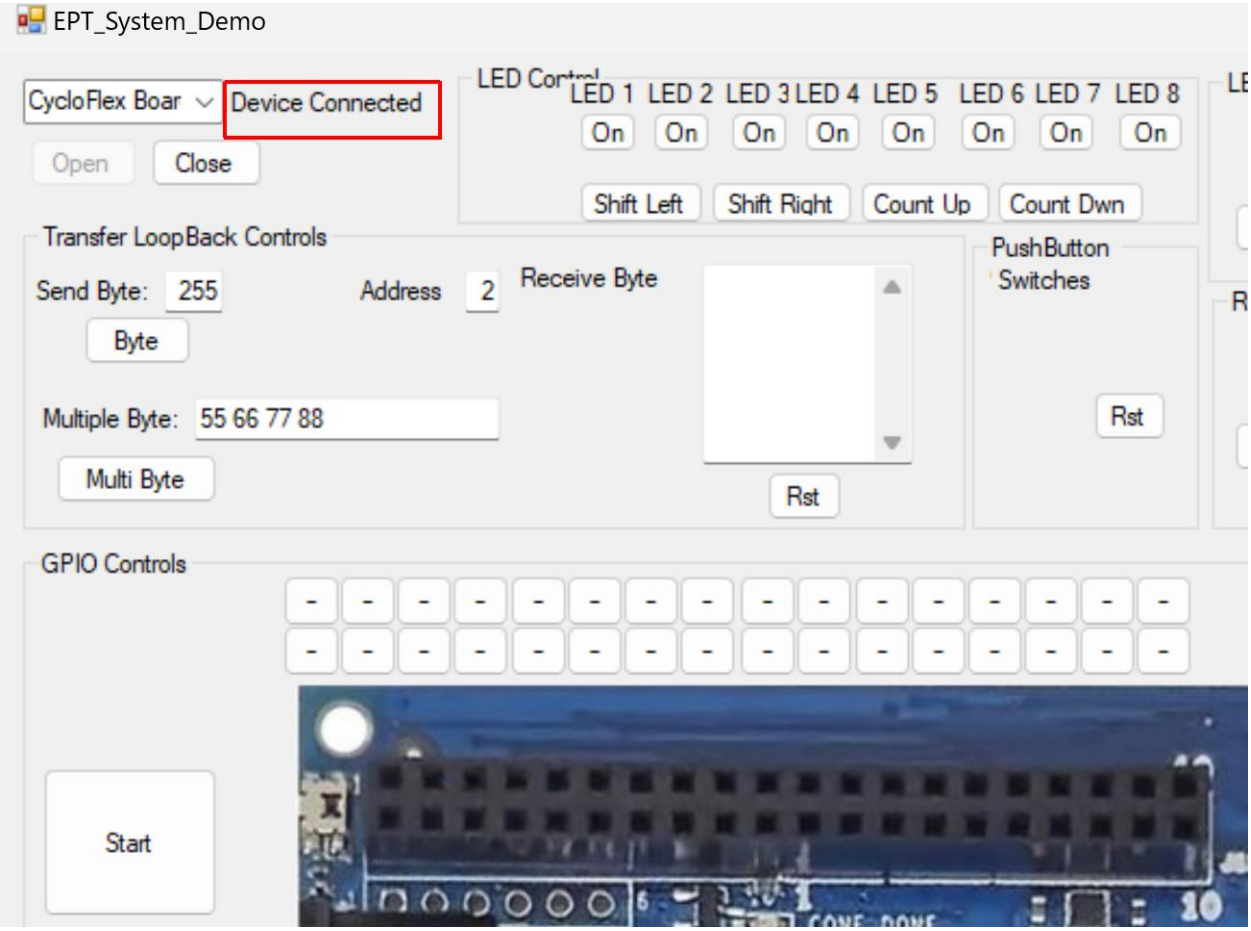
Double click on the EPT_System_Demo.exe. The application should load with a Windows form.



With the application loaded, select the FPGA board (EPT CycloFlex x) from the dropdown combo box and click on the “Open” button.

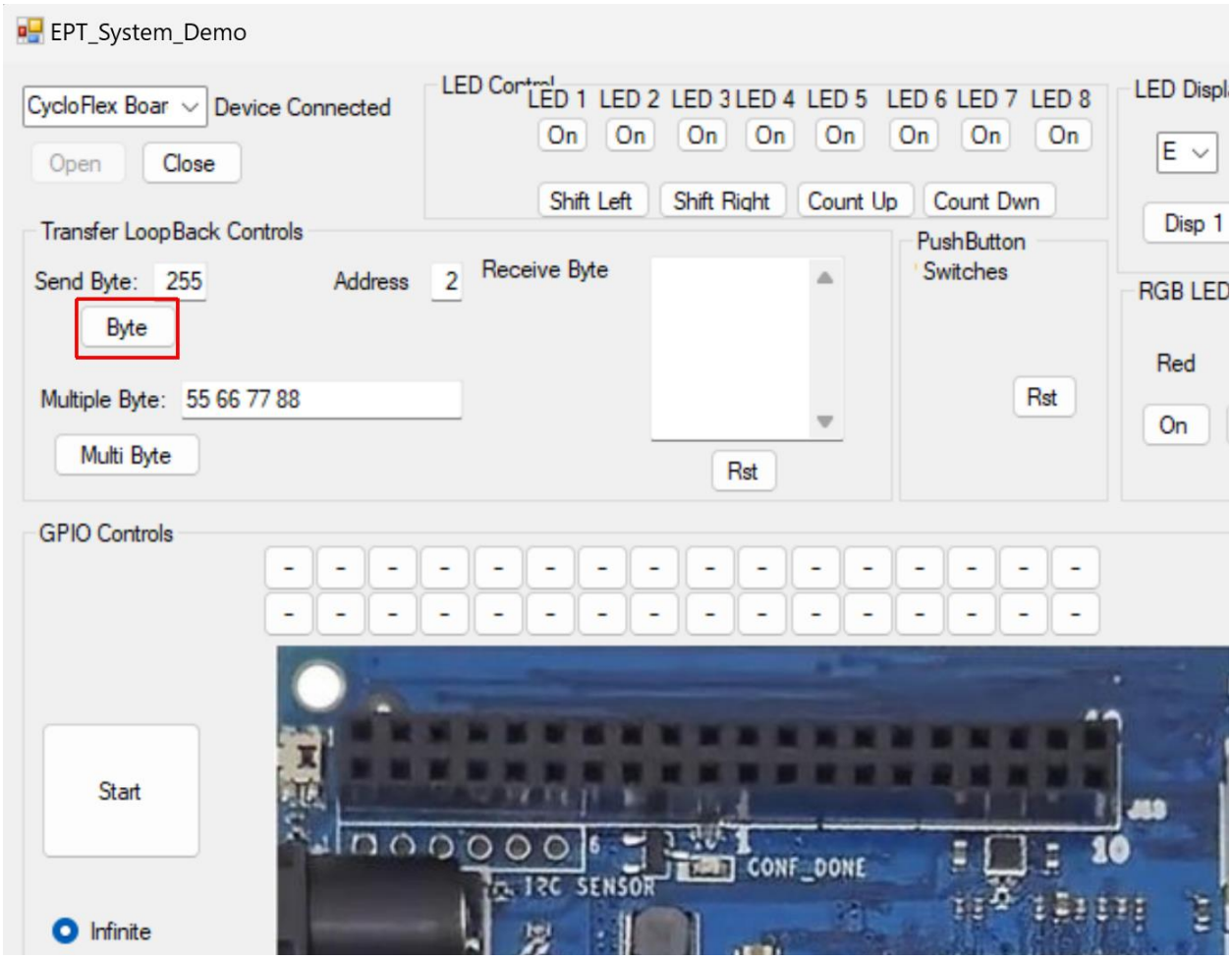


When the EPT System Demo software has properly accessed the board, the “Device Connected” label will appear.

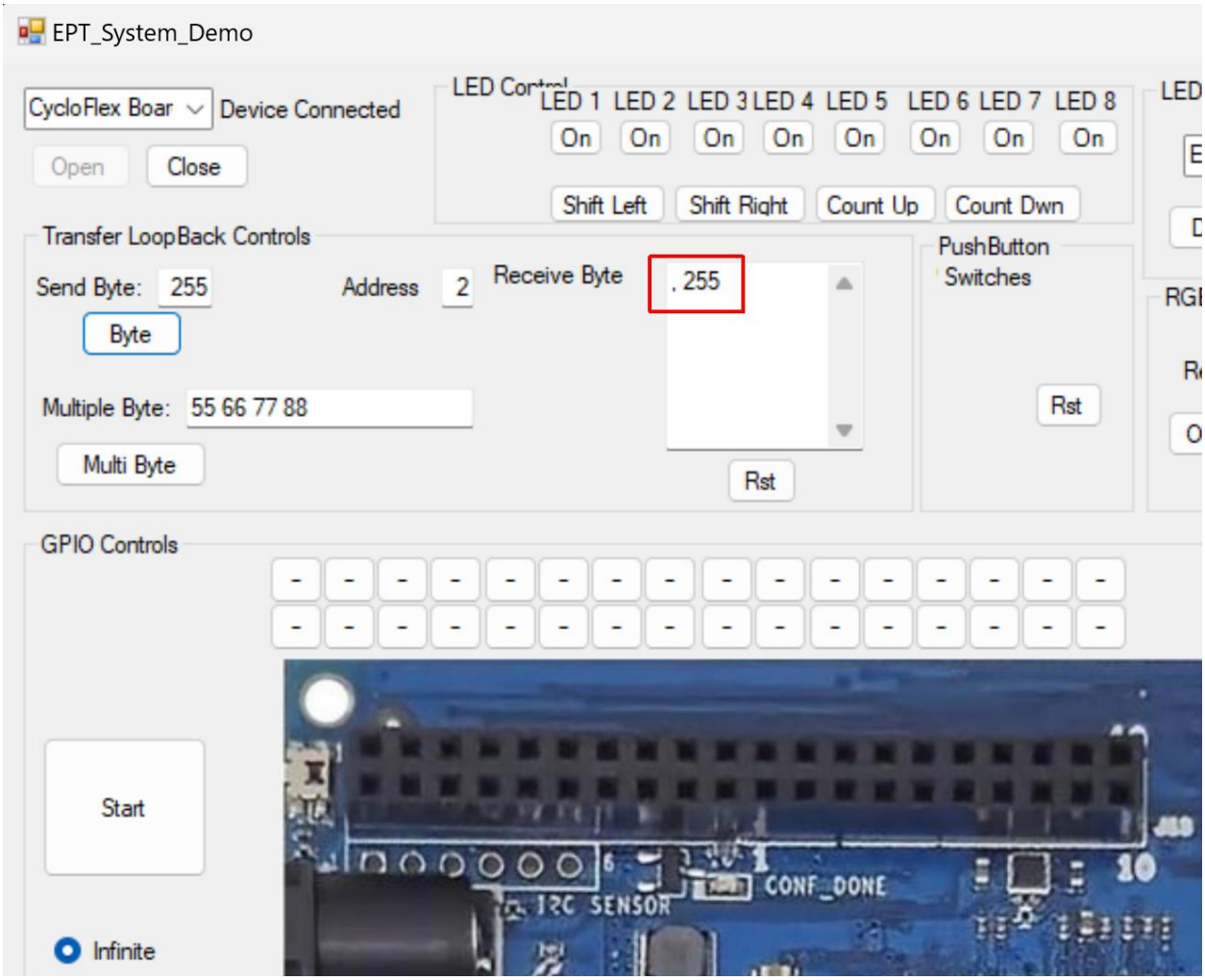


In the “Transfer LoopBack Controls” Group, leave the Address set at 2.

To exercise the Single Byte Transfer EndTerm, click the “Byte” button in the Transfer Controls group.

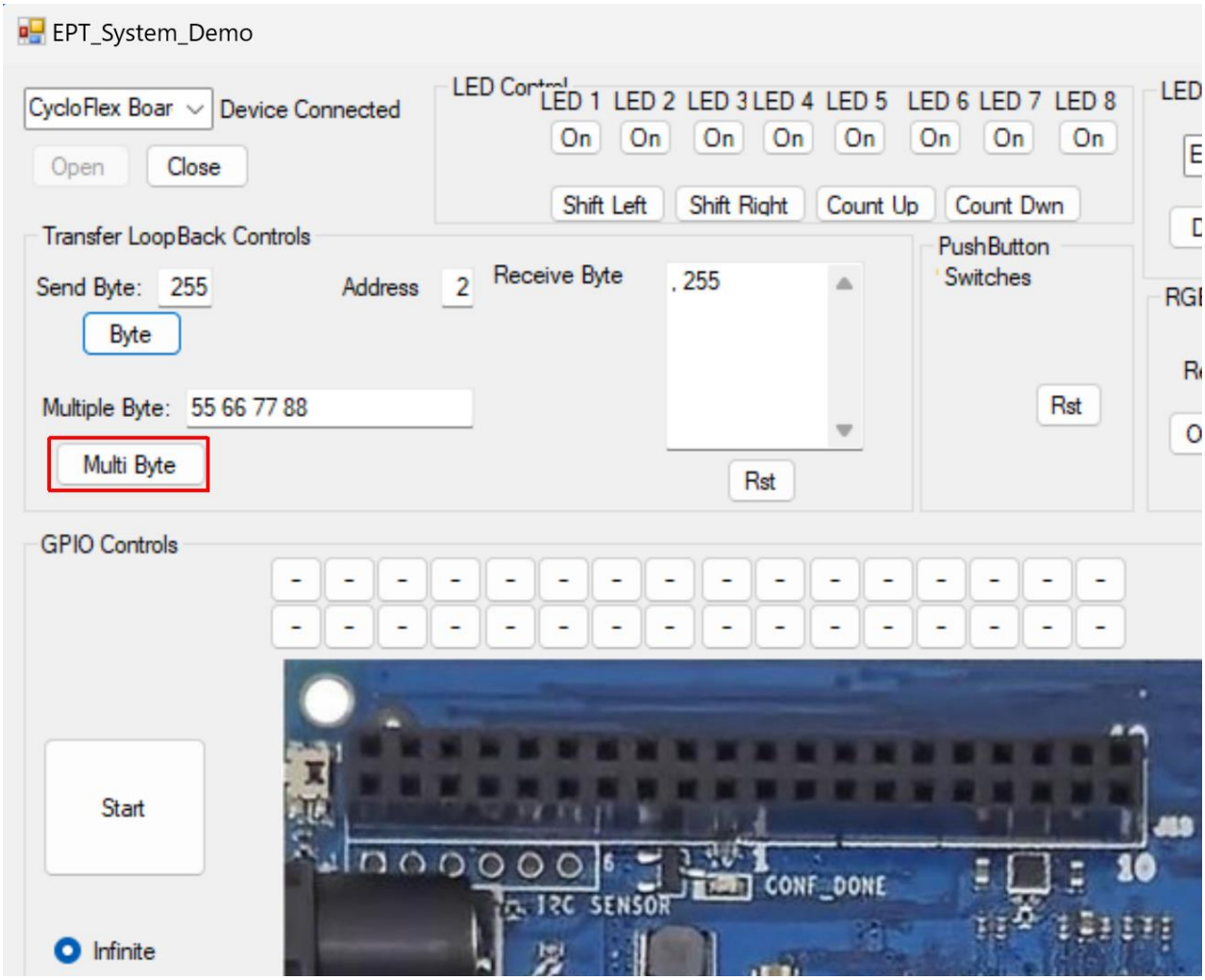


The byte in the drop down box will be transmitted to the CycloFlex. Then, the CycloFlex will receive the byte and re-send it back to the Host PC to be displayed in the “Receive Byte” Text Box.

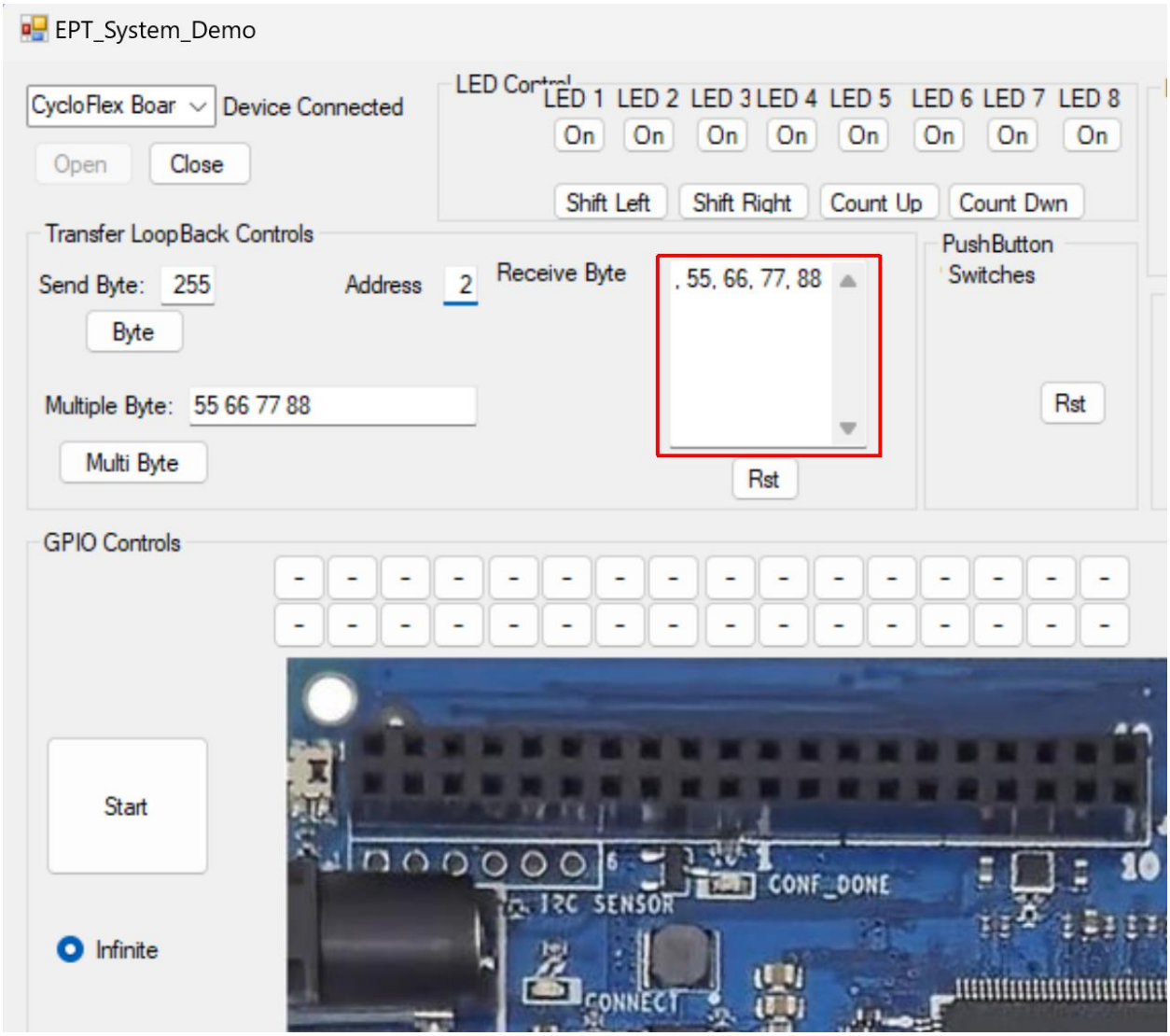


The user can change the byte in the “Send Byte:” Text Box and attempt the loopback again. This box is in decimal format, so the byte is limited to 0 to 255 decimal. In hexadecimal, it is 0x00 to 0xff.

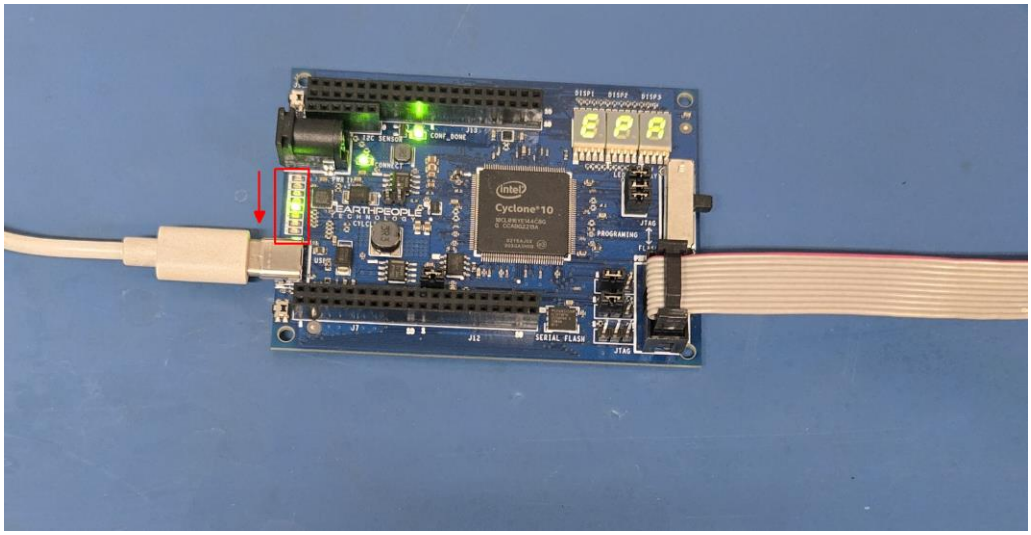
Type in several numbers separated by a space and less 256 into the “Multiple Byte” textbox.



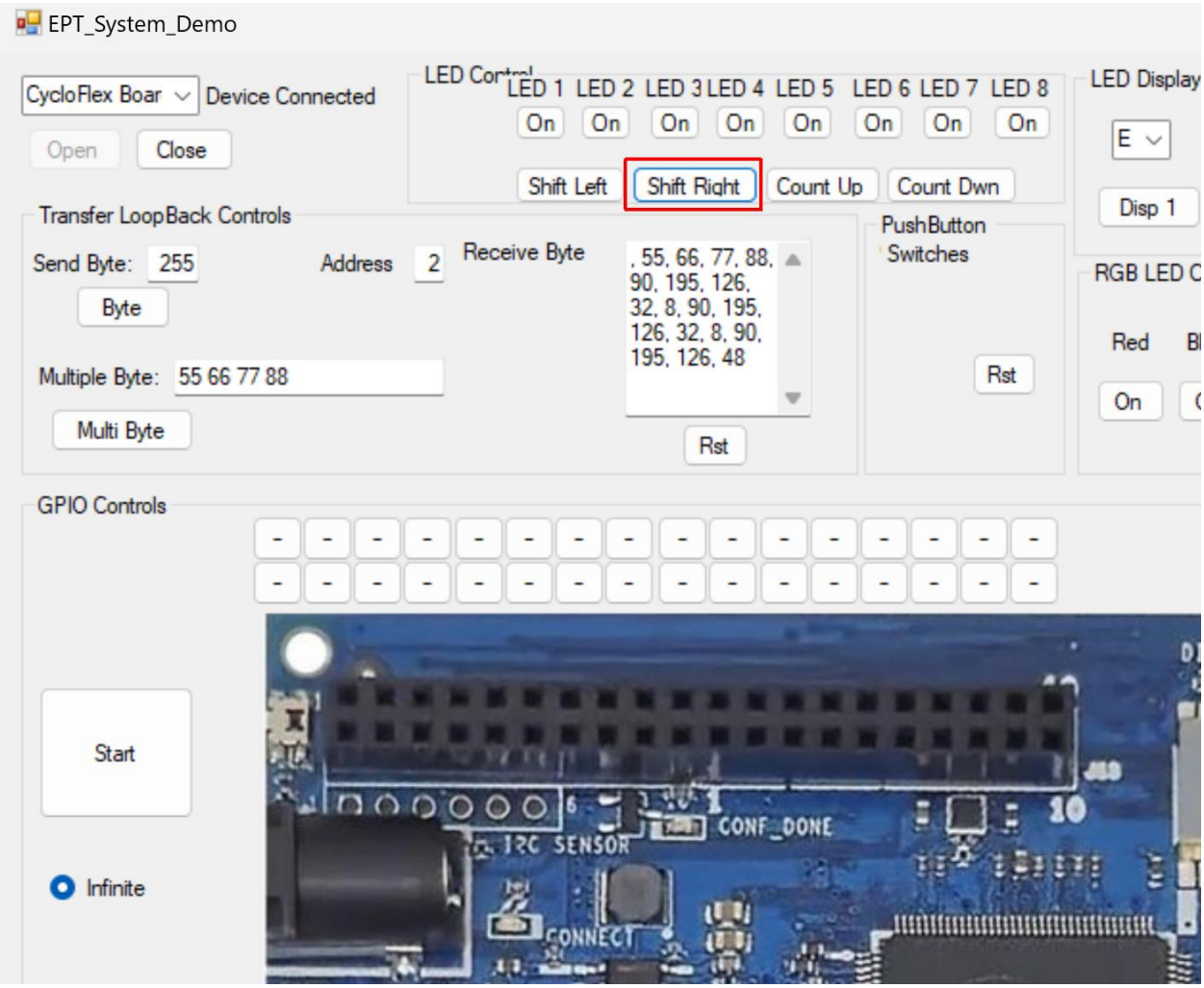
Then hit the “Multi Byte” button. The numbers appear in the “Receive Byte” textbox.


EPT_System_Demo
CycloFlex Boar
Device Connected
Open
Close
LED Control
LED 1 LED 2 LED 3 LED 4 LED 5 LED 6 LED 7 LED 8
On On On On On On On On
Shift Left
Shift Right
Count Up
Count Dwn
Transfer LoopBack Controls
Send Byte: 255
Byte
Address 2
Receive Byte
, 55, 66, 77, 88
Rst
Multiple Byte: 55 66 77 88
Multi Byte
PushButton Switches
Rst
GPIO Controls
Start
Infinite
I2C SENSOR
CONF_DONE
CONNECT

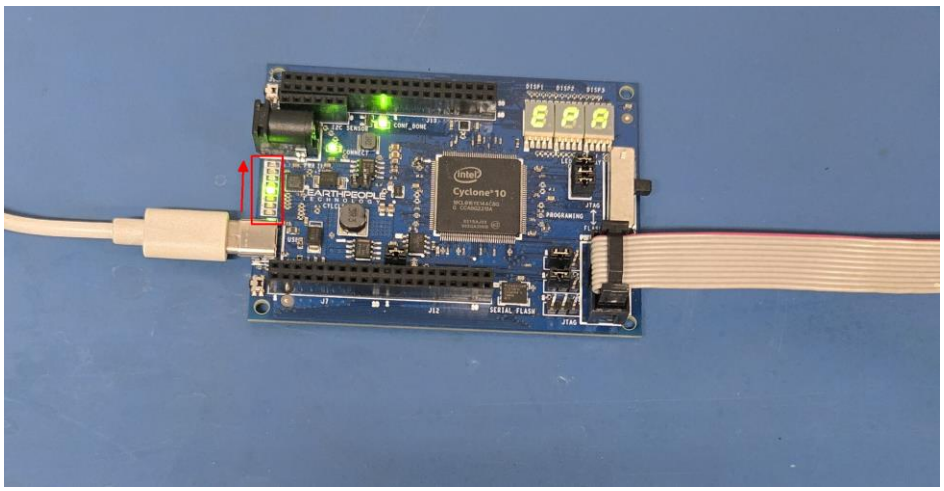
Ensure that the “Heartbeat” function is displayed on the Green User LEDs (repeating sweep of the LEDs)



Under LED Controls, press the “Shift Right” button, Notice the change in direction of the LED Heartbeat.

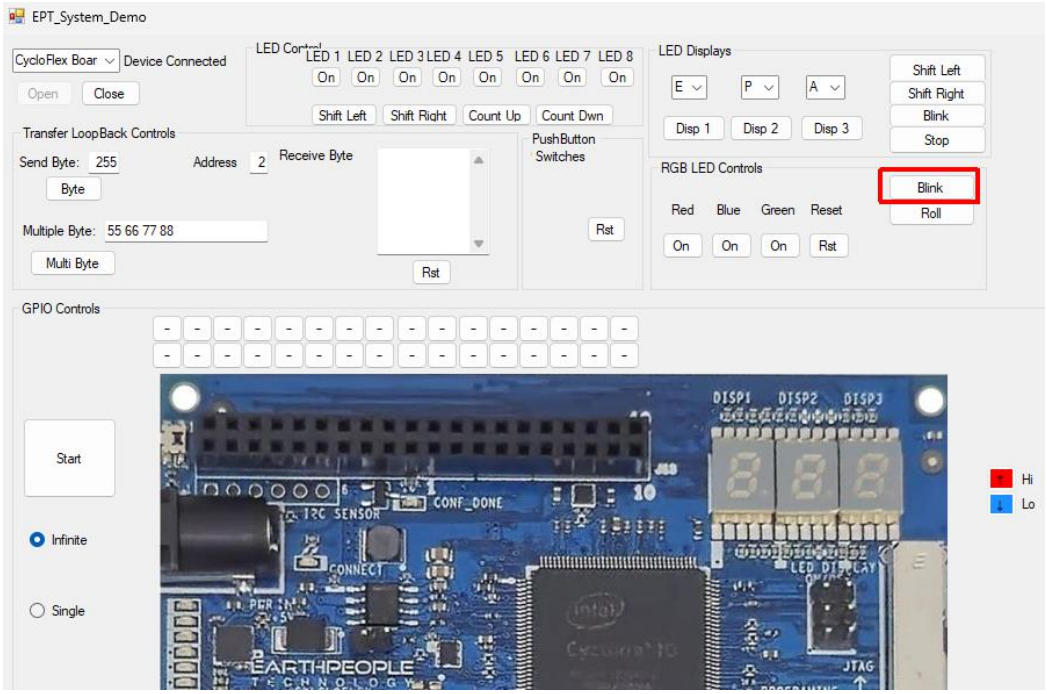


Ensure that the “Heartbeat” function changes direction.

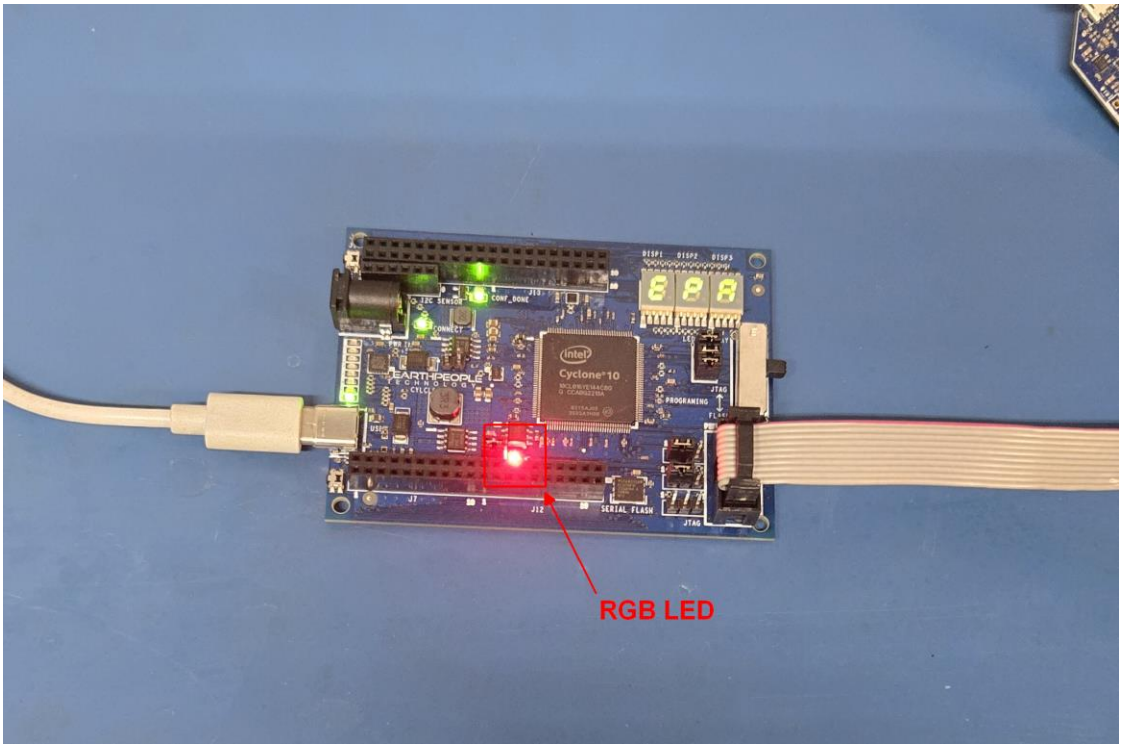


The user can click on the other buttons in the LED Controls Group to see the change to LED pattern.

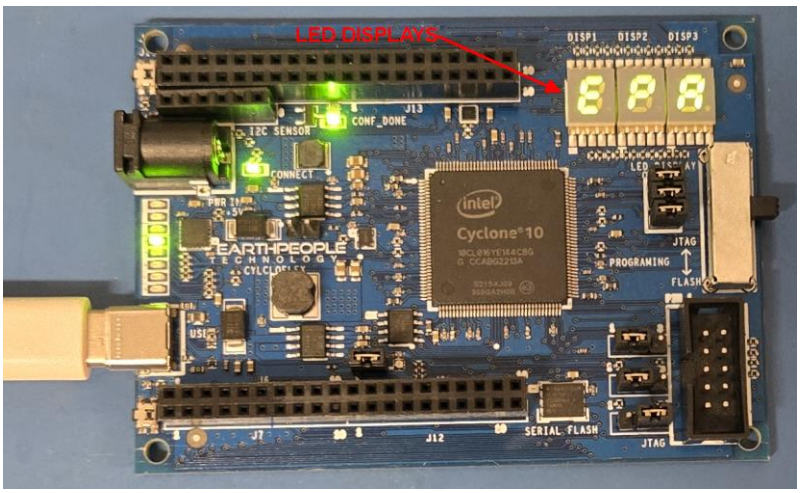
Next, under the “RGB LED Controls”, click the “Blink” button



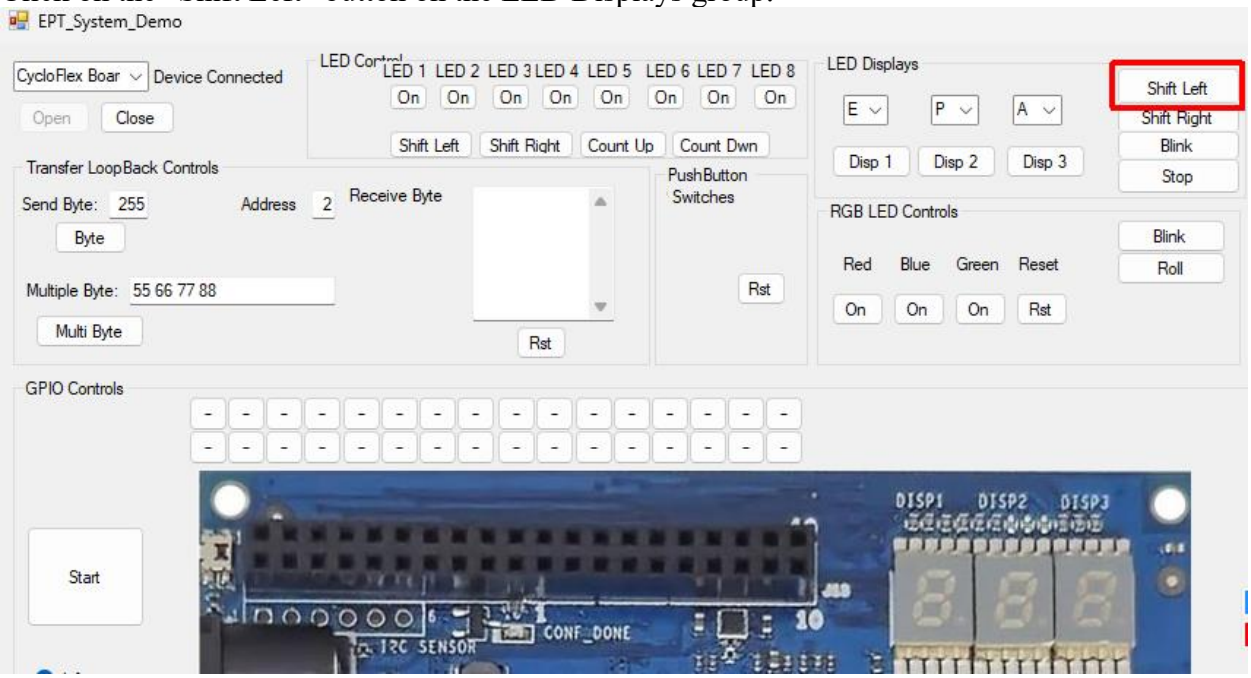
Ensure that the RGB LED goes through its full test of Red, Blue and Green colors



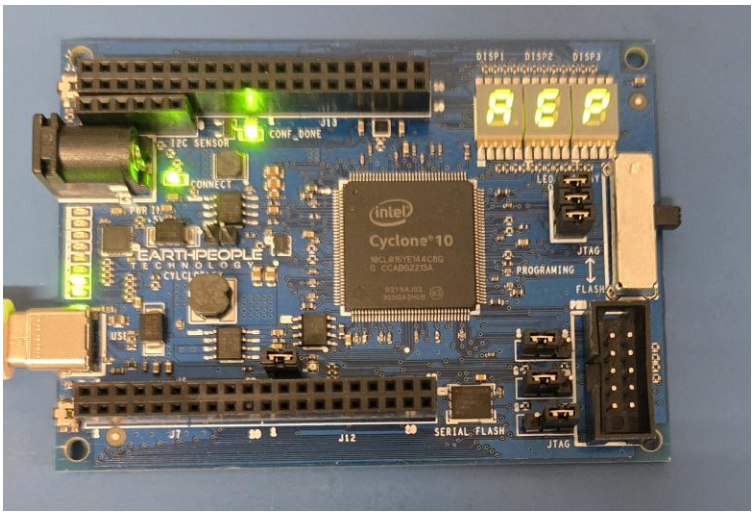
Under the “LED Displays” group box, the characters “EPA” will be displayed on the three LED Displays. Locate the three displays on the CycloFlex in the upper right corner.



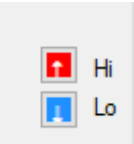
Click on the “Shift Left” button on the LED Displays group.

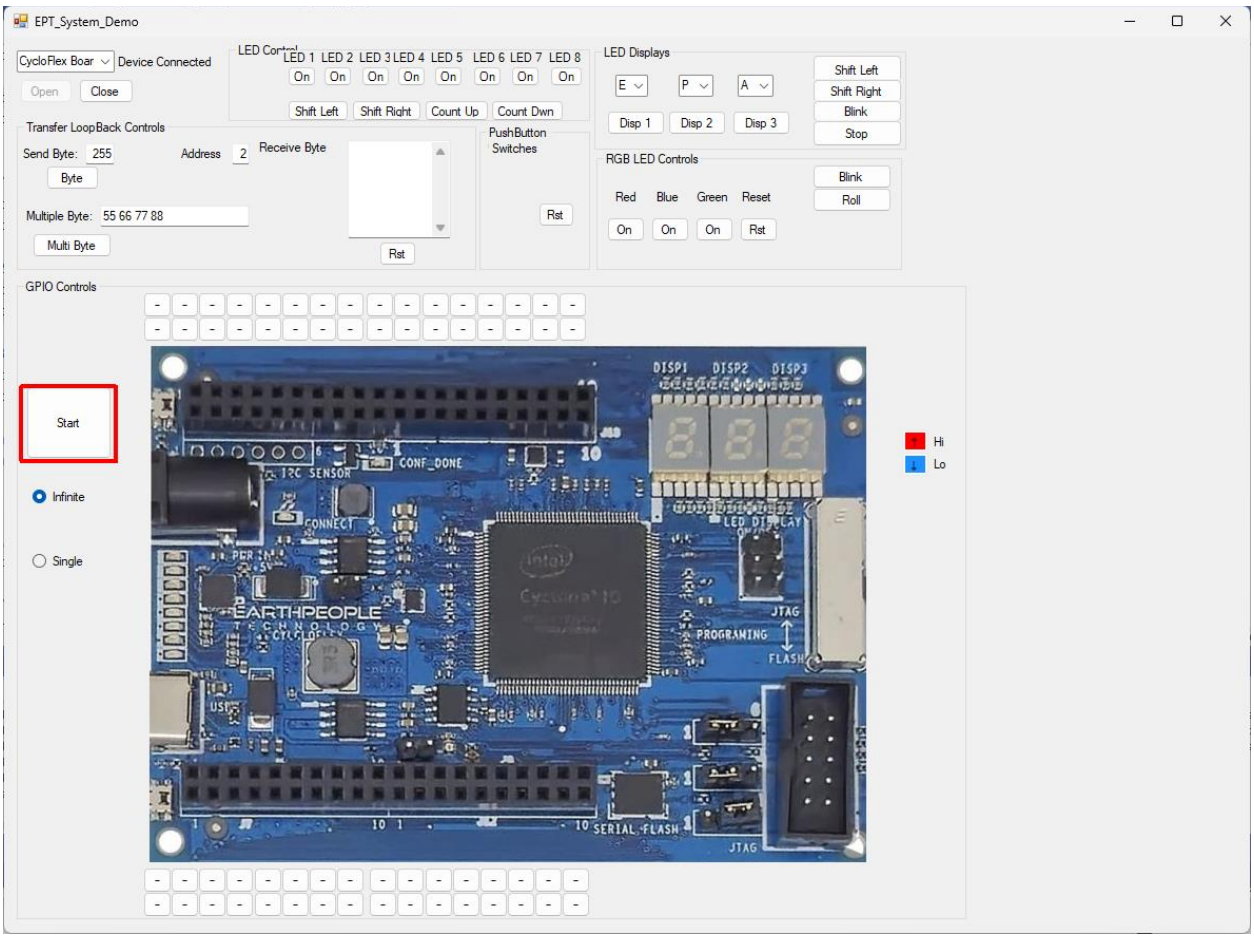


Take note that the LED Display Characters shift to the Left at a 1 second rate.

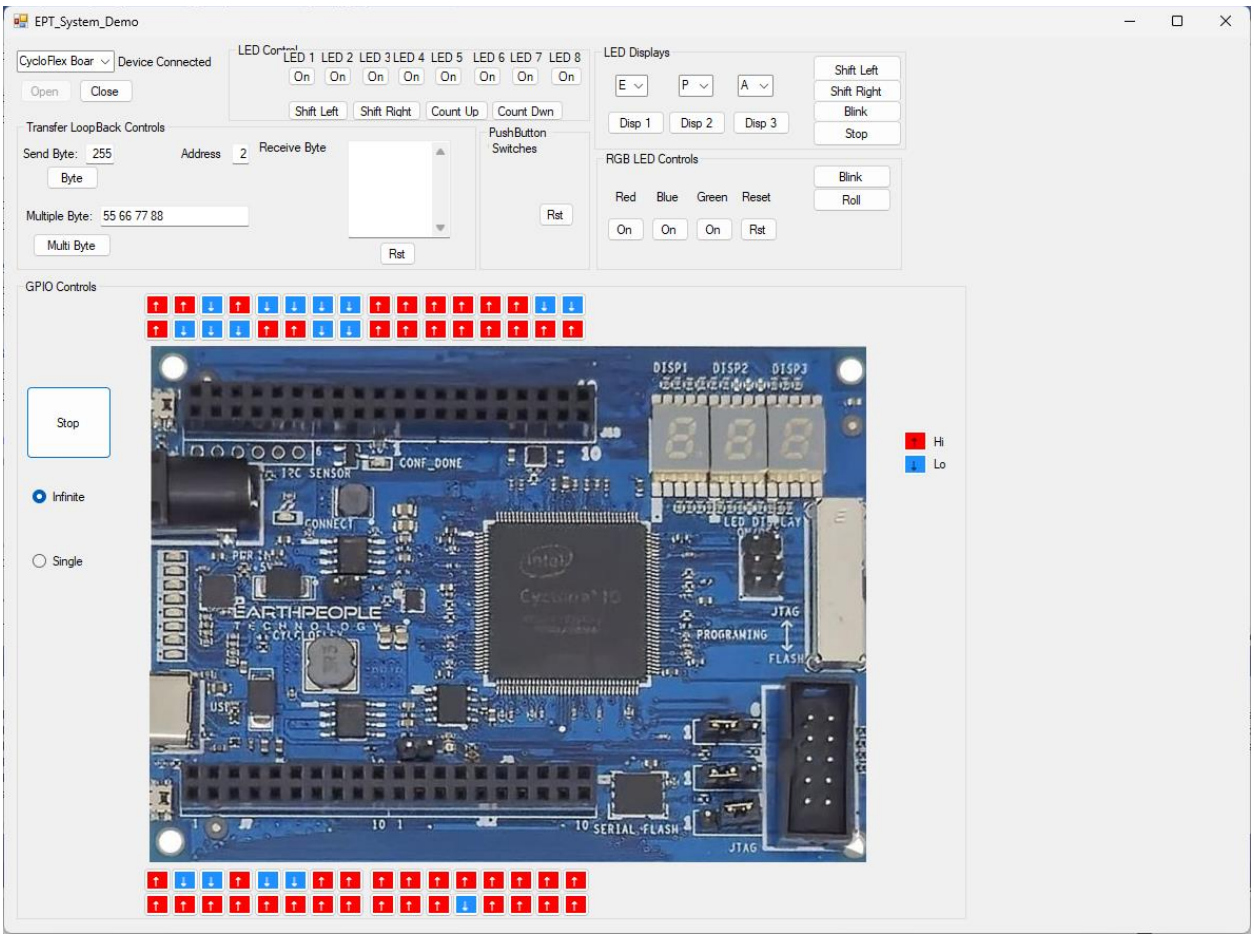


To exercise the Block Transfer EndTerm, click the “Start” button in the GPIO Controls group. The CycloFlex will sample the state of selected Input pins of the FPGA that is connected to a board edge connector. The System Demo Window will display the results of each pin, Hi or Lo



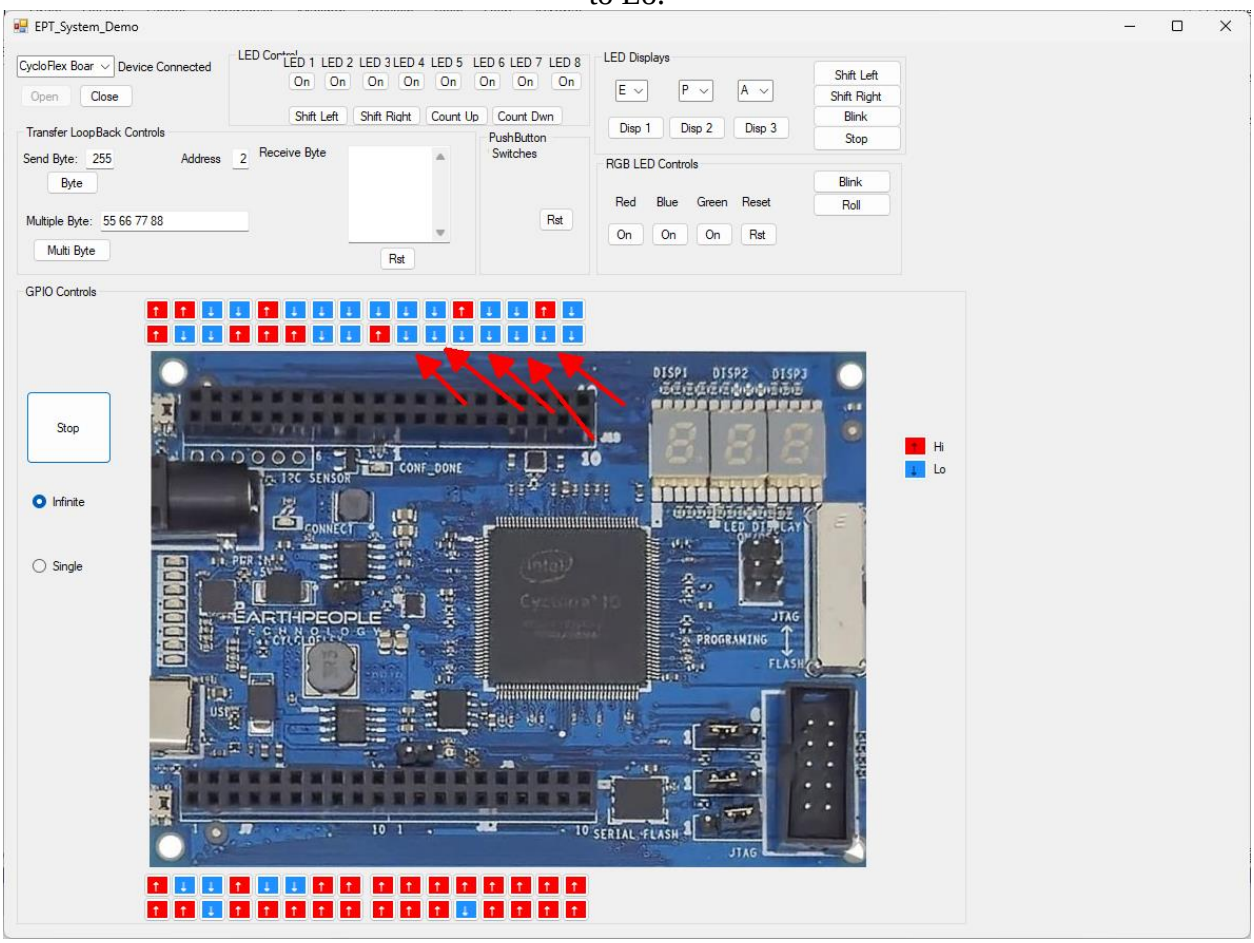


The results of each pin are displayed next to the image of the CycloFlex in separate buttons.



Selected inputs are attached to the User LEDs, the RGB LED, and the LED Displays. So, a mixture of alternating Hi and Lo indications is normal. The user can make contact with the GPIO connector pins at the bottom of the board at the XIO_4. Notice the GPIO pins will move from Hi

to Lo.



The user can also connect +3.3V or Ground using a jumper to any of the GPIO's in the board edge connector. Be sure NOT to connect +3.3V to a Ground pin or Ground to a +3.3V pin. Consult the Data Sheet for the location of the Inputs on the connectors.

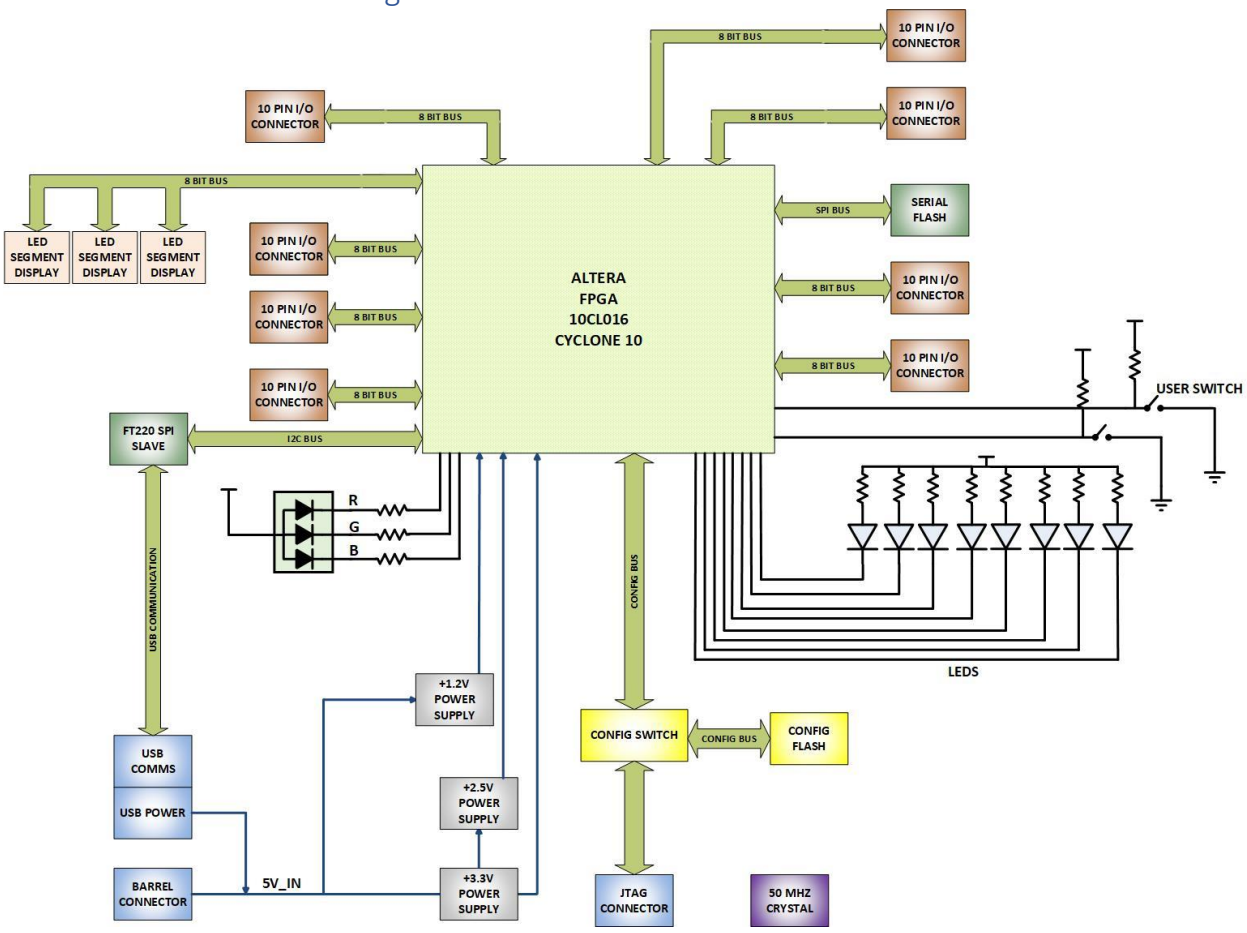
This descibes the EPT System Demo Software for the CycloFlex Board. All source code for the FPGA and Host PC are available on the EPT_FPGA_SYSTEM_PROJECT_1.0_DVD. The following sections of this user manual can be used to determine how to make changes to the EPT System Demo FPGA code as well as the Visual C# project.

3 CycloFlex Description

3.1 PCB Footprint

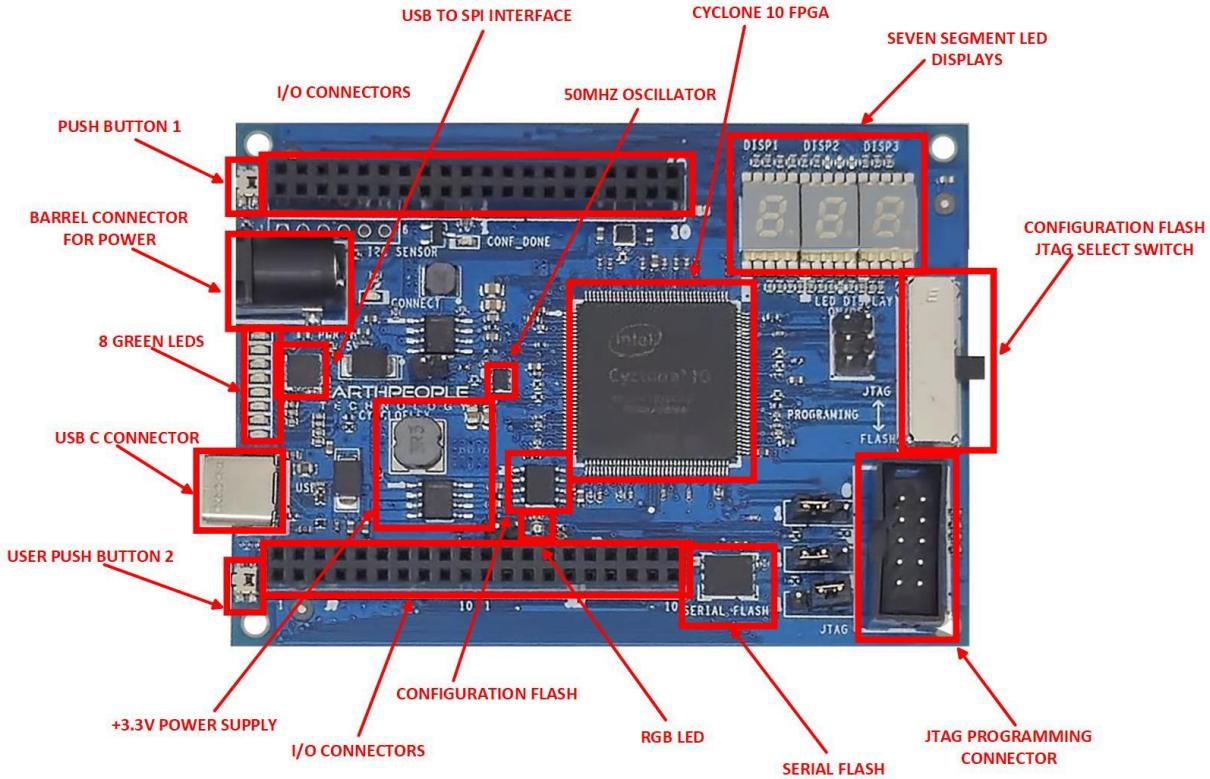
The PCB is 94mm x 65mm with four mounting holes (M2 metric screws) spaced as shown in the figure. These mounting holes are electrically isolated from all signals on the CycloFlex. The two connectors (USB and DC power) overhang the PCB by approximately 1mm in order to accommodate mounting within an enclosure.

3.2 Functional Block Diagram



3.3 CycloFlex Specifications

- Cyclone 10 10CL016 FPGA From Altera
- 16,000 Logic Elements, 504 Kbits of SRAM memory and four clock module PLLs;
- 4 Mbit On Board Configuration Flash
- 64 Inputs/Outputs available at connectors on board
- 8 Green User configurable LEDs 2 Pushbutton Switches
- Three Seven Segment LED Displays
- On Board Serial Flash
- Two Power options: Standard USB (+5V @ 2Amp) Using USB-C connector;
- 5mm Barrel Connector Accepts +5V @ 3Amp
- 50MHz Oscillator
- On board USB to Slave SPI Adapter for high speed communications



3.4 Cyclone 10 FPGA

The CycloFlex includes the Altera 10CL016YE144C8G FPGA (Cyclone 10). It is an EQFP 144 pin package. The chip is optimized for low cost and low static power, making them ideal for high-volume and cost-sensitive applications. The Cyclone 10 is based on 20nm chip architecture from Intel. It has internal speed of 402MHz Clock Tree performance. The parameters for the 10CL016:

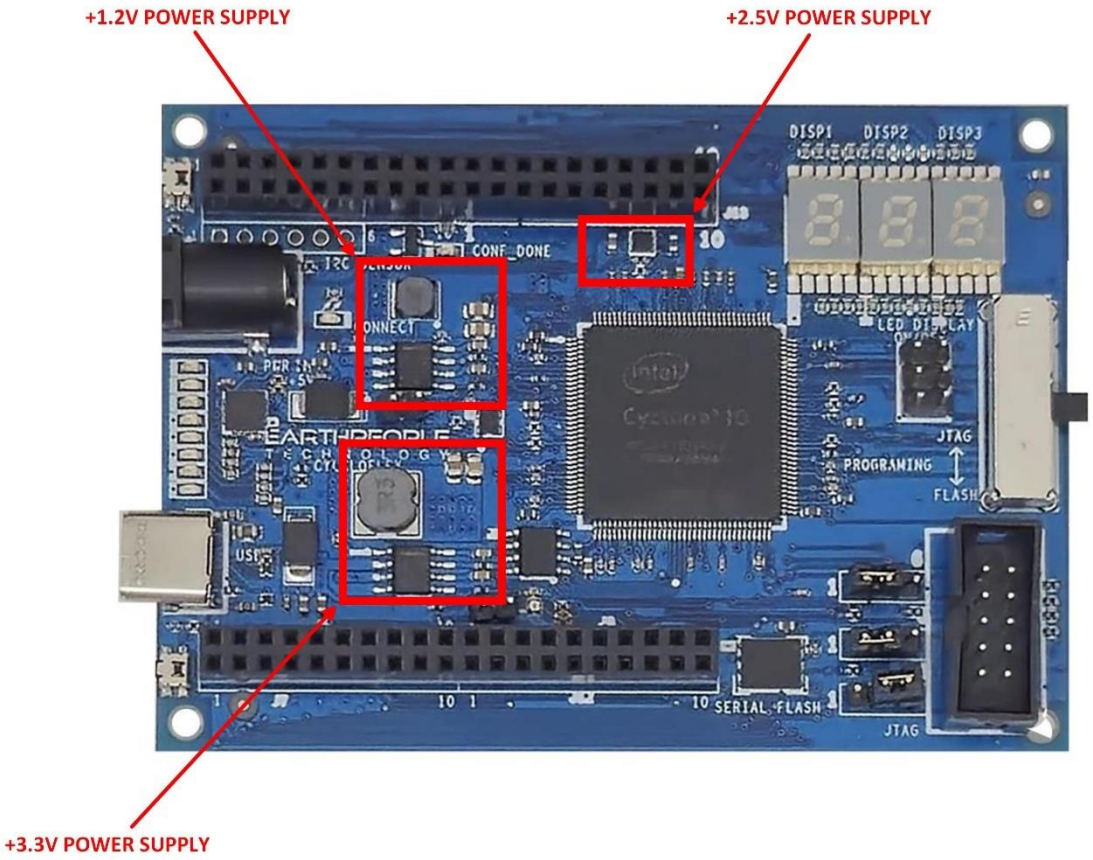
Parameter	**10CL016**
LEs (Logic Elements)	16,000
Block memory (Kb)	504
18 x 18 multipliers	56
Phase-locked loops (PLLs)	4
Maximum I/O	78

3.5 Power Supply

The CycloFlex board has three discrete power supplies on board.

- +3.3VDC
- +2.5VDC
- +1.2VDC

These supplies provide power for the FPGA and the various components on the board. The +2.5V and +1.2V supplies are only used internal to the board. The +3.3V supply is available to the user to power off board electronics. The +3.3V supply is available at the following User I/O connections.



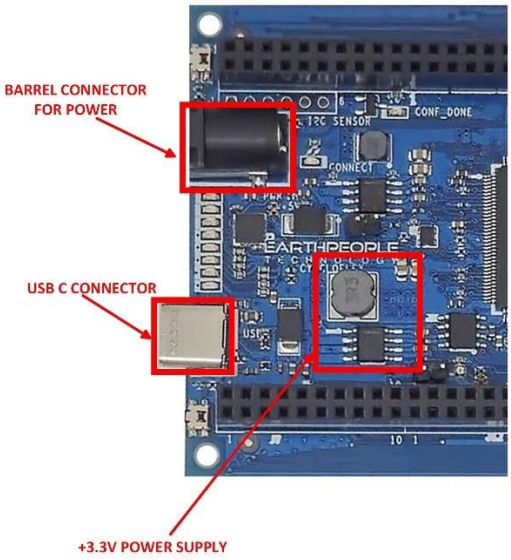
Connector	Pin Number	Power Signal	Description
J8	9	+3.3V	The User +3.3V pins of the connectors have full access to the Power Supply with No Current Limiting. The User must take caution not to sink
	10	+3.3V	
J11	9	+3.3V	

	10	+3.3V	more than 1.5A from a combination of all +3.3V pins.
J14	1	+3.3V	
J6	9	GROUND	
	10	GROUND	
J7	9	GROUND	
	10	GROUND	
J9	9	GROUND	
	10	GROUND	
J10	9	GROUND	
	10	GROUND	
J12	9	GROUND	
	10	GROUND	
J14	2	GROUND	

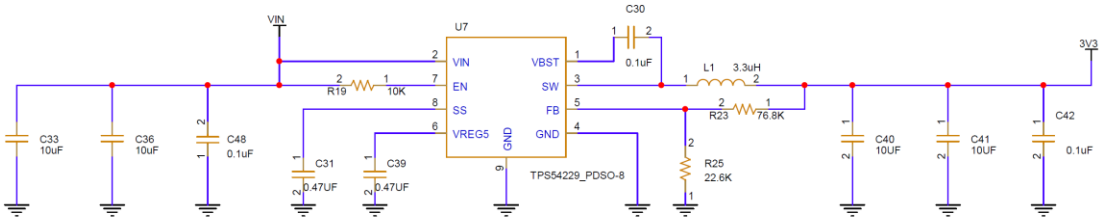
The CycloFlex is designed to be operated from one of two different power sources:

- USB-C cable from Laptop/PC.
- +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.

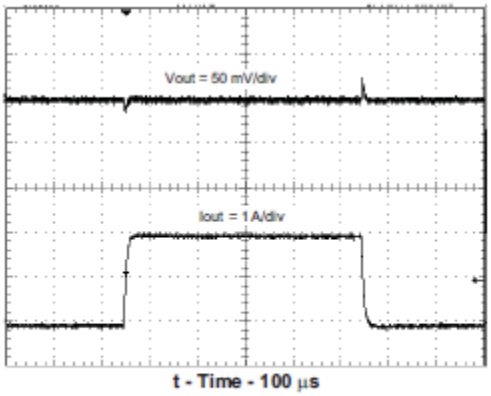


The core of the +3.3VDC Power Supply is the Texas Instruments TPS54229 chip.



CycloFlex +3.3VDC Switching Power Supply

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

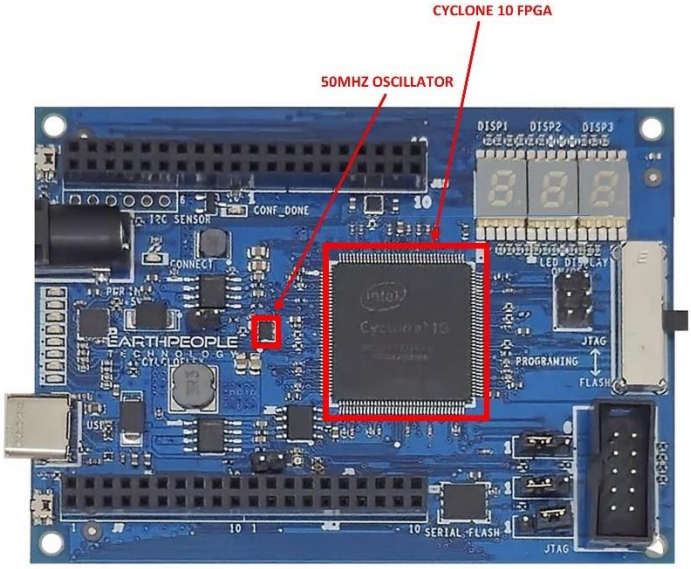


1.05-V, 50-mA to 2-A LOAD TRANSIENT RESPONSE

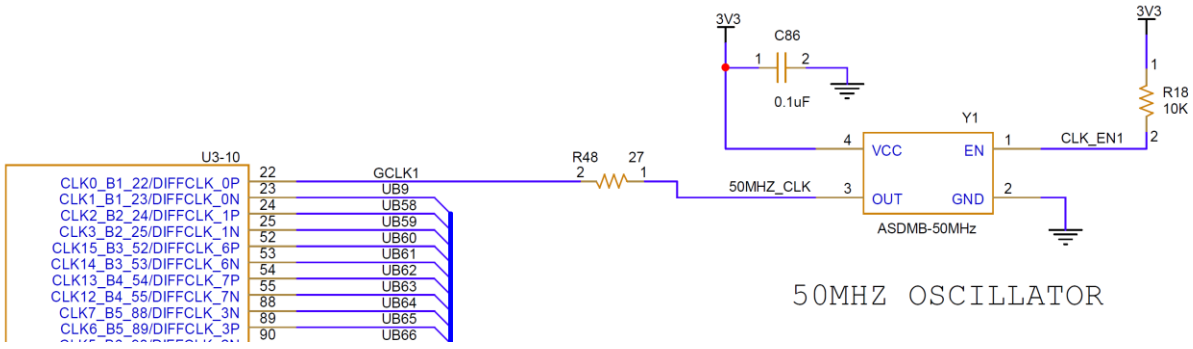
The +3.3VDC includes a 3.3uH Inductor capable of 3.2A RMS. Coupled with two 10uF 25V Ceramic Capacitors, this supply can deliver ultra stable load to the FPGA while at the same time providing power to User needs via the I/O connector pins. This supply provides a lot of power and takes up a small amount of real estate on the CycloFlex board.

3.6 Clock Domain

The CycloFlex provides an external clock domain to the Cyclone 10 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.

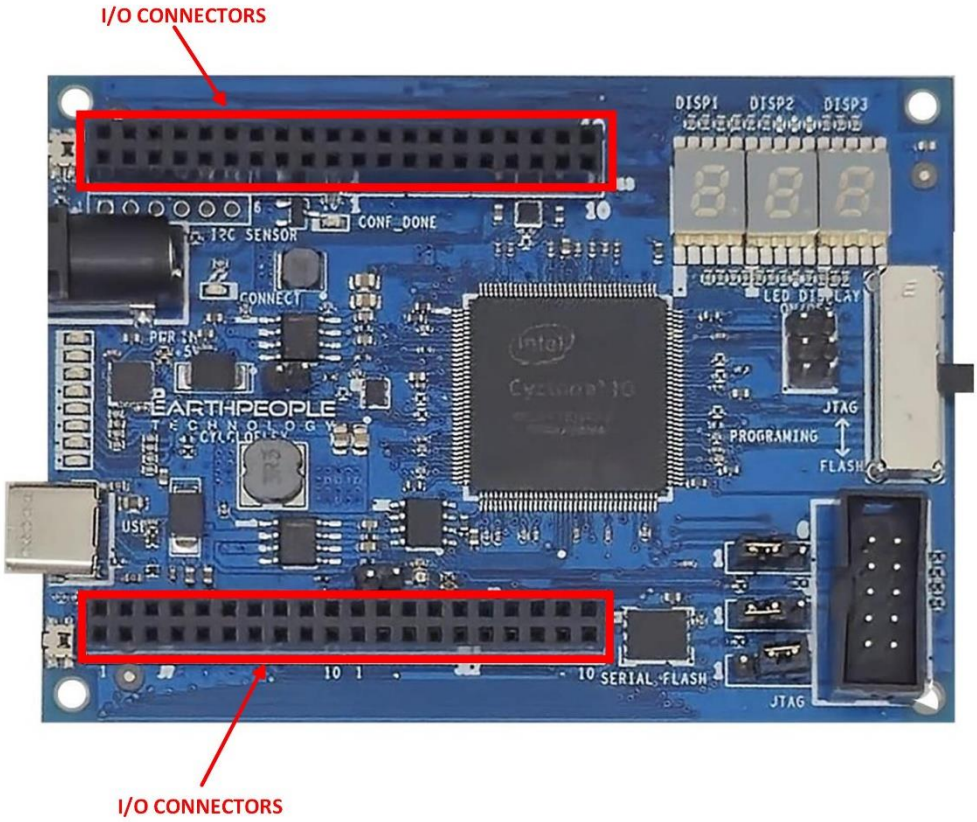


The oscillator is held operational when the enable signal is high.

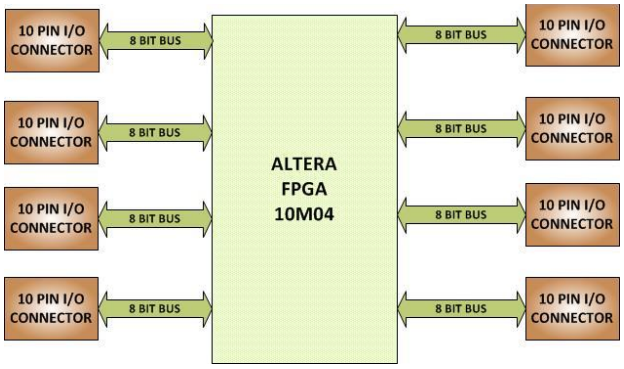
Parameter	Min	Max	Units
Overall Frequency Stability:	-50	+50	ppm
Operating Temperature:	0	+70	°C
Output Load:		25	pF
Supply Current	9	16	mA

3.7 Digital I/Os

The CycloFlex 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.

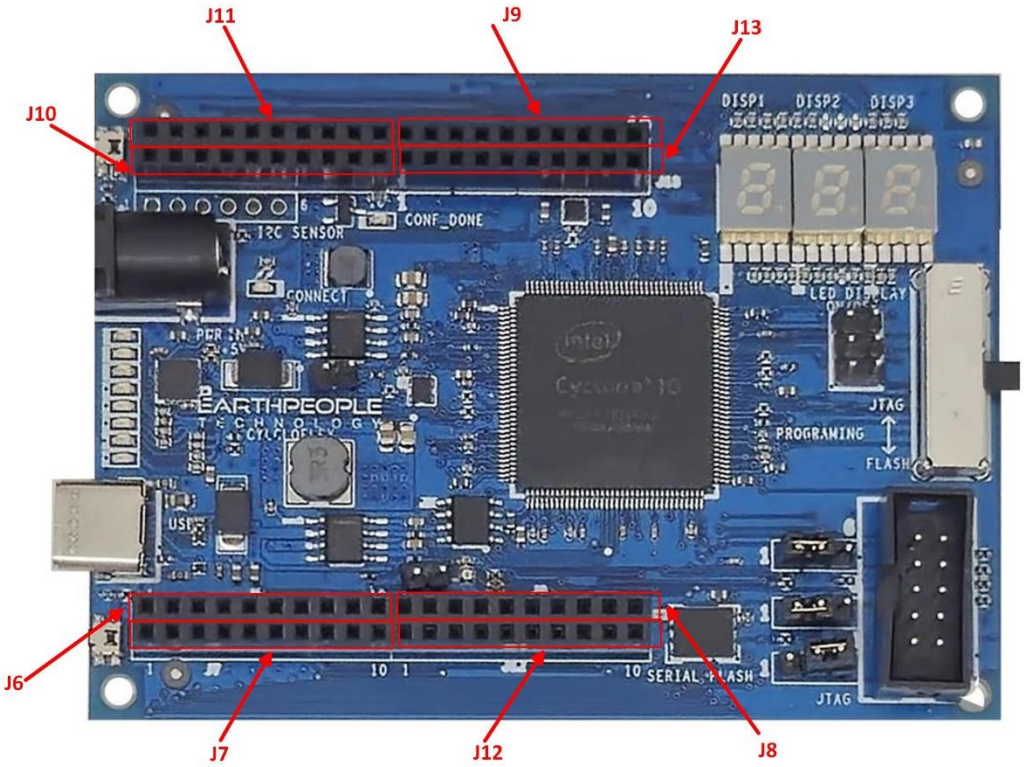


All I/O's are organized into separate banks of the FPGA. There are eight banks. These different banks provide different output speed technologies. Programmable Open Drain The optional open-drain output for each I/O pin is equivalent to an open collector output. If it is configured as an open drain, the 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.

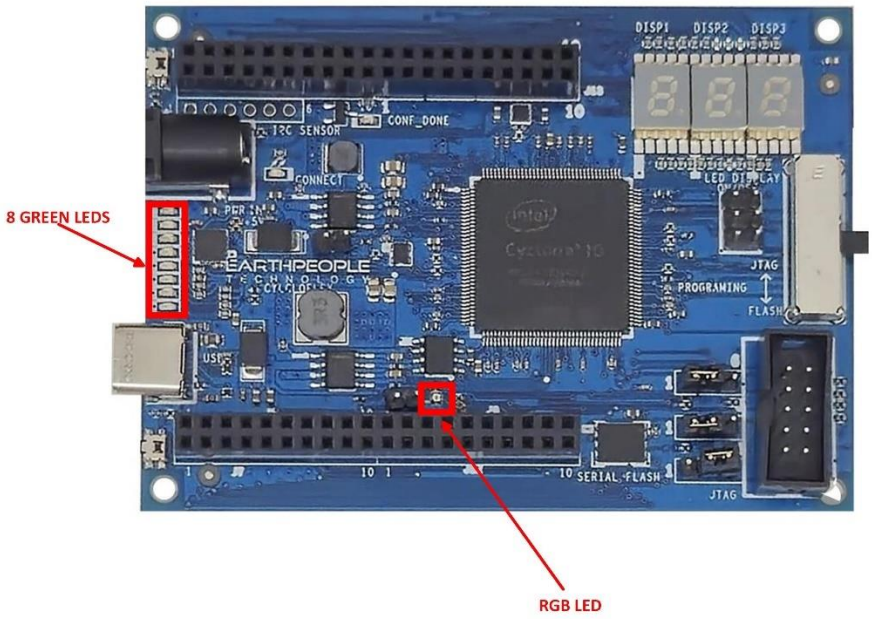
To provide maximum flexibility to system designers, all FPGA I/O are intrinsically bidirectional. Based on the I/O direction specified in the configuration file, each I/O can be configured as input-only, output-only, or bidirectional. While the output buffer of a bidirectional I/O has always included 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.

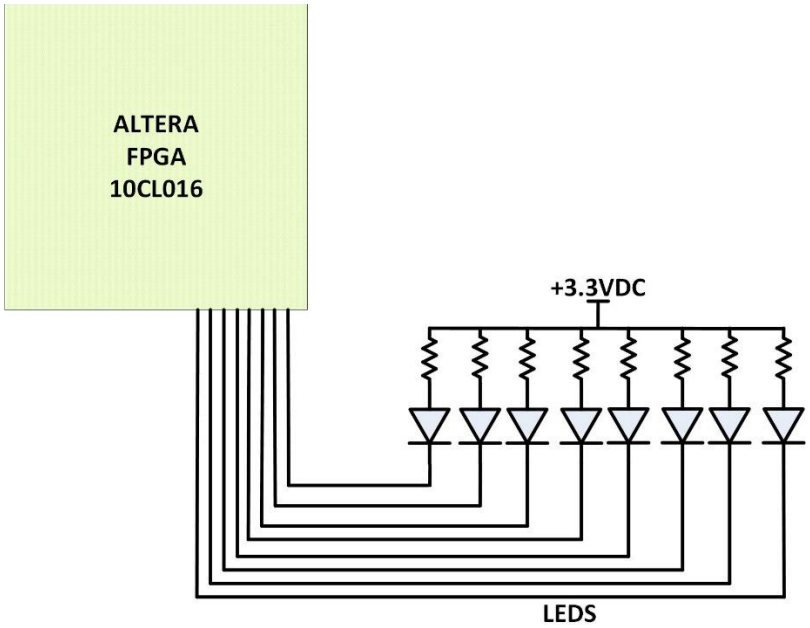


Connector	CycloFlex FPGA Label	
J11	XIO_1	
J9	XIO_2	
J10	XIO_3	
J13	XIO_4	
J6	XIO_5	
J8	XIO_6	
J7	XIO_7	
J12	XIO_8	

3.8 LEDs

The CycloFlex includes eight user LEDs and One RGB LED. These LEDs are fully controllable by the FPGA. They are directly driven from the FPGA. Each LED is connected to its own series





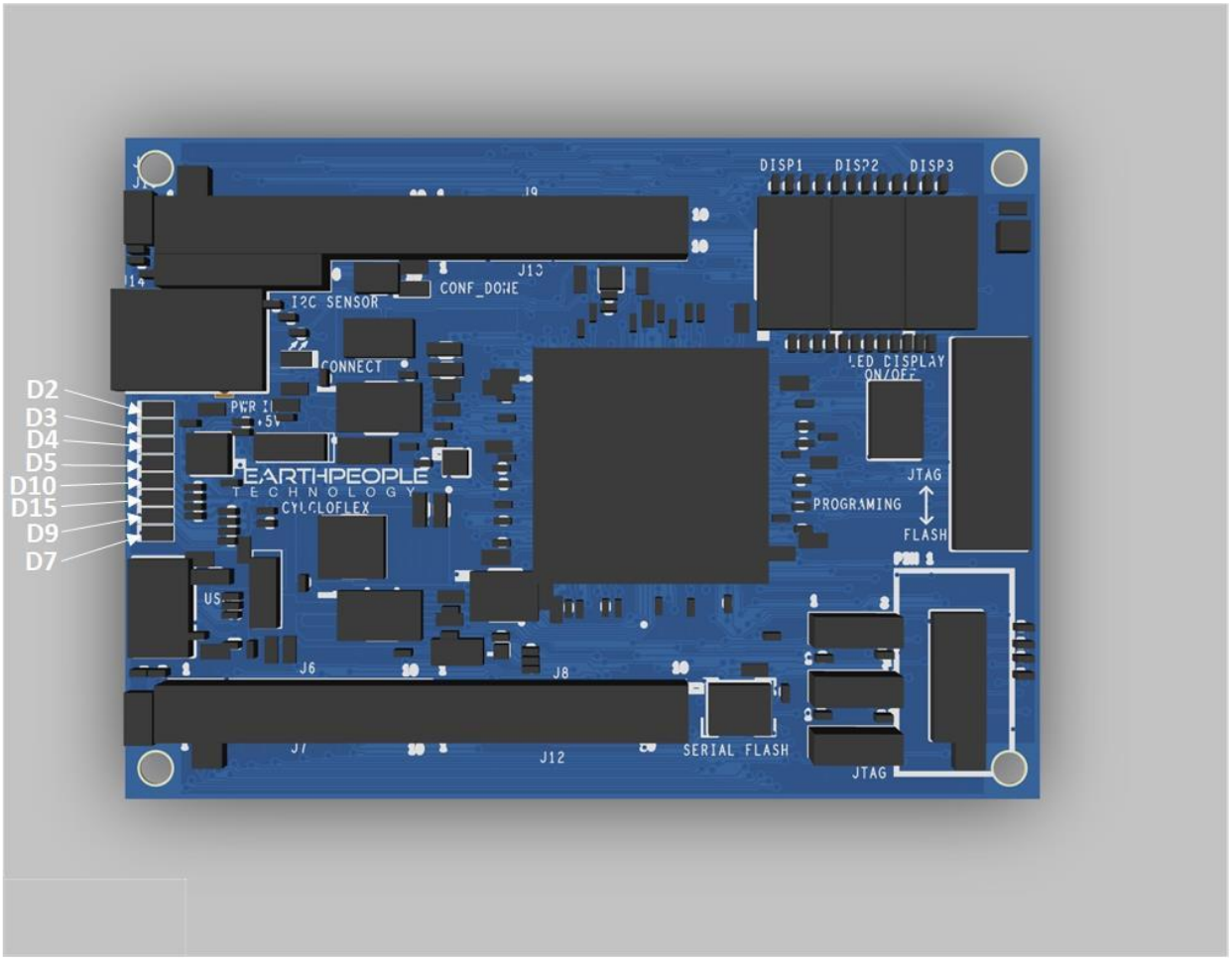
Resistor which provides current limiting. The LEDs are attached in a Current Sink configuration. This means the Anode of the LED is permanently connected to +3.3V via resistor. The Cathode is connected to Ground via a GPIO pin from the FPGA. Each LED Anode is connected with a 220 Ohm series resistor for each LED. This provides the following current through the LEDS

$$I_{LED} = \text{ Current flowing through the LED}$$
$$V_0 = Voltage\ applied\ to\ the\ LED$$
$$V_F = Forward\ Voltage\ Drop\ of\ the\ LED$$
$$R \quad = Series\ Resistor\ Value$$
$$I_{LED} = \frac{V_O - V_F}{R}$$

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

$$I_{LED} = 5.9mA$$

LED Number	CycloFlex Schematic Signal	LED Signal Name	Cyclone 10 Pin Number
D2	UB0	USER_LEDS[0]	10
D3	UB1	USER_LEDS[1]	32
D4	UB2	USER_LEDS[2]	33
D5	UB3	USER_LEDS[3]	42
D10	UB25	USER_LEDS[4]	114
D15	UB23	USER_LEDS[5]	112
D9	UB24	USER_LEDS[6]	113
D7	UB22	USER_LEDS[7]	106



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.

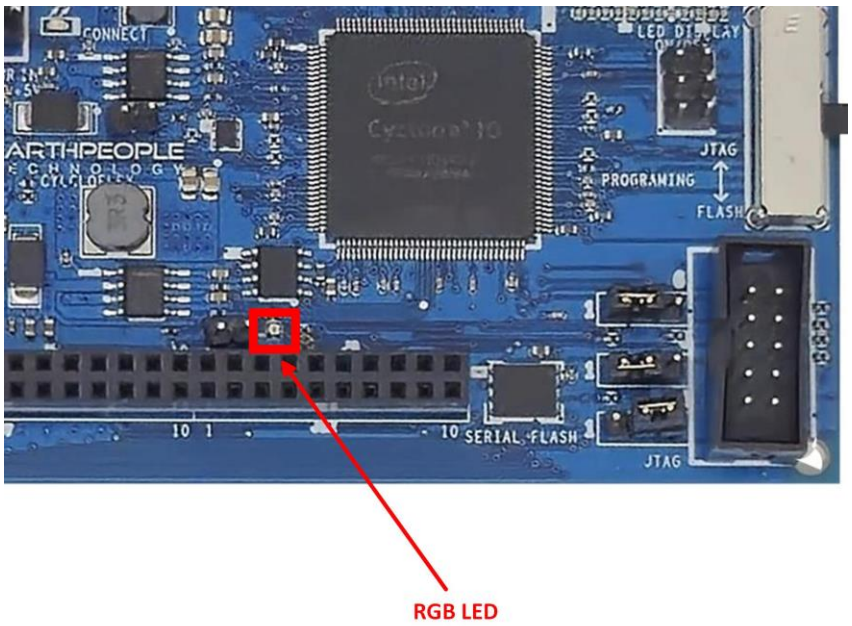
```
//*****************************************************************
//* Module Declaration
//*****************************************************************

module EPT_10CL016_AF_T2_Top (


    input   wire                CLK_50MHZ,
    input   wire                RST_N,

    //SPI Communications FT-220X
    output  wire                SPI_SCLK,
    input   wire                SPI_MISO,
    inout   wire                SPI_MIOSIO_1,
    output  wire                SPI_SS,

    //User LED Control Registers
    output reg   [7:0]          USER_LEDS,

    //User GPIOs
    input   wire  [3:0]         XIO_1,       //XIO -- UB

    //User RGB LED Control Registers
    output  reg                 RGB_LED_BLUE,
    output  reg                 RGB_LED_RED,
    output  reg                 RGB_LED_GREEN,
```

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.

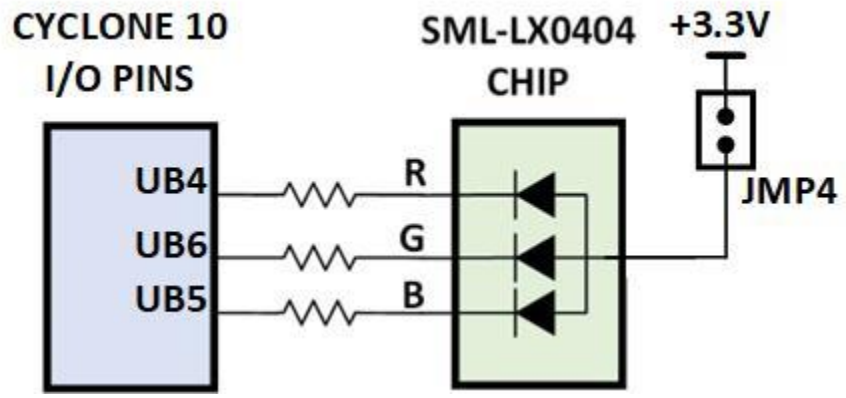
```
//-------------------------------------------------
// Set the USER_LEDS outputs
//-------------------------------------------------
always @(posedge CLK_50MHZ or negedge RST)
begin
  if(!RST)
      USER_LEDS <= 8'hz;
  else
  begin
    if(led_reset)
       USER_LEDS <= 8'hz;
    else if(state[LOAD_LEDS])
    begin
      if ( led_reg[0] )
          USER_LEDS[0] = 1'b0;
      else
          USER_LEDS[0] = 1'bz;
      if ( led_reg[1] )
          USER_LEDS[1] = 1'b0;
      else
          USER_LEDS[1] = 1'bz;
      if ( led_reg[2] )
          USER_LEDS[2] = 1'b0;
      else
          USER_LEDS[2] = 1'bz;
      if ( led_reg[3] )
          USER_LEDS[3] = 1'b0;
      else
          USER_LEDS[3] = 1'bz;
```

3.9 RGB LED

The CycloFlex also includes one RGB LED. It is a SML-LX0404 LED RGB chip. Each LED with the chip is attached to a current limiting resistor. Each LED leg is controlled by a pin from the Cyclone 10 FPGA.



The SML-LX0404 chip is a current sink and are connected to pins on the Cyclone 10. The anode is connected to +3.3V. The series resistors are calculated for current limiting based on +3.3V.



Each series resistor uses a 220 Ohm in a resistor array. The GPIO from the FPGA can be used to control the On/Off of each LED individually. The LED RGB chip uses +3.3V supply and each LED is configured in a Current Sink scheme.

In order to light up the each LED, the user code must assert a zero on the associated signal for the LED. To turn off the LED, assert High Z on the signal.

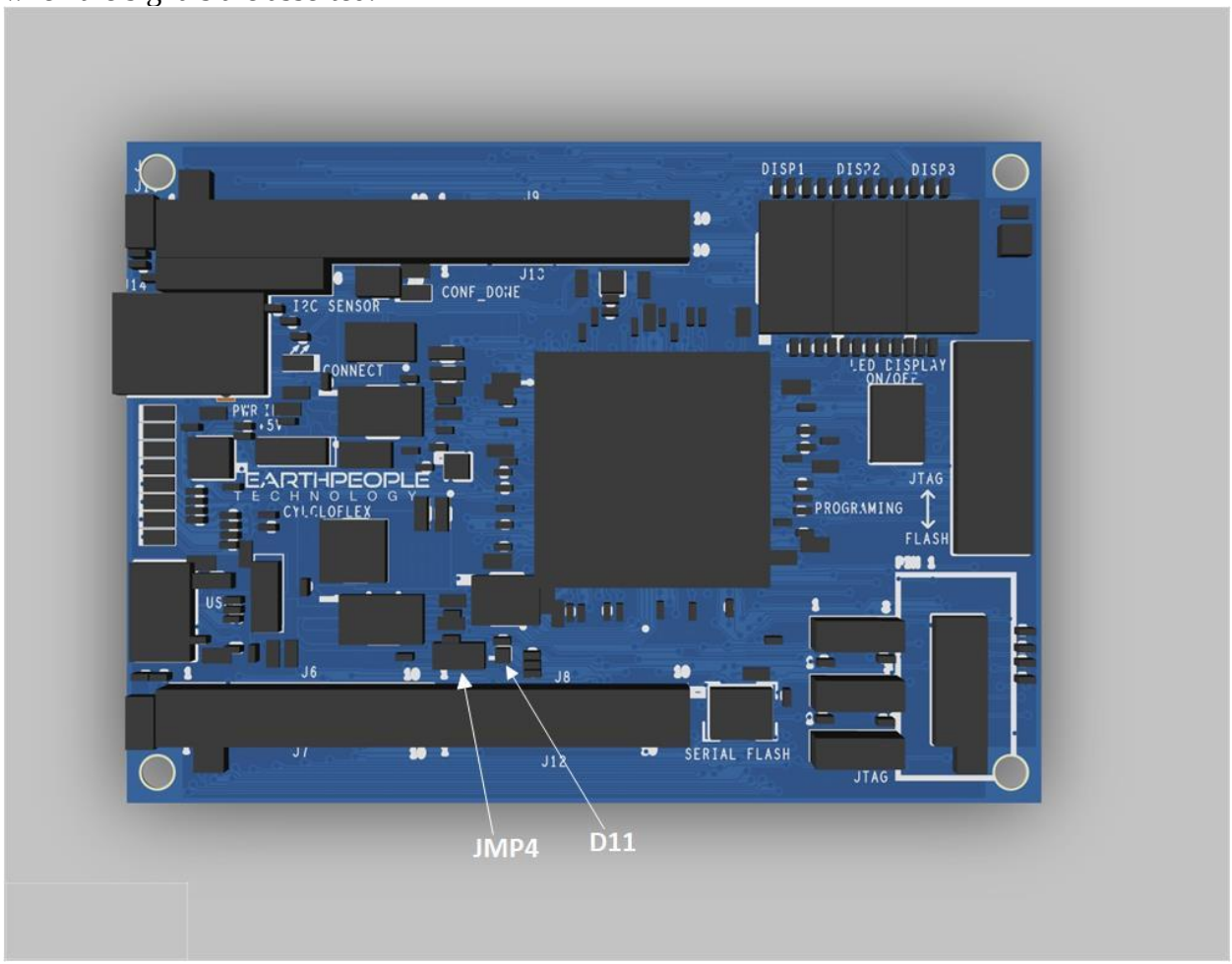
The LED RGB signals are organized on the following pins from the Cyclone 10 chip:

LED Number	CycloFlex Schematic Signal	RGB Signal Name	Cyclone 10 Pin Number
D11	UB4	LED_RED	43
D11	UB5	LED _BLUE	44
D11	UB6	LED_GREEN	49

The RGB LED also includes a jumper to turn off the entire chip. This jumper is inline with the +3.3V source to the chip. If the user would like to use the User Bus signals:

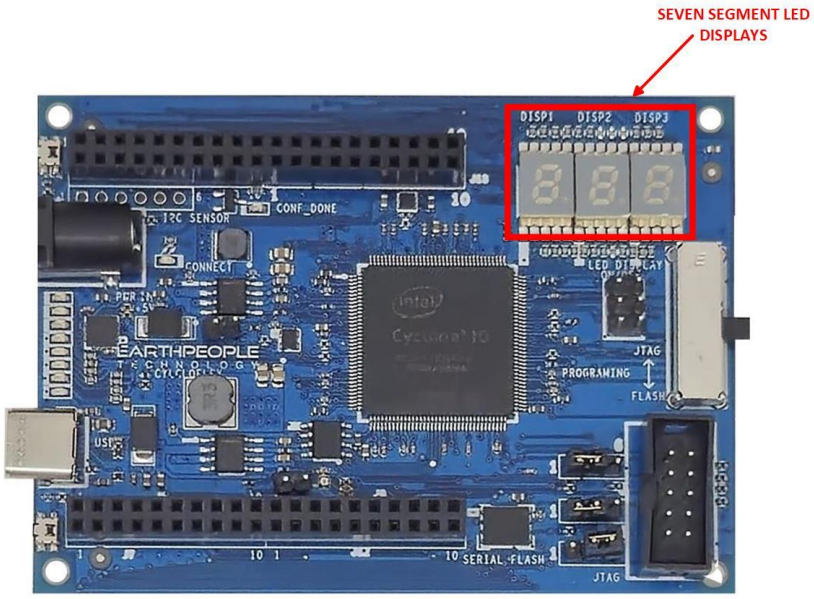
- UB4
- UB5
- UB6

For other IO activities, remove the jumper JMP4 and the RGB LED will not light up when the signals are asserted.

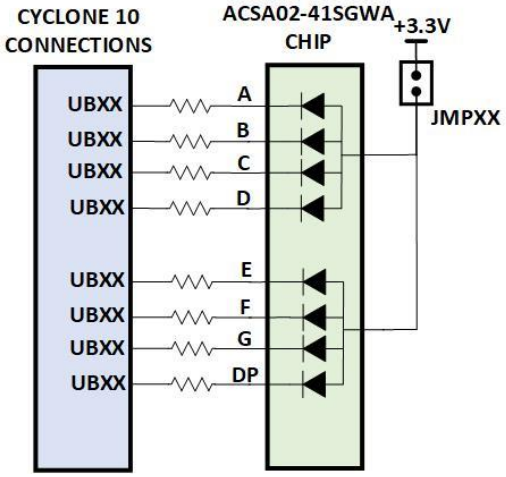


3.10 Seven Segment LED Displays

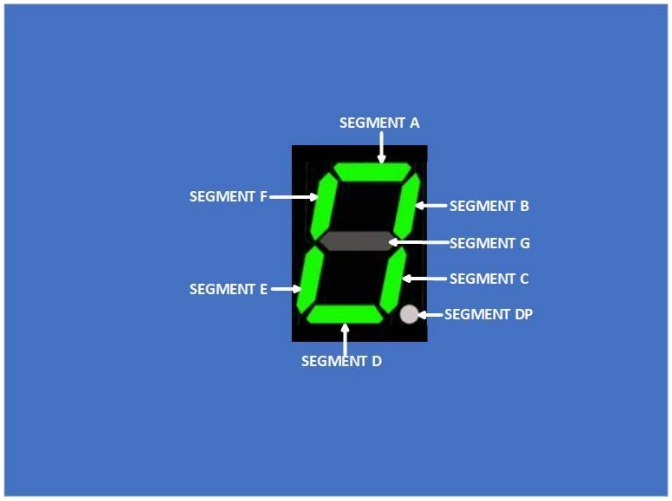
The Seven Segment LED board uses one ACSA02-41SGWA-F01 manufactured by Kingbright. The Display is 0.2 inch digit height. It has low current operation and excellent character appearance. The Super Bright Green source color devices are made with Gallium Phosphide Green Light Emitting Diode.



The GPIOs from the Cyclone 10 are used to control the On/Off of each LED individually. The Seven Segment LED Board is for use only with +3.3V.



The ACSA02-41SGWA-F01 chip is a current sink connected to Cyclone 10 IO pins. The anode should be connected to +3.3V. The reason for this is the Series resistors are calculated for current limiting based on +3.3V. Each Segment of the Display is mapped



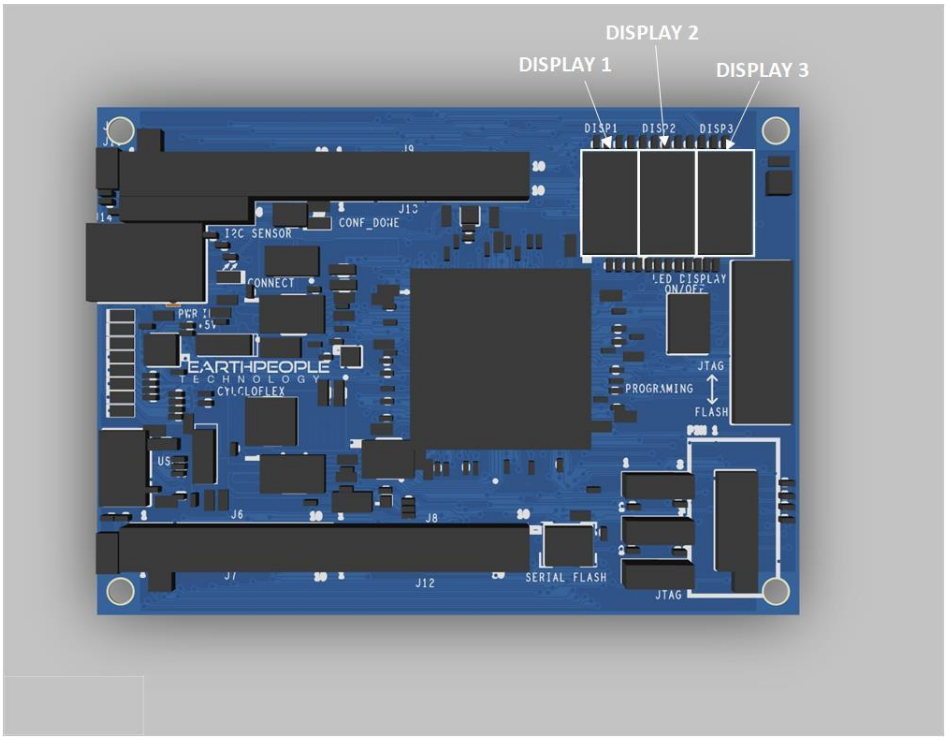
To a pin on the Cyclone 10 FPGA. When the user would like to form a character on the LED Display, the correct segment must be asserted inside the FPGA.

LED Display Number	CycloFlex Schematic Signal	Segment Name	Cyclone 10 Pin Number
1	UB48	Segment A	142
1	UB30	Segment B	141
1	UB56	Segment C	135
1	UB57	Segment D	137
1	UB31	Segment E	136
1	UB47	Segment F	143
1	UB46	Segment G	144
1	UB55	Segment DP	133

LED Display Number	CycloFlex Schematic Signal	Segment Name	Cyclone 10 Pin Number
2	UB18	Segment A	72

2	UB17	Segment B	71
2	UB38	Segment C	69
2	UB16	Segment D	68
2	UB15	Segment E	67
2	UB54	Segment F	132
2	UB14	Segment G	66
2	UB39	Segment DP	65

LED Display Number	CycloFlex Schematic Signal	Segment Name	Cyclone 10 Pin Number
3	UB34	Segment A	105
3	UB35	Segment B	100
3	UB51	Segment C	99
3	UB52	Segment D	101
3	UB53	Segment E	103
3	UB33	Segment F	111
3	UB32	Segment G	119
3	UB50	Segment DP	98



Each segment of each display uses a current sink method to turn on the segment LED. This means the Anode of each segment LED is connected to the +3.3V (via a Jumper) and the Cathode is connected to the IO of the Cyclone 10. To turn on the Segment LED, the FPGA code must apply a low to the signal.

```
//-----------------------------------------------
always @(posedge CLK or negedge RST)
begin
  if(!RST)
  begin
       SEVEN_SEGMENT_LED_1 = 8'hz;
       SEVEN_SEGMENT_LED_2 = 8'hz;
       SEVEN_SEGMENT_LED_3 = 8'hz;
  end
  else
  begin
    if(LED_DISPLAY_ENABLE_1)
    begin
      //LED Display 1 Segment A
      if ( LED_DISPLAY_BYTE_1[0] )
          SEVEN_SEGMENT_LED_1[0] <= 1'b0;
      else
          SEVEN_SEGMENT_LED_1[0] <= 1'bz;

      //LED Display 1 Segment B
      if ( LED_DISPLAY_BYTE_1[1] )
          SEVEN_SEGMENT_LED_1[1] <= 1'b0;
      else
          SEVEN_SEGMENT_LED_1[1] <= 1'bz;

      //LED Display 1 Segment C
      if ( LED_DISPLAY_BYTE_1[2] )
          SEVEN_SEGMENT_LED_1[2] <= 1'b0;
      else
          SEVEN_SEGMENT_LED_1[2] <= 1'bz;

      //LED Display 1 Segment D
      if ( LED_DISPLAY_BYTE_1[3] )
          SEVEN_SEGMENT_LED_1[3] <= 1'b0;
      else
          SEVEN_SEGMENT_LED_1[3] <= 1'bz;
```

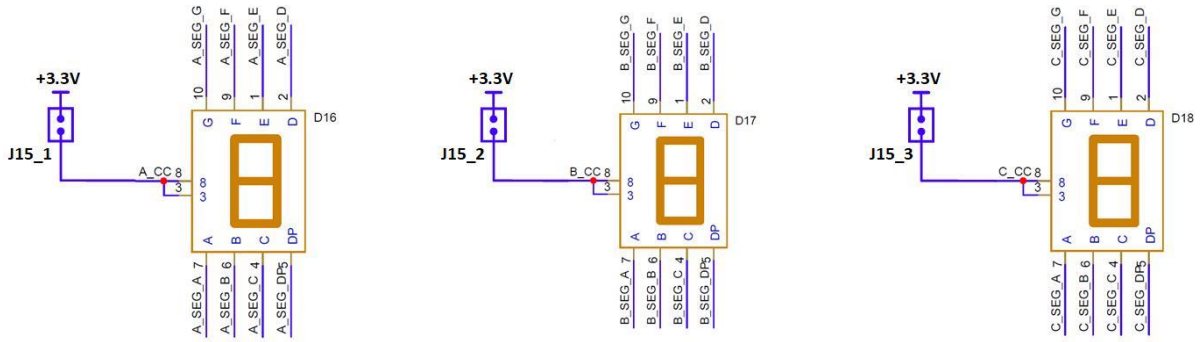
To turn off each segment LED, the FPGA code must apply a tri-state (1'bz) to the signal. In the code section above, the LED Display Byte is received from the Host Software running on the PC (LED_DISPLAY_BYTE_1[7:0]). Each array index represents a segment of the LED Display. The Host PC software and the FPGA code must be aligned with which segment of the Display corresponds to A, B, C, D etc. In the code section above:

- LED_DISPLAY_BYTE_1[0] = LED Display 1 Segment A

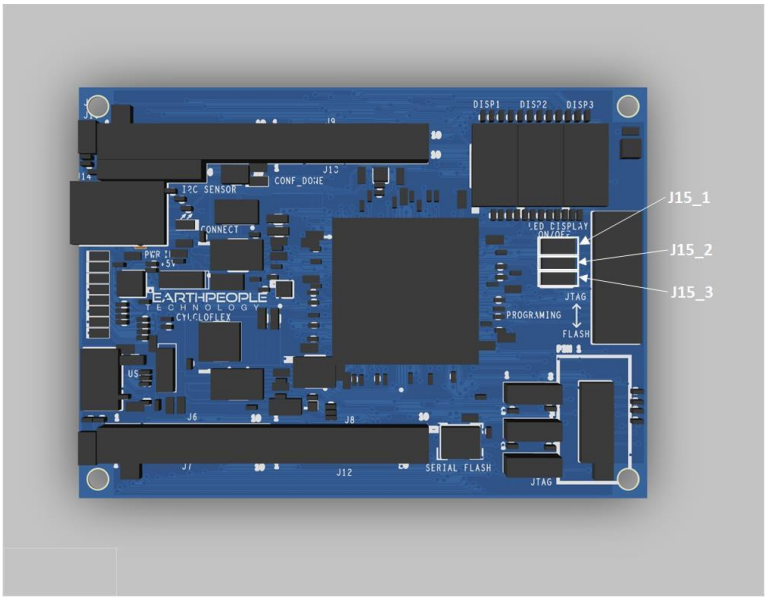
- LED_DISPLAY_BYTE_1[1] = LED Display 1 Segment B
- LED_DISPLAY_BYTE_1[2] = LED Display 1 Segment C
- LED_DISPLAY_BYTE_1[3] = LED Display 1 Segment D
- LED_DISPLAY_BYTE_1[4] = LED Display 1 Segment E
- LED_DISPLAY_BYTE_1[5] = LED Display 1 Segment F
- LED_DISPLAY_BYTE_1[6] = LED Display 1 Segment G
- LED_DISPLAY_BYTE_1[7] = LED Display 1 Segment DP

The Seven Segment LED Displays also include a jumpers to turn off each individual display. Each jumper is inline with the +3.3V source to the chip. The following jumpers control the on/off the following:

LED Display	CycloFlex Schematic Signal	Jumper Schematic Number	J15 Pin Number
LED Display 1	A_CC	J15_1	1,2
LED Display 2	B_CC	J15_2	3,4
LED Display 3	C_CC	J15_3	5,6

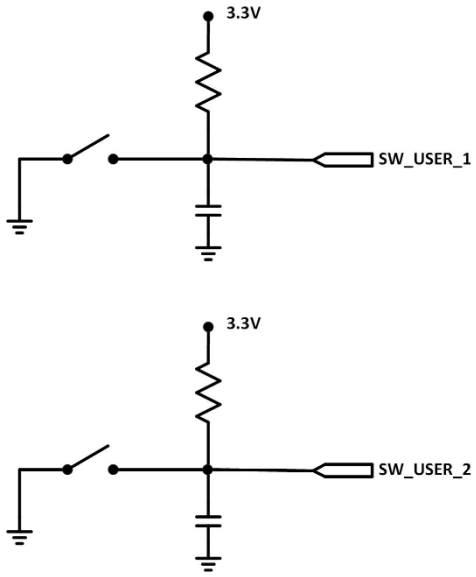


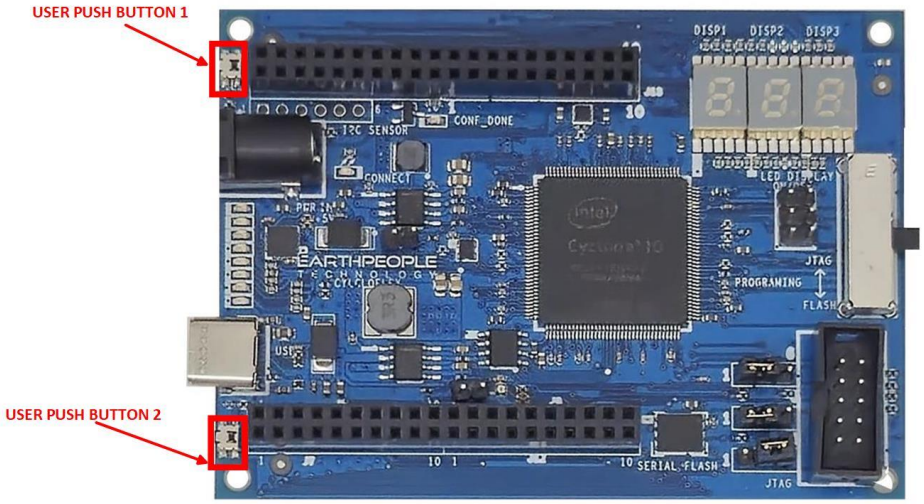
Each of the LED segment signals from the Cyclone 10 FPGA are used as IO's on the connectors. If the user would like to use the IO's for other purposes, the jumper selects will turn off the LED Displays. For other IO activities, remove the jumper J15_X jumpers and the associated LED Display will not light up when the segment signals are asserted.



3.11 PushButtons

The CycloFlex includes two push button switches. Both are momentary contact switches. They include a 1uF cap to ground to debounce both switches.


3.3V
SW_USER_1
3.3V
SW_USER_2



Component	Net Name	Pin Number on Cylcone 10	Signal in EPT Project Pinout
SW1	PB_SWITCH_1	126	PB_SWITCH__1
SW2	PB_SWITCH_2	91	PB_SWITCH__2

Both of these Push Button Switches are configurable as inputs only to the Cyclone 10. The user code can read the state of the switch or use it as an event to trigger some action in the code.

```
//-----------------------------------------------
// Trigger Out Byte. Device can
// send the trigger byte to the host.
//-----------------------------------------------

always @(posedge CLK_50MHZ or negedge RST)
begin
   if(!RST)
       trigger_out <= 0;
   else
   begin
       // Push Button swtiches, a button press will send
       // send the trigger byte to the host.
       if(!PB_SWITCH_1)
          trigger_out <= PB_SWITCH_1_ASSERT;
       else if(!PB_SWITCH_2)
          trigger_out <= PB_SWITCH_2_ASSERT;
       else
          trigger_out <= 0;
   end

end
```

In the above code section, at each rising clock edge of the 50MHz Oscillator the PB_SWITCH_1 and PB_SWITCH_2 are sampled. If either switch are determined low, the "trigger_out' byte assumes the XX_ASSERT byte value.

3.12 Serial Flash

The CycloFlex includes a Serial Flash chip. The chip is P/N: S25FL127SABNFI100Z from Macronix. The S25FL127S device is a flash non-volatile memory product. This device connects to a host system via an SPI. Traditional SPI single bit serial input and output (Single I/O or SIO) is supported as well as optional two bit (Dual I/O or DIO) and four bit (Quad I/O or QIO) serial commands.

All communication between the host system and S25FL-S family of memory devices is in the form of units called commands. All commands begin with an instruction that selects the type of information transfer or device operation to be performed. Commands may also have

an address, instruction modifier, latency period, data transfer to the memory, or data transfer from the memory.

All instruction, address, and data information is transferred serially between the host system and memory device.

All instructions are transferred from host to memory as a single bit serial sequence on the SI signal. Single bit wide commands may provide an address or data sent only on the SI signal. Data may be sent back to the host serially on the SO signal. Dual or Quad Output commands provide an address sent to the memory only on the SI signal. Data will be returned to the host as a sequence of bit pairs on IO0 and IO1 or four bit (nibble) groups on IO0, IO1, IO2, and IO3. Dual or Quad Input/Output (I/O) commands provide an address sent from the host as bit pairs on IO0 and IO1 or, four bit (nibble) groups on IO0, IO1, IO2, and IO3. Data is returned to the host similarly as bit pairs on IO0 and IO1 or, four bit (nibble) groups on IO0, IO1, IO2, and IO3. Commands are structured as follows: • Each command begins with CS# going LOW and ends with CS# returning HIGH. The memory device is selected by the host driving the Chip Select (CS#) signal LOW throughout a command. • The serial clock (SCK) marks the transfer of each bit or group of bits between the host and memory. • Each command begins with an eight bit (byte) instruction. The instruction is always presented only as a single bit serial sequence on the Serial Input (SI) signal with one bit transferred to the memory device on each SCK rising edge. The instruction selects the type of information transfer or device operation to be performed.

• The instruction may be standalone or may be followed by address bits to select a byte location within one of several address spaces in the device. The instruction determines the address space used. The address may be either a 24-bit or a 32-bit address. The address transfers occur on SCK rising edge.

• The width of all transfers following the instruction are determined by the instruction sent. Following transfers may continue to be single bit serial on only the SI or Serial Output (SO) signals, they may be done in two bit groups per (dual) transfer on the IO0 and IO1 signals, or they may be done in 4 bit groups per (quad) transfer on the IO0-IO3 signals. Within the dual or quad groups the least significant bit is on IO0. More significant bits are

placed in significance order on each higher numbered IO signal. Single bits or parallel bit groups are transferred in most to least significant bit order.

• Some instructions send an instruction modifier called mode bits, following the address, to indicate that the next command will be of the same type with an implied, rather than an explicit, instruction. The next command thus does not provide an instruction byte, only a new address and mode bits. This reduces the time needed to send each command when the same command type is repeated in a sequence of commands. The mode bit transfers occur on SCK rising edge.

• The address or mode bits may be followed by write data to be stored in the memory device or by a read latency period before read data is returned to the host. • Write data bit transfers occur on SCK rising edge.

• SCK continues to toggle during any read access latency period. The latency may be zero to several SCK cycles (also referred to as dummy cycles). At the end of the read latency cycles, the first read data bits are driven from the outputs on SCK falling edge at the end of the last read latency cycle. The first read data bits are considered transferred to the host on the following SCK rising edge. Each following transfer occurs on the next SCK rising edge.

• If the command returns read data to the host, the device continues sending data transfers until the host takes the CS# signal HIGH. The CS# signal can be driven HIGH after any transfer in the read data sequence. This will terminate the command.

At the end of a command that does not return data, the host drives the CS# input HIGH. The CS# signal must go HIGH after the eighth bit, of a standalone instruction or, of the last write data byte that is transferred. That is, the CS# signal must be driven HIGH when the number of clock cycles after CS# signal was driven LOW is an exact multiple of eight cycles. If the CS# signal does not go HIGH exactly at the eight SCK cycle boundary of the instruction or write data, the command is rejected and not executed.

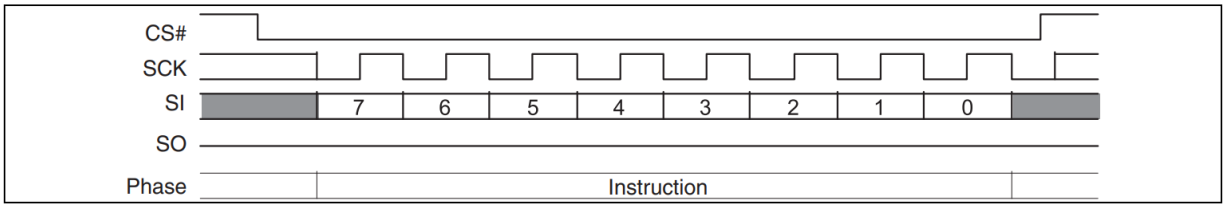
• All instruction, address, and mode bits are shifted into the device with the Most Significant bits (MSb) first. The data bits are shifted in and out of the device MSb first. All

data is transferred in byte units with the lowest address byte sent first. Following bytes of data are sent in lowest to highest byte address order i.e. the byte address increments.

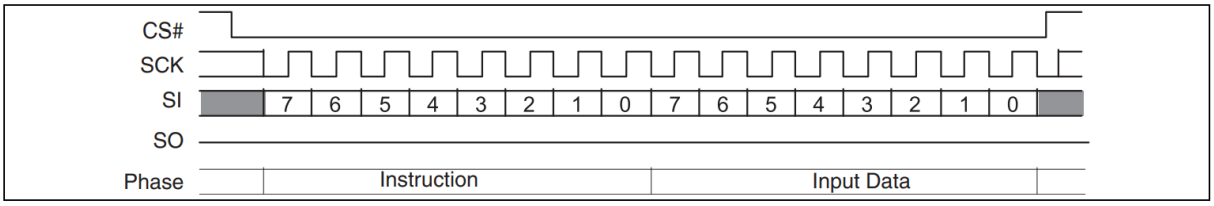
• All attempts to read the flash memory array during a program, erase, or a write cycle (embedded operations) are ignored. The embedded operation will continue to execute

without any affect. A very limited set of commands are accepted during an embedded operation. These are discussed in the individual command descriptions.

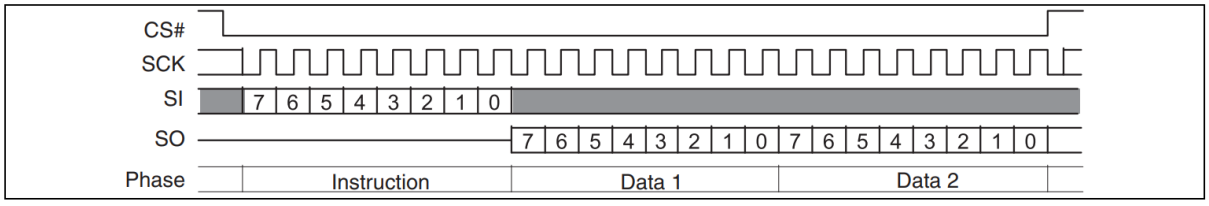
• Depending on the command, the time for execution varies. A command to read status information from an executing command is available to determine when the command completes execution and whether the command was successful.



Standalone Instruction Command



Single Bit Wide Input Command



Single Bit Wide Output Command

3.12.1 S25FL127S Command Set

Similar to most flash chips, the S25FL127S supports a command set. The user will write a command via SPI protocol and the flash chip will respond.

Function	Command name	Command description	Instruction value (Hex)	Maximum frequency (MHz)
Read Device Identification	READ_ID (REMS)	Read Electronic Manufacturer Signature	90	108
	RDID	Read ID (JEDEC Manufacturer ID and JEDEC CFI)	9F	108
	RSFDP	Read JEDEC Serial Flash Discoverable Parameters	5A	108
	RES	Read Electronic Signature	AB	50
Register Access	RDSR1	Read Status Register 1	05	108
	RDSR2	Read Status Register 2	07	108
	RDCR	Read Configuration Register 1	35	108
	WRR	Write Register (Status-1, Configuration-1)	01	108
	WRDI	Write Disable	04	108
	WREN	Write Enable	06	108
	CLSR	Clear Status Register 1 - Erase/Prog. Fail Reset	30	108
	ECCRD	ECC Read (4-byte address)	18	108
	ABRD	AutoBoot Register Read	14	108 (QUAD=0) 108 (QUAD=1)
	ABWR	AutoBoot Register Write	15	108
	BRRD	Bank Register Read	16	108
	BRWR	Bank Register Write	17	108
	BRAC	Bank Register Access (Legacy Command formerly used for Deep Power Down)	B9	108
	DLPRD	Data Learning Pattern Read	41	108
	PNVDLR	Program NV Data Learning Register	43	108
	WVDLR	Write Volatile Data Learning Register	4A	108

3.12.2 Read Identification (RDID 9Fh)

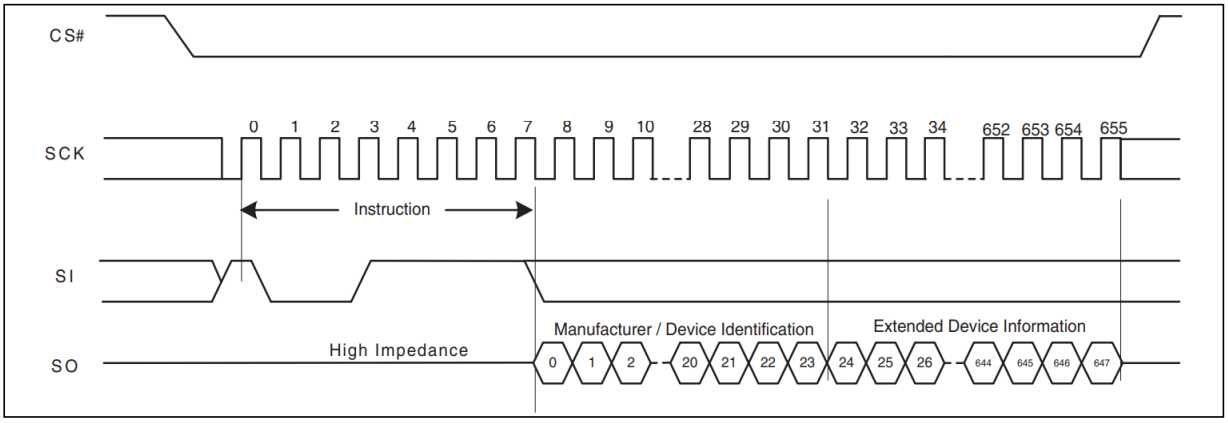
The Read Identification (RDID) command provides read access to manufacturer identification, device identification, and Common Flash Interface (CFI) information. The manufacturer identification is assigned by JEDEC. The CFI structure is defined by JEDEC standard. The device identification and CFI values are assigned by Cypress

The JEDEC Common Flash Interface (CFI) specification defines a device information structure, which allows a vendor-specified software flash management program (driver) to be used for entire families of flash devices. Software support can then be device-independent, JEDEC manufacturer ID independent, forward and backward-compatible for the specified flash device families. System vendors can standardize their flash drivers for long-term software compatibility by using the CFI values to configure a family driver from the CFI information of the device in use.

Any RDID command issued while a program, erase, or write cycle is in progress is ignored and has no effect on execution of the program, erase, or write cycle that is in progress. The RDID instruction is shifted on SI. After the last bit of the RDID instruction is shifted into the device, a byte of manufacturer identification, two bytes of device identification, extended device identification, and CFI information will be shifted sequentially out on SO. As a whole this information is referred to as ID-CFI.

Continued shifting of output beyond the end of the defined ID-CFI address space will provide undefined data. The RDID command sequence is terminated by driving CS# to the logic high state anytime during data output. The maximum clock frequency for the RDID command is 108 MHz.

Read Identification (RDID) Command Sequence



Device ID and Common Flash Interface (ID-CFI) address map

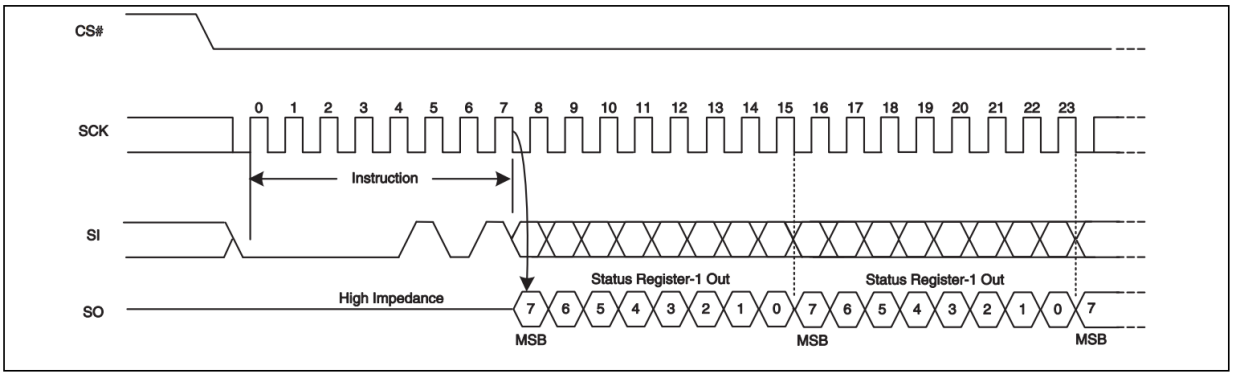
Byte address	Data	Description
00h	01h	Manufacturer ID for Cypress
01h	20h (128 Mb)	Device ID Most Significant Byte - Memory Interface Type
02h	18h (128 Mb)	Device ID Least Significant Byte - Density
03h	4Dh	ID-CFI Length - number bytes following. Adding this value to the current location of 03h gives the address of the last valid location in the legacy ID-CFI address map. This only includes up to the end of the Primary Vendor Specific table. The Alternate Vendor Specific table contains additional information.
04h	00h (Uniform 256-KB sectors) 01h (4-KB parameter sectors with uniform 64-KB sectors)	Sector Architecture
05h	80h (FL-S Family)	Family ID

06h	xxh	Reserved
07h	xxh	Reserved
08h	xxh	Reserved
09h	xxh	Reserved

3.12.3 Register Access commands

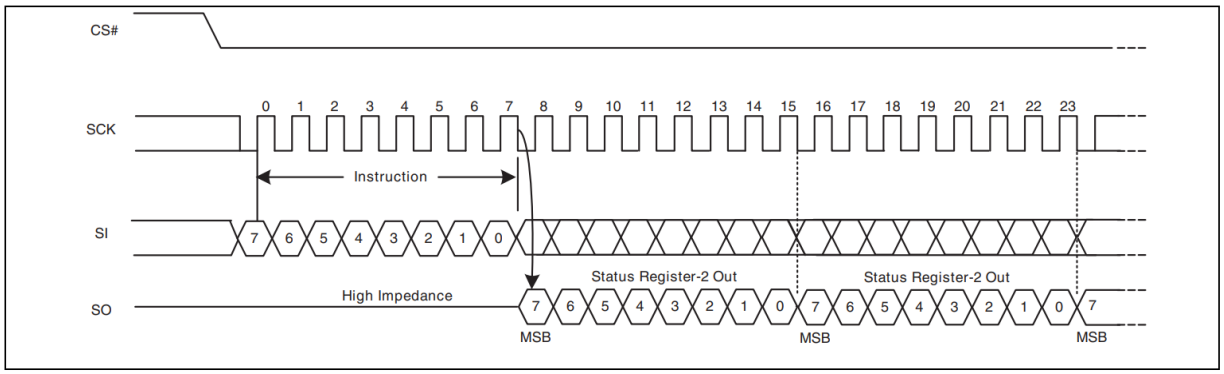
Read Status Register 1 (RDSR1 05h)

The Read Status Register 1 (RDSR1) command allows the Status Register 1 contents to be read from SO. The Status Register 1 contents may be read at any time, even while a program, erase, or write operation is in progress. It is possible to read the Status Register 1 continuously by providing multiples of eight clock cycles. The status is updated for each eight cycle read. The maximum clock frequency for the RDSR1 (05h) command is 108 MHz.



Read Status Register 2 (RDSR2 07h)

The Read Status Register (RDSR2) command allows the Status Register 2 contents to be read from SO. The Status Register 2 contents may be read at any time, even while a program, erase, or write operation is in progress. It is possible to read the Status Register 2 continuously by providing multiples of eight clock cycles. The status is updated for each eight cycle read. The maximum clock frequency for the RDSR2 command is 108 MHz.



Read Configuration Register (RDCR 35h)

For the remainder of the access to the registers of the S25FL127S, please see the data sheet.

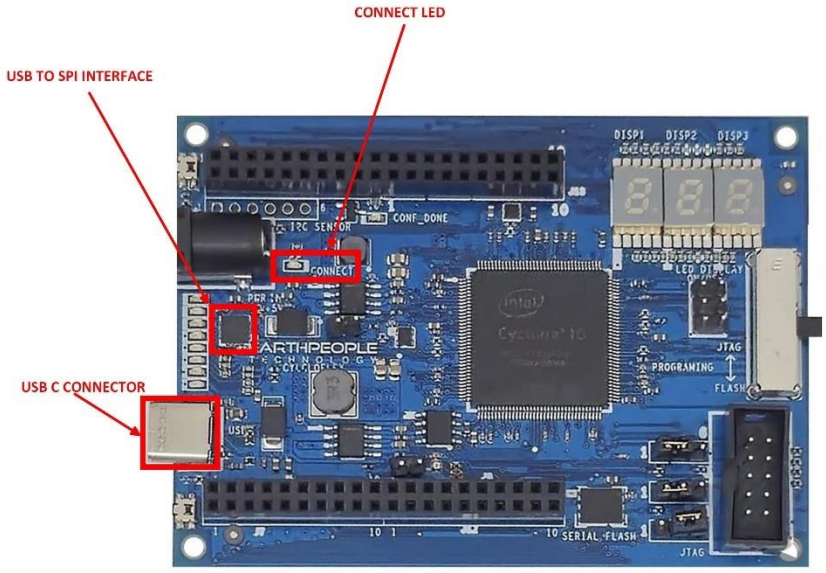
3.13 Power Input

The CycloFlex is designed to be operated from one of two different power sources:

- Standard USB cable from Laptop/PC.
- +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.

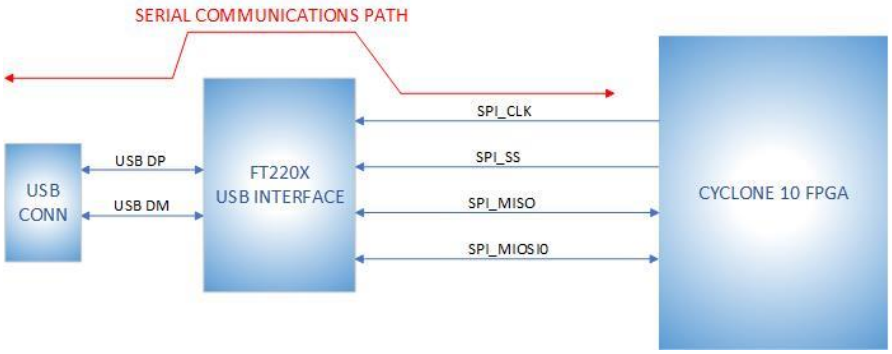
3.14 Communications Interface

The CycloFlex is equipped with a bi-directional communications mechanism. This pathway is via an FT220X USB to SPI bus chip.



The FT220X chip connects the Cyclone 10 FPGA to the Host PC using USB. The FT220X chip provides all of the USB protocol and is transparent to the user. The Input/Output side of the chip provides a modified version of the accepted SPI bus. It uses FTDI's proprietary FT1248 protocol. It has the following features:

- USB to Slave SPI interface in 1-bit mode using the FT1248 protocol
- Data transfer rates to 500 kByte/s
- 512 byte receive buffer and 512 byte transmit buffer utilizing buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development.

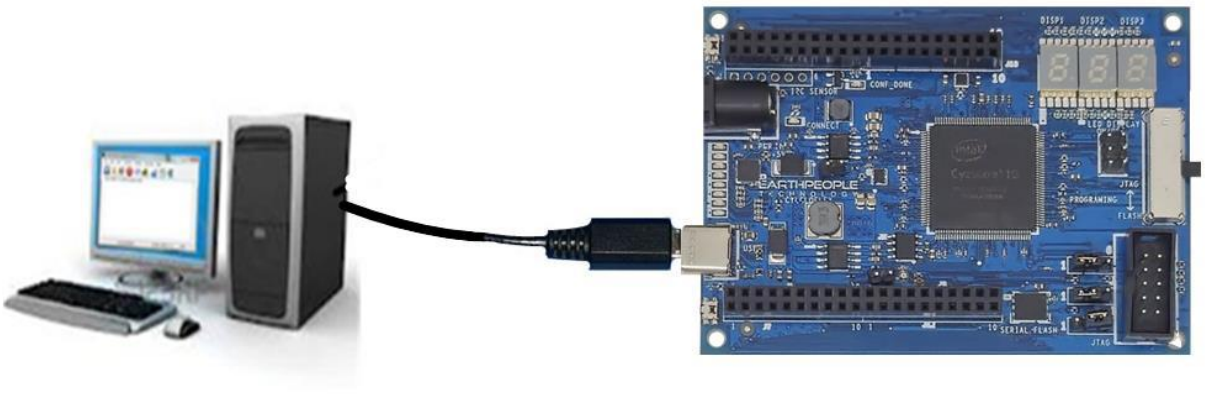


There are two things that are extremely important to note in the features above:

- SPI interface using the FT1248 protocol
- FTDI's Virtual Com driver

The FT1248 protocol is a deviation from the typical SPI bus scheme. It keeps the SCLK and Chip Select signals typical of SPI. However, it changes the MOSI and MISO application. The FT220X chip changes the MOSI signal to a bi-directional communication port. Earth People Technology has created a Verilog Library to make this change transparent to user code. So, the user is free to create a Master SPI interface within the FPGA code. The second item to note is the FTDI Virtual Com Driver. This driver creates a transparent pathway between the user code in the FPGA and the Host PC. The user can create quite advanced communications schemes very easily with little coding effort.

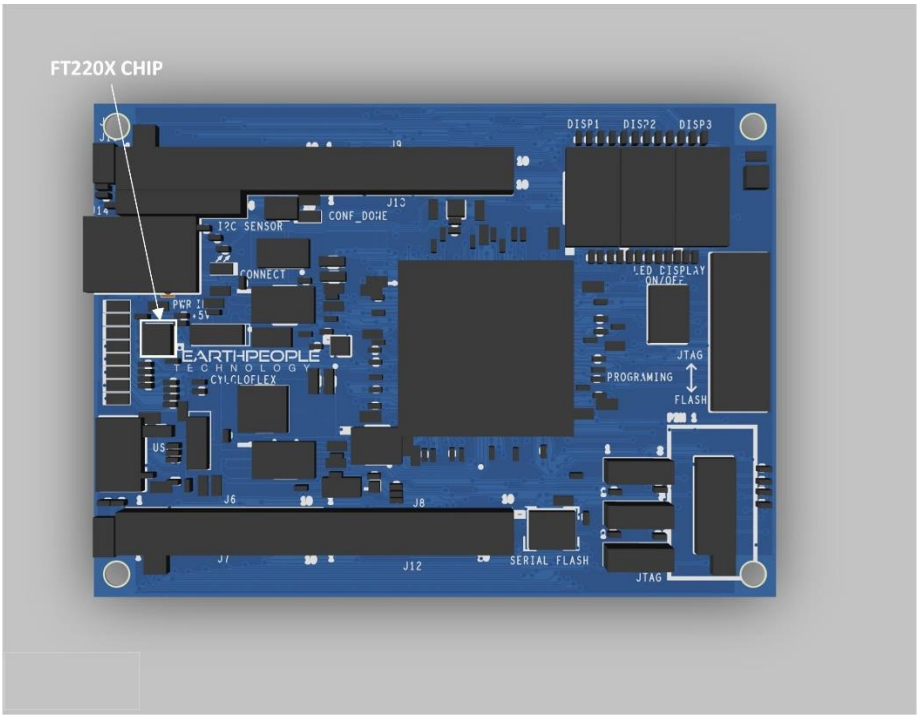
Earth People Technology has created demonstration software that provides a bi-directional communication pathway that can transfer data at 500Kbytes/second. All source code is provided and the user is encouraged to use this software to as a base to create custom projects for their need.



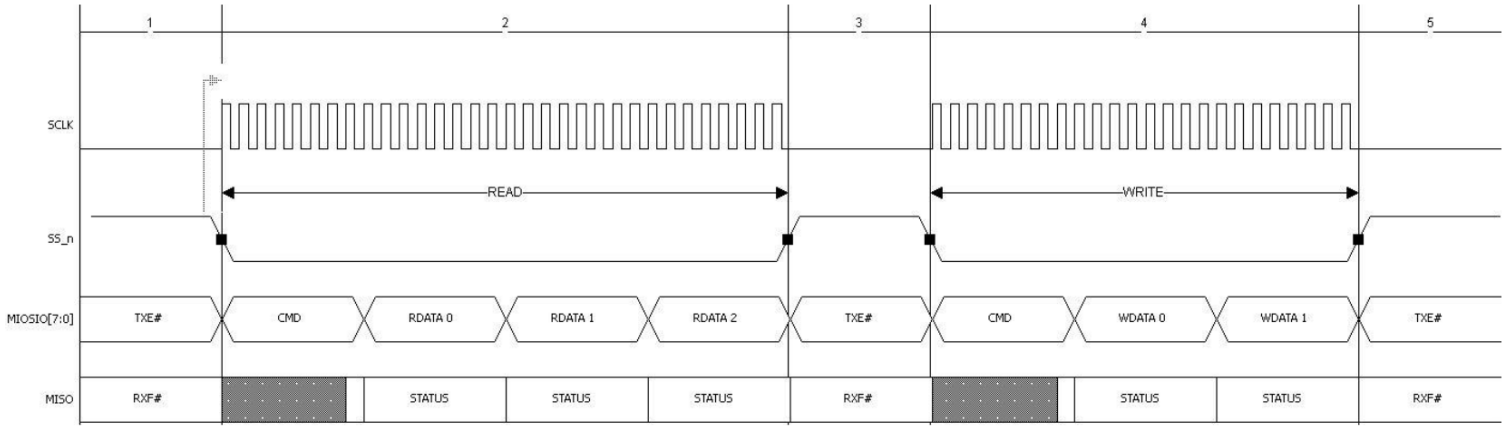
The USB-C connector provides a path between the Host PC and the CycloFlex Board. It provides a transparent means to power the board and bi-directional communications. The communications occurs over USB. The FTDI driver provides a convient transparent pathway to the CycloFlex Board. The user code in the FPGA is only required to meet the requirements of the FT220X chip.

Component	Function	Net Name	Pin Number on Cylcone 10	Signal in EPT Project
U1	MIOSIO0	UB27	120	SPI_MIOSIO_1
U1	MISO	UB26	115	SPI_MISO
U1	SCLK	UB29	125	SPI_SCLK
U1	SS	UB28	121	SPI_SS

The FT220X chip is located in the upper left corner of the CycloFlex board. It has four connections to the FPGA.



The FT220X chip DOES NOT use the typical SPI bus protocol. It uses a modified protocol, the FT1248. The difference between normal SPI and the FT1248 protocol is the MOSI data line is NOT a single direction data line. The FT1248 protocol makes this line a bi-direction data line. For data transmitted from the FPGA to the FT220X, the host (FPGA user code) must send a command via the MOSI line then read back the data from the MOSI line. The FT220X renames the MOSI line to the MIOSIO[x] signal. “x” Because the chip has four of these IO lines available to increase bandwidth. Here is the timing diagram for the FT220X basic Read/Write cycle.



During the Read Cycle, the Host (FPGA user code) sends a command to the FT220X, then reads the data bytes from the chip. All via the MIOSIO[0] signal.

During the Write Cycle, the Host (FPGA user code) sends a command to the FT220X, then transmits the number of bytes to the chip. All via the MIOSIO[0] signal.

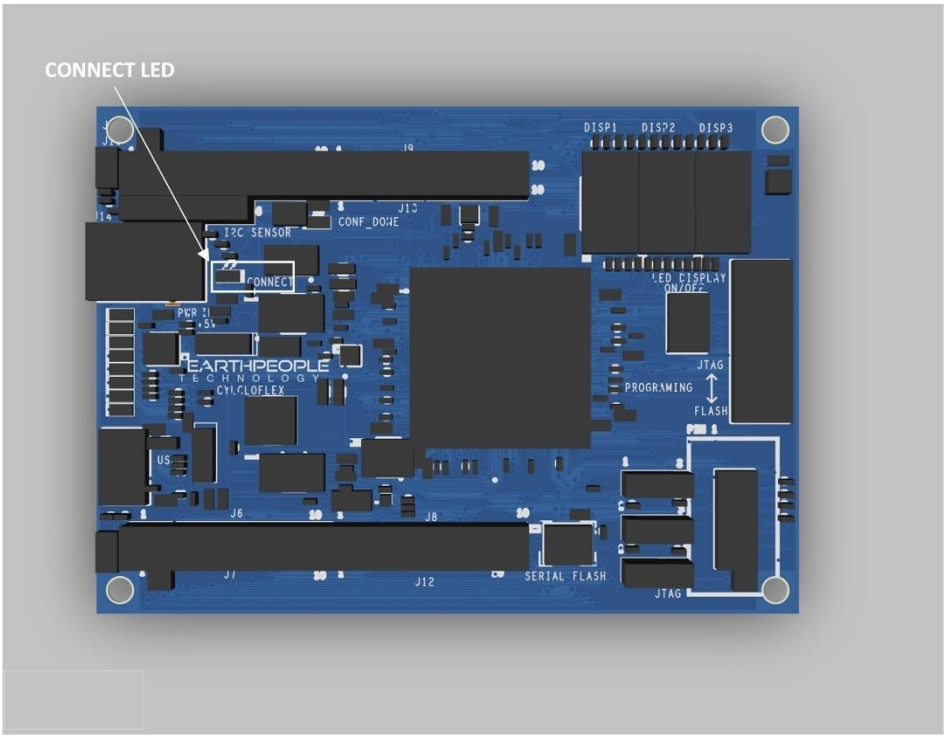
The commands that the Host must send to the FT220X chip are:

Command	Identifier	Description
Write	0x00	Write request command
Read	0x01	Read request command
Read modem status	0x02	Read modem status command, users may wish to emulate modem status control. A RMS command returns status bits RTS and DTR
Write modem status	0x03	Write modem status command, users may wish to emulate modem status control. A WMS command allows users to set status bits: DCD, RI, DSR, CTS

Write buffer flush	0x04	Write buffer flush request – This command is used to indicate to the FT1248 slave that its write buffers should be flushed rather than wait for any latency timers to expire. If this command is received the FT1248 block will flag the upstream controllers (USB FIFOs etc) to flush their write buffers regardless of what content is present in the FT1248 write buffer
Address eeprom	0x05	Address EEPROM command sets the address users wish to write or read from
Write eeprom	0x06	Write EEPROM command sets the write data to be written into the EEPROM
Read eeprom	0x07	Read EEPROM command reads
Read USB status	0x08	Read USB Status: 00 = suspended, 01 = default, 10 = addressed, 11 = configured
Reserved	0x09 – 0xF	Unused Commands

3.14.1 CONNECT LED

The CycloFlex has a “CONNECT” LED installed on the board. If the USB Driver is installed properly on the Host PC (See the Load EPT Drivers Section), once the USB cable is connected to the USB-C connector, this LED will light up.



The purpose of this LED is to inform the user that the following has occurred:

- USB Driver is properly installed on the Host PC
- The USB-C cable is properly installed
- The +3.3V power supply is functioning properly
- The FT220X is functioning properly

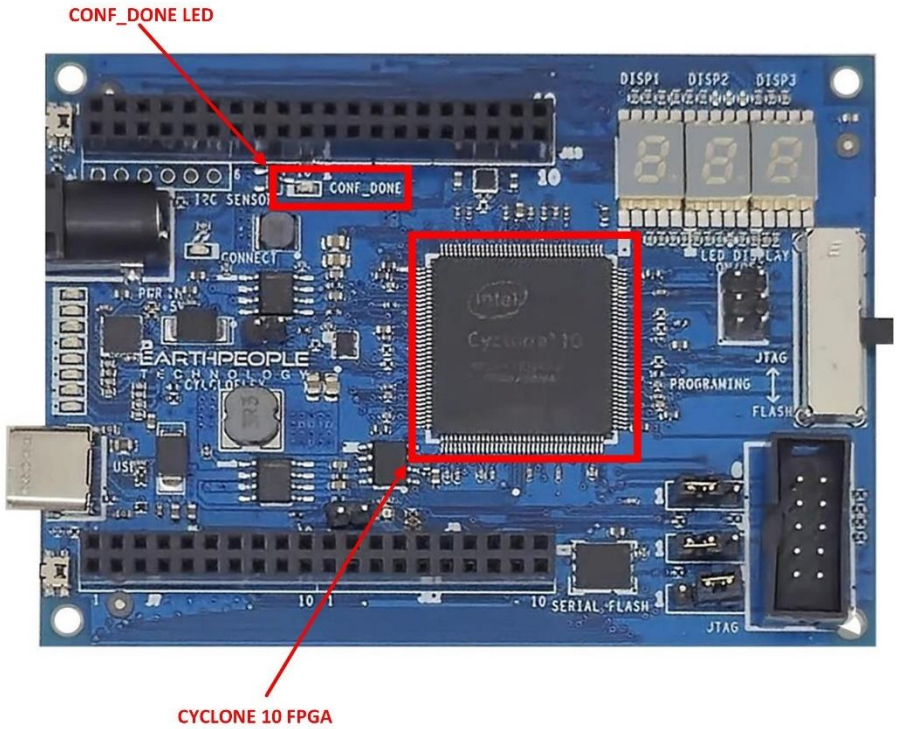
These four items will provide reasonable assurance that the USB section of the CycloFlex is functioning properly. The user can further use the pre-installed FPGA project to ensure that other parts of the CycloFlex board are functioning properly. See the section “Getting Started with the CycloFlex” for the use of the pre-installed FPGA project and its capabilities.

3.14.2 Active Host Communications

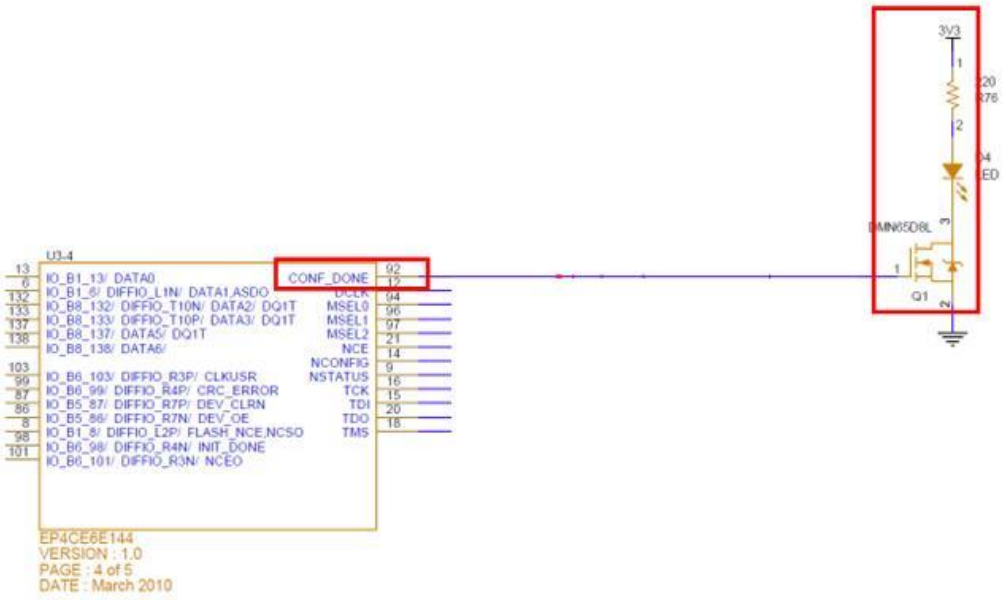
Earth People Technology has developed a robust software SDK that allows the user to easily create a bi-directional communications scheme with minimal coding effort.

3.14.3 CONF DONE LED

The CONF_DONE LED is connected directly to the FPGA CONF_DONE signal.



When this LED is lit, it signals to the user that the FPGA has been correctly configured from the configuration flash or the JTAG connection.



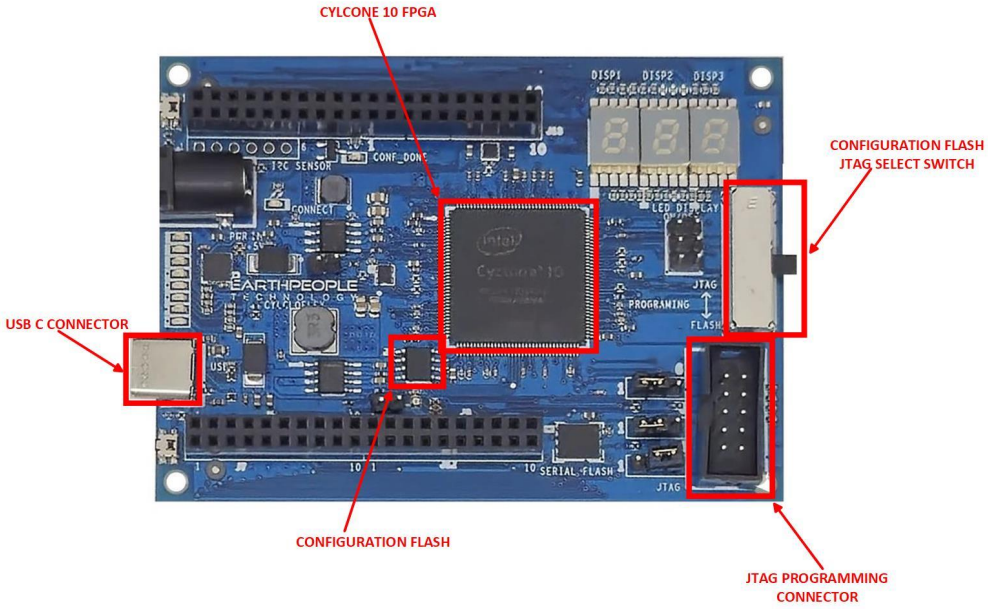
The purpose of this LED is to inform the user that the following has occurred:

- Power to the CycloFlex has been successful
- The +3.3V, +2.5V and +1.2V power supplies are functioning properly
- The Cyclone 10 chip is functioning properly
- The Cyclone 10 chip has been properly programmed with a valid FPGA project

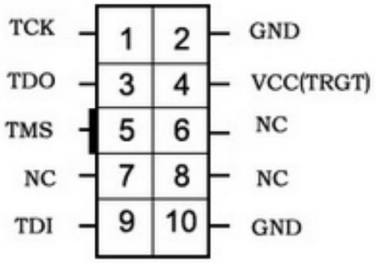
These four items will provide reasonable assurance that the power and programming section of the CycloFlex is functioning properly. The user can further use the pre-installed FPGA project to ensure that other parts of the CycloFlex board are functioning properly. See the section “Getting Started with the CycloFlex” for the use of the pre-installed FPGA project and its capabilities.

3.15 JTAG Interface

The CycloFlex has a 5x2 header for use in programming the Cyclone 10 FPGA via JTAG. The connector is located in the bottom right corner of the CycloFlex. It is shrouded and keyed to allow easier insertion.

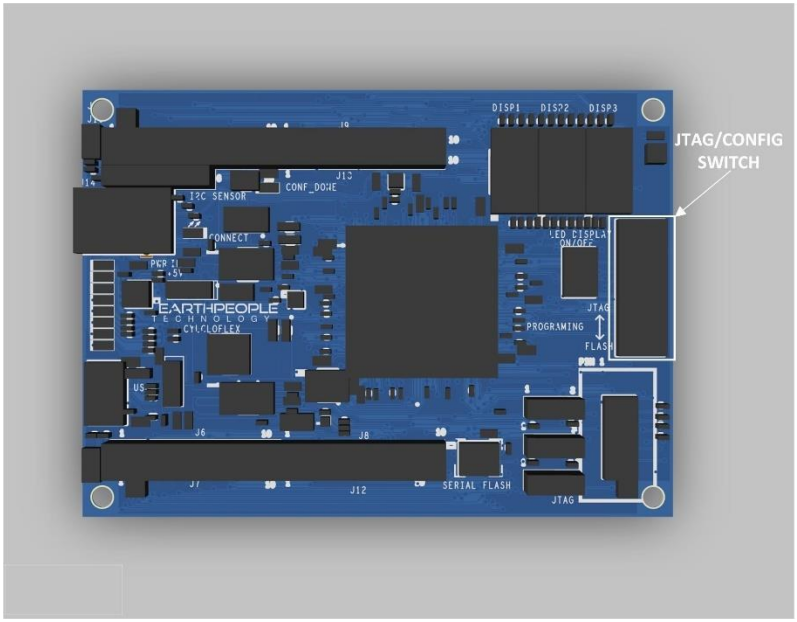


This connector uses the standard Altera Blaster connector pinout.

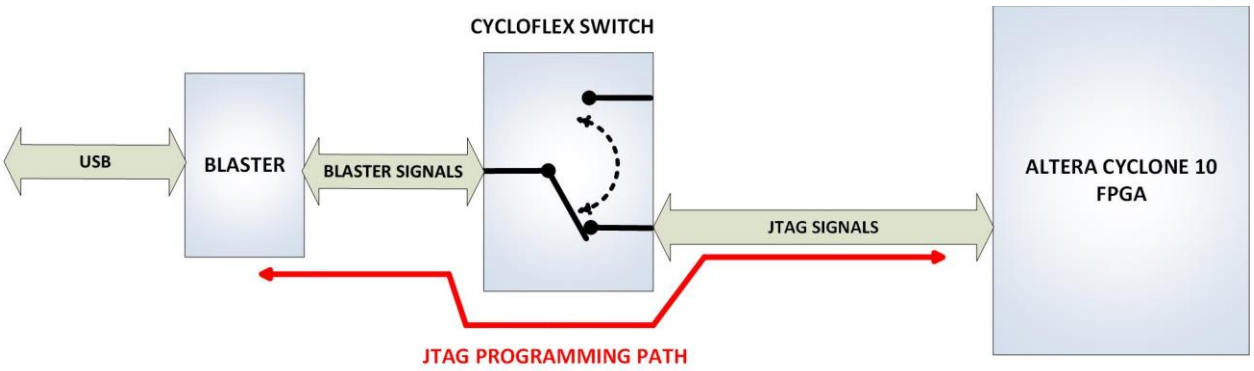


The VCC(TRGT) is set to +3.3V on the CycloFlex. This voltage is used to set the output voltage I/O level on the Altera Blaster. For the CycloFlex, +3.3V is the I/O level required. The board is 100% compatible with all Altera Blasters. The board requires power from either Barrel Connector or USB-C Connector. Then connect the Blaster.

The CycloFlex includes programming switch that capable of six Dual Pole Single Throw connections. The JTAG/CONFIG SWITCH.



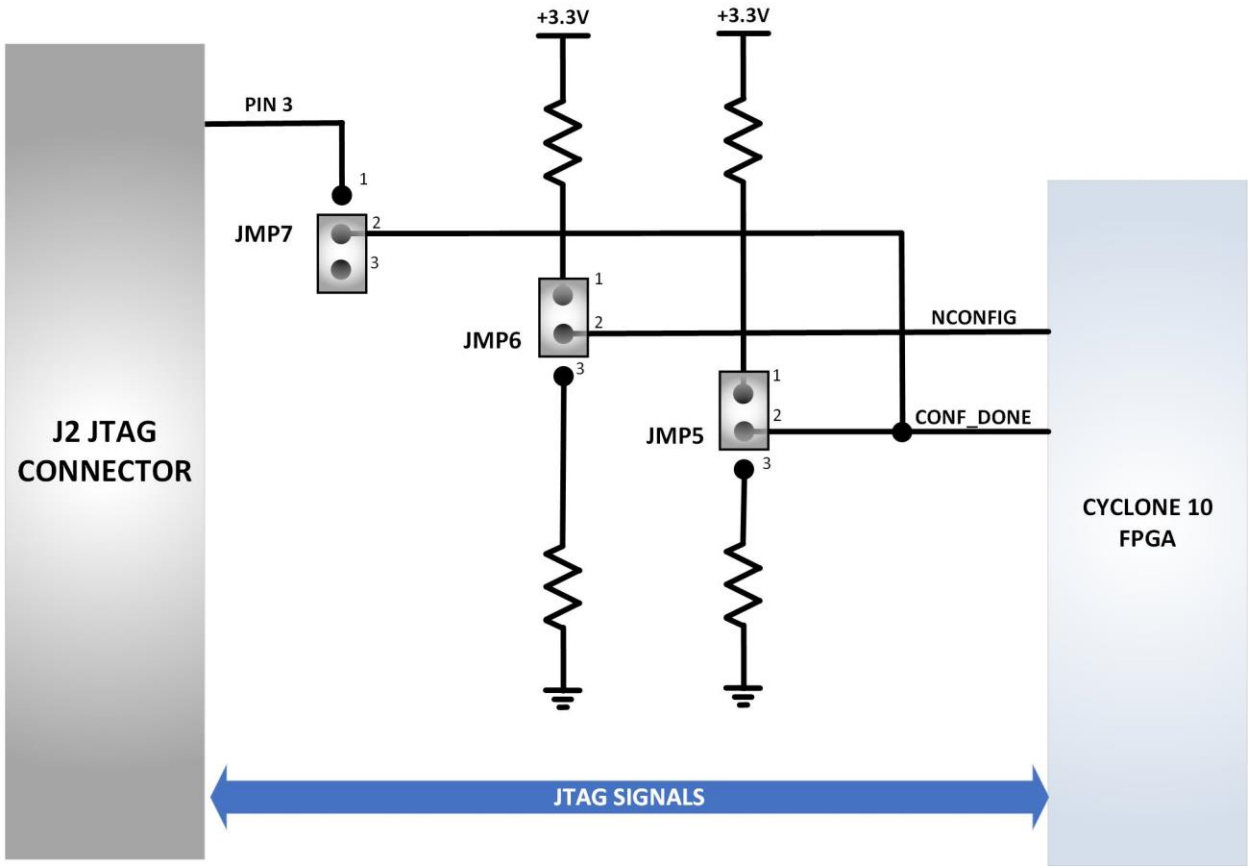
This switch is designed to allow the external Blaster Programmer to either program the Cyclone 10 chip directly using JTAG or program the Config Flash chip. Changing the position of the switch will allow the same Blaster to be used for either JTAG programming or Config Flash programming.



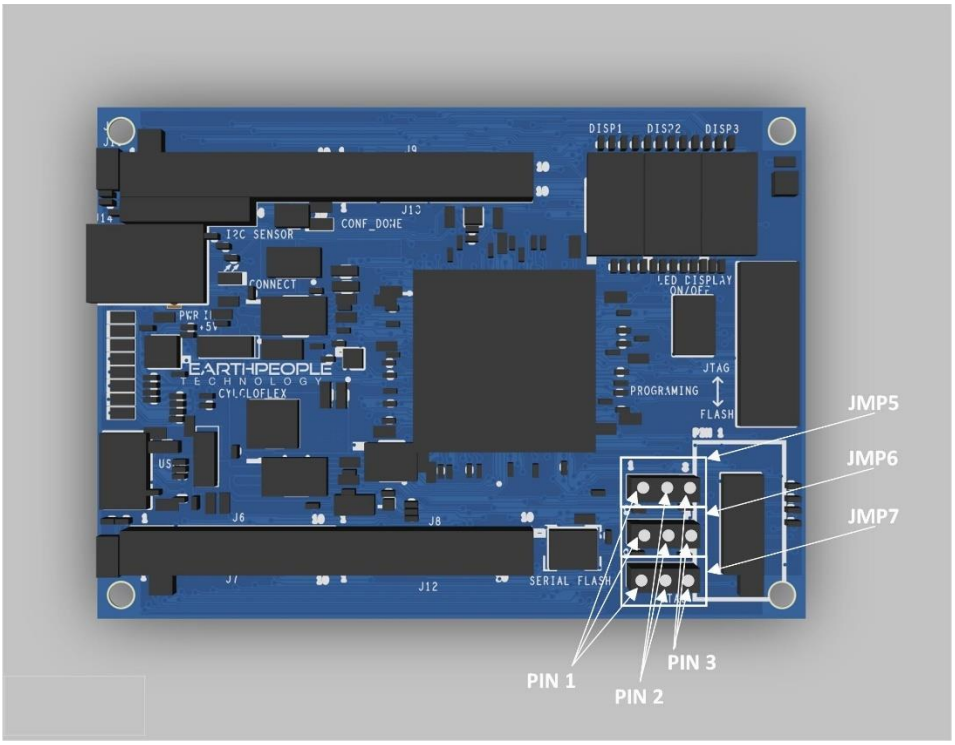
The JTAG Programming Path must have certain configuration pins of the Cyclone 10 in fixed positions (pulled up or pulled down) during programming. The following jumpers must be installed for correct programming of the Cyclone 10 chip using JTAG:

- JMP7
- JMP6
- JMP5

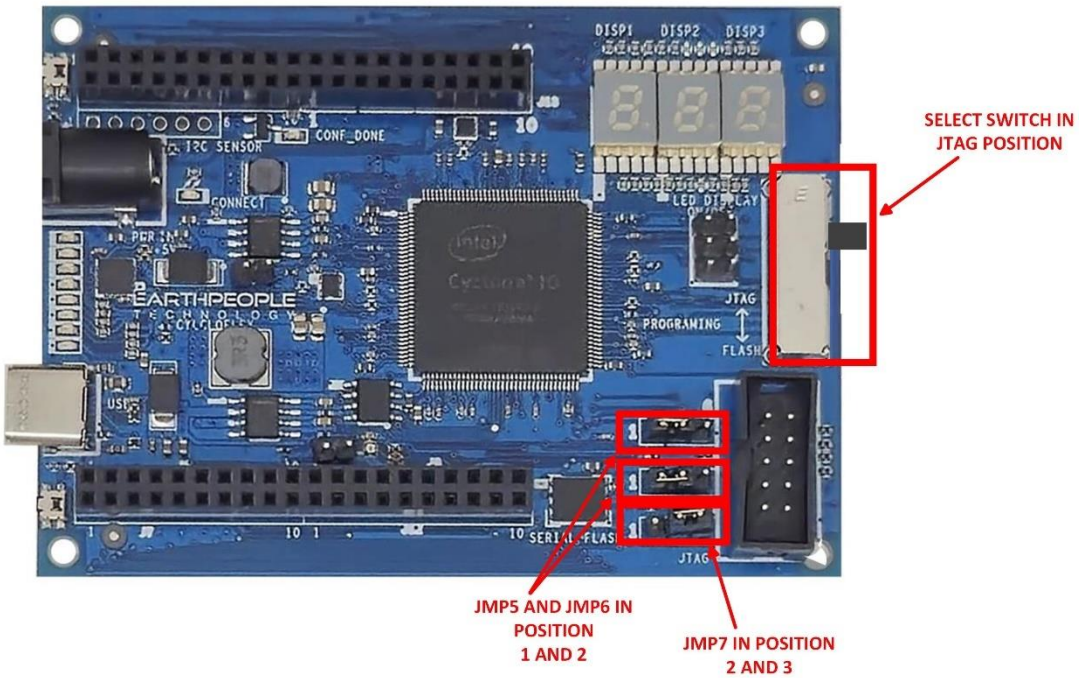
A jumper is a two pin socket used to short two header pins together.



The jumper locations are:



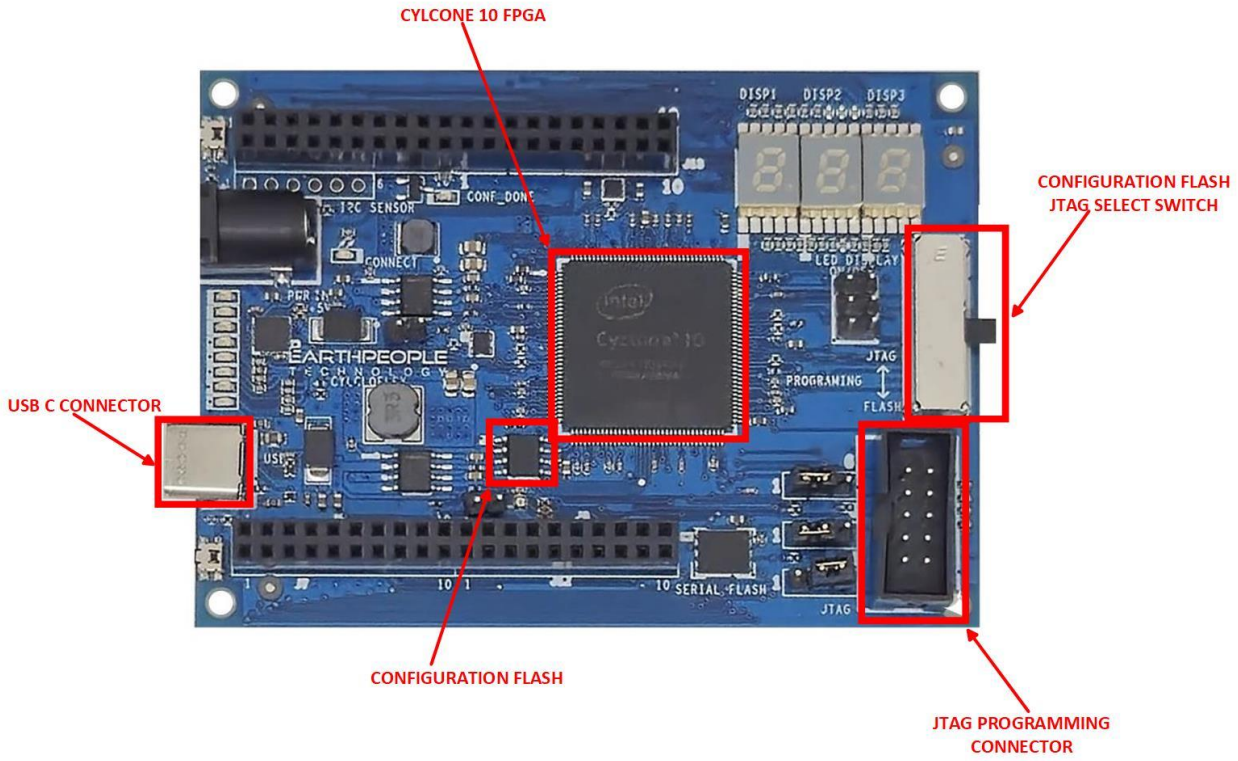
The correct settings for JTAG Path Programming:



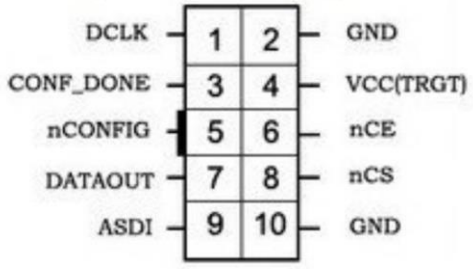
Once the jumpers and JTAG/Flash switch have been set, refer the section “Programming the CycloFlex” for instructions on how to use the Quartus Prime Lite software to program the Cyclone 10 chip.

3.16 Configuration Flash Programming

The Configuration Flash is used to store the users synthesized object file for use when the power to the CycloFlex board is cycled (on/off).

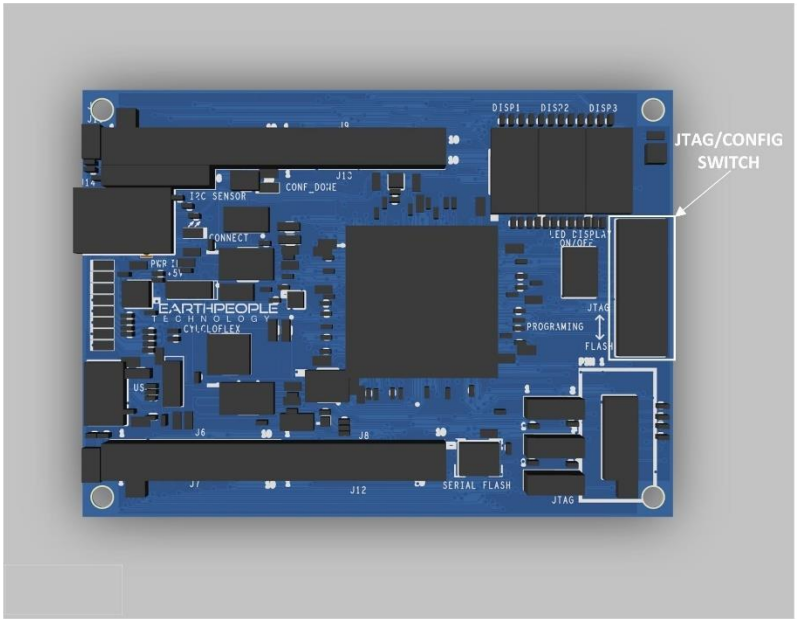


The CycloFlex uses the same 10 pin connector for Configuring the Flash chip with the following pinout:

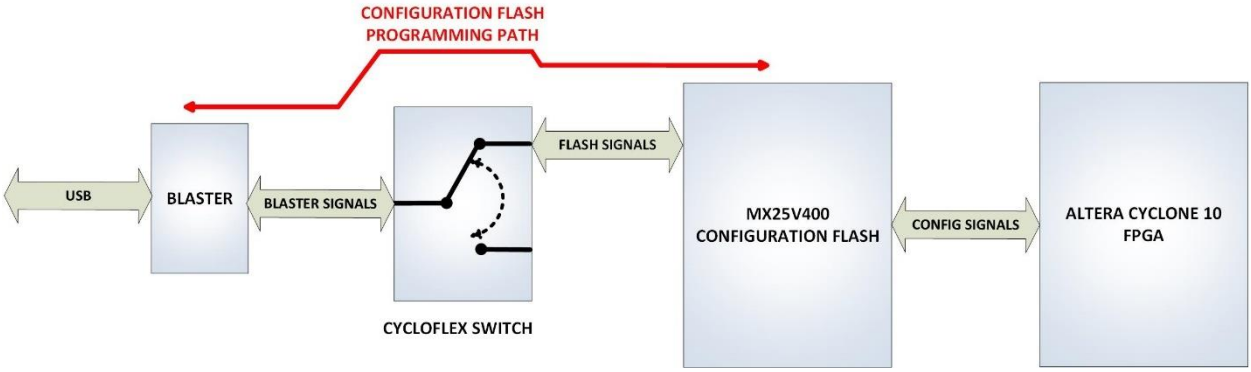


The board requires power from either Barrel Connector or USB-C Connector. Then connect the Blaster.

The CycloFlex includes programming switch that capable of six Dual Pole Single Throw connections. The JTAG/CONFIG SWITCH.



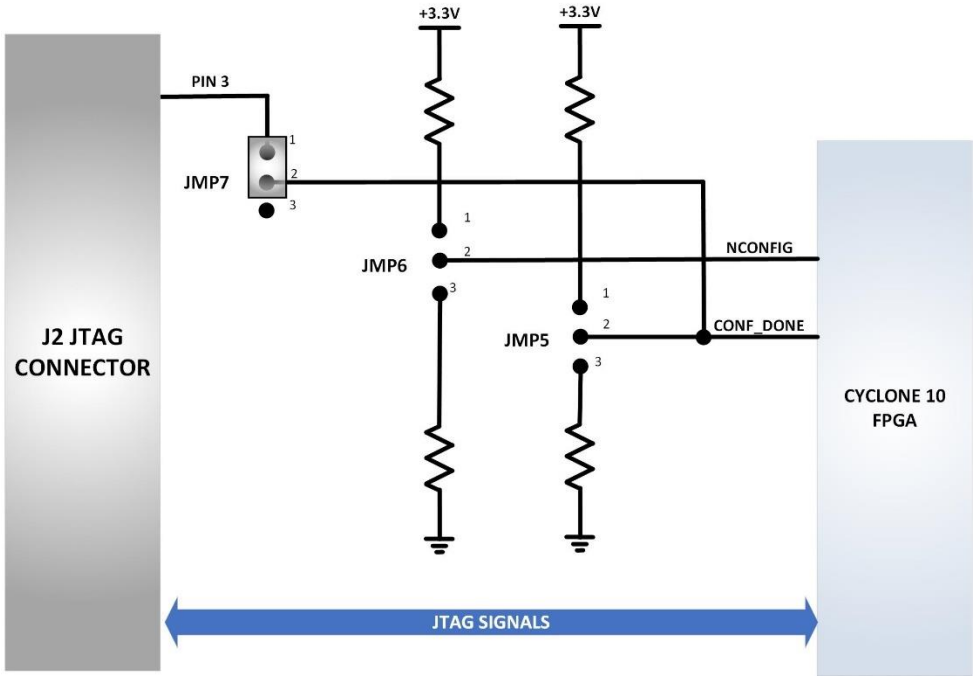
This switch is designed to allow the external Blaster Programmer to either program the Cyclone 10 chip directly using JTAG or program the Config Flash chip. Changing the position of the switch will allow the same Blaster to be used for either JTAG programming or Config Flash programming.



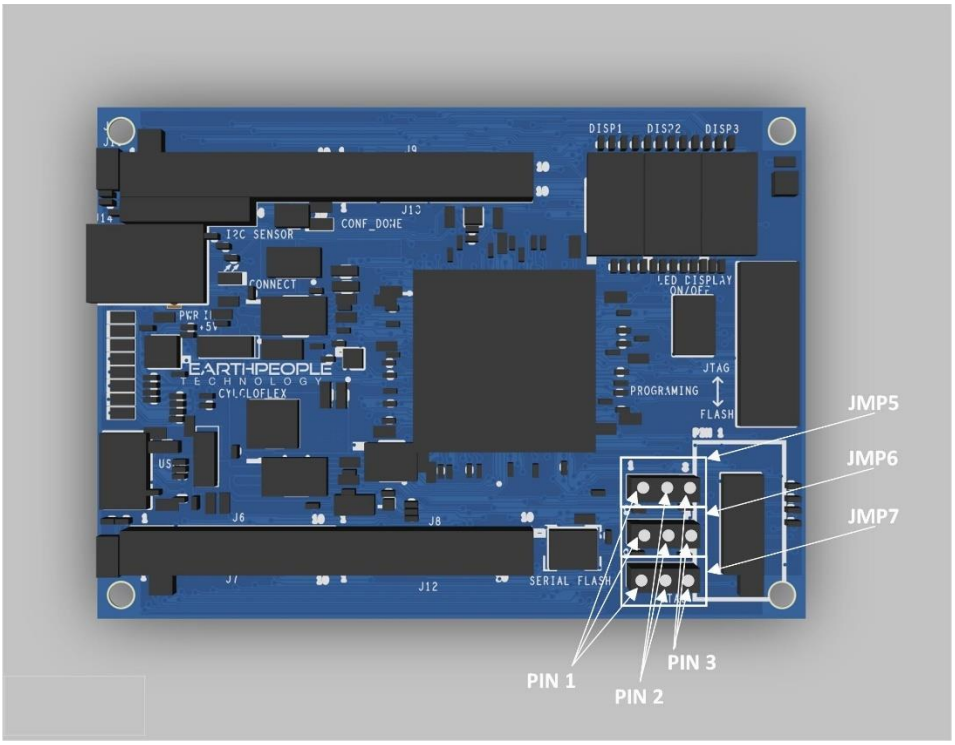
The JTAG Programming Path must have certain configuration pins of the Cyclone 10 in fixed positions (pulled up or pulled down) during programming. The following jumpers must be installed for correct programming of the Configuration Flash:

- JMP7
- JMP6
- JMP5

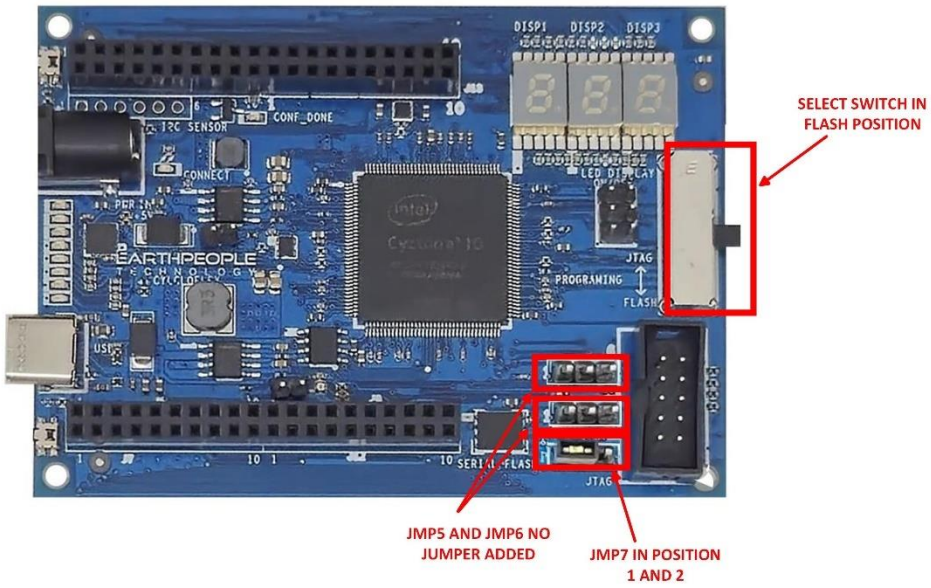
A jumper is a two pin socket used to short two header pins together.



The jumper locations are:



The CycloFlex should be configure with the following:

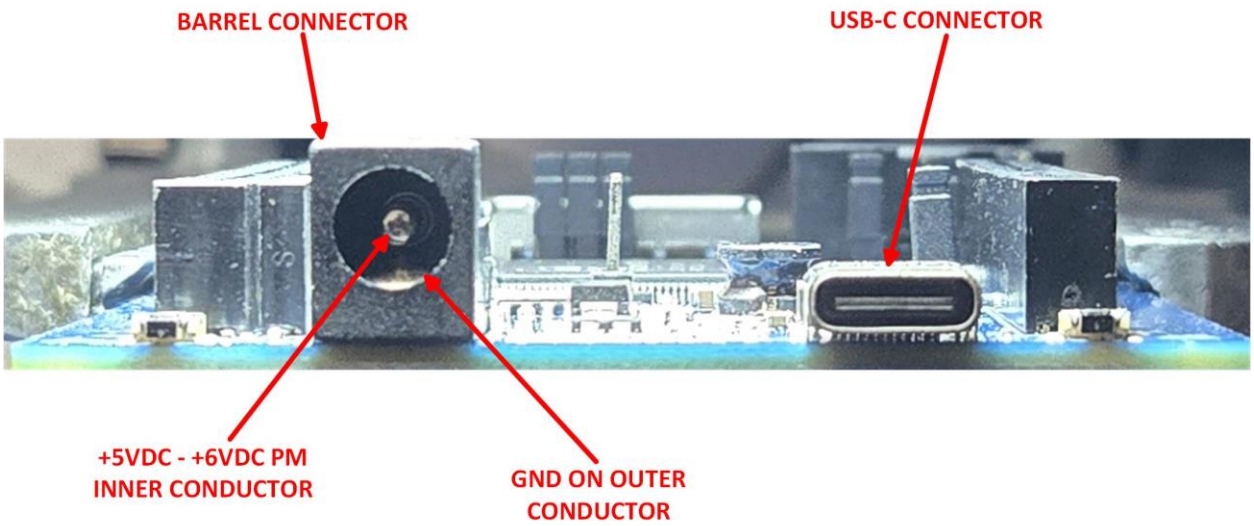


Once the jumpers and JTAG/Flash switch have been set, refer the section “Programming the CycloFlex” for instructions on how to use the Quartus Prime Lite software to program the Cyclone 10 chip.

4 Powering the CycloFlex

There are a few options for powering the CycloFlex Board. You can run the CycloFlex 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.

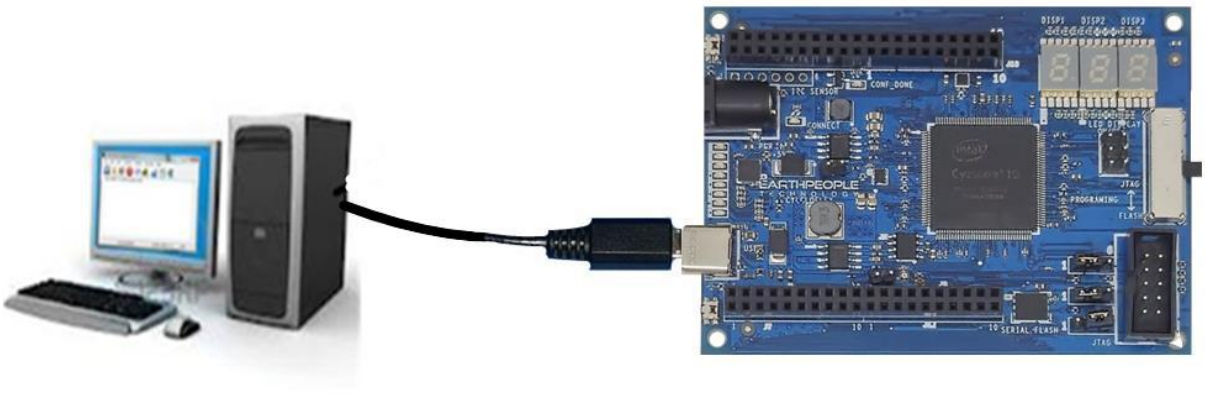
- Standard USB cable from Laptop/PC.
- +5 VDC wall charger (phone charger) through USB cable.
- +4.5 to +5.5 VDC supplied through the DC power jack.



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 CycloFlex. 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 CycloFlex is only designed to handle 2Amps of current. So, damage may occur to the CycloFlex if the user does not exercise care in design and use of the Inputs/Outputs.



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

Power the CycloFlex directly from the wall charger. +5V~6V@2A

5 EPT Drivers

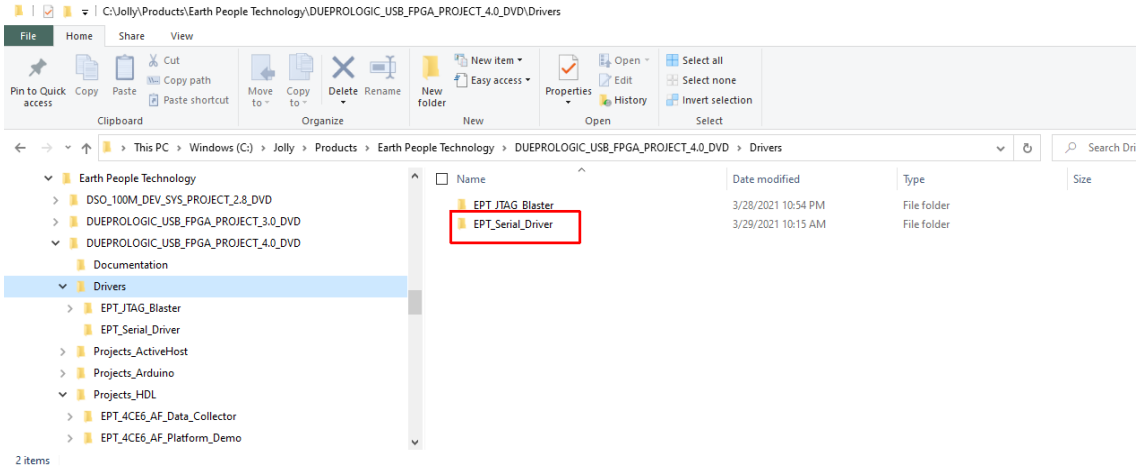
The CycloFlex Development system requires drivers for any interaction between PC and the board. The communication between the two consists of programming the FPGA and data

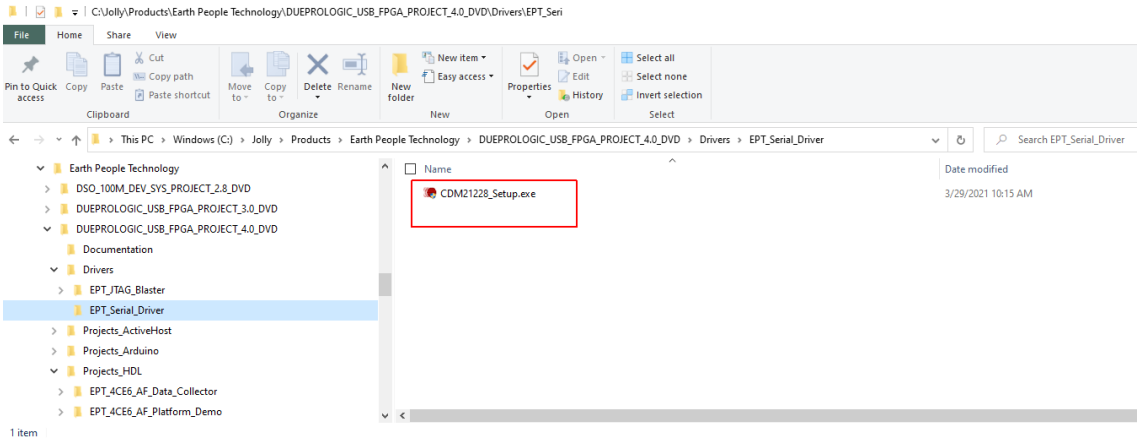
transfer. In both cases, the USB Driver is required. This will allow Windows to recognize the USB Chip and setup a pathway for Windows to communicate with the USB hardware.

5.1 USB Driver

The CycloFlex uses an FTDI FT220X USB to Slave SPI chip. This chip provides the USB interface to the PC and the serial/FIFO interface to the FPGA. The FT220X requires the use of the FTDI USB driver. To install the driver onto your PC, use the EPT_Serial_Driver Folder. The installation of the FTDI 2.12.xx driver is easily accomplished by double clicking the CDM212xx_Setup.exe.

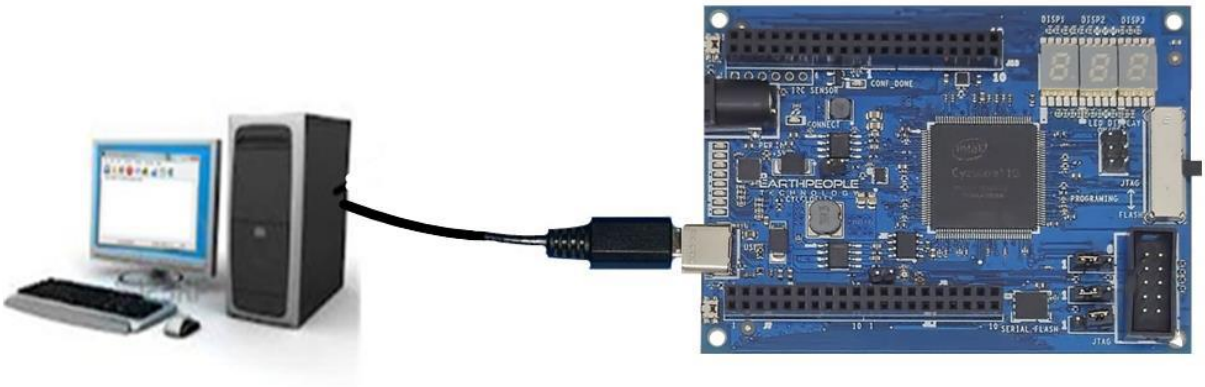
Locate the EPT_Serial_Driver folder in the Drivers folder of the CycloFlex_USB_FPGA_PROJECT_x.x_DVD using Windows Explorer.





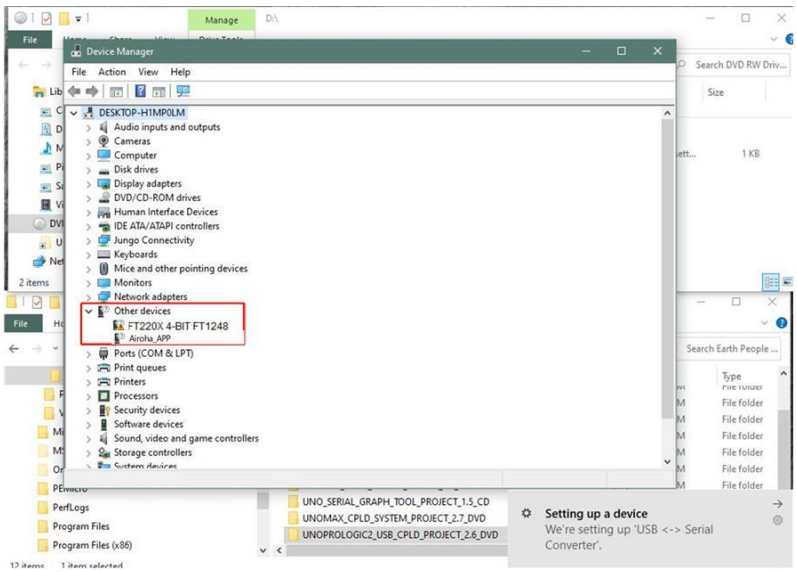
Double click on the *.exe file and select the default settings when the software tool queries the user.

Plug in the CycloFlex into an available USB port.



Windows will attempt to locate a driver for the USB device. Allow Windows to install the driver for the CycloFlex.

If Windows cannot load a driver for the CycloFlex, a notification window will inform the user that the driver load has failed for the device.

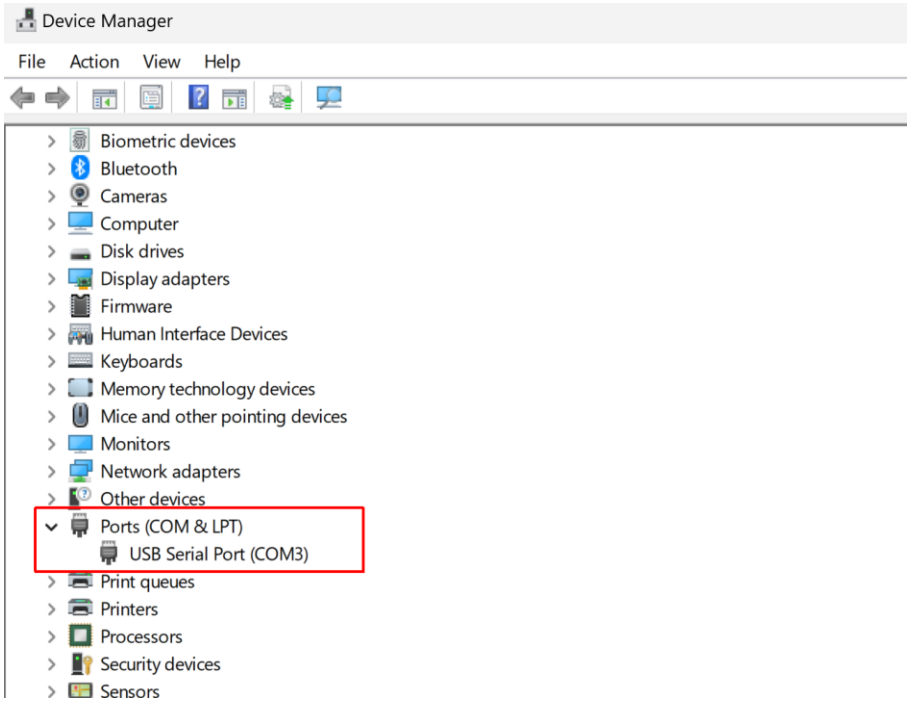


Right click on the “FT220X 4-BIT FT1248” icon that failed to load. Then select the “Load driver from Known Location”. Navigate to the CYCLOFLEX_USB_FPGA_PROJECT_x.x_DVD/drivers/EPT_Serial_Driver folder. Allow the Windows PC to locate and install the driver.

If this method continues to fail, contact support at Earth People Technology using the following methods:

www.earthpeopletechnology.com/Forums
support@earthpeopletechnology.com
sales@earthpeopletechnology.com

If the driver is successfully installed, Windows will inform the user. The user can check Device Manager to ensure the correct driver was installed for the CycloFlex. The CycloFlex will show up as two COM Ports under the “Ports (COM &LPT)” under the Device Manager.



When this is complete, the drivers are installed and the CycloFlex can be used for programming and USB data transfers.

6 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.

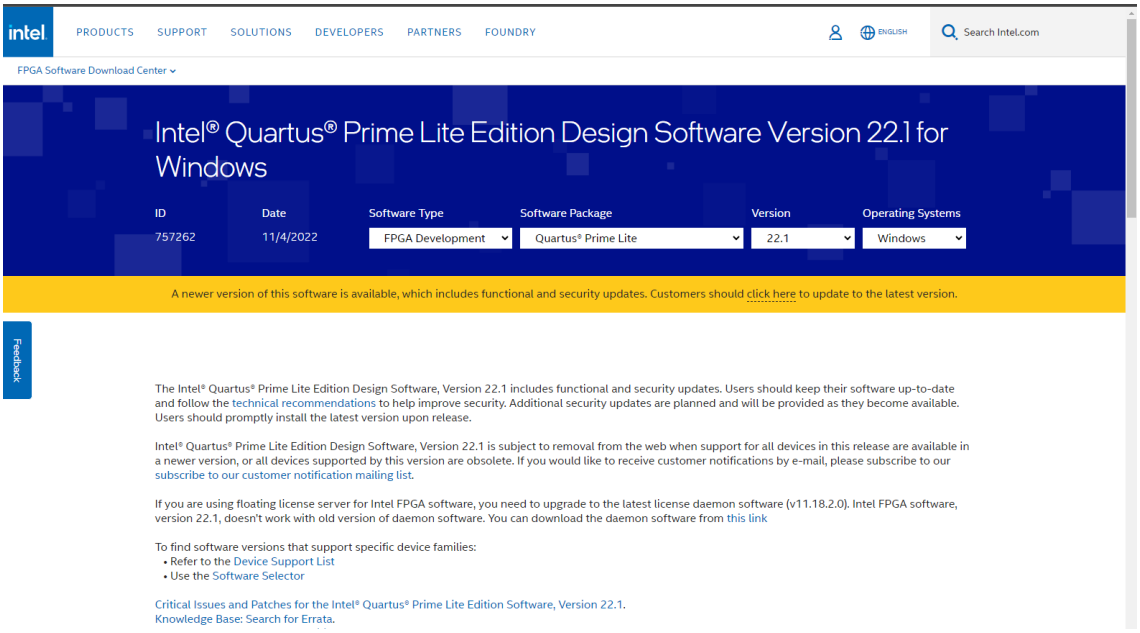
6.1.1 Downloading Quartus

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:

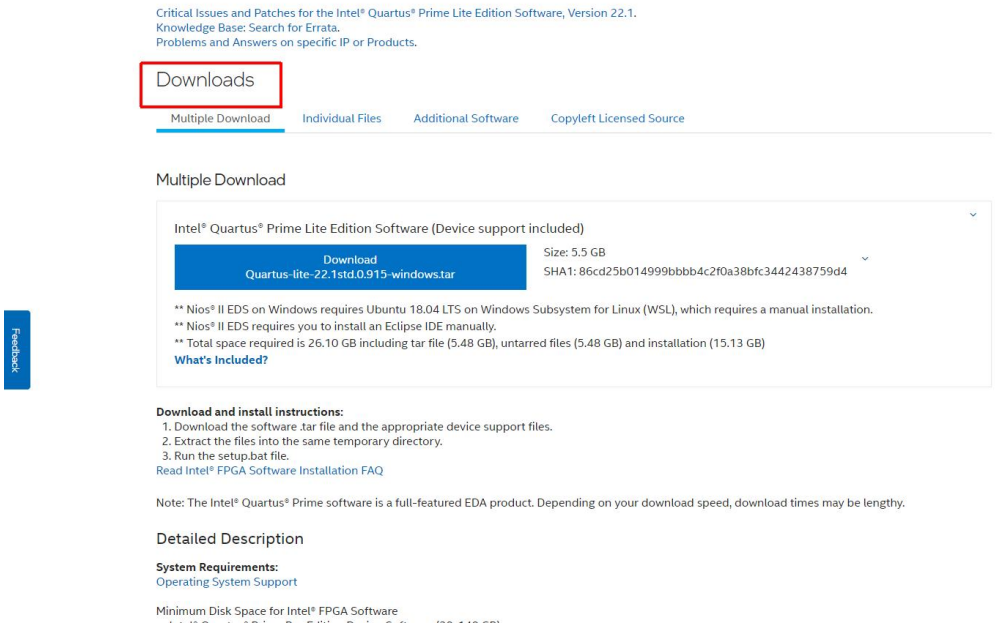
Intel FPGA Quartus Prime Lite

****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.****

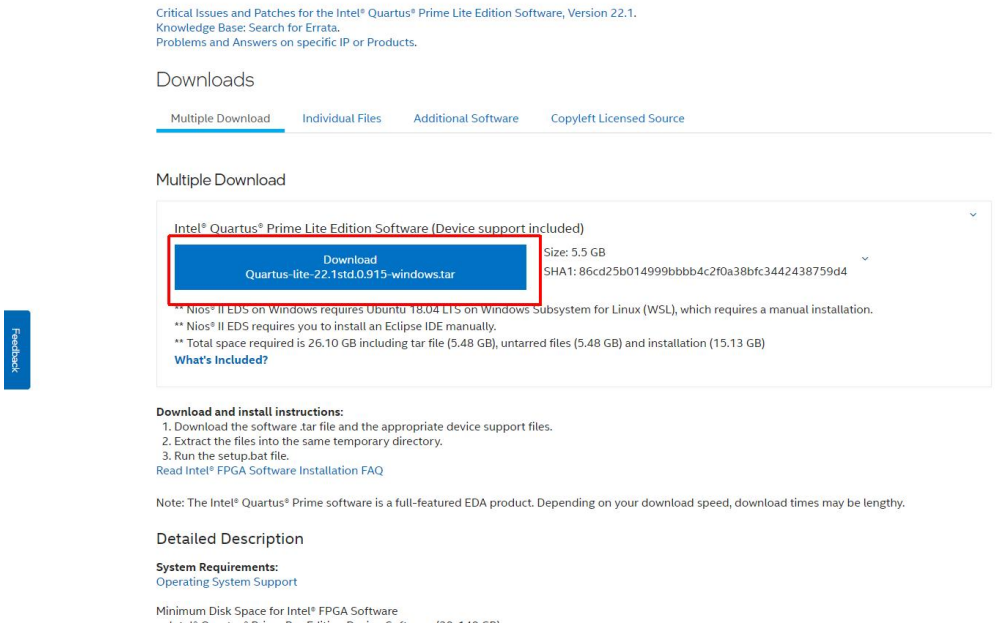
You will first need to apply for an account with Intel. Then use your login and password to access the download site. Click on the Download Windows Version.



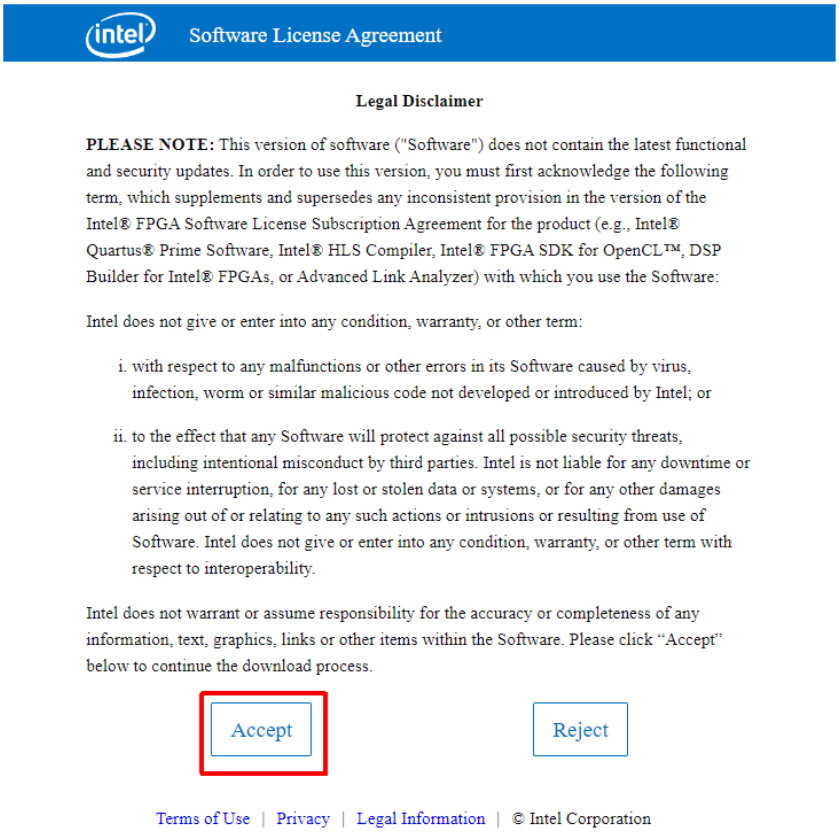
Scroll down the page to the “Downloads” section.



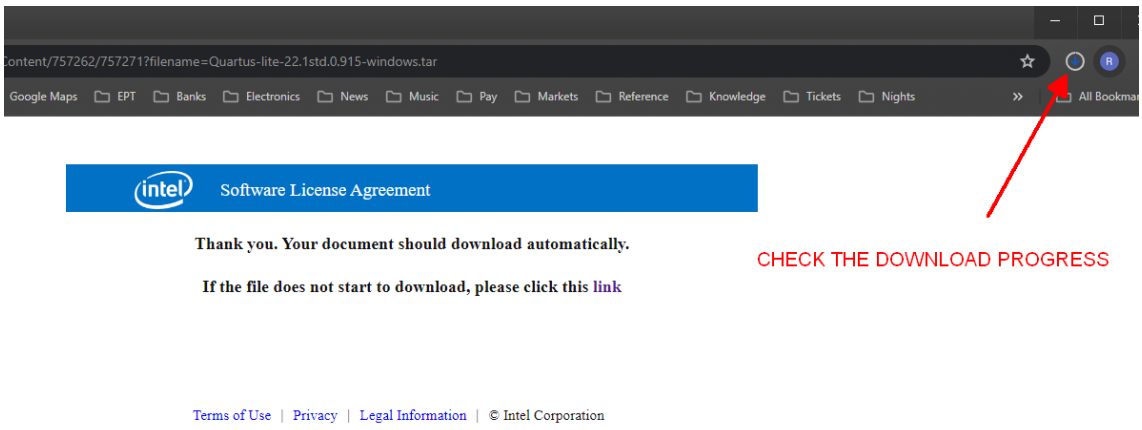
Click on the “Download Quartus-lite-22.1 xxxxxx Windows Tar.



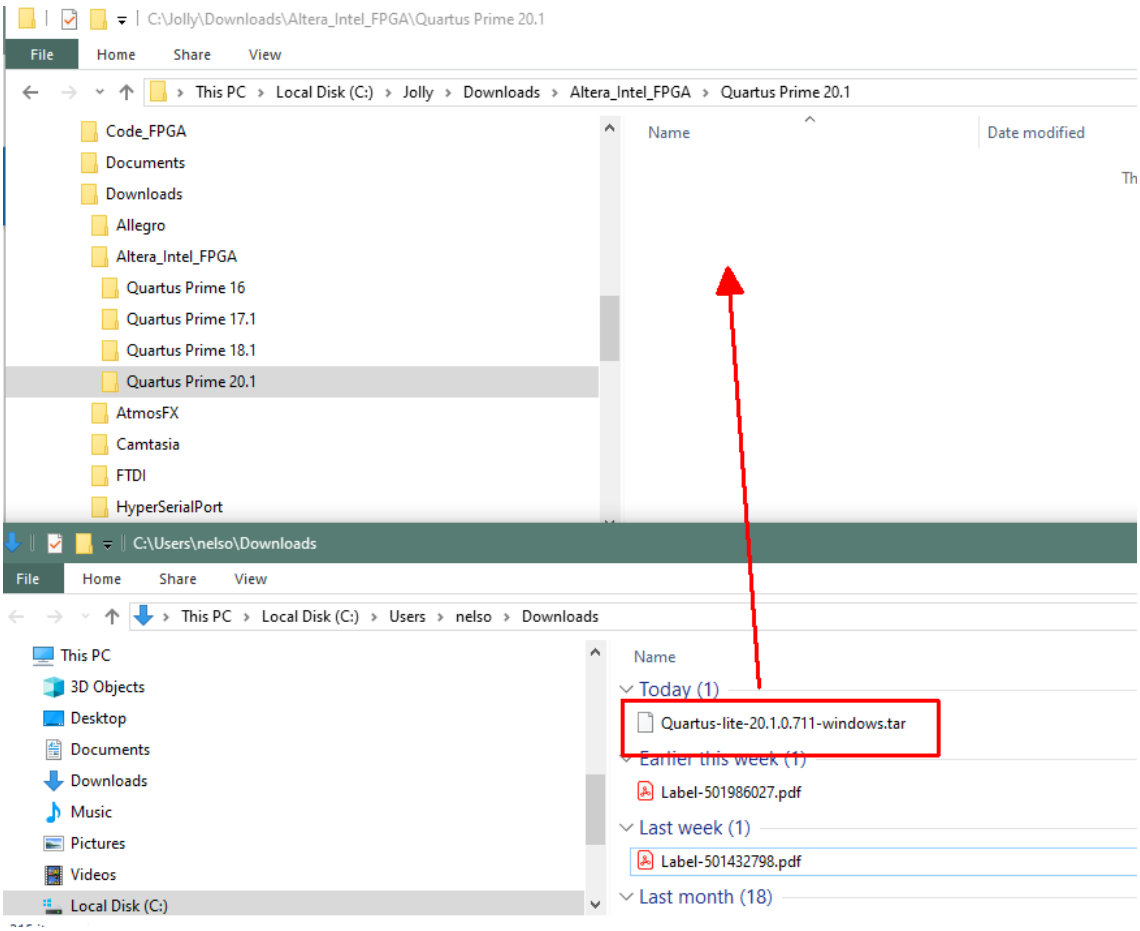
Click through the Legal Stuff.



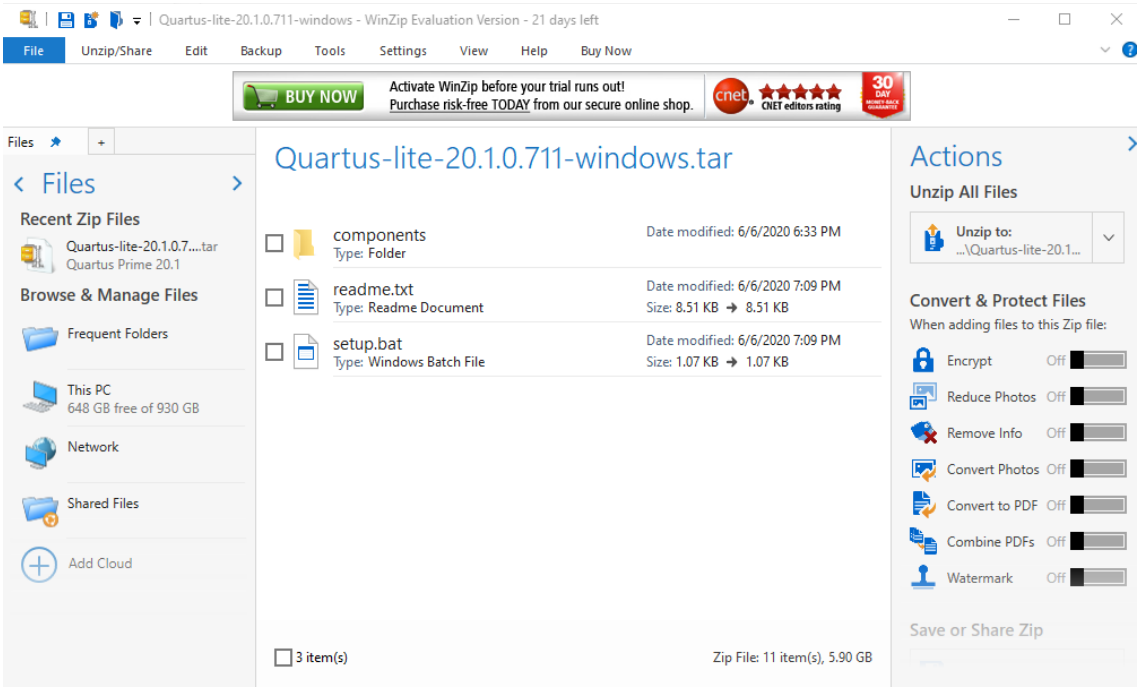
This will start the download.



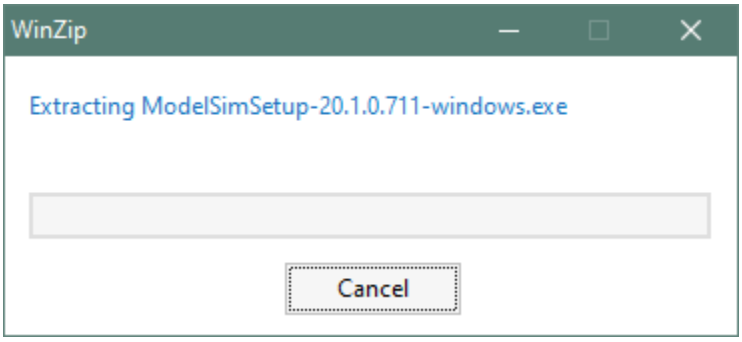
The file is 13.9 GB (or greater), so this could take a couple of hours depending on your internet connection. When download is complete, store the *.tar file in a directory on your PC.



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

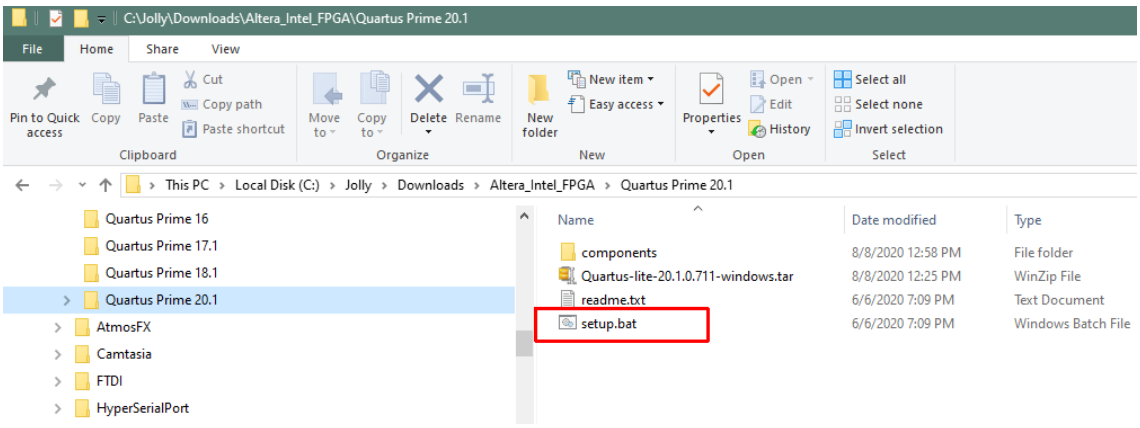


The tool will unpack all files.

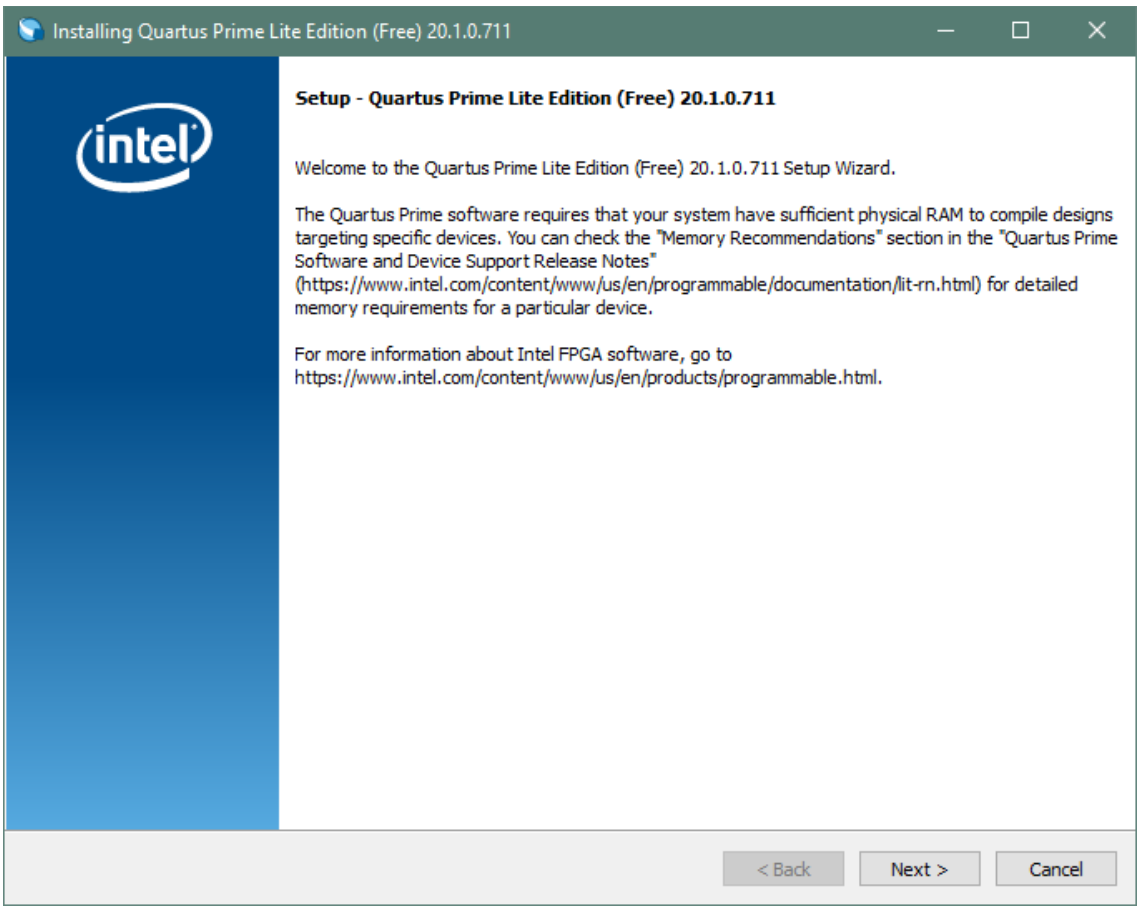


6.1.2 Quartus Installer

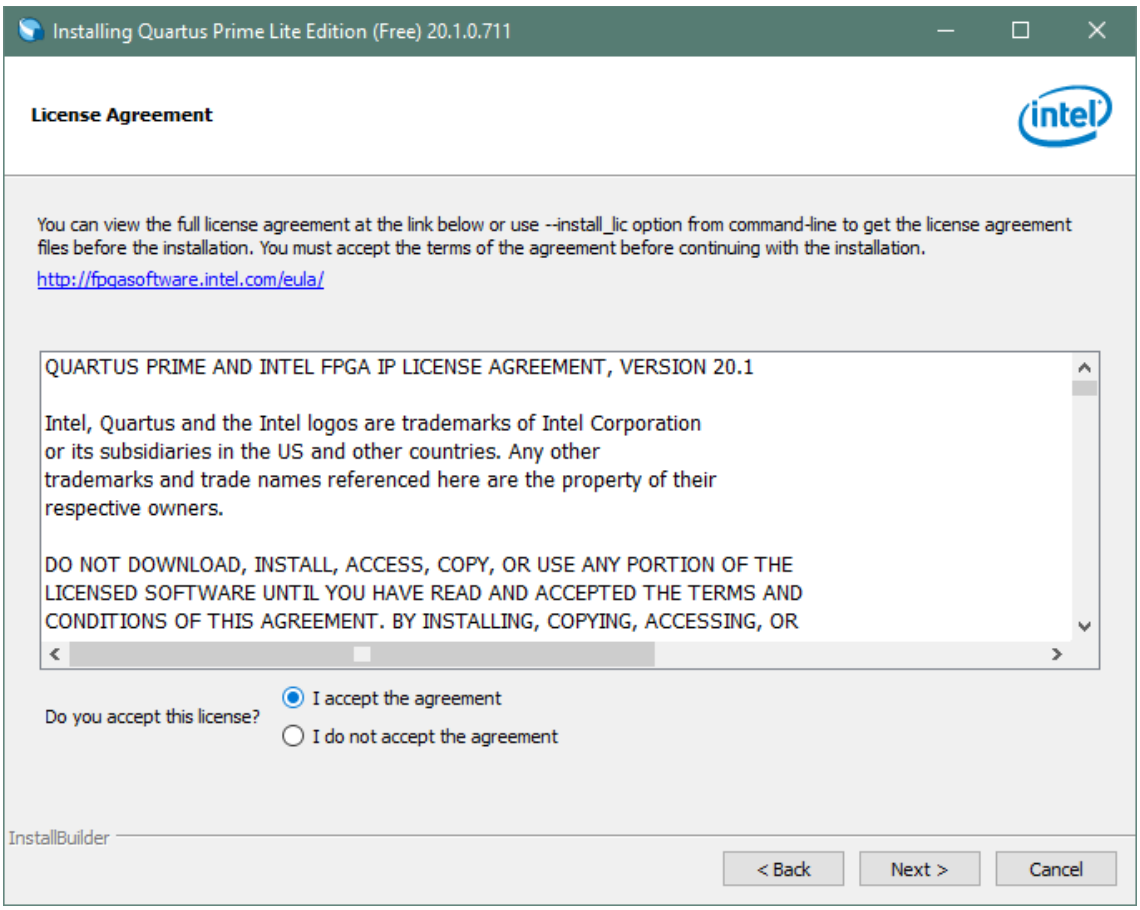
When the unpacking finishes from the previous section, double click the setup.bat file in the download folder.



Click “Next” on the Introduction Window.

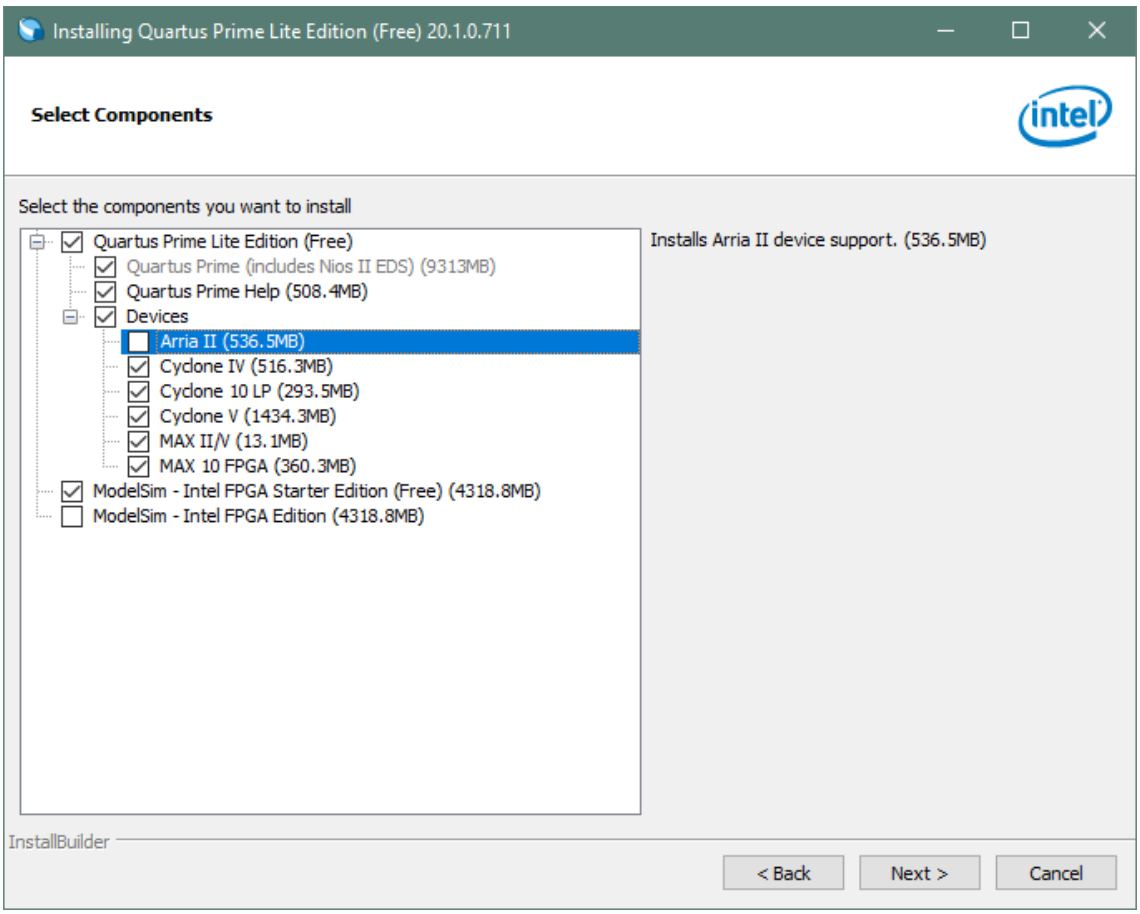


Click the checkbox to agree to the license terms. Then click “Next”.

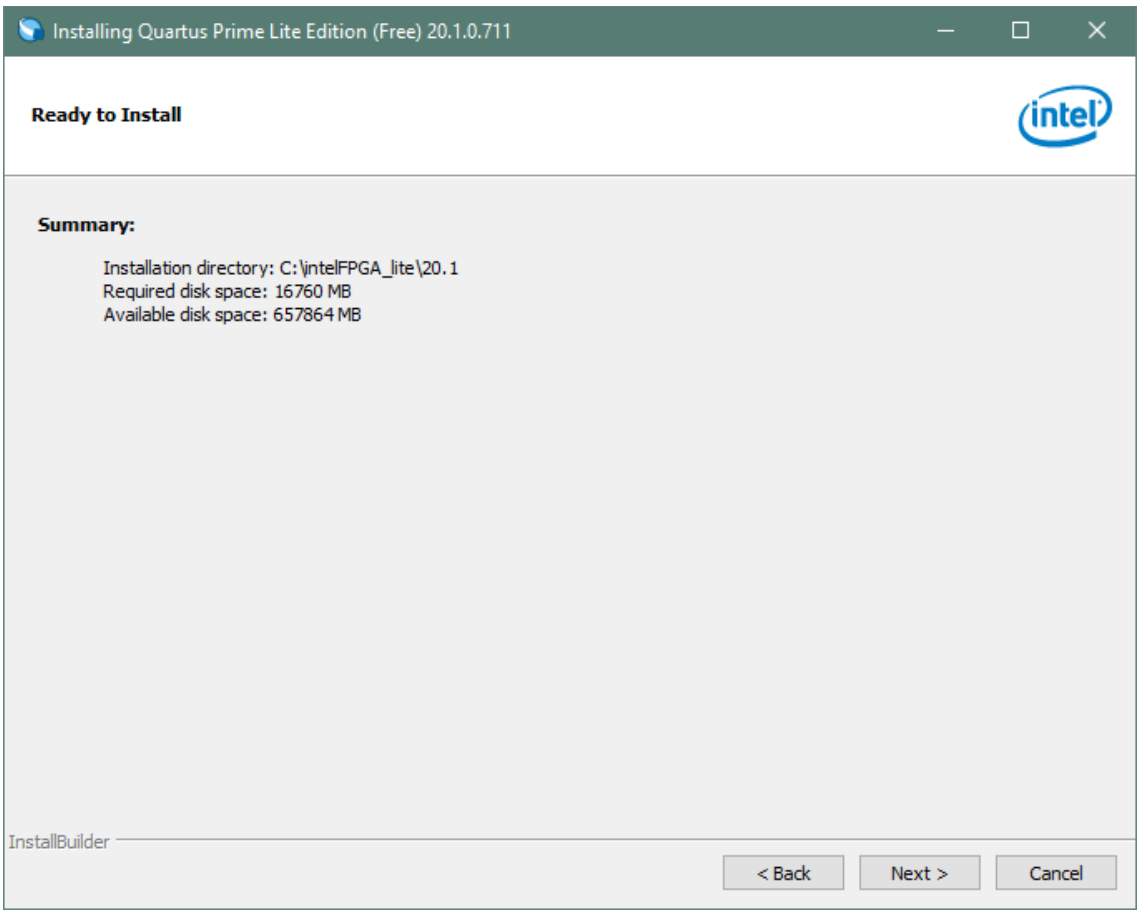


Click “Next” and accept the defaults.

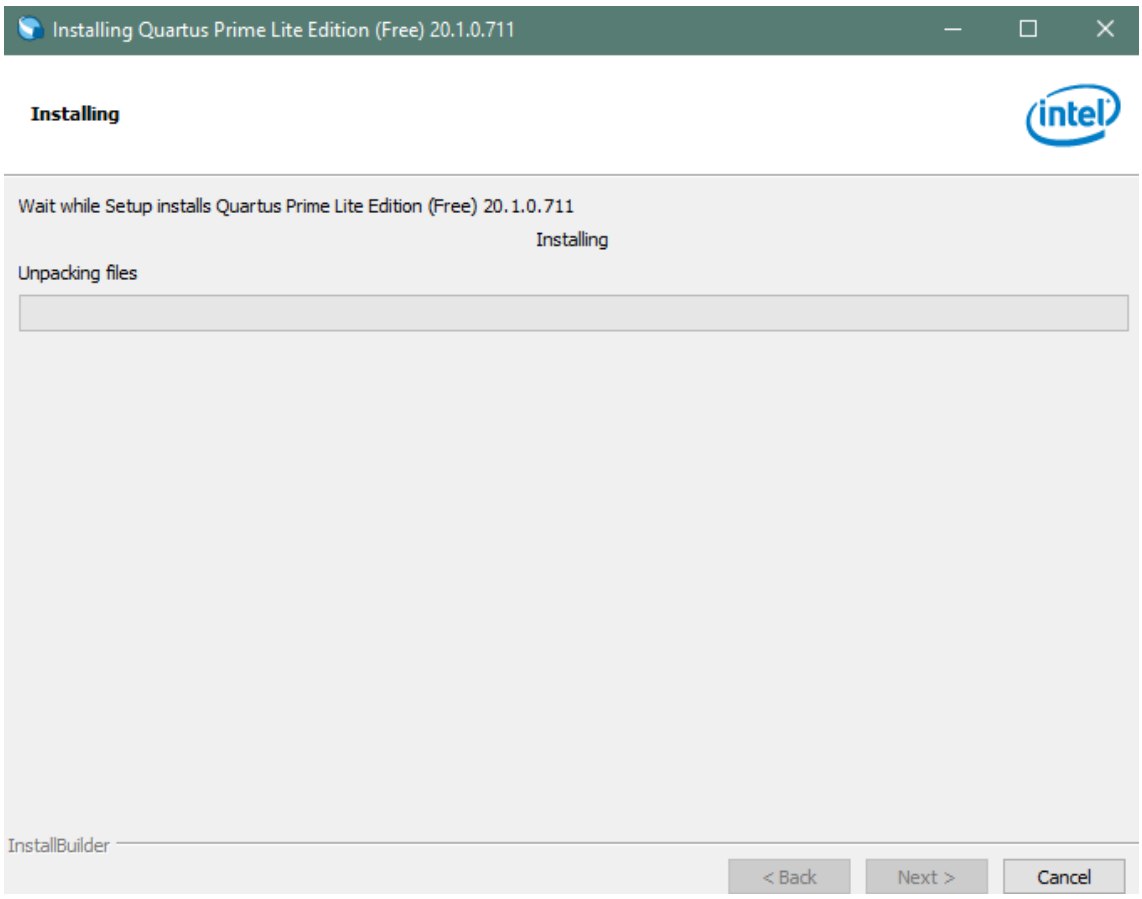
At the Select Products Window, de-select the Quartus Prime Supbscription Edition by clicking on its check box so that the box is not checked. Then click on the check box by the Quartus Prime Web Edition (Free).



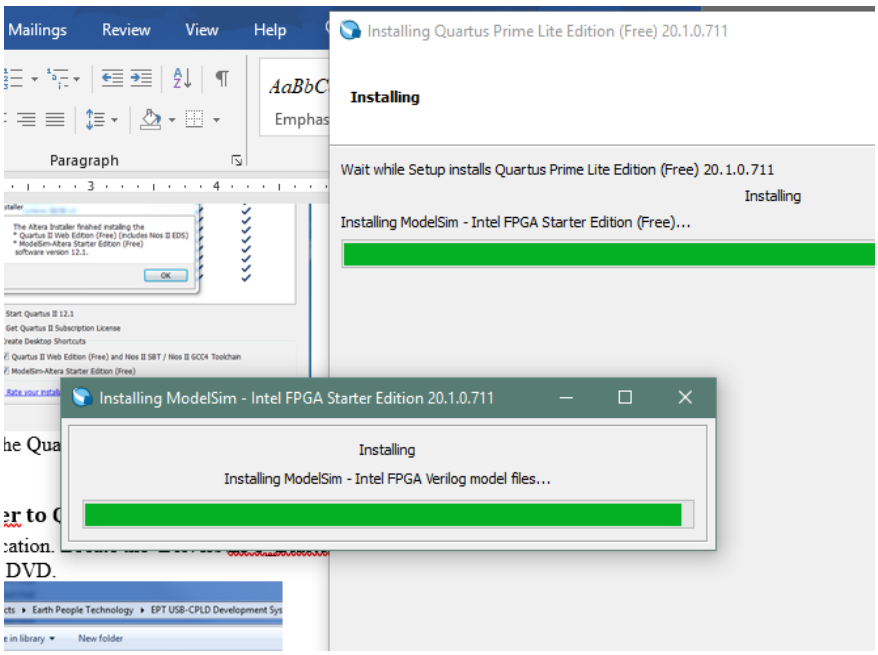
Click “Next” to accept the defaults

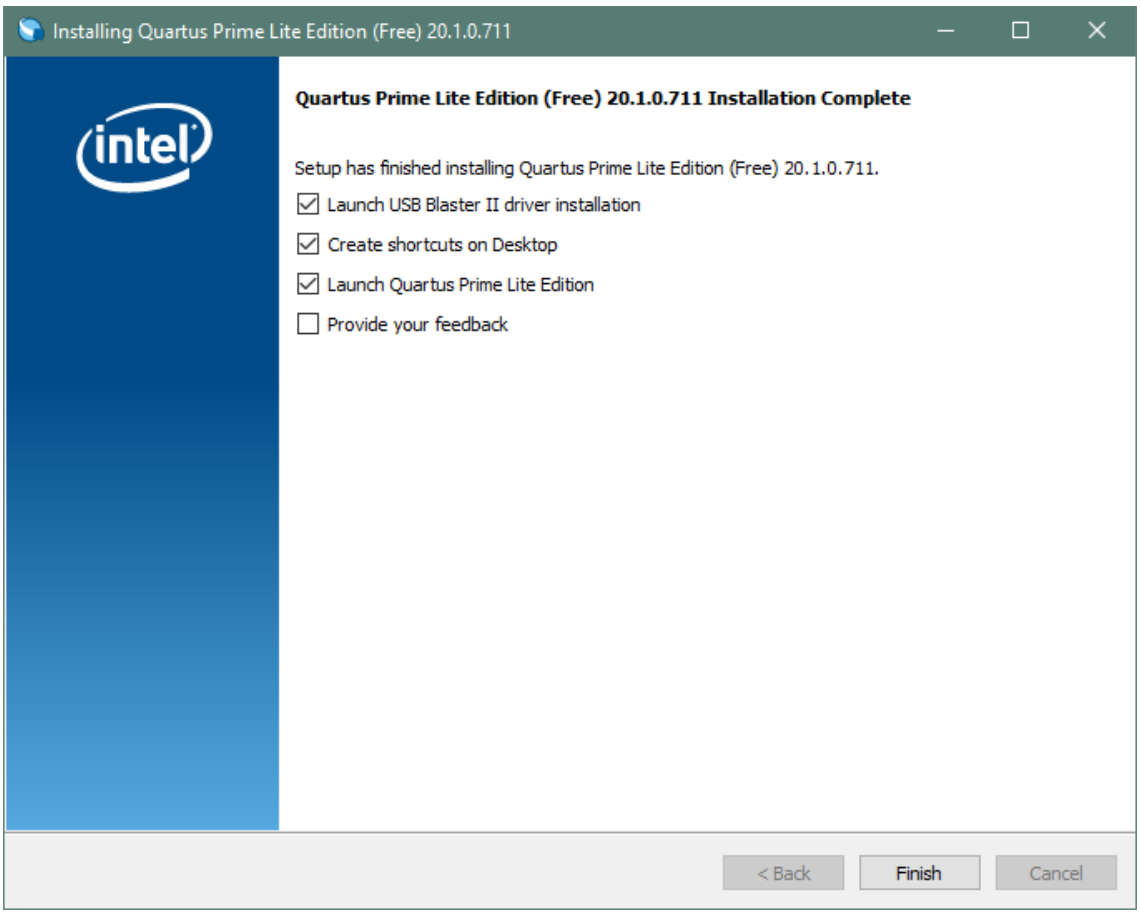


Click “Next” to accept the defaults

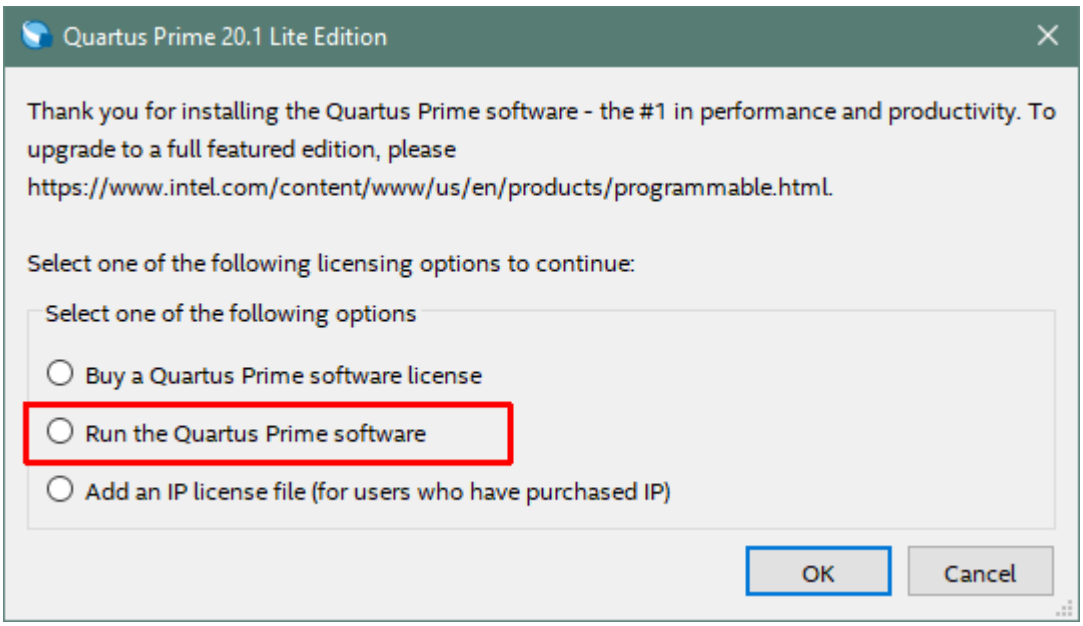


Wait for the installation to complete.

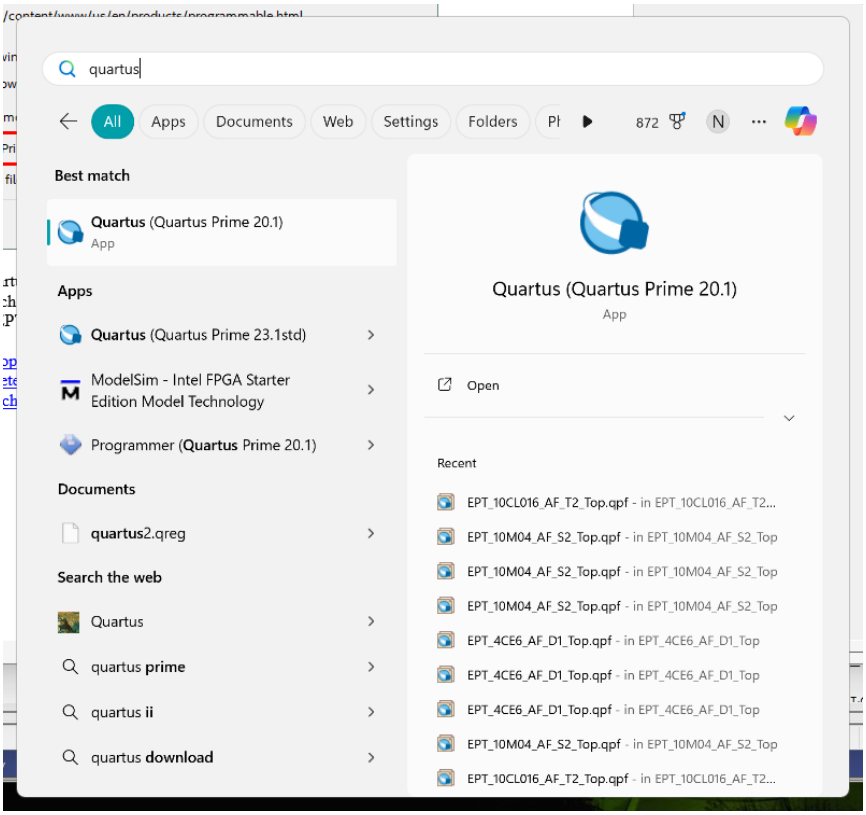

Mailings
Review
View
Help
Paragraph
Installing Quartus Prime Lite Edition (Free) 20.1.0.711
Installing
Wait while Setup installs Quartus Prime Lite Edition (Free) 20.1.0.711
Installing
Installing ModelSim - Intel FPGA Starter Edition (Free)...
Installing ModelSim - Intel FPGA Starter Edition 20.1.0.711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

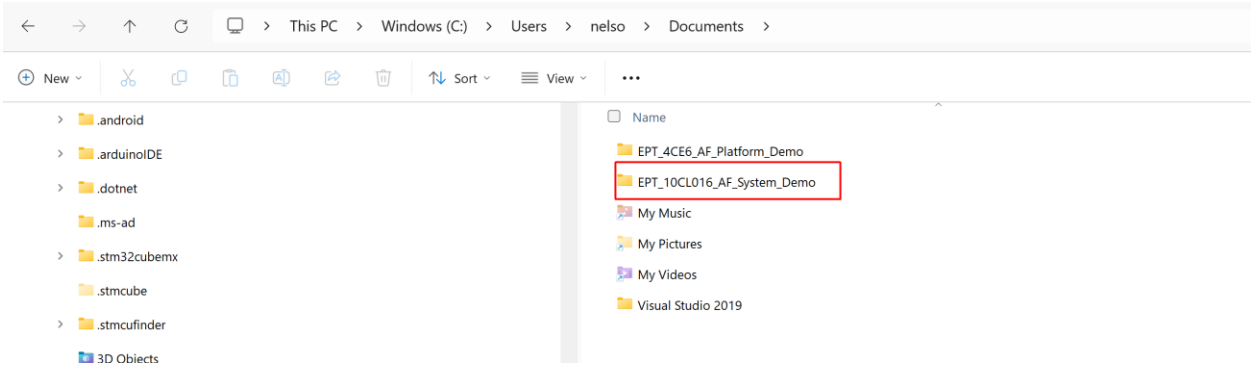
https://www.earthpeopletechnology.com->Forums
support@earthpeopletechnology.com
sales@earthpeopletechnology.com

7 Compiling and Synthesizing the FPGA Project

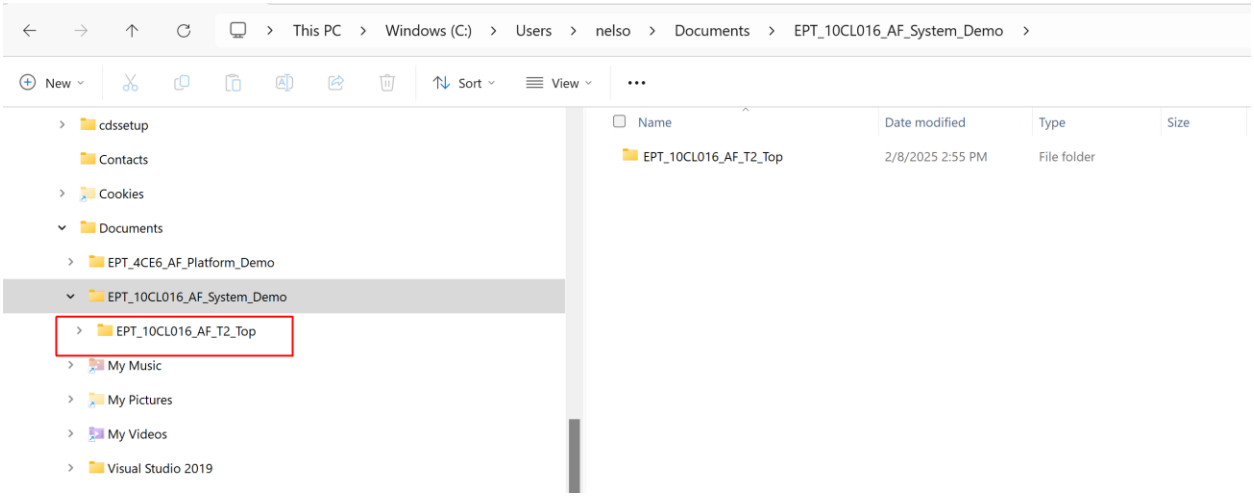
With the Quartus Prime Lite installed on the user PC, setting up and project and compiling and synthesizing is the next step. Once the user HDL code is written according to the syntax rules of the language (Verilog and VHDL) it can be compiled and synthesized using the Quartus Prime software. This manual will not focus on HDL coding or proper coding techniques, instead it will use the example code to compile, synthesize and program the FPGA.

7.1 Setting up the Project and Compiling

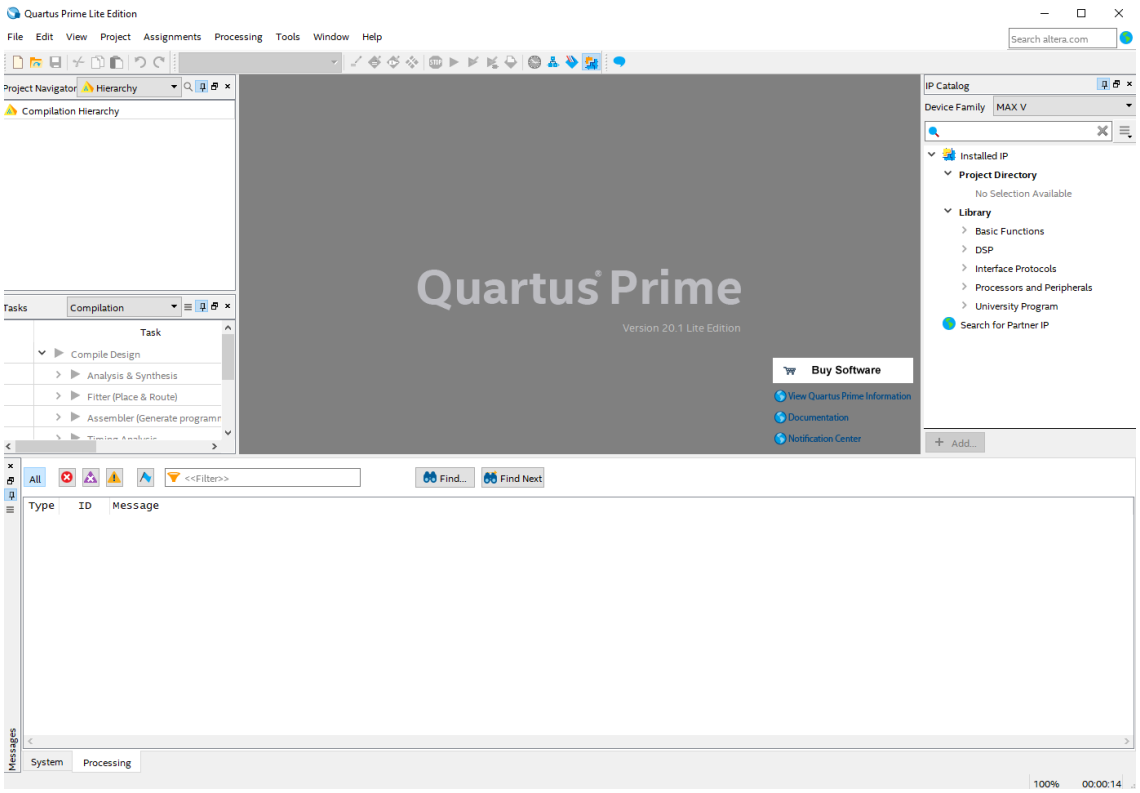
Once the HDL code (Verilog or VHDL) is written and verified using a simulator, a project can be created using Quartus Prime. Writing the HDL code and simulating it will be covered in later sections. Bring up Quartus Prime, then use Windows Explorer to browse to C:\Users\nelso\Documents\ to create a new directory called: “EPT_10CL016_AF_System_Demo”.



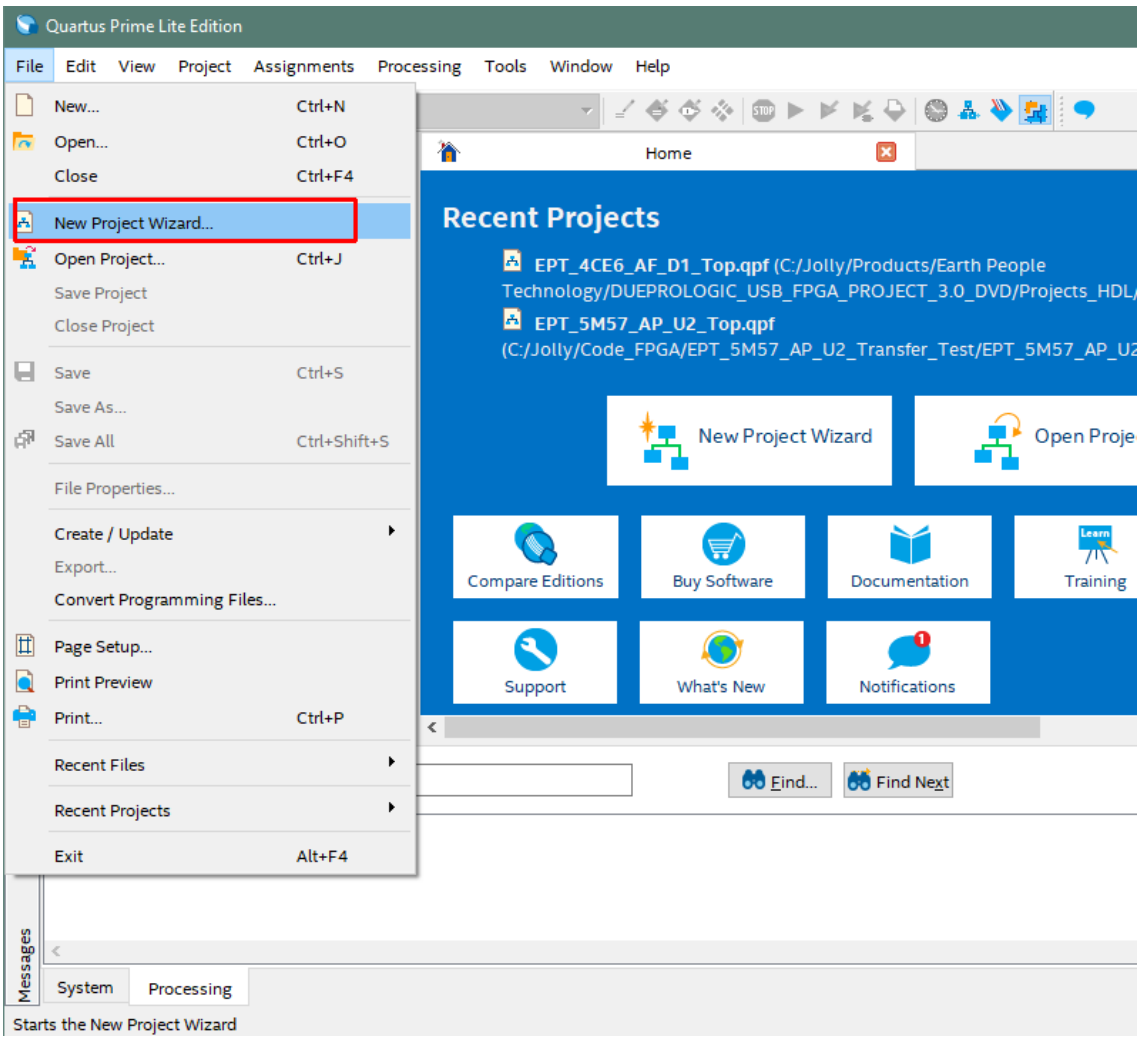
Create another directory under the EPT_10CL016_AF_System_Demo called: EPT_10CL016_AF_T2_Top



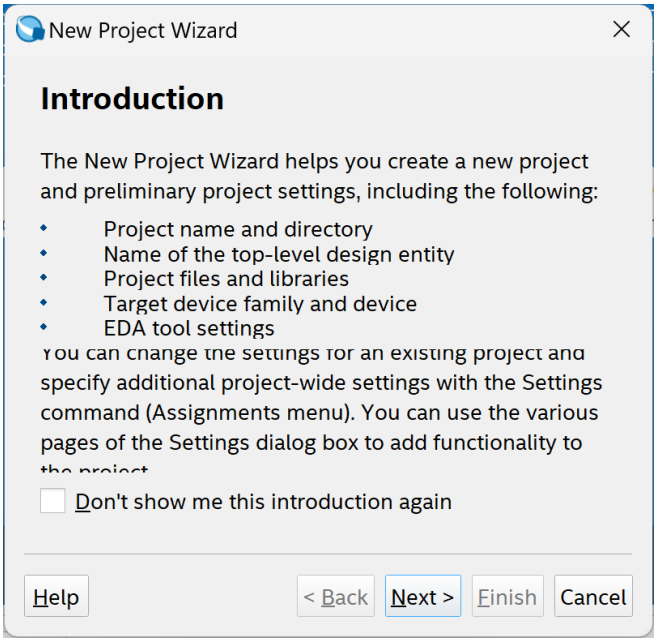
Open Quartus Prime by clicking on the icon.



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

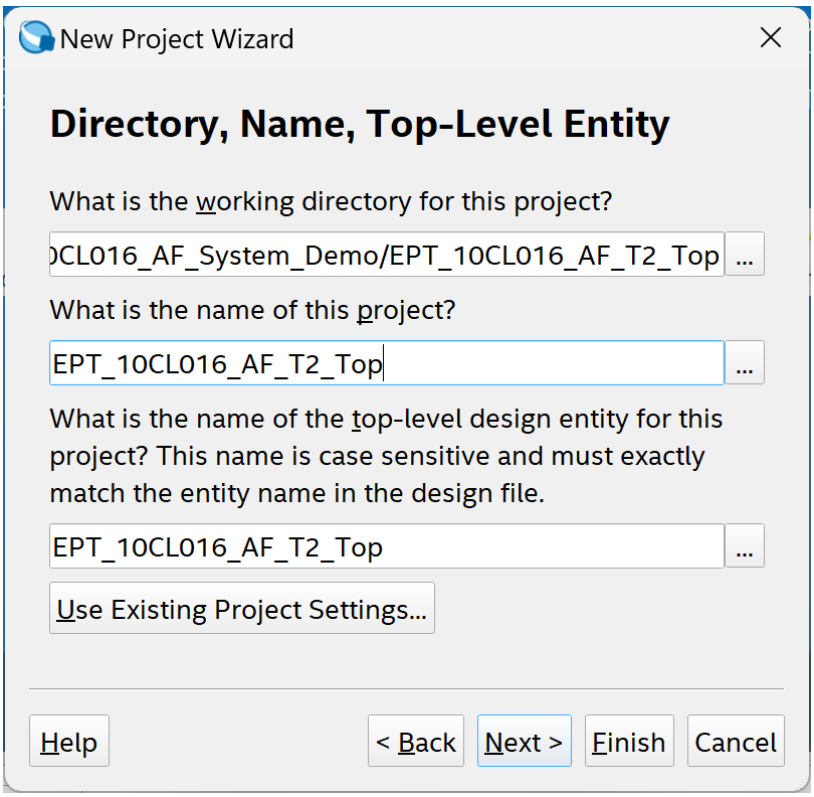


The “Introduction” window will open. Click “Next”.

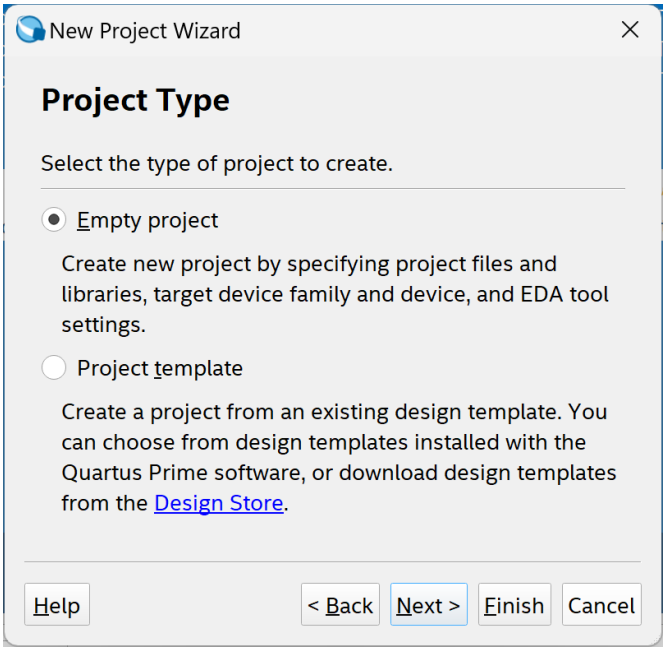


At the Top-Level Entity page, browse to the C:\Users\nelso\Documents\EPT_10CL016_AF_System_Demo\EPT_10CL016_AF_System_Demo

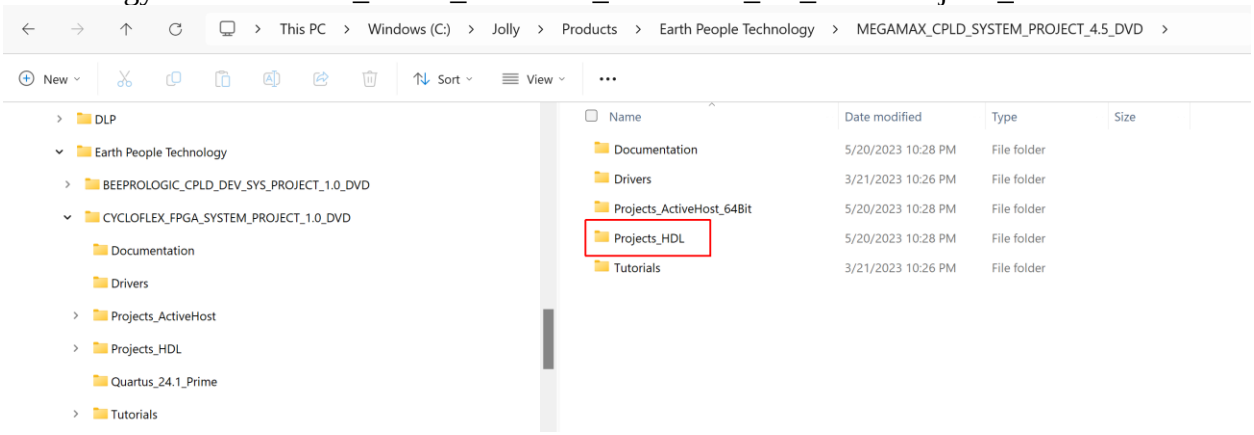
directory to store your project. Type in a name for your project “EPT_10CL016_AF_T2_Top”.



Make sure you selected the “EPT_10CL016_AF_T2_Top” folder
Select Next.
Choose the “Empty project”

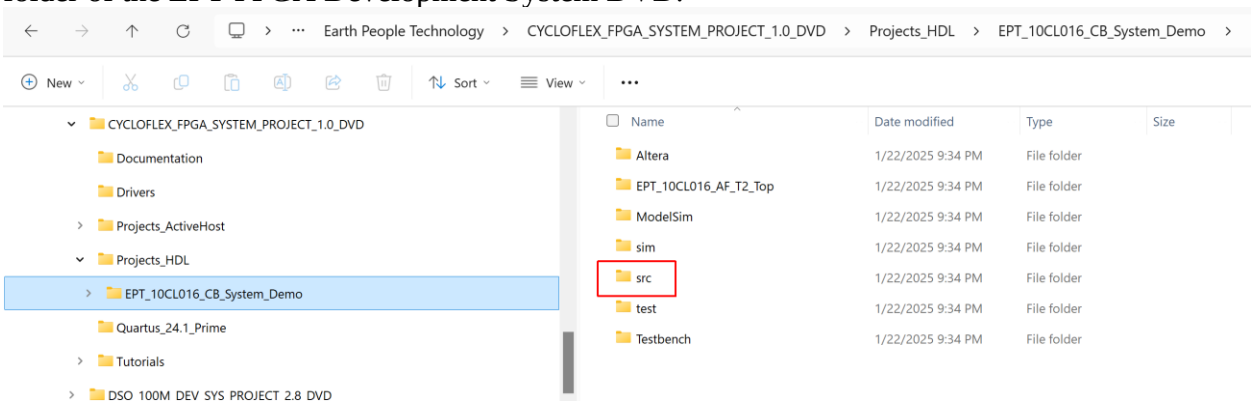


Next, the “Add Files” window appears. Here we need to bring up a Windows Explorer window and browse to the C:\$\$\Earth People Technology\CYCLOFLEX_FPGA_SYSTEM_PROJECT_1.0_DVD\Projects_HDL folder.

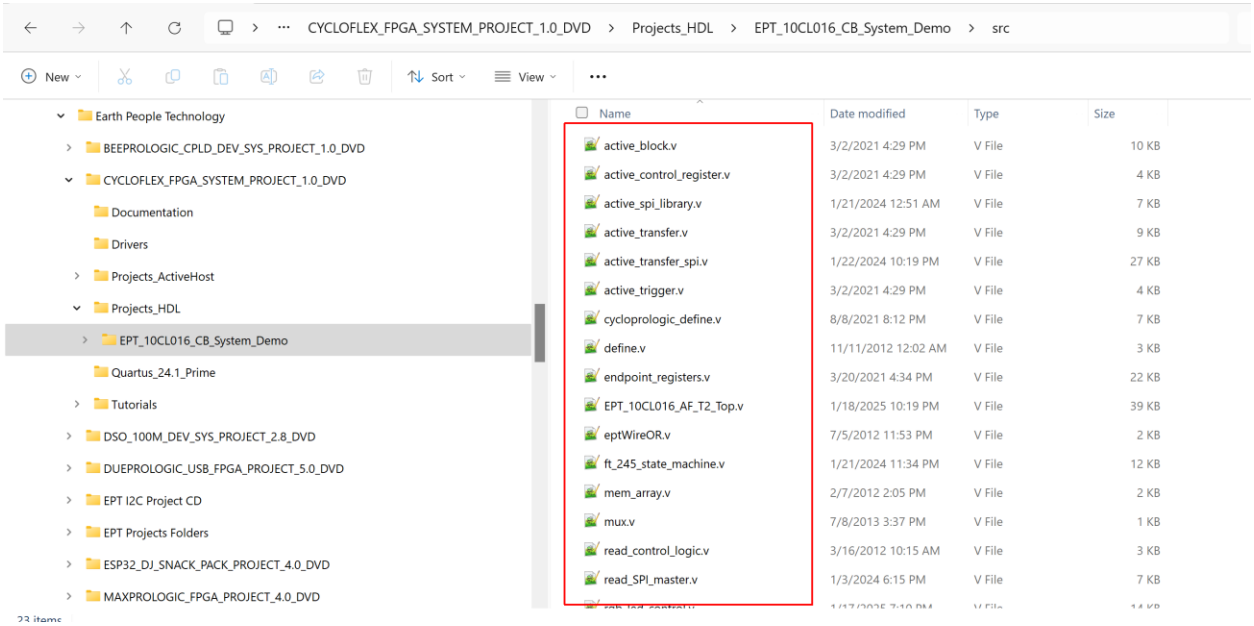


In the Projects_HDL\EPT_10CL016_AF_System_Demo\src

folder of the EPT FPGA Development System DVD.



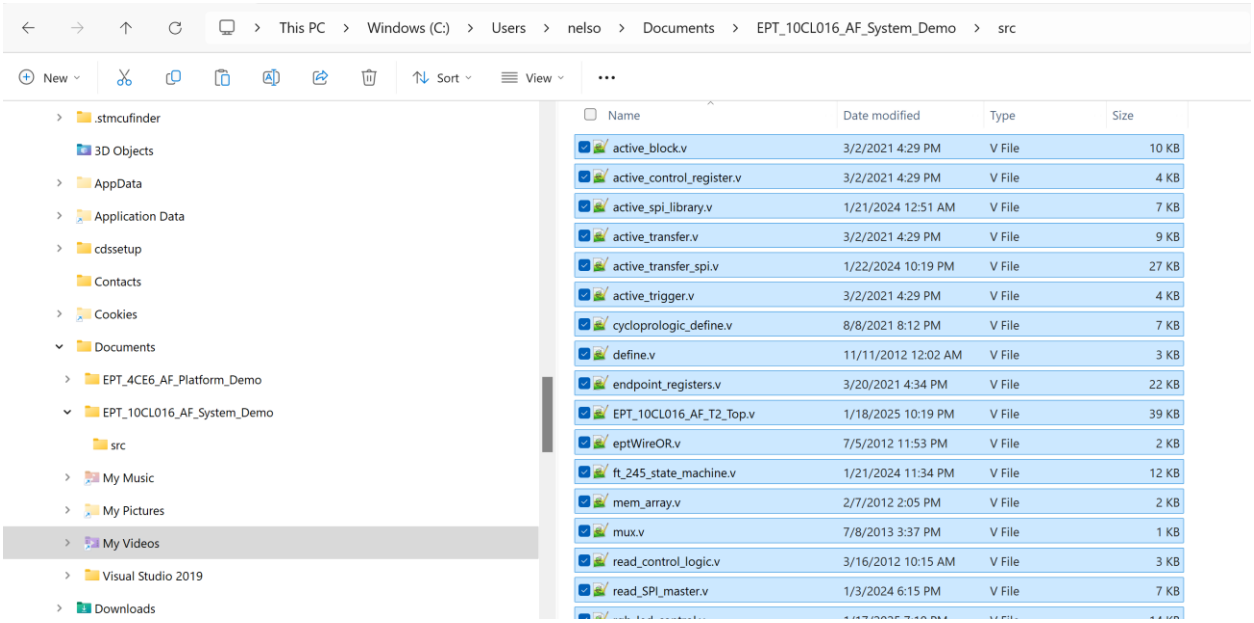
Copy the files from the \src directory.



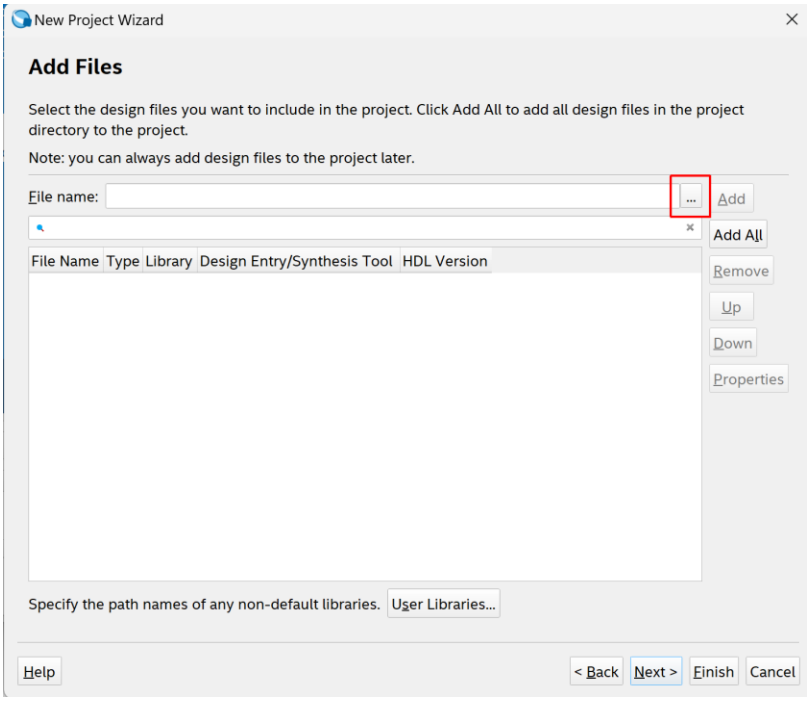
- active_block.v
- active_control_register.v

- active_spi_library.v
- active_transfer.v
- active_transfer_spi.v
- active_trigger.v
- cycloprologic_define.v
- define.v
- endpoint_registers.v
- EPT_10CL016_AF_T2_Top.v
- eptWireOR.v
- ft_245_state_machine.v
- mem_array.v
- mux.v
- read_control_logic.v
- read_SPI_master.v
- rgb_led_control.v
- serial_clock.v
- seven_segment_led_display.v
- spi_iface_master.v
- sync_fifo.v
- write_control_logic.v
- write_SPI_master.v

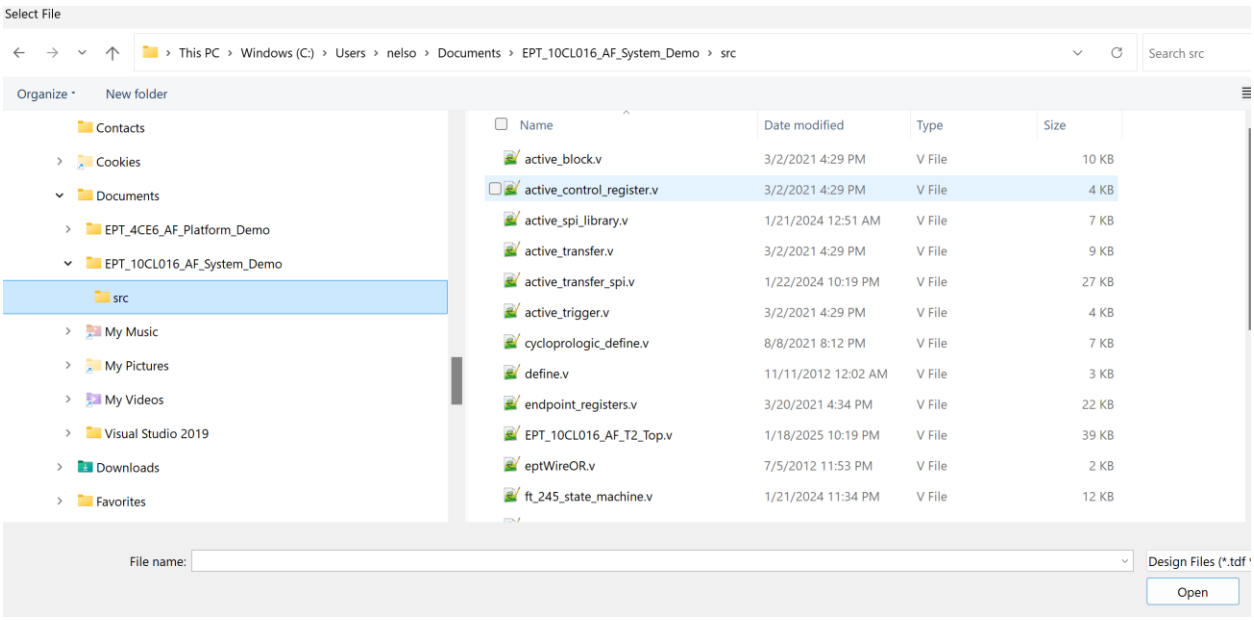
Then paste them into a “src” folder on the C:\Users\nelso\Documents\EPT_10CL016_AF_System_Demo\src



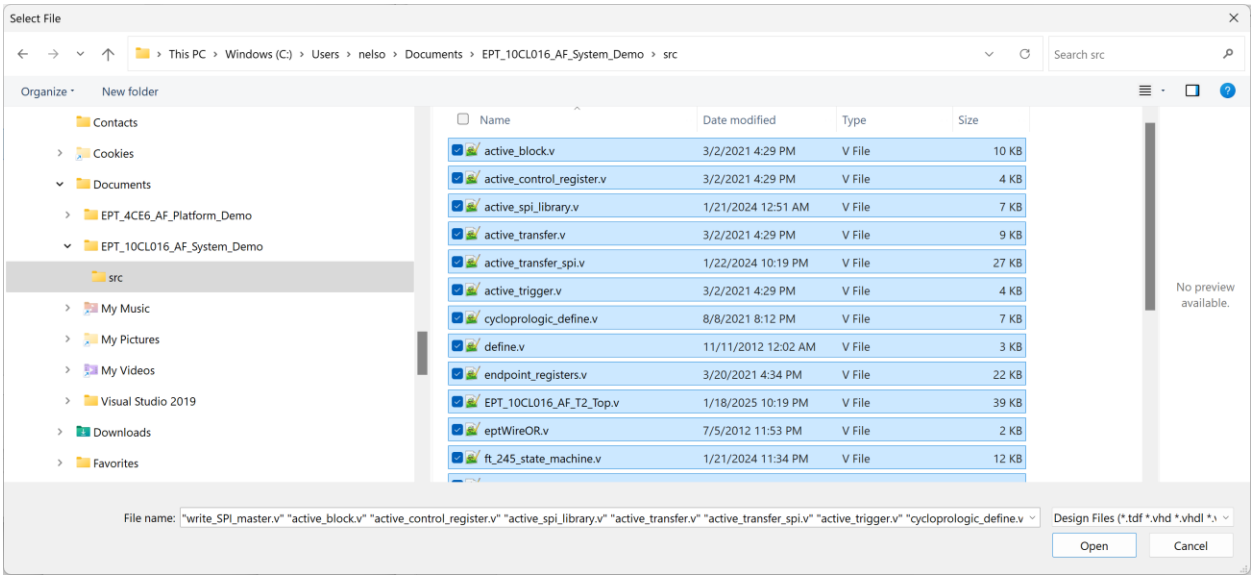
Then, go back to the Quartus “Add Files” window. Click on the “…” button across from the “File name:” textbox.



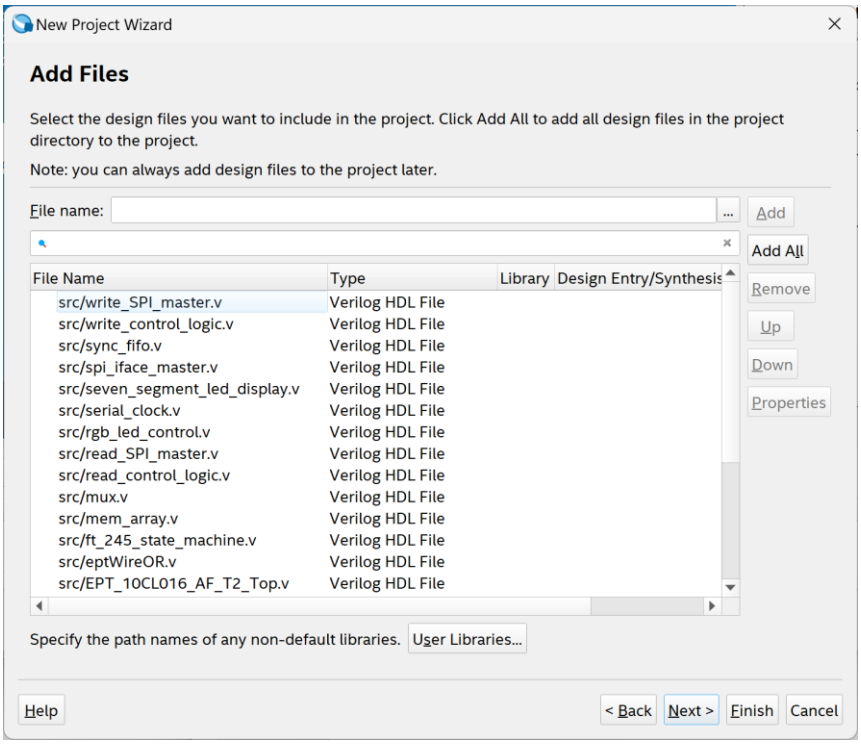
Navigate to the C:\Users\nelso\Documents\EPT_10CL016_AF_System_Demo\src folder



Then select all files in the “src” folder.

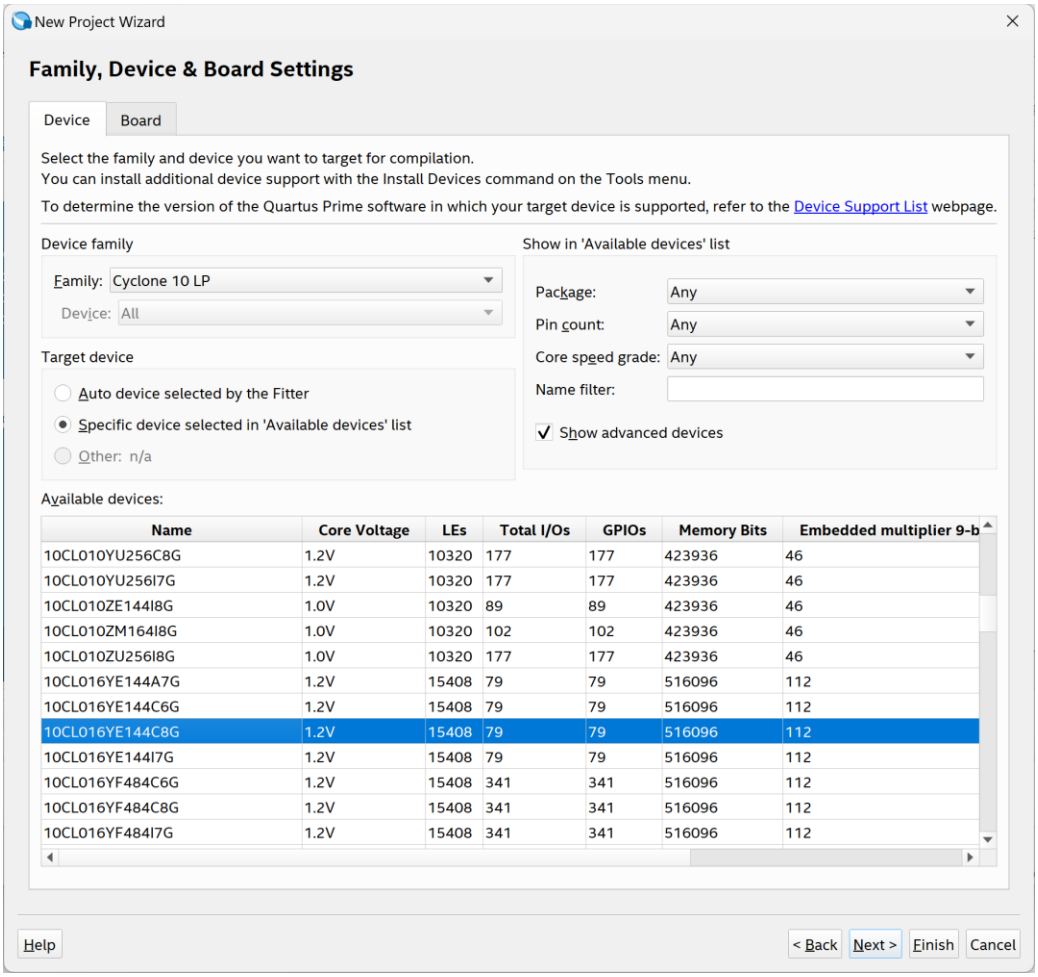


Click “Open”

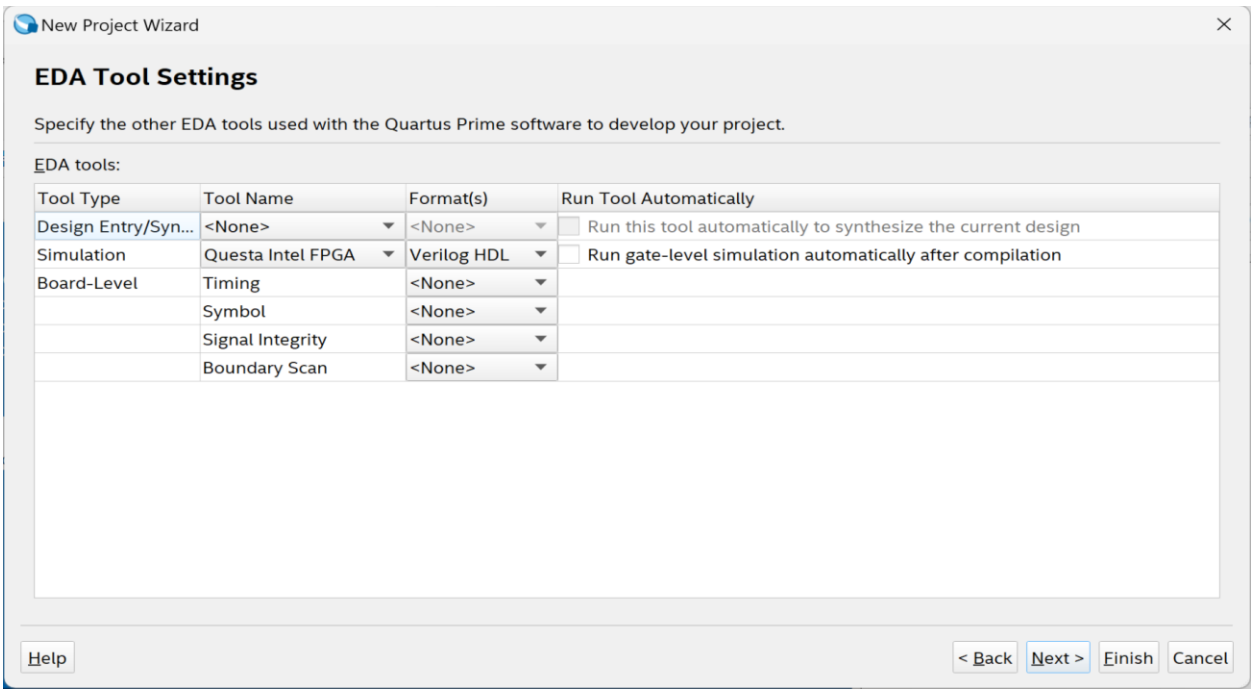


Now, all source files for the project are in the Quartus Prime project.

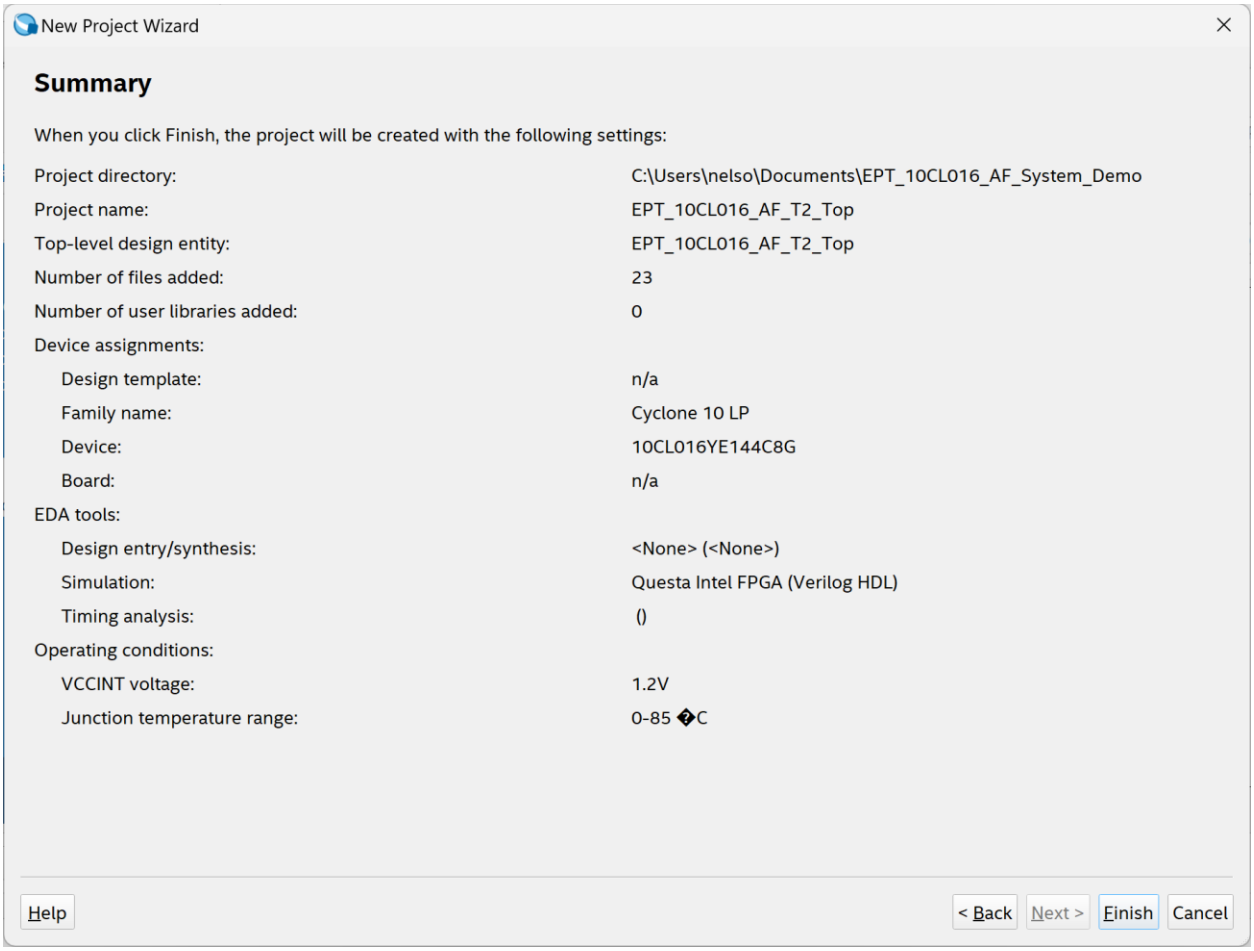
Select Next, at the Device Family group, select Cyclone 10 for Family. In the Available Devices group, browse down to 10CL016YE144C8G for Name.



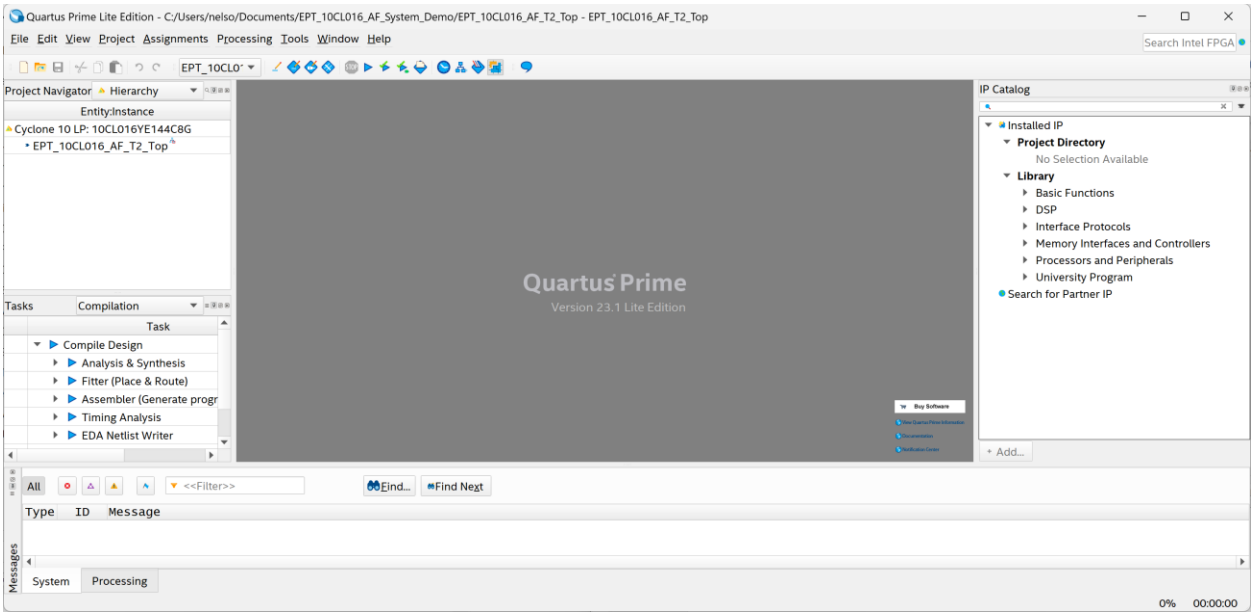
Select Next, leave defaults for the EDA Tool Settings.



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



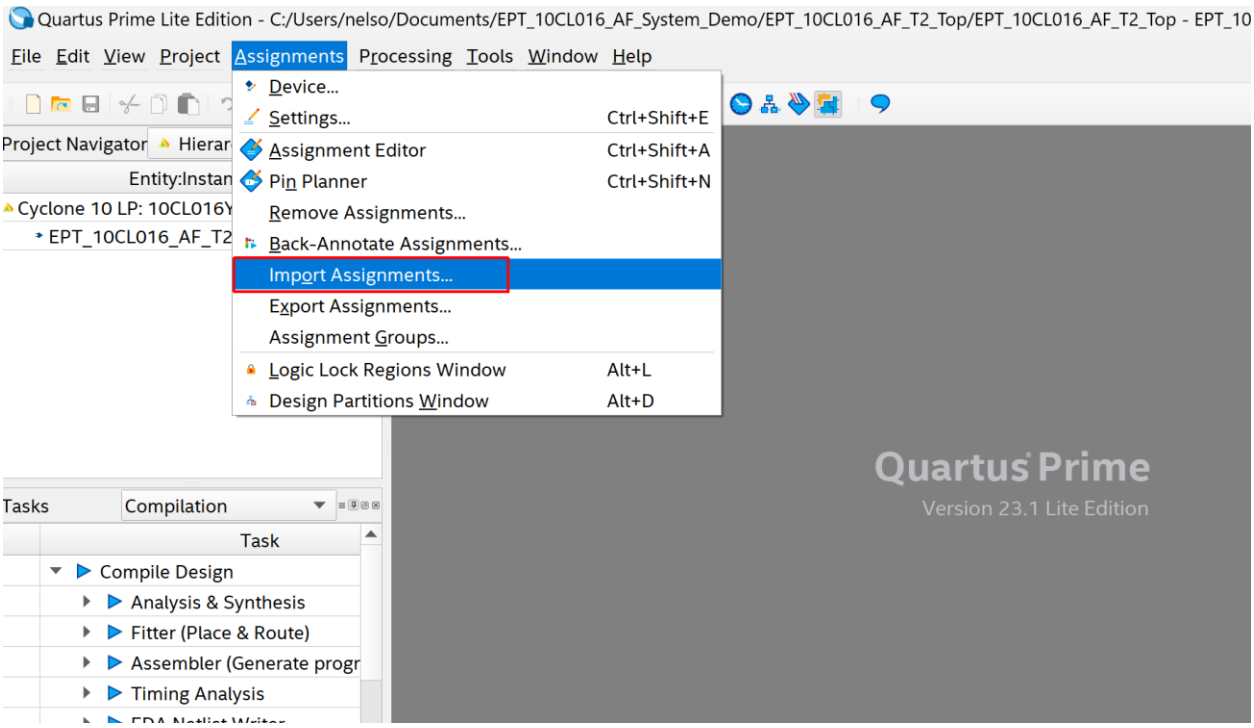
Now, your project is ready to go.



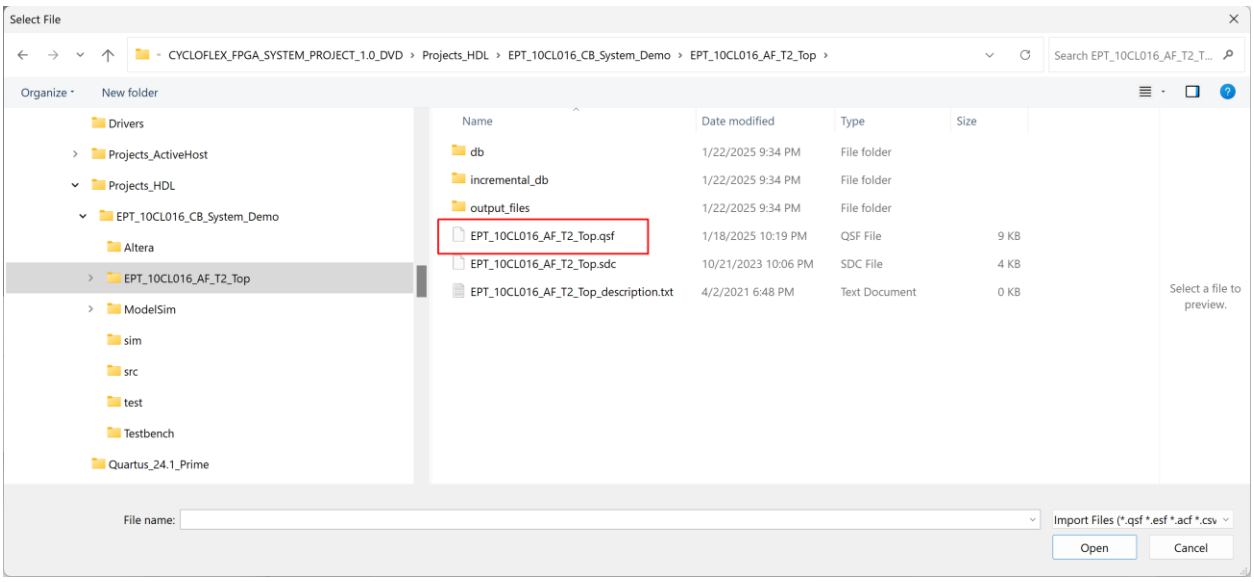
Next, we will select the pins and synthesize the project.

7.1.1 Selecting Pins and Synthesizing

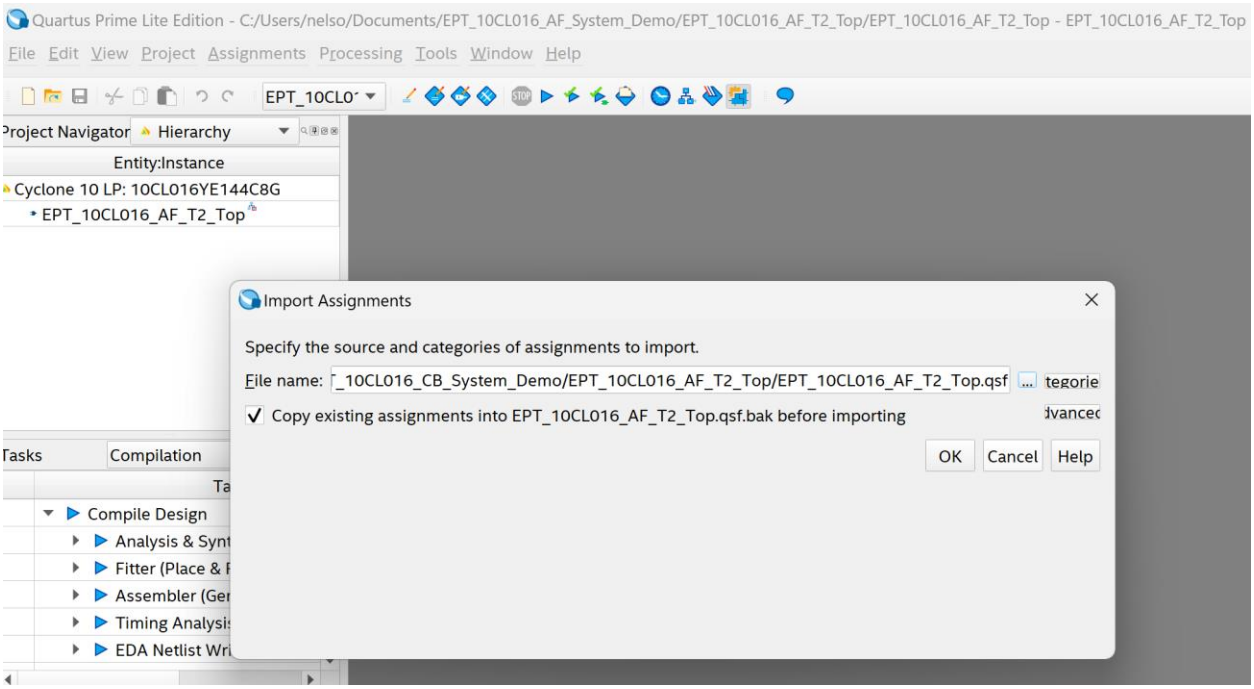
With the project created, we need to assign pins to the project. The signals defined in the top level file (in this case: EPT_10CL016_AF_T2_Top.v) will connect directly to pins on the FPGA. The Pin Planner Tool from Quartus Prime will add the pins and check to verify that our pin selections do not violate any restrictions of the device. In the case of this example we will import pin assignments that created at an earlier time. Under Assignments, Select Import Assignments.



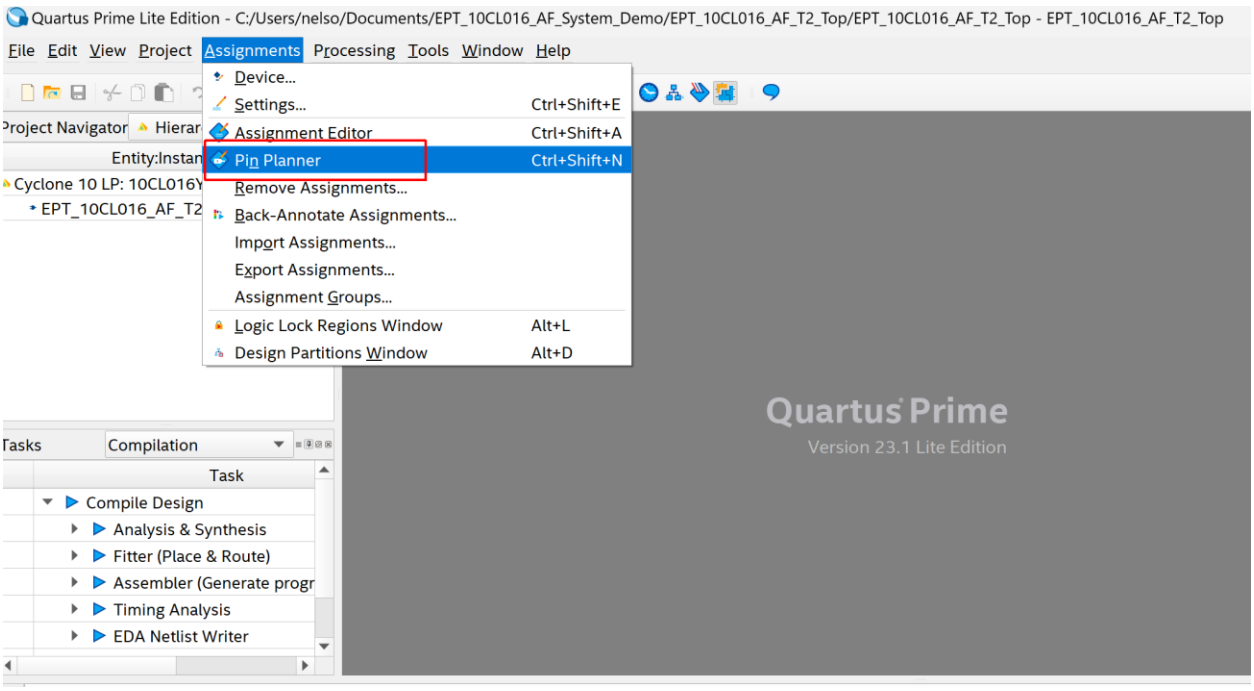
At the Import Assignment dialog box, Browse to the \Projects_HDL\EPT_System_Demo \ EPT_10CL016_AF_T2_TOP folder of the EPT FPGA Development System DVD. Select the "EPT_10CL016_AF_T2_Top.qsf" file.


Select File
CYCLOFLEX_FPGA_SYSTEM_PROJECT_1.0_DVD › Projects_HDL › EPT_10CL016_CB_System_Demo › EPT_10CL016_AF_T2_Top ›
Search EPT_10CL016_AF_T2_T...
Organize
New folder
Drivers
Projects_ActiveHost
Projects_HDL
EPT_10CL016_CB_System_Demo
Altera
EPT_10CL016_AF_T2_Top
ModelSim
sim
src
test
Testbench
Quartus_24.1_Prime
Name
Date modified
Type
Size
db
1/22/2025 9:34 PM
File folder
incremental_db
1/22/2025 9:34 PM
File folder
output_files
1/22/2025 9:34 PM
File folder
EPT_10CL016_AF_T2_Top.qsf
1/18/2025 10:19 PM
QSF File
9 KB
EPT_10CL016_AF_T2_Top.sdc
10/21/2023 10:06 PM
SDC File
4 KB
EPT_10CL016_AF_T2_Top_description.txt
4/2/2021 6:48 PM
Text Document
0 KB
Select a file to preview.
File name:
Import Files (*.qsf *.esf *.acf *.csv
Open
Cancel

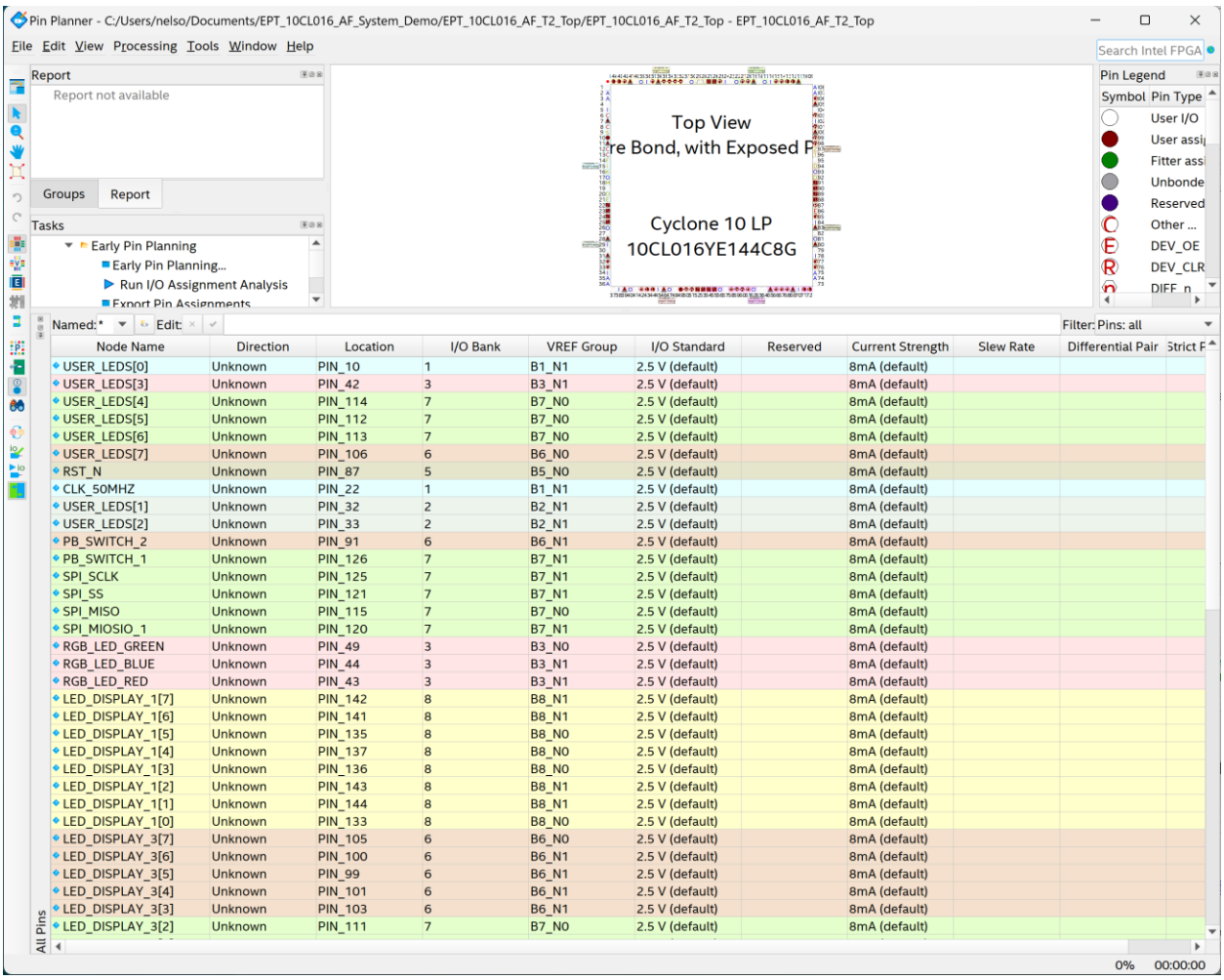
Click “Open”



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



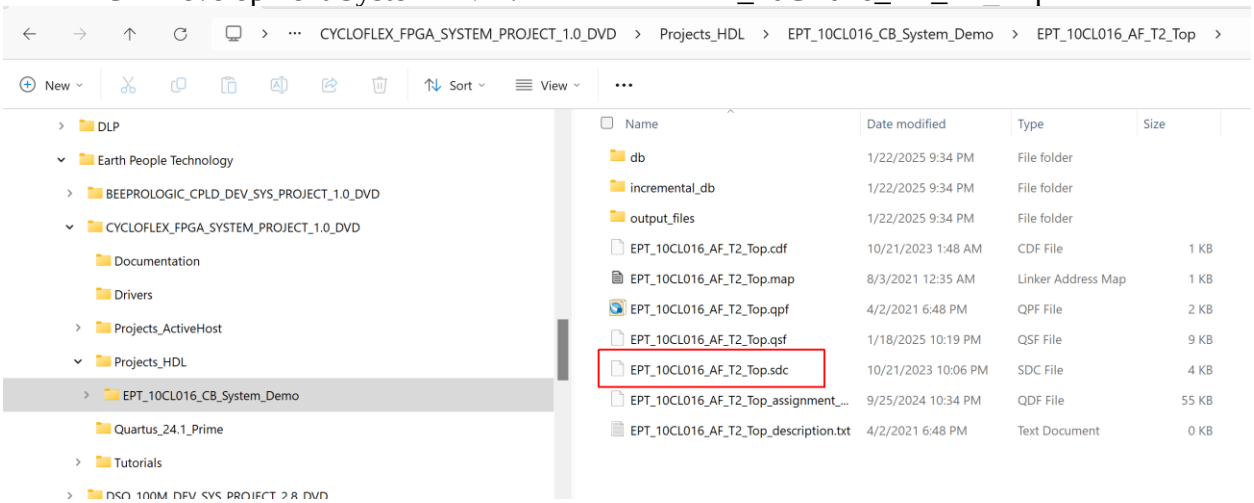
The pin locations should not need to be changed for EPT USB FPGA Development System. However, if you need to change any pin location, just click on the “location” column for the particular node you wish to change. Then, select the new pin location from the drop down box.



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

Quest Timing Analyzer Quick Start Guide

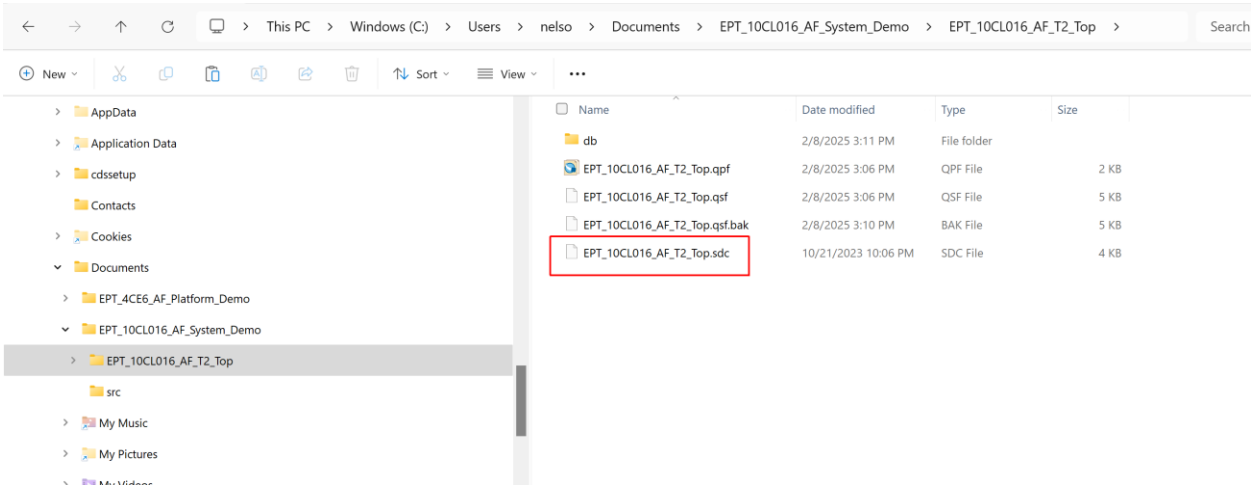
Browse to the \Projects_HDL\EPT_Platform_Demo \ EPT-10CL016-AF-T2_TOP folder of the EPT FPGA Development System DVD. Select the "EPT_10CL016_AF_T2_Top.sdc" file.



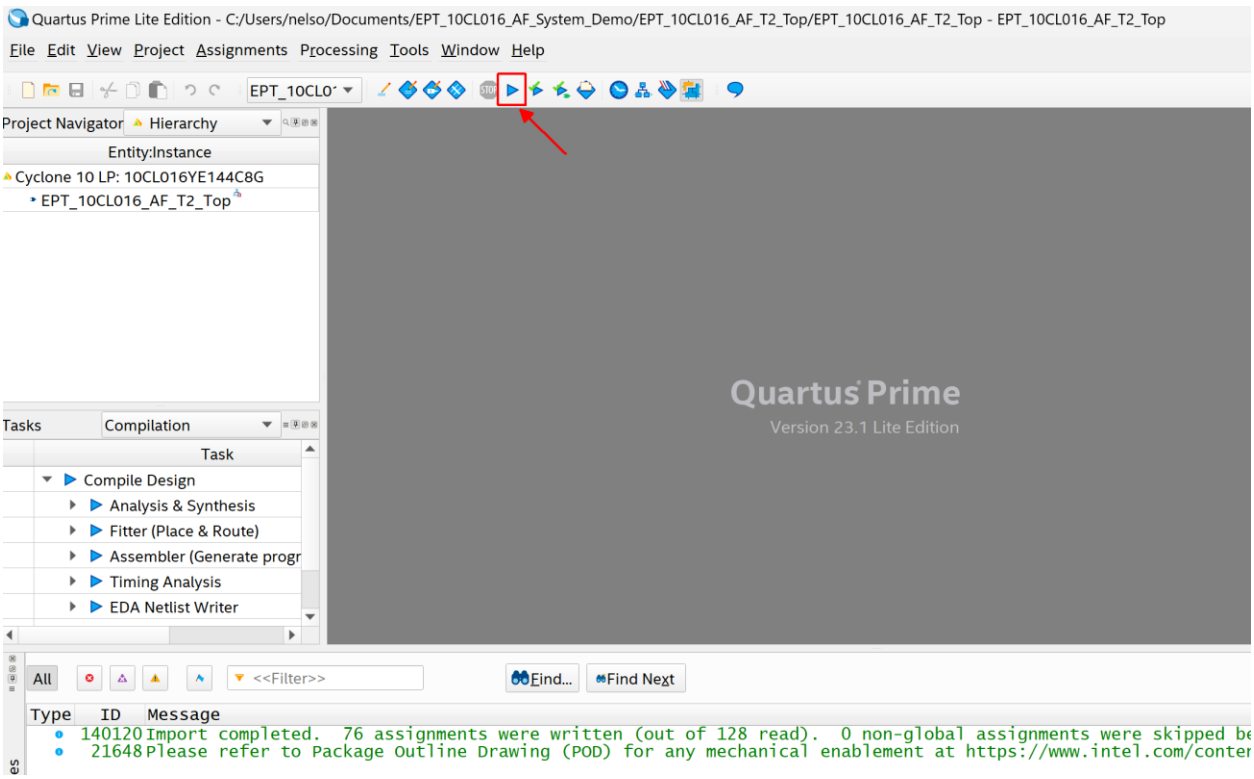
Copy the file and browse to

C:\Users\nelso\Documents\ EPT_10CL016_AF_T2_Platform_Demo \ EPT_10CL016_AF_T2_Top

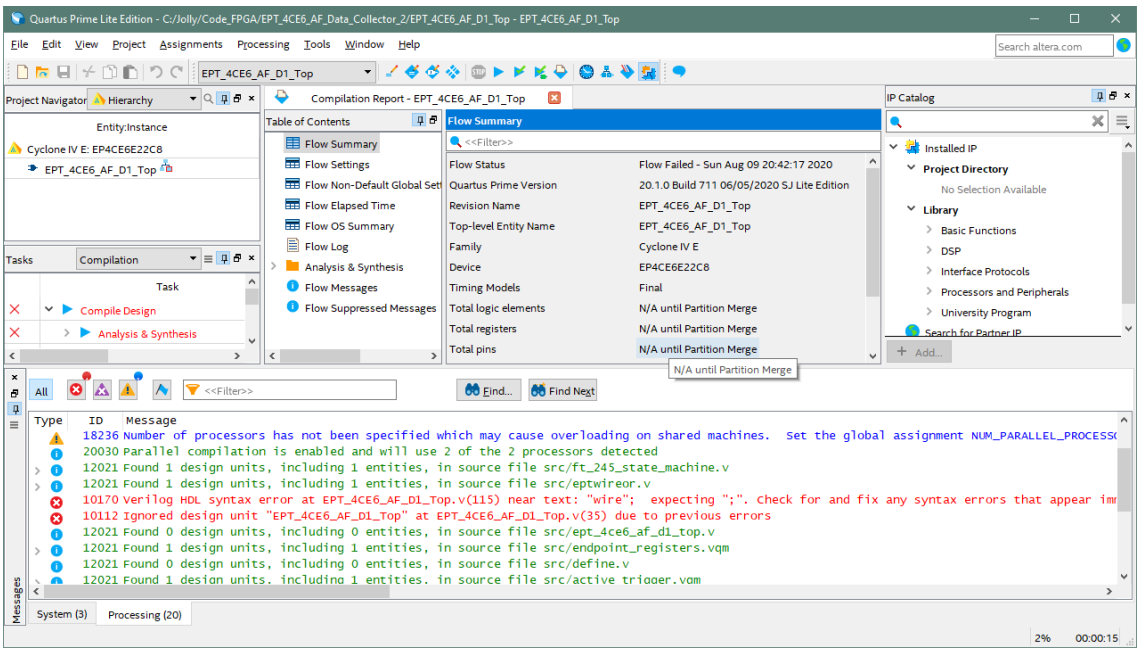
directory. Paste the file.



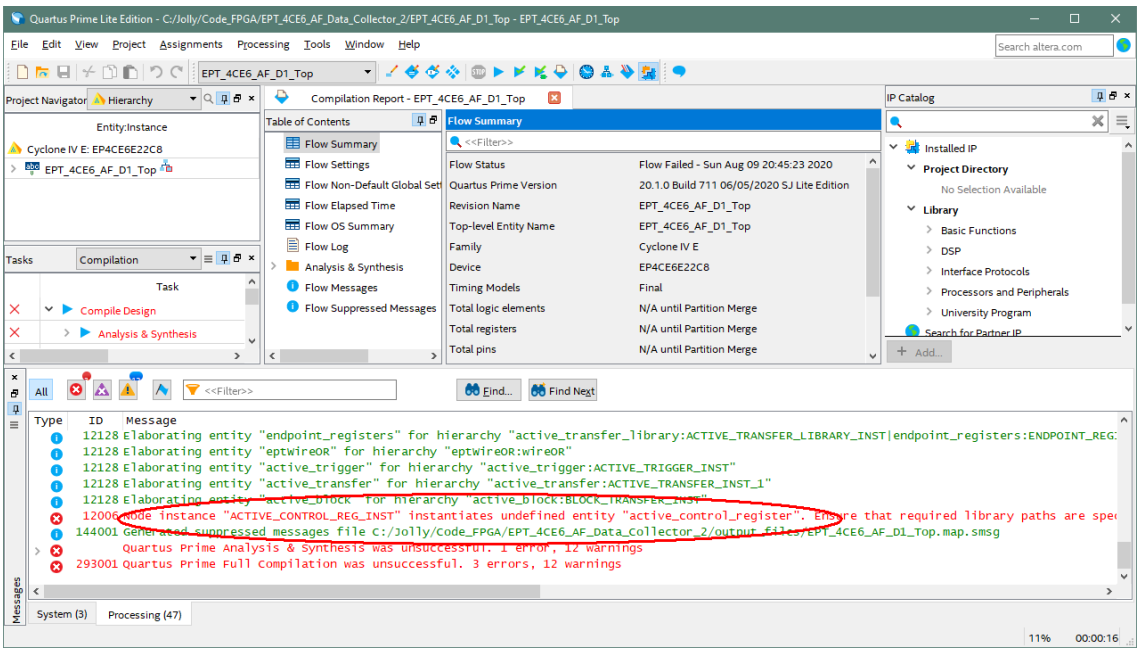
Select the Start Compilation button.



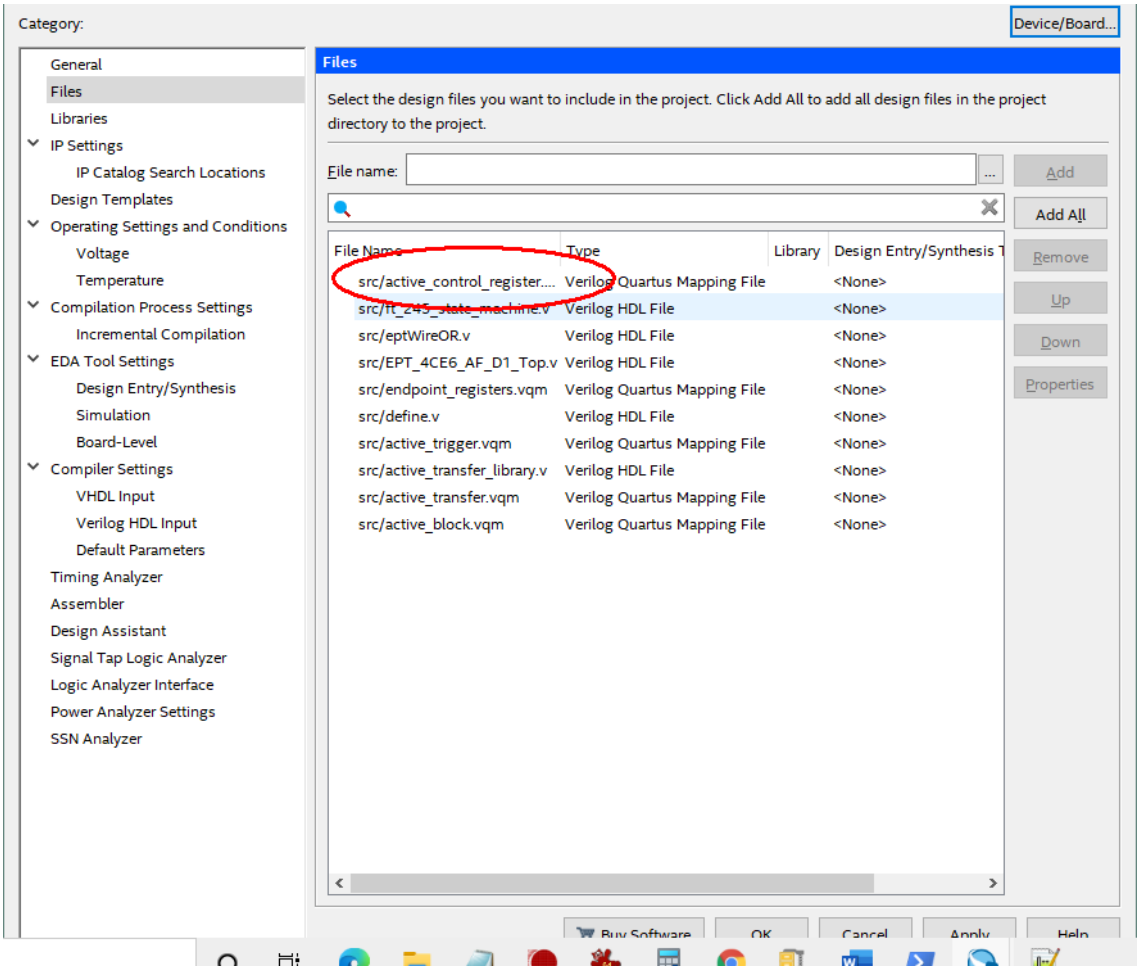
If you forget to include a file or some other error you should expect to see a screen similar to this:



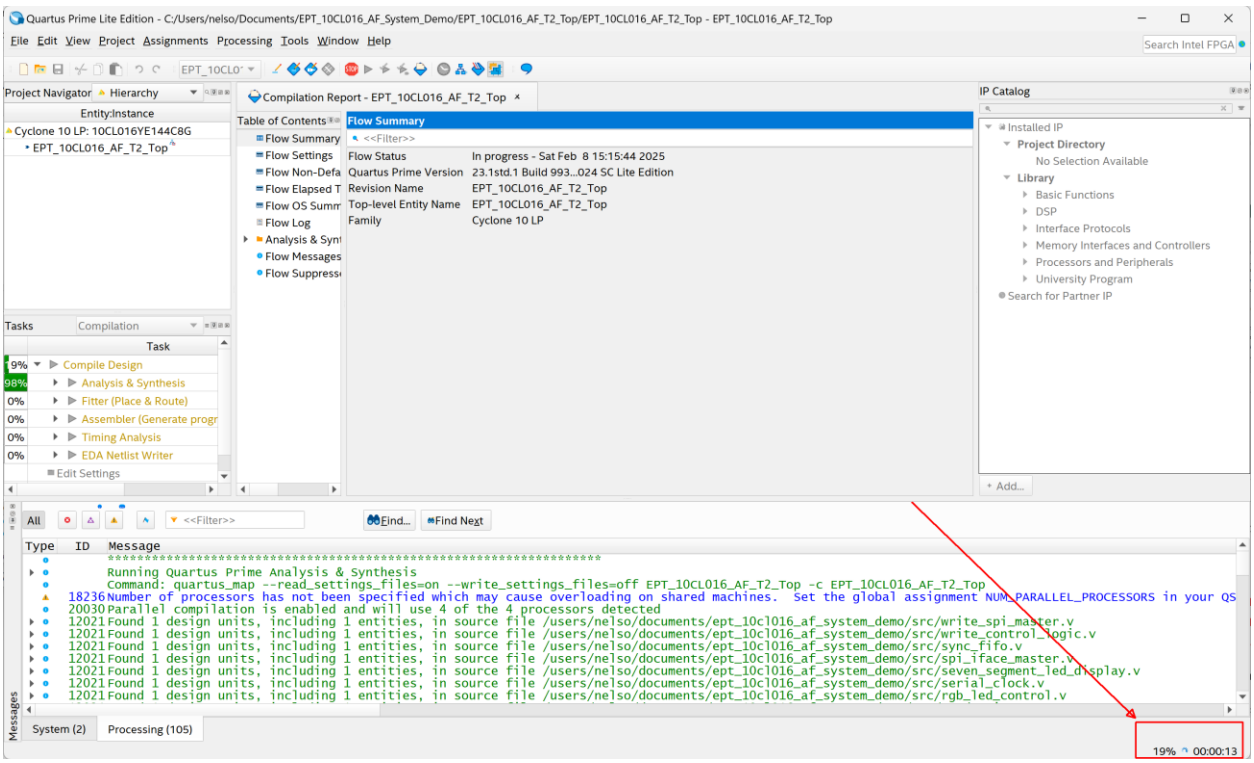
Click Ok, the select the “Error” tab to see the error.



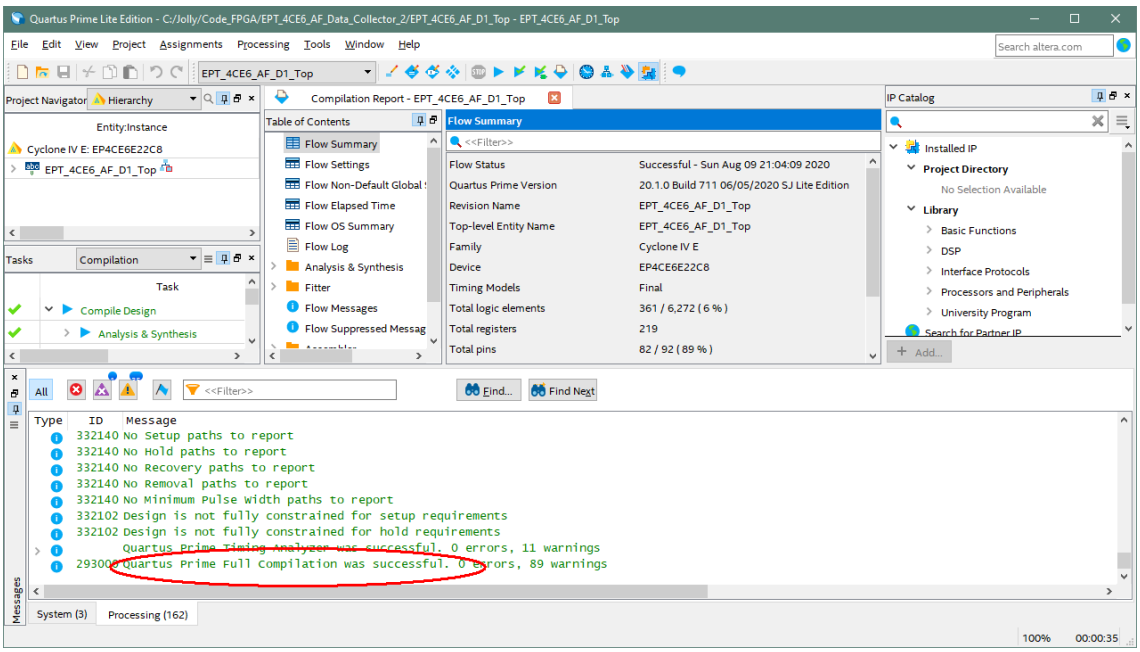
The error in this case is the missing file “active_control_register”. Click on the Assignment menu, then select Settings, then select Files. Add the “active_control_register.v” file from the database.



Click Ok then re-run the Compile process. Compilation progress is noted in bottom right corner.



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.

8 Configuring the FPGA

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

Configuring the Cyclone 10 FPGA on the CycloFlex is the process of programming the compiled/synthesized user project into the FPGA. Once the FPGA is configured with the user project, the CycloFlex board is usable with the code. There are two methods to configure the Cyclone 10 FPGA

- Direct JTAG access
- On Board Configuration Flash chip

Direct JTAG access allows the Quartus Prime Lite software to program the object file directly into the FPGA using JTAG. This will configure the FPGA with the user project using volatile memory. If the power is removed from the Cyclone 10, the chip will lose its configuration.

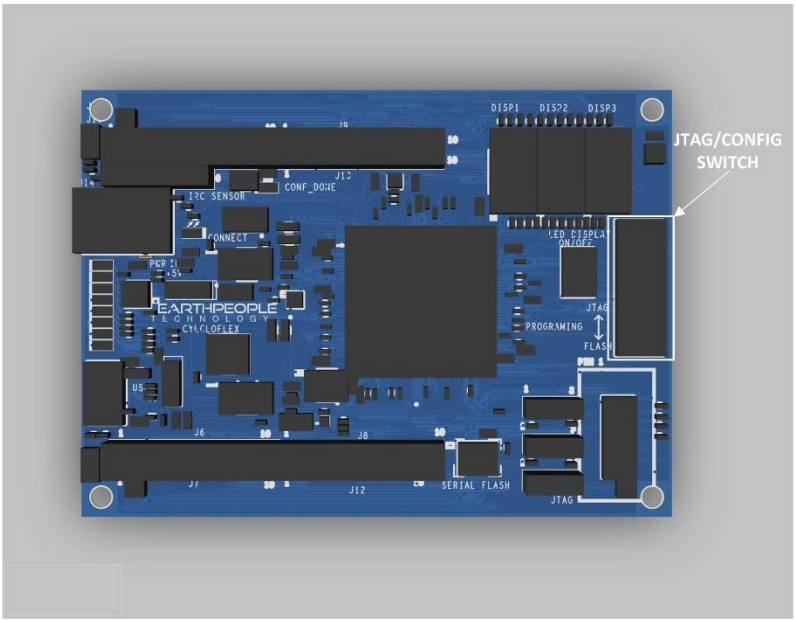
On Board Configuration Flash chip programming allows the Quartus Prime Lite software to write to the CycloFlex Configuration Flash chip. The users compiled/synthesized project is loaded into the on board flash. Once the flash chip loading is complete, the CycloFlex will reset the FPGA. During the reset, the Cyclone 10 will load the contents of the Configuration Flash chip into the FPGA. If the FPGA is power cycled (Off/On), the chip will perform the loading from the flash chip.

The next two sections describe how to implement the JTAG direct and On Board Configuration Flash programming of the CycloFlex.

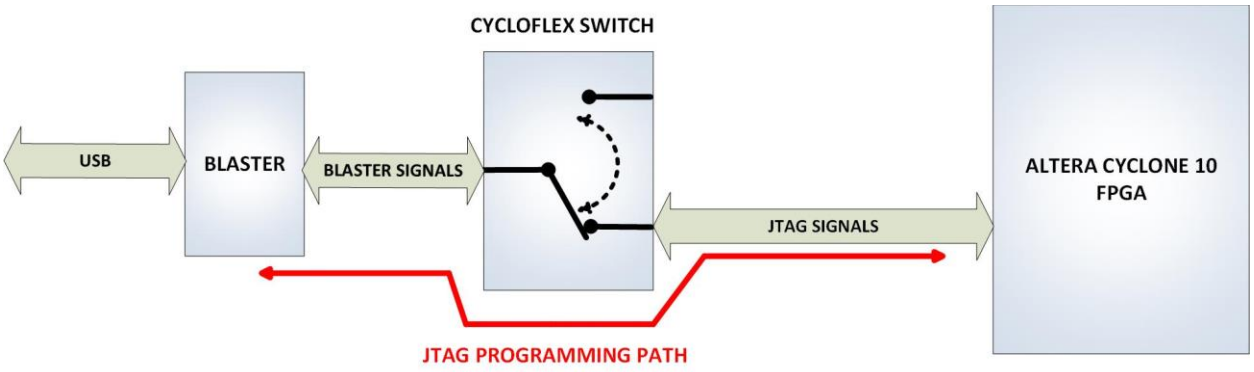
8.1 JTAG Direct Programming of the FPGA

Connect the CycloFlex to the PC via the USB-C connection, Connect a JTAG Blaster compatible programmer to the PC, connect the 10 pin cable from the Blaster to the JTAG connector of the CycloFlex board, open up Quartus Prime, open the programmer tool.

The CycloFlex includes programming switch that capable of six Dual Pole Single Throw connections. The JTAG/CONFIG SWITCH.



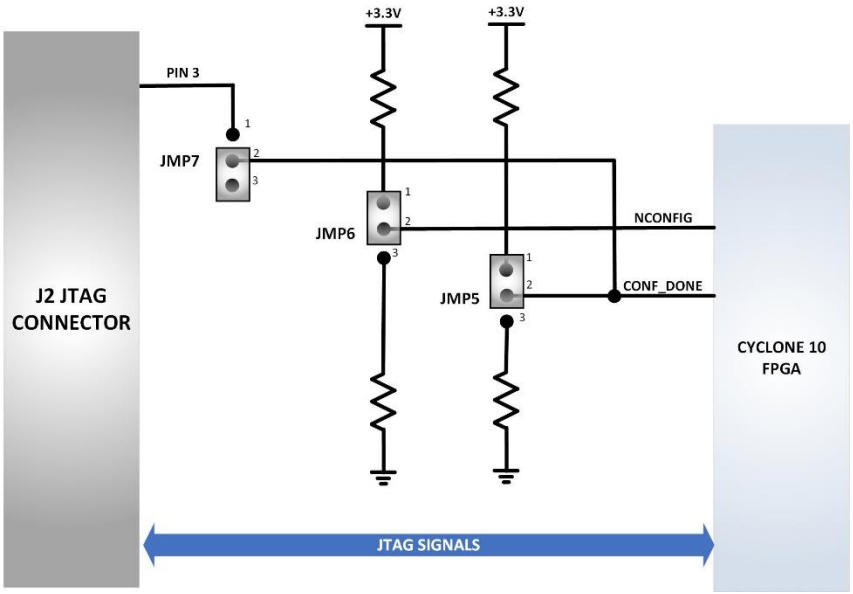
This switch is designed to allow the external Blaster Programmer to either program the Cyclone 10 chip directly using JTAG or program the Config Flash chip. Changing the position of the switch will allow the same Blaster to be used for either JTAG programming or Config Flash programming.



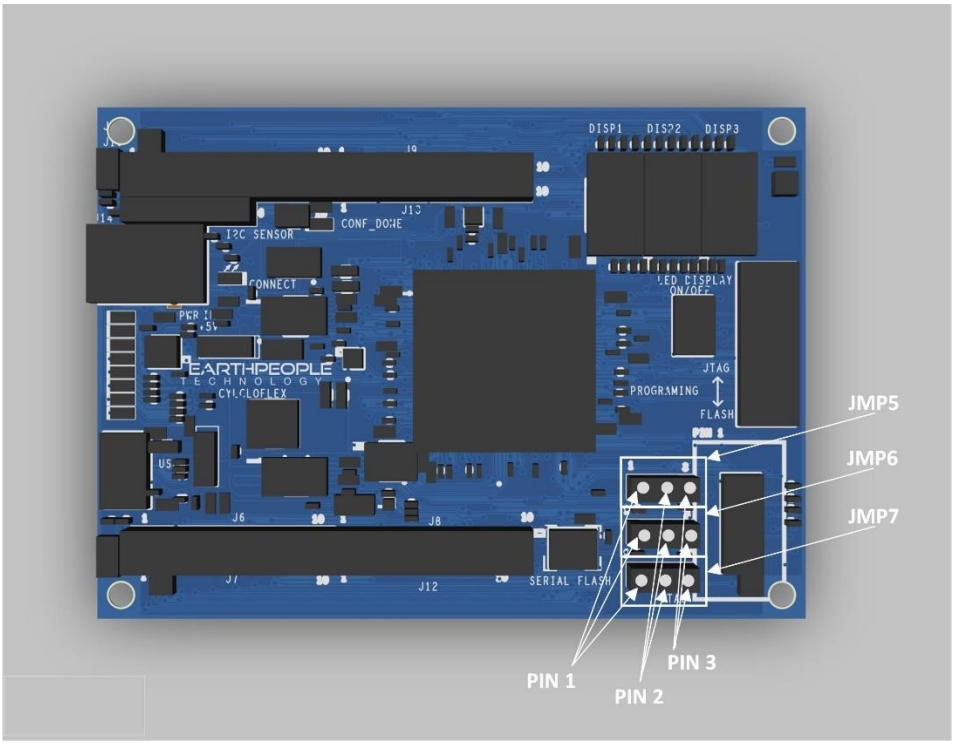
The JTAG Programming Path must have certain configuration pins of the Cyclone 10 in fixed positions (pulled up or pulled down) during programming. The following jumpers must be installed for correct programming of the Cyclone 10 chip using JTAG:

- JMP7
- JMP6
- JMP5

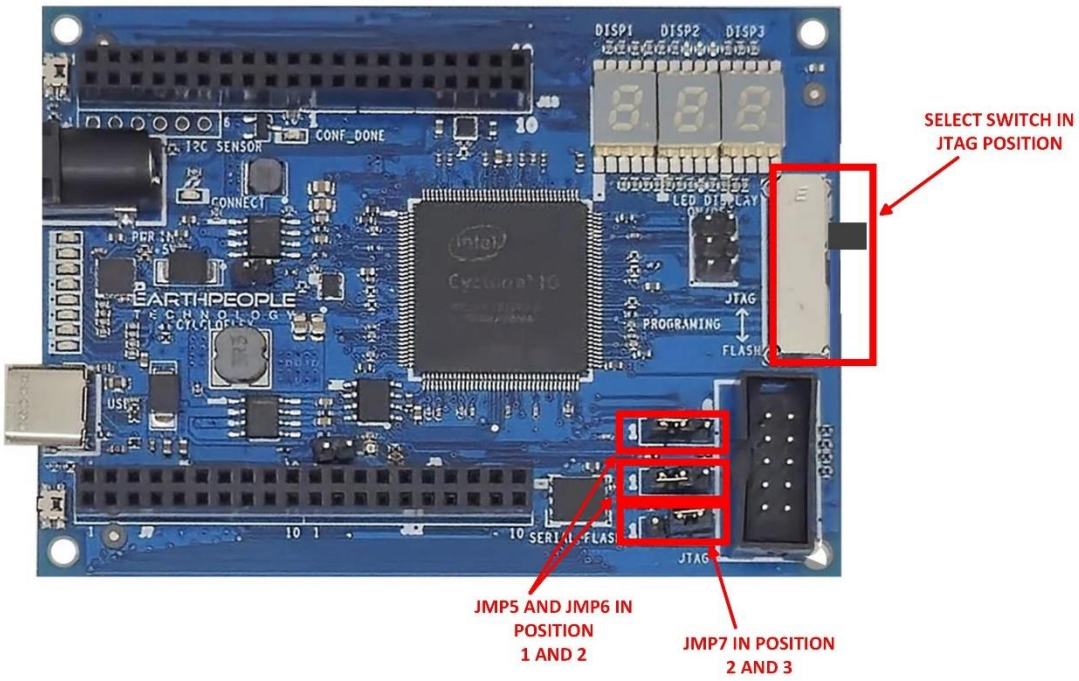
A jumper is a two pin socket used to short two header pins together.



The jumper locations are:

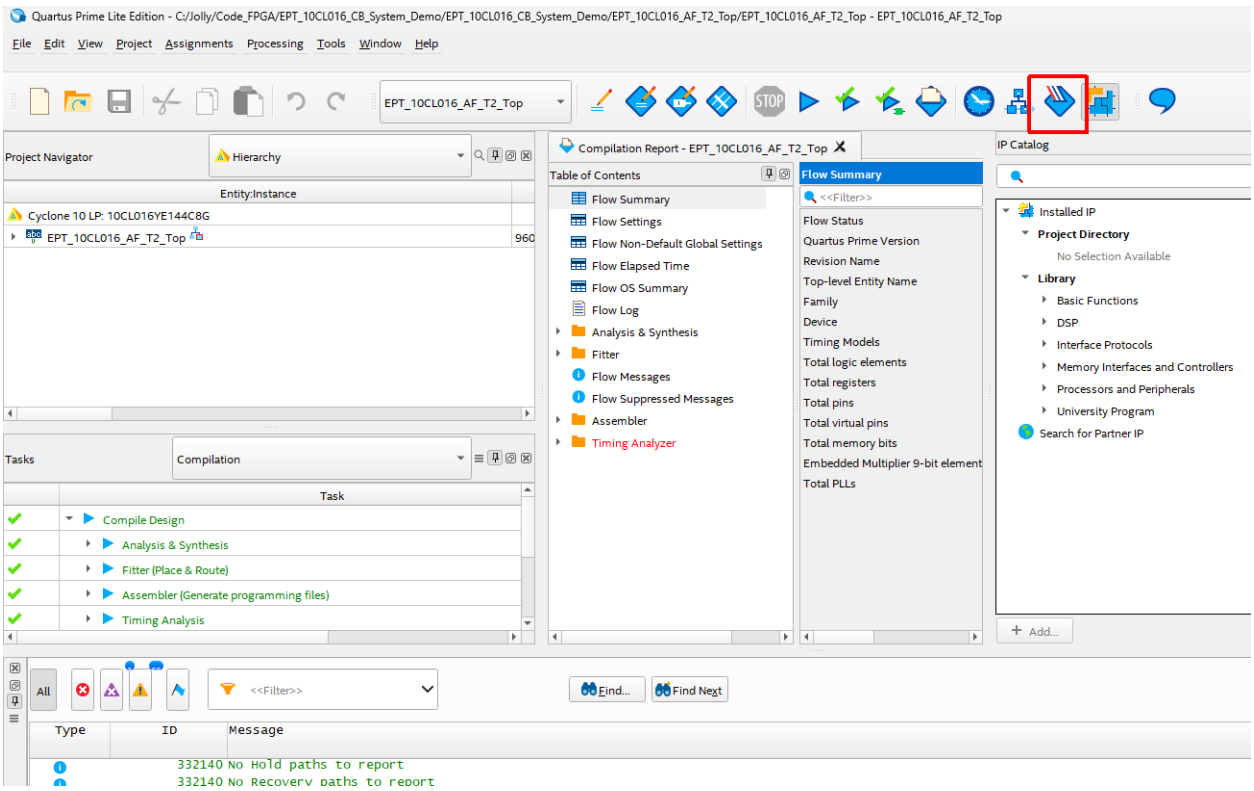


The correct settings for JTAG Path Programming:

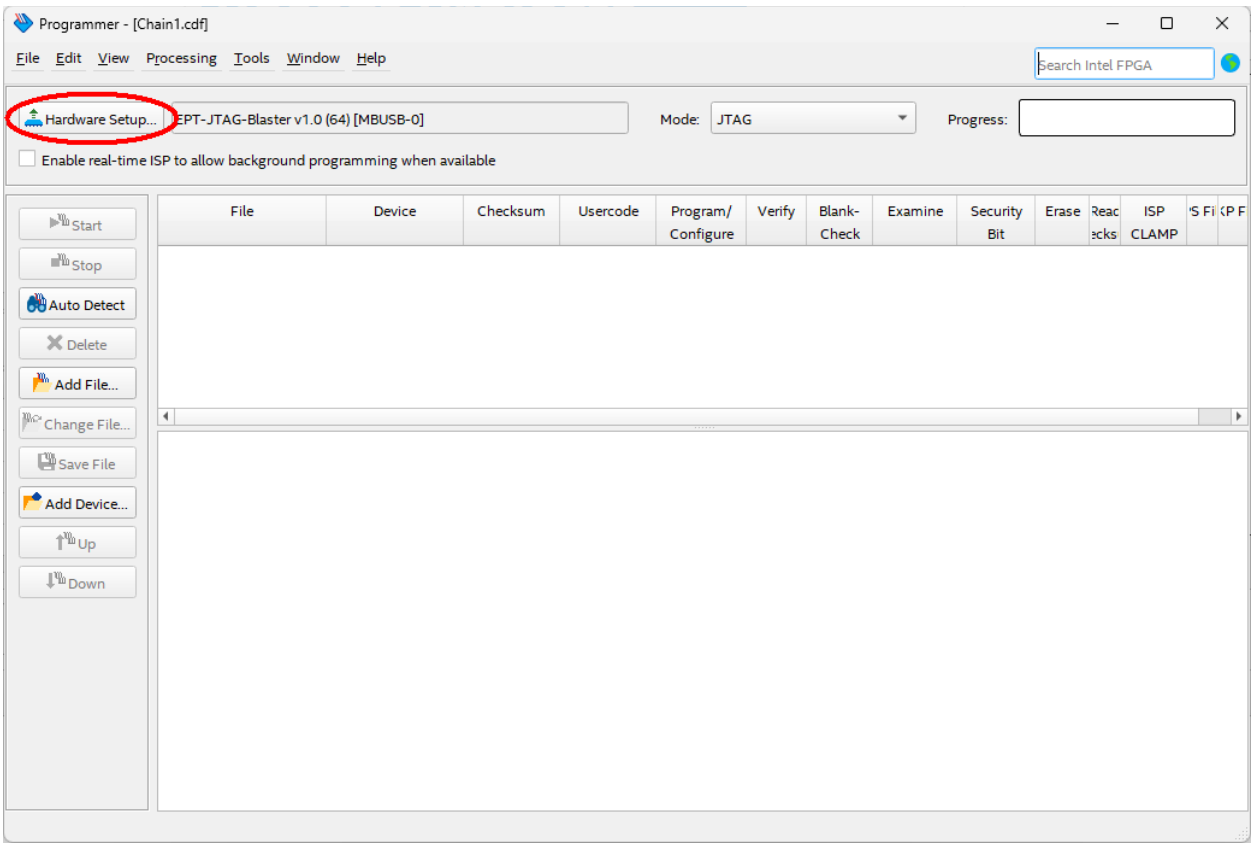


Once the jumpers and JTAG/Flash switch have been set, refer the section “Programming the CycloFlex” for instructions on how to use the Quartus Prime Lite software to program the Cyclone 10 chip.

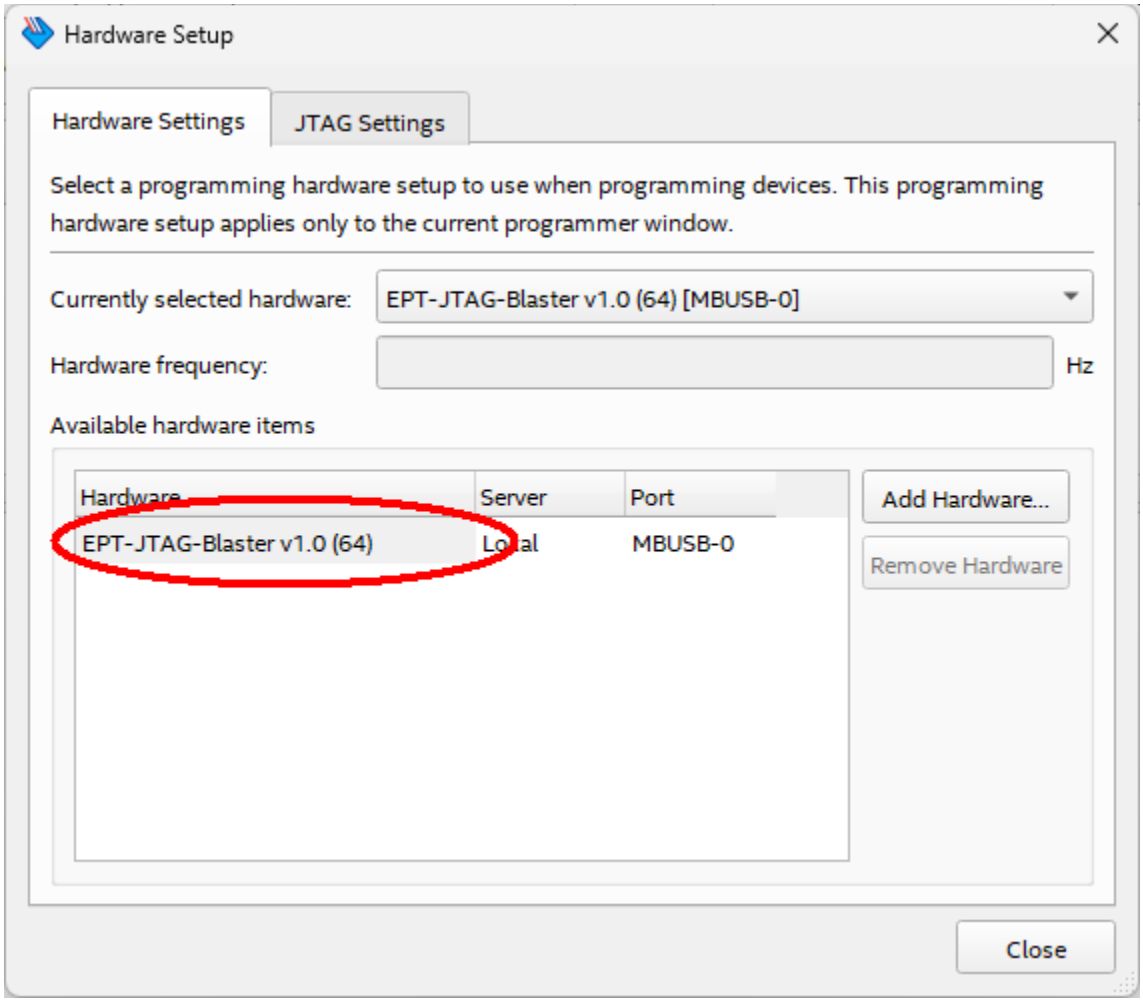
Click on the Programmer button.



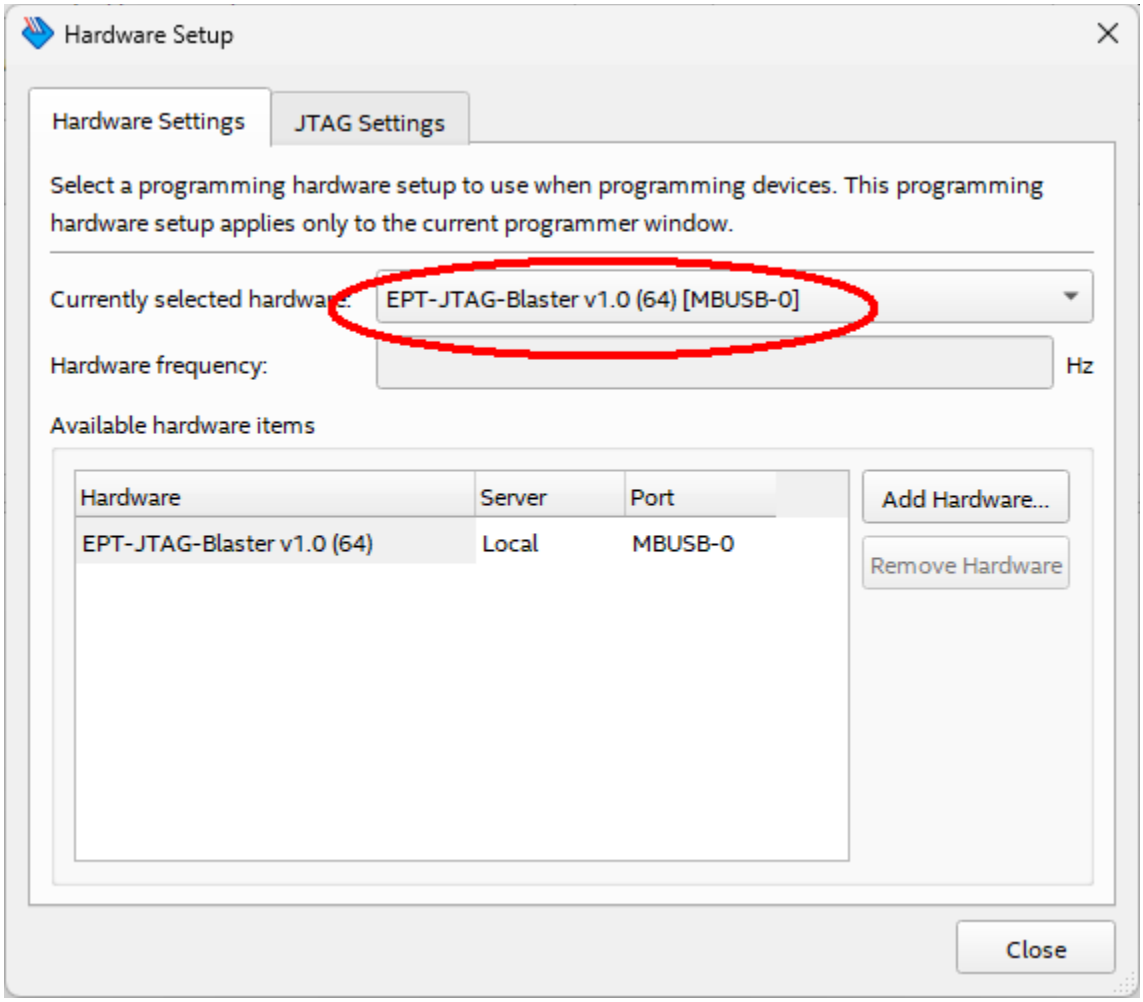
The Programmer Window will open up with the programming file selected. Click on the Hardware Setup button in the upper left corner.



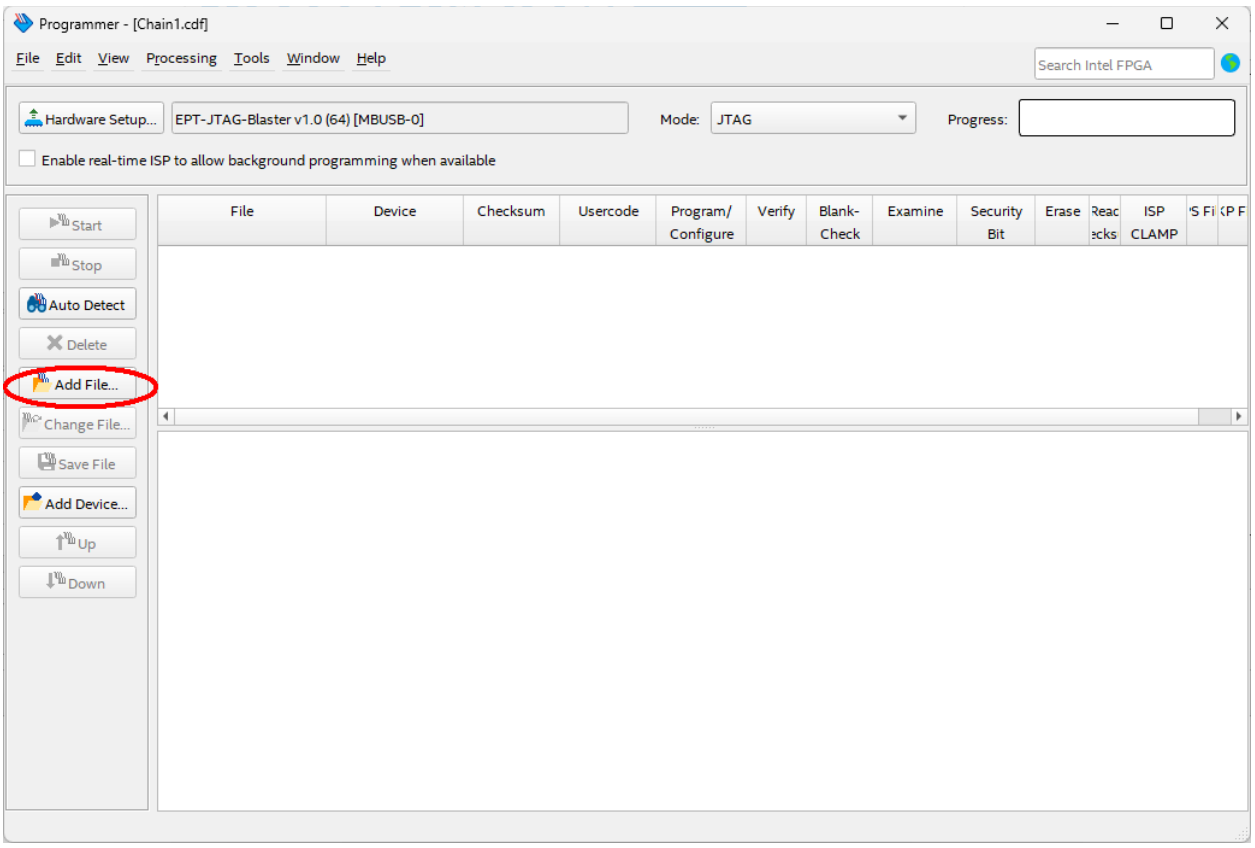
The Hardware Setup Window will open. In the “Available hardware items”, double click on “EPT-JTAG-Blaster v1.0b (64)”.

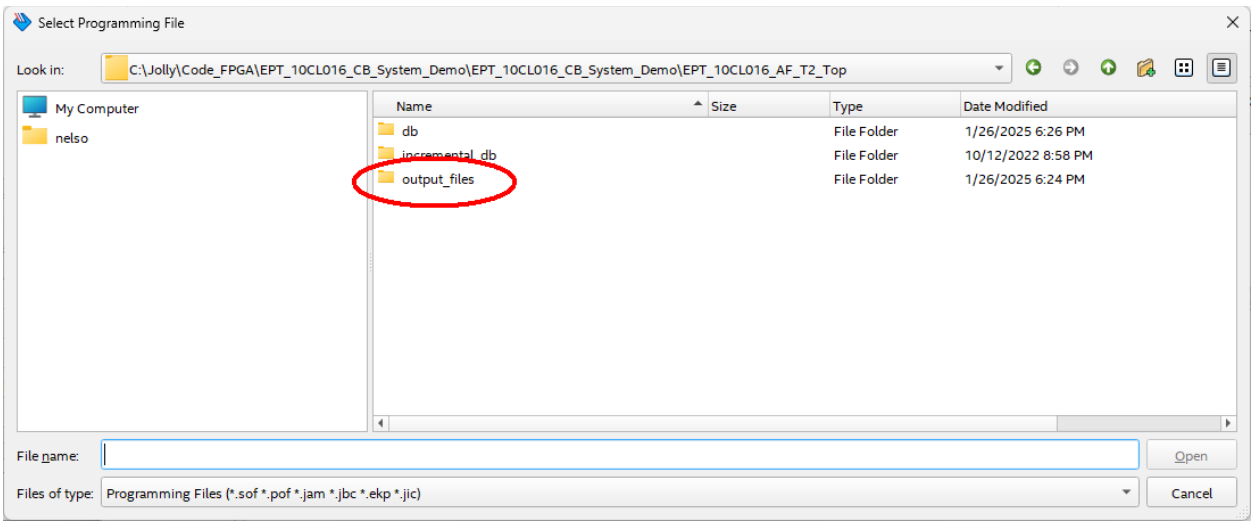


If you successfully double clicked, the “Currently selected hardware:” dropdown box will show the “EPT-JTAG-Blaster v1.0b (64)”.

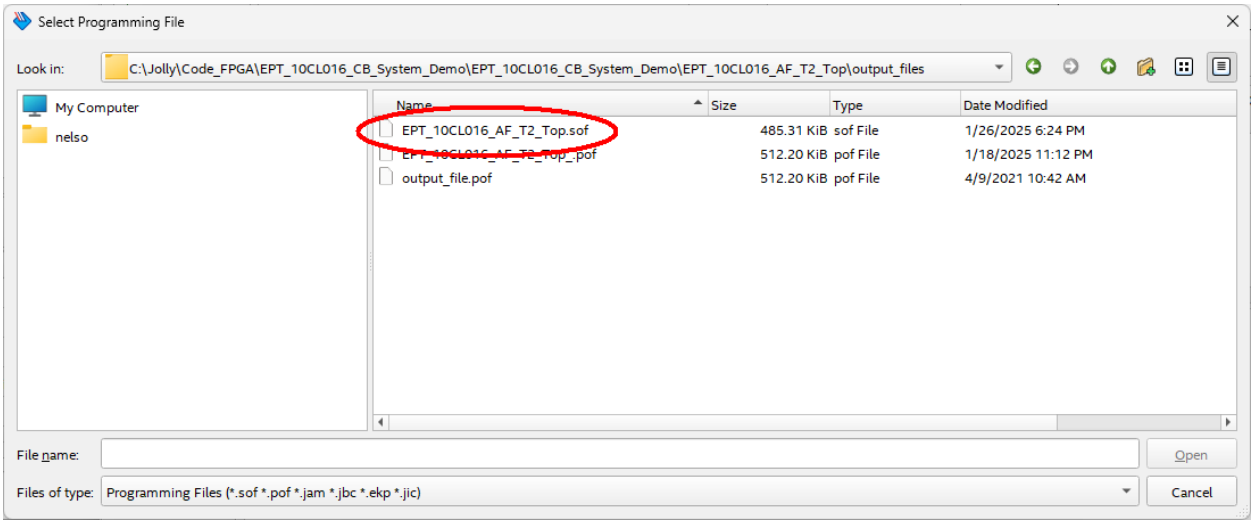


Click on the “Add File” button

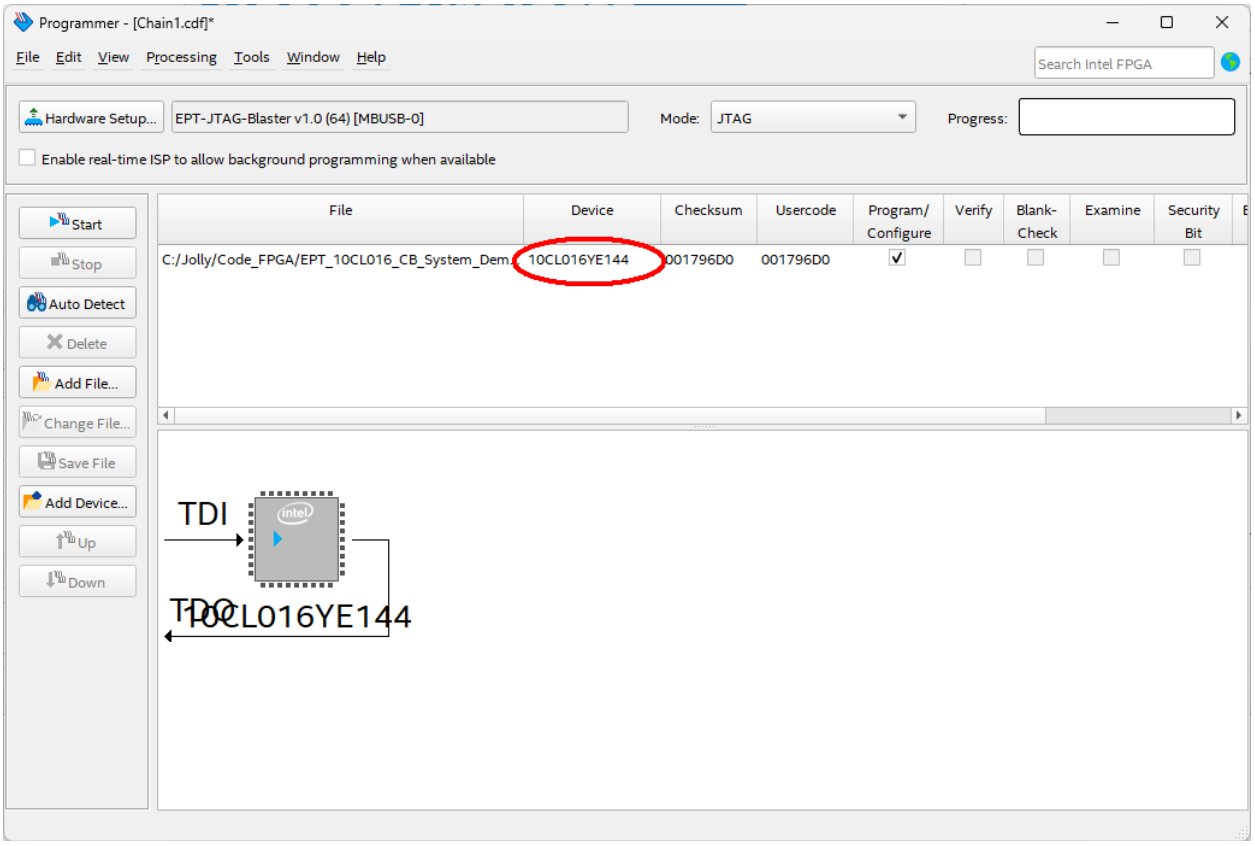

Programmer - [Chain1.cdf]
File Edit View Processing Tools Window Help
Search Intel FPGA
Hardware Setup...
EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]
Mode: JTAG
Progress:
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add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



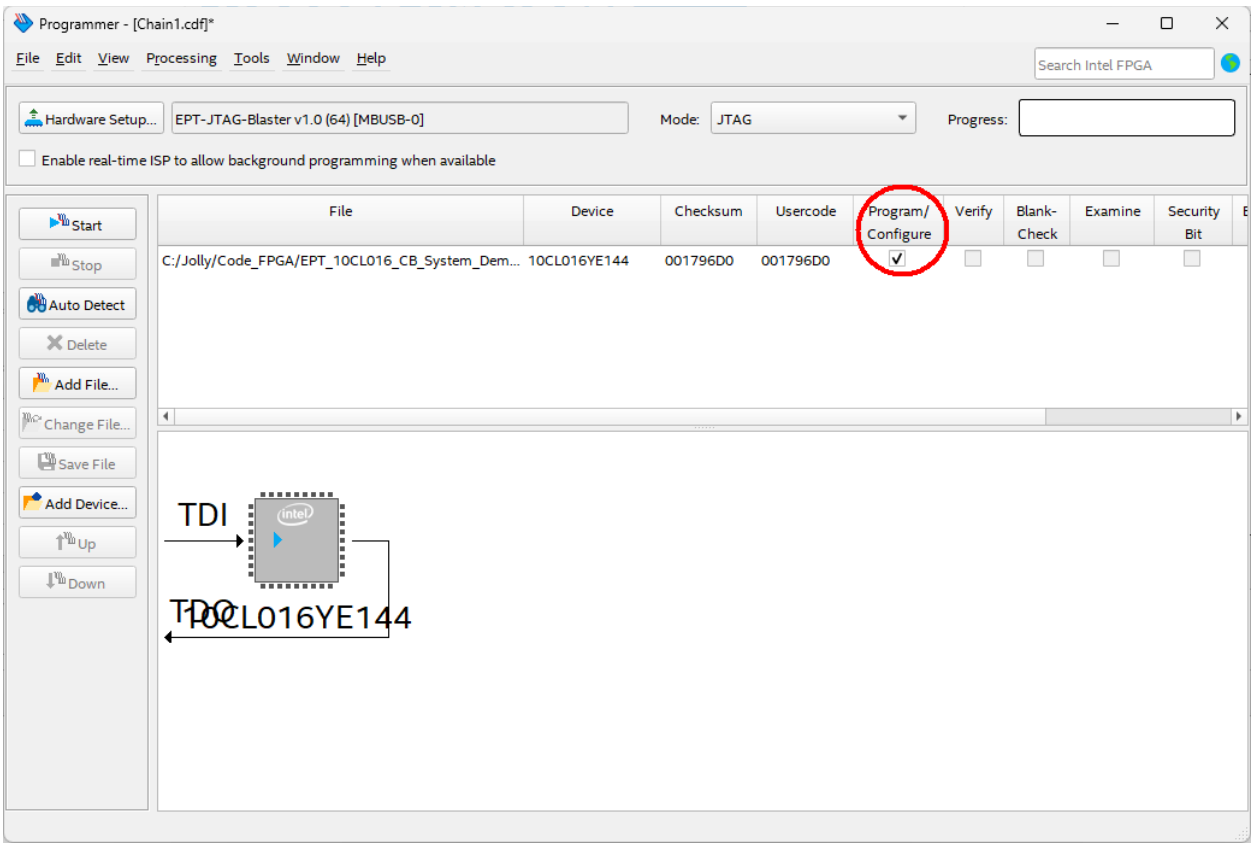
At the Browse window, double click on the output files folder.



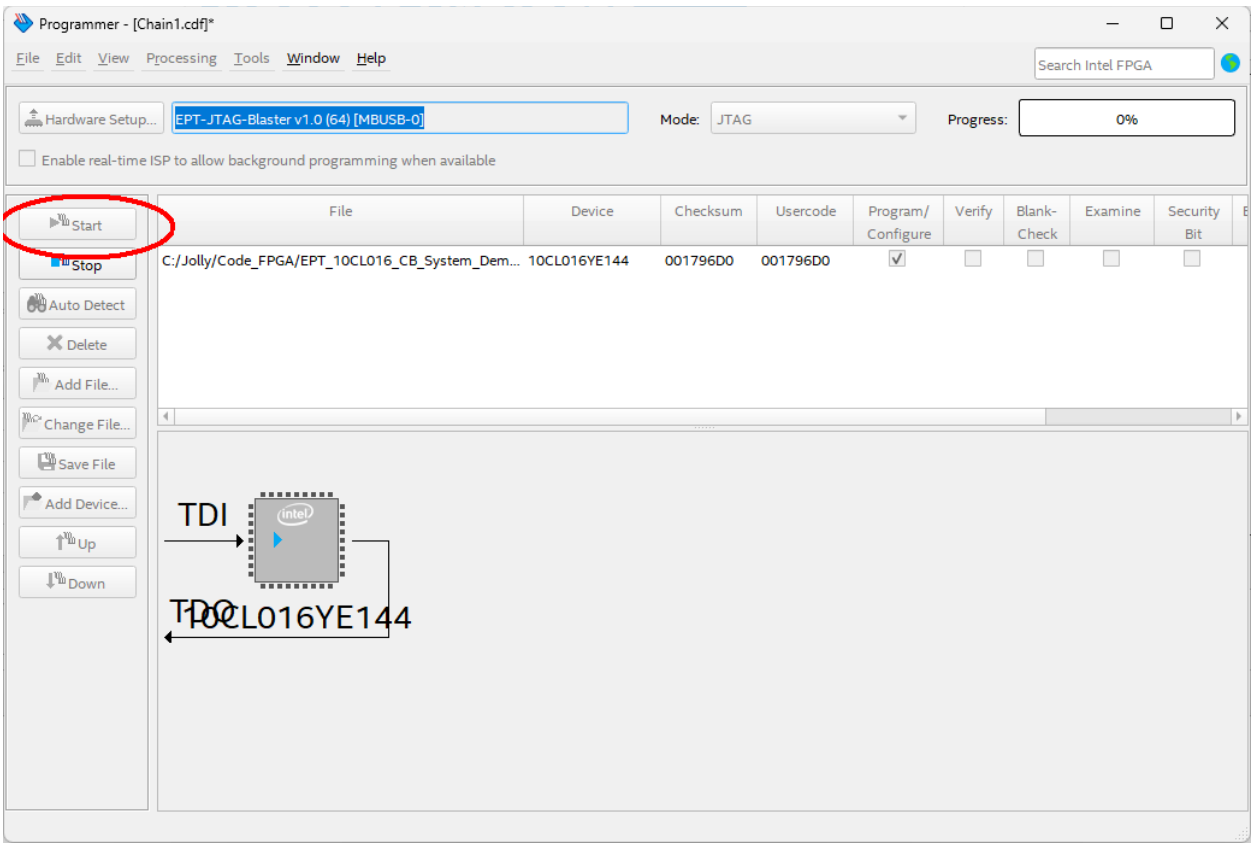
Double click on the “EPT_10CL016_AF_T2_Top.sof” file. Click the Open button in the lower right corner.



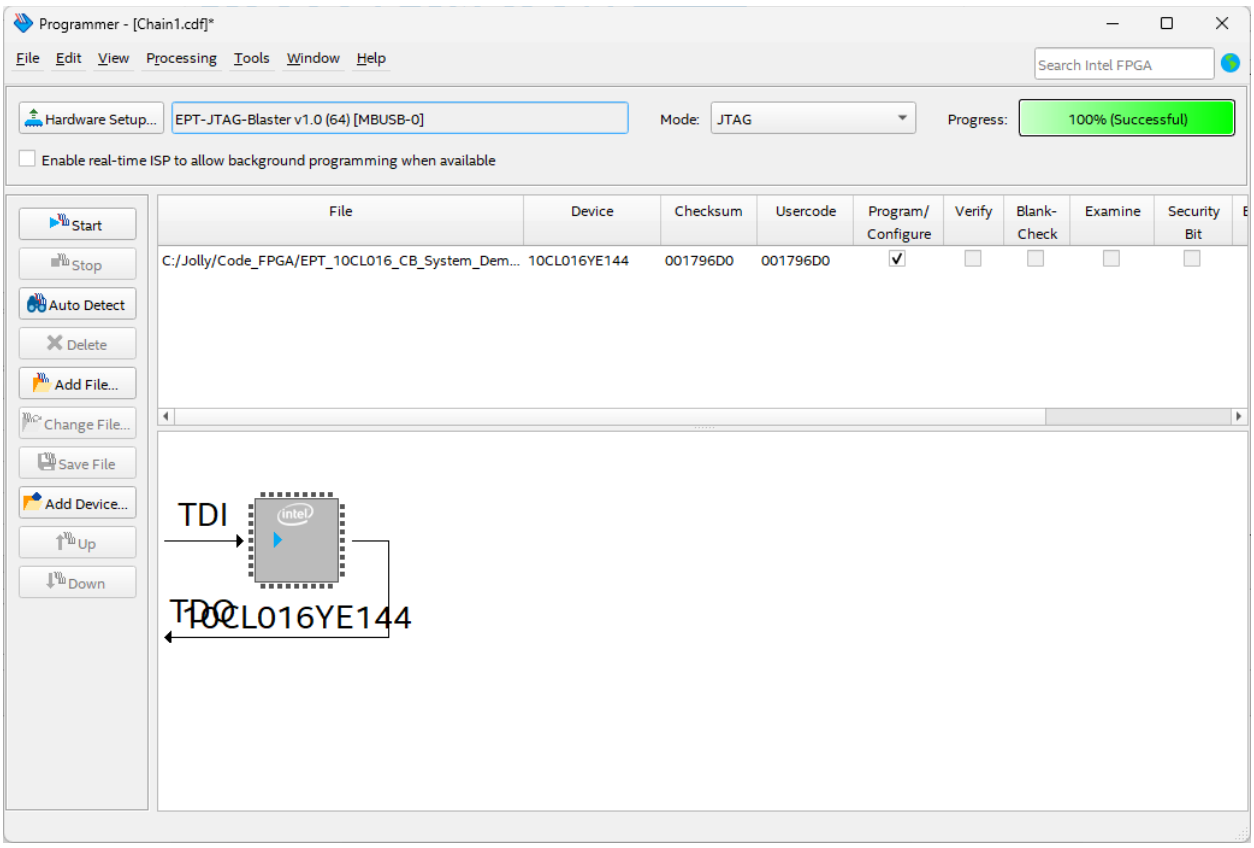
Next, selet the checkbox under the “Program/Configure” of the Programmer Tool.



Click on the Start button to to start programming the FPGA. The Progress bar will indicate the progress of programming.



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



If the Programming window indicates that the “Progress:” is 100% (Successful), the FPGA has been programmed directly though JTAG. The CycloFlex is now ready for use or debugging code.

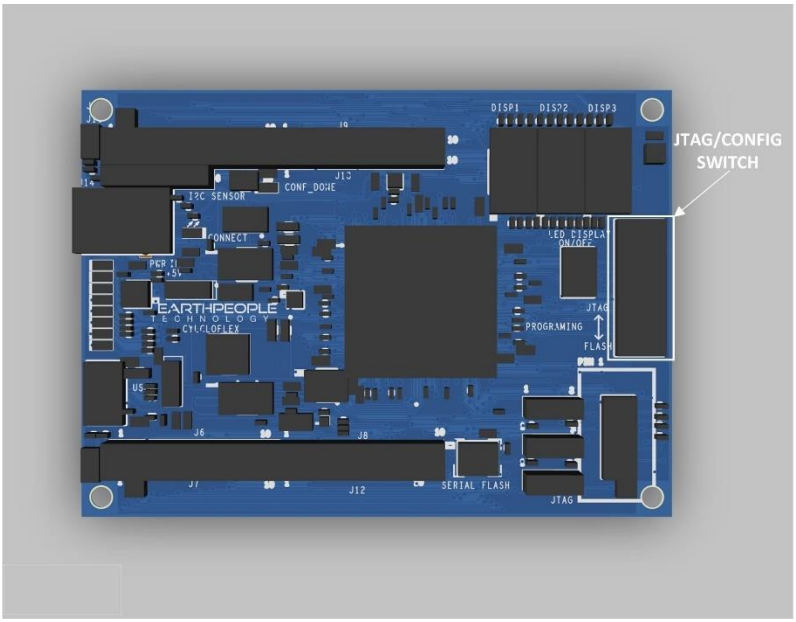
If the power is cycled (Off/On), the FPGA will lose the *.sof file and the FPGA becomes a blank chip. Repeat the previous steps to program the user project into the FPGA directly with the JTAG.

8.2 Configuration Flash Programming of the FPGA

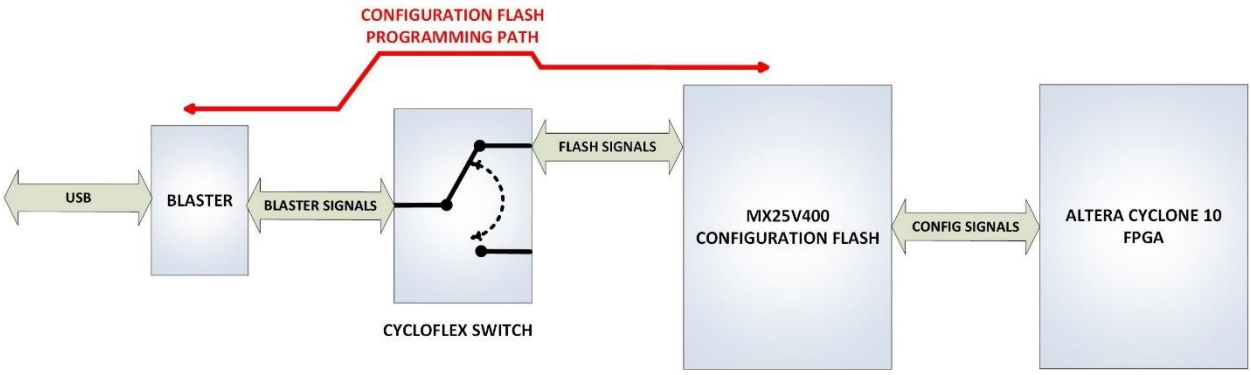
The Cyclone 10 FPGA requires configuration before it can be used to perform the functions of the user code. The CycloFlex board includes a 4Mbit Configuration Flash chip that is designed to store the user code. Then, when the power is cycled (On/Off), the FPGA will read the contents of the Configuration Flash and set up the functionality of the user code within the chip. The user will program a *.pof file into the flash chip using Quartus.

The board requires power from either Barrel Connector or USB-C Connector. Then connect the Blaster.

The CycloFlex includes a programming switch that is capable of six Dual Pole Single Throw connections. The JTAG/CONFIG SWITCH.



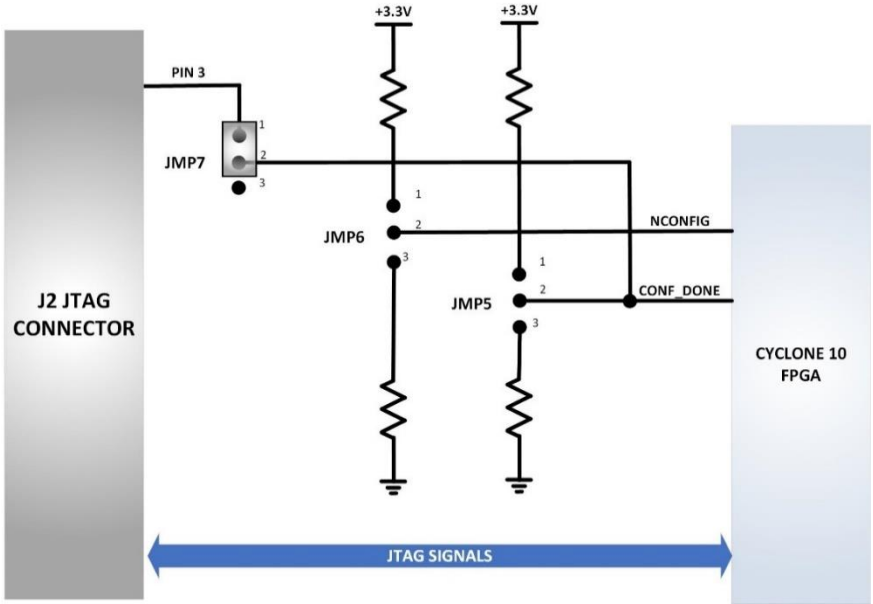
This switch is designed to allow the external Blaster Programmer to either program the Cyclone 10 chip directly using JTAG or program the Config Flash chip. Changing the position of the switch will allow the same Blaster to be used for either JTAG programming or Config Flash programming.



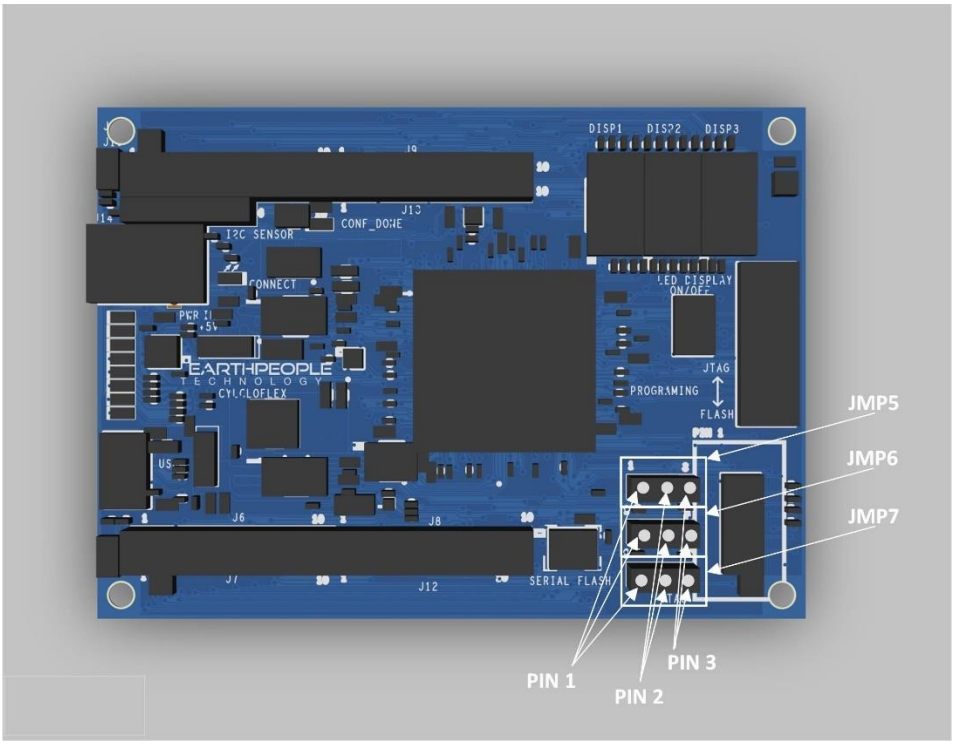
The JTAG Programming Path must have certain configuration pins of the Cyclone 10 in fixed positions (pulled up or pulled down) during programming. The following jumpers must be installed for correct programming of the Configuration Flash:

- JMP7
- JMP6
- JMP5

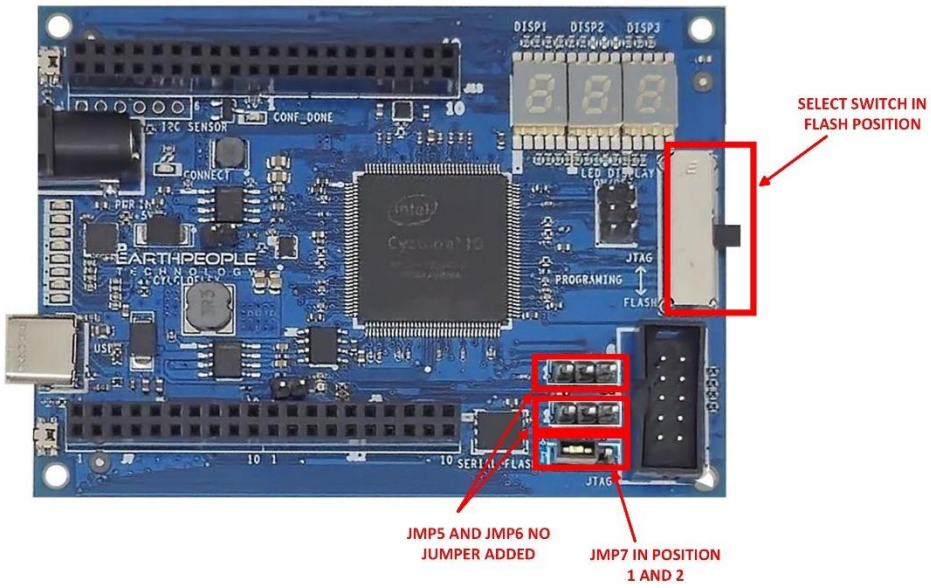
A jumper is a two pin socket used to short two header pins together.



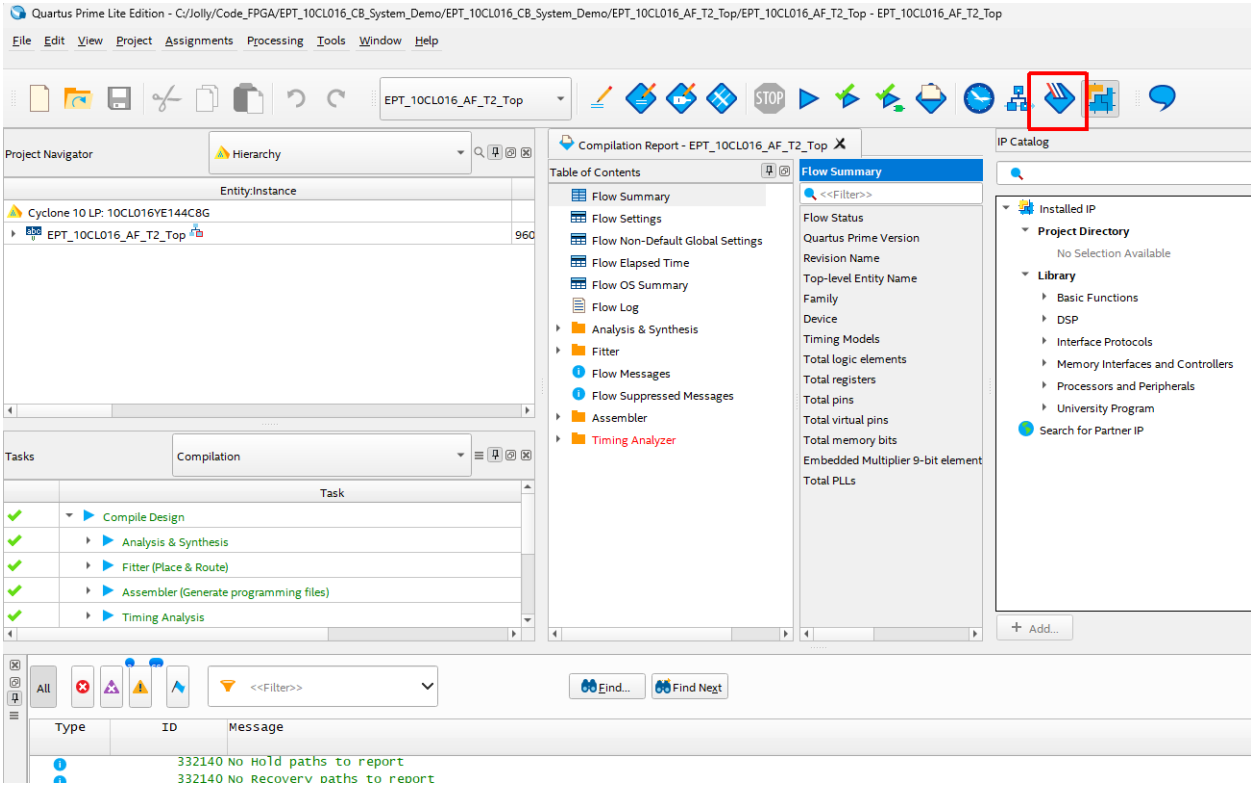
The jumper locations are:



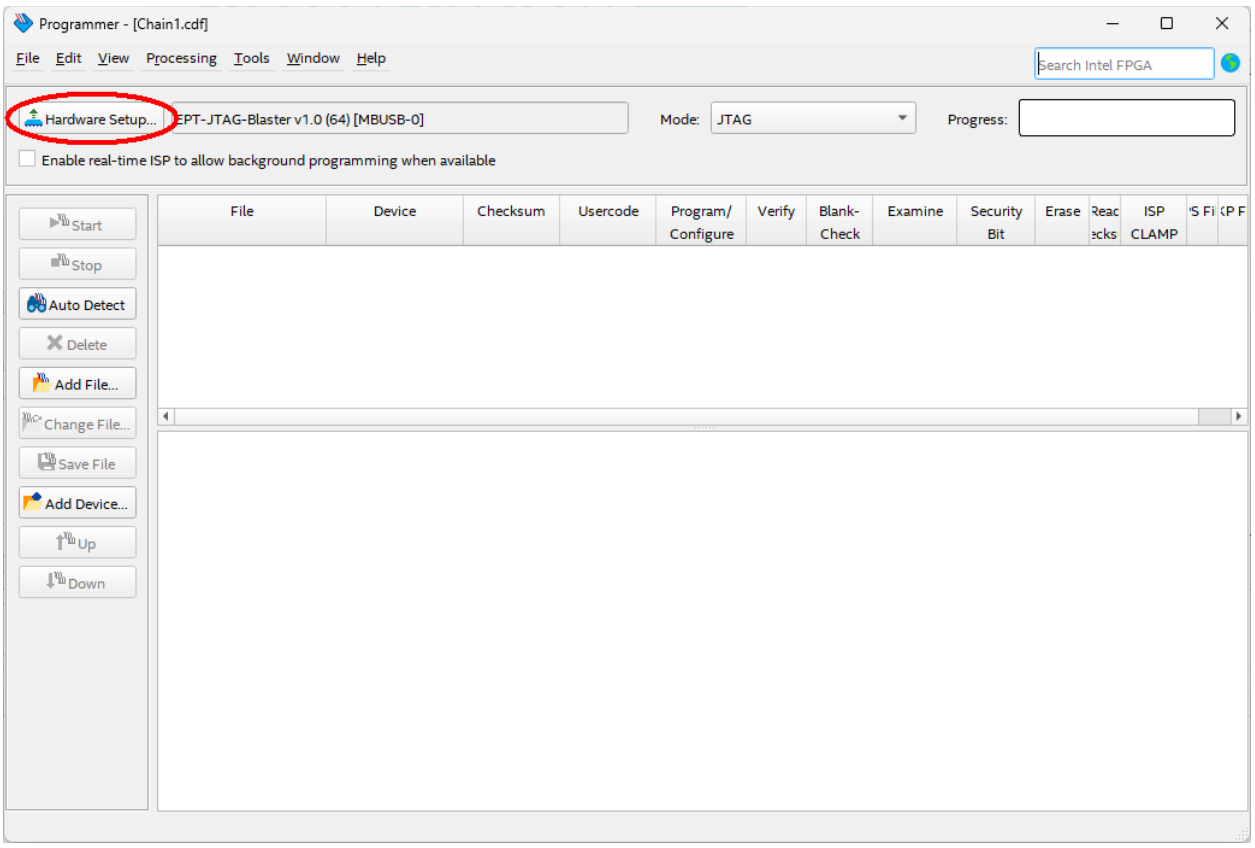
The CycloFlex should be configure with the following:



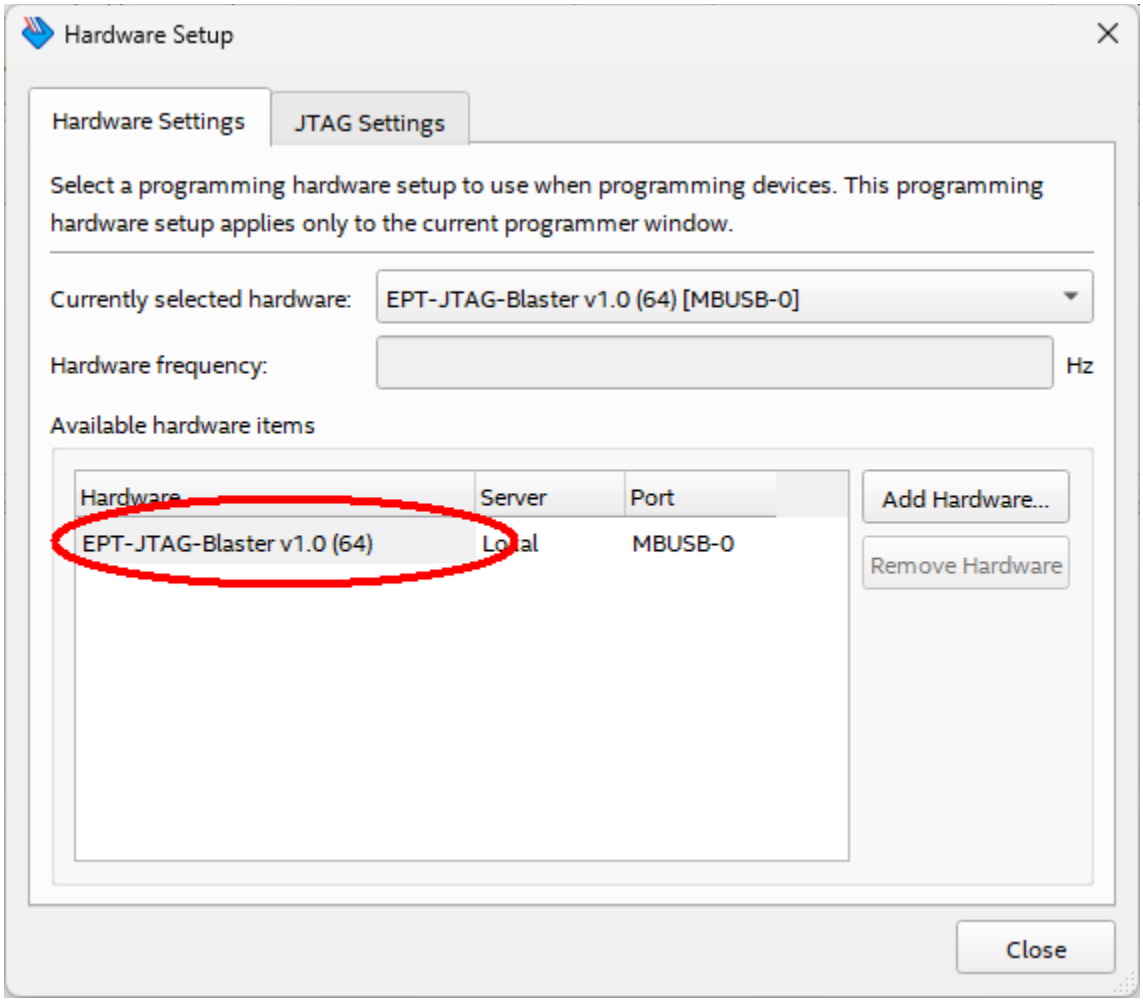
Once the jumpers and JTAG/Flash switch have been set, refer the section “Programming the CycloFlex” for instructions on how to use the Quartus Prime Lite software to program the Cyclone 10 chip.
Click on the Programmer button.



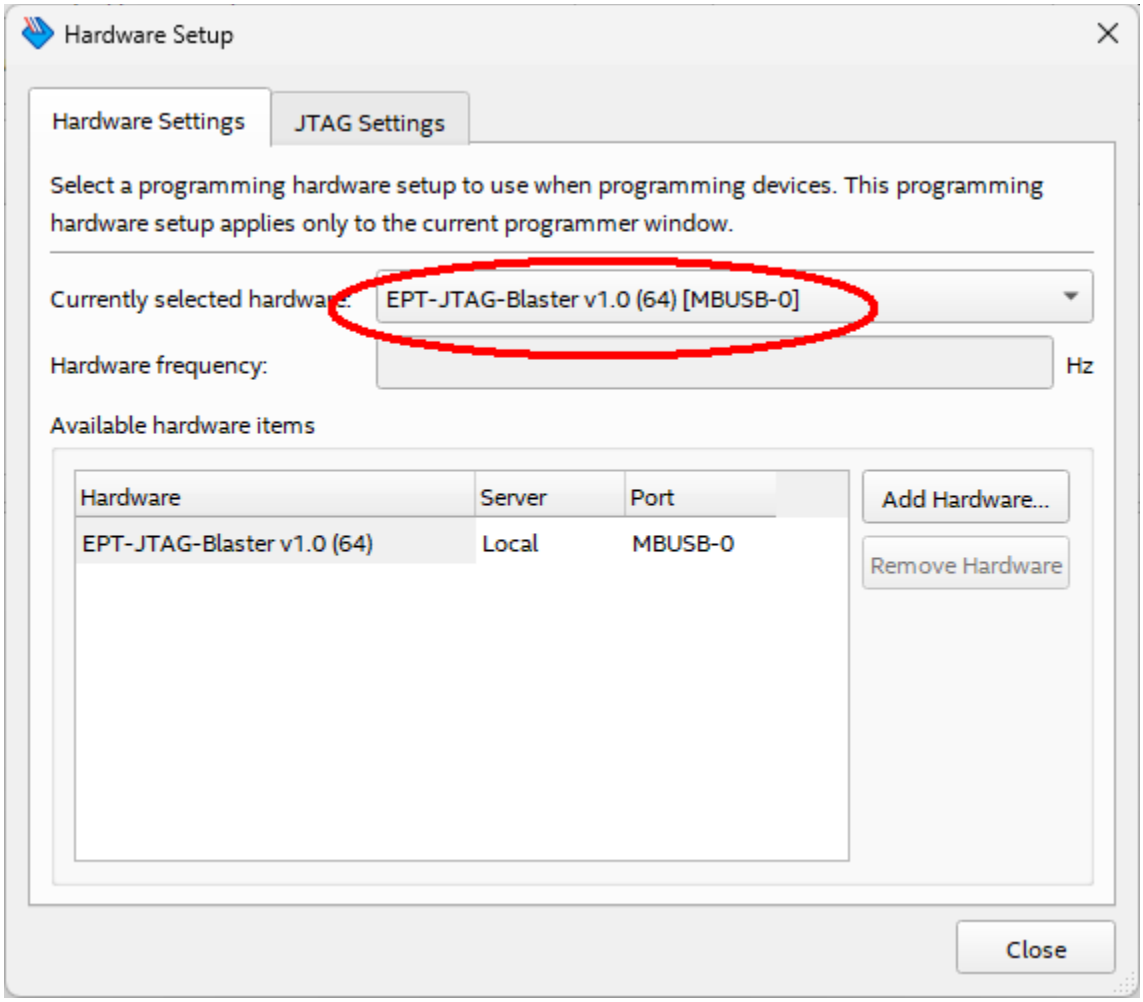
The Programmer Window will open up with the programming file selected. Click on the Hardware Setup button in the upper left corner.



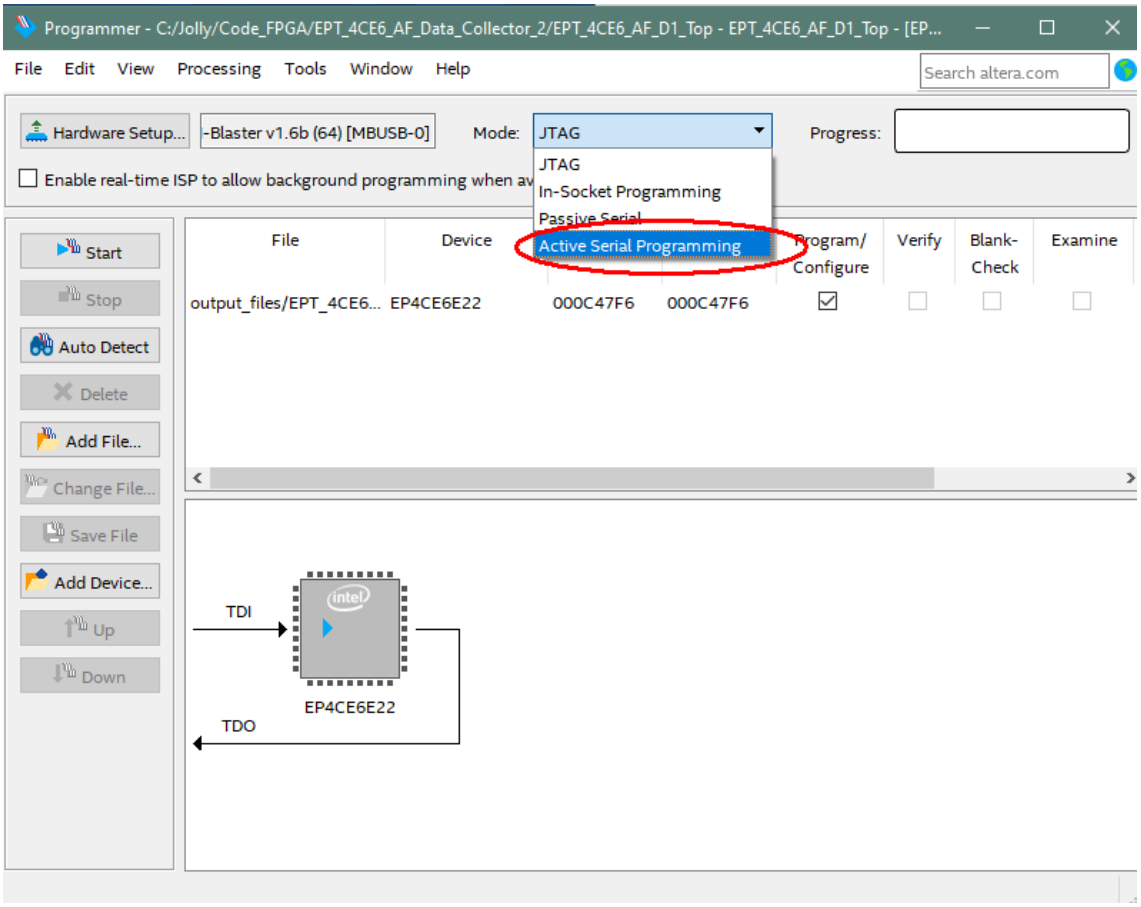
The Hardware Setup Window will open. In the “Available hardware items”, double click on “EPT-JTAG-Blaster v1.0b (64)”.



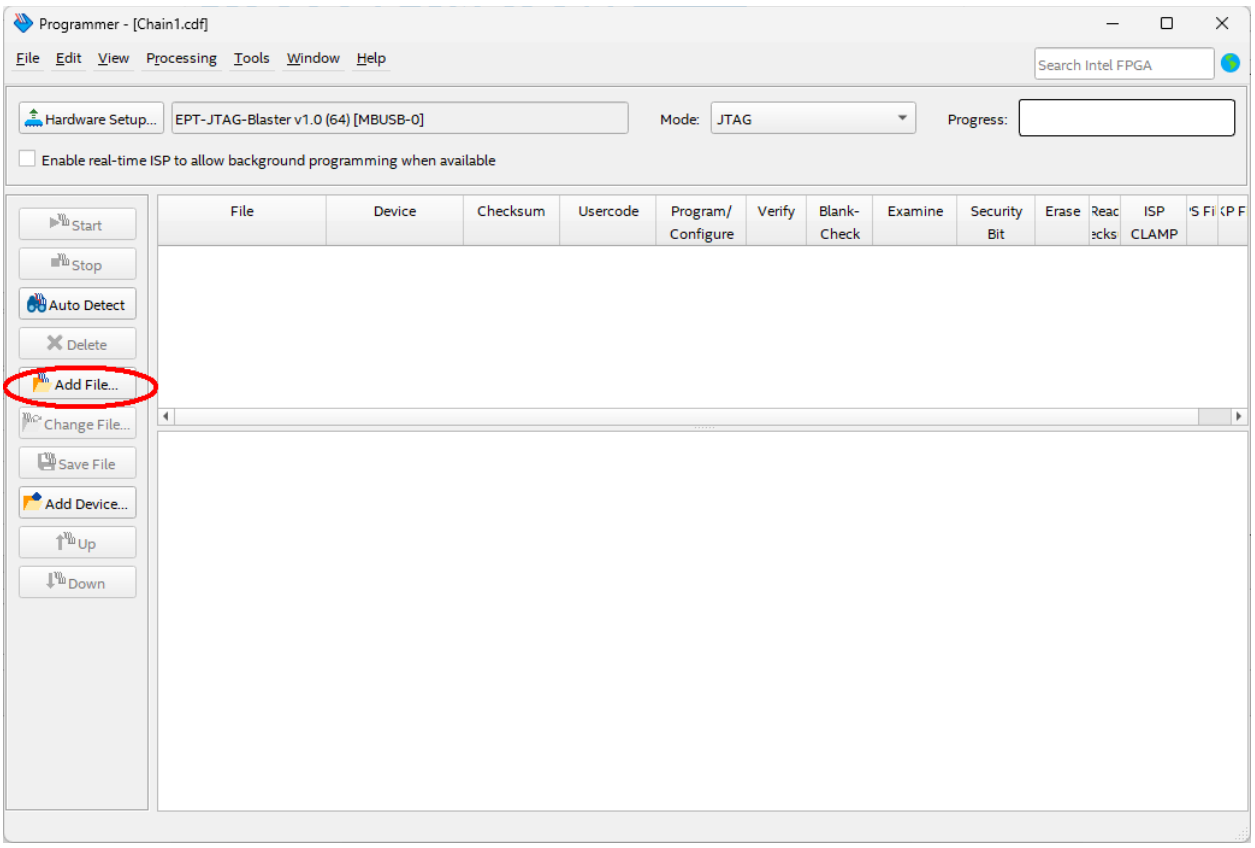
If you successfully double clicked, the “Currently selected hardware:” dropdown box will show the “EPT-JTAG-Blaster v1.0b (64)”.



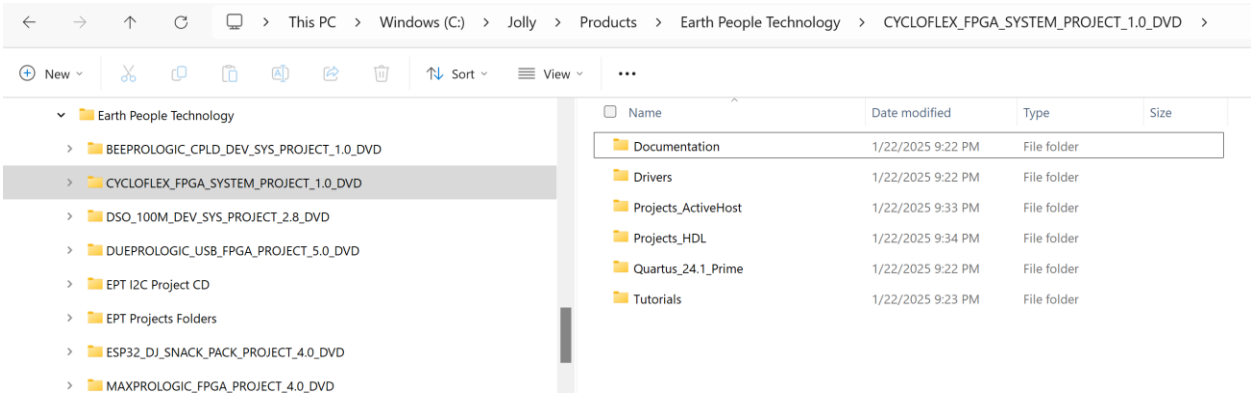
Click on the “Mode:” drop down box. Select the “Active Serial Programming” option.



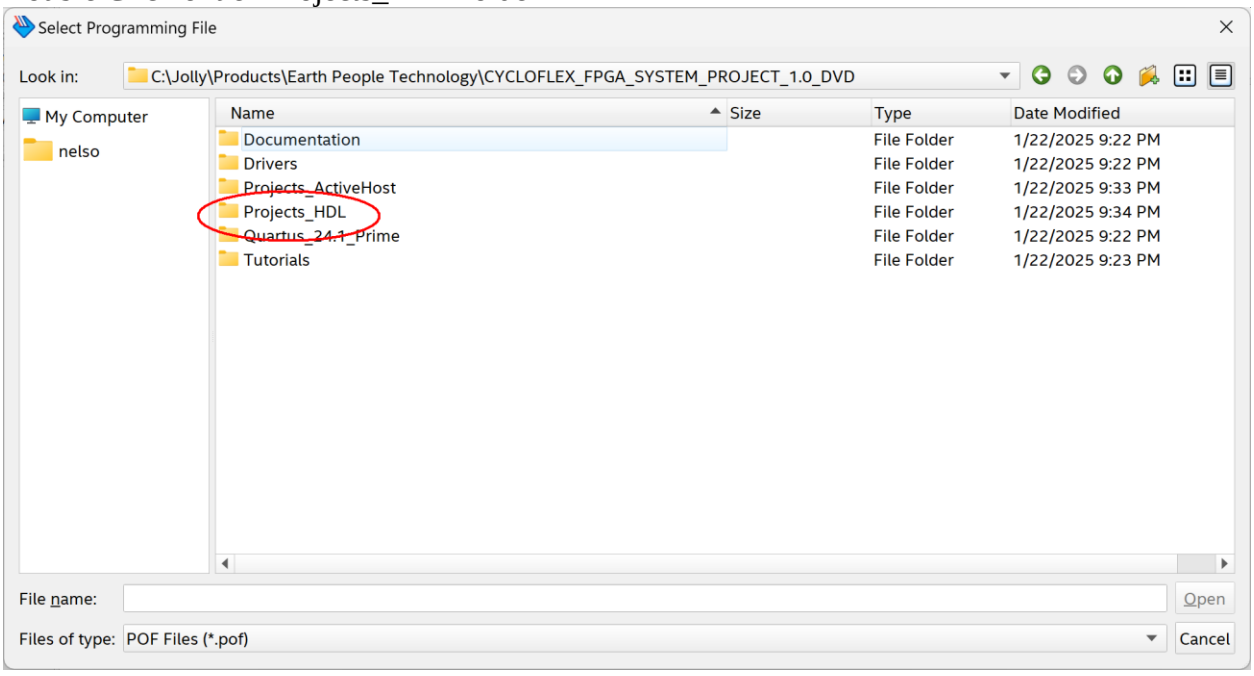
Click on the “Add File” button



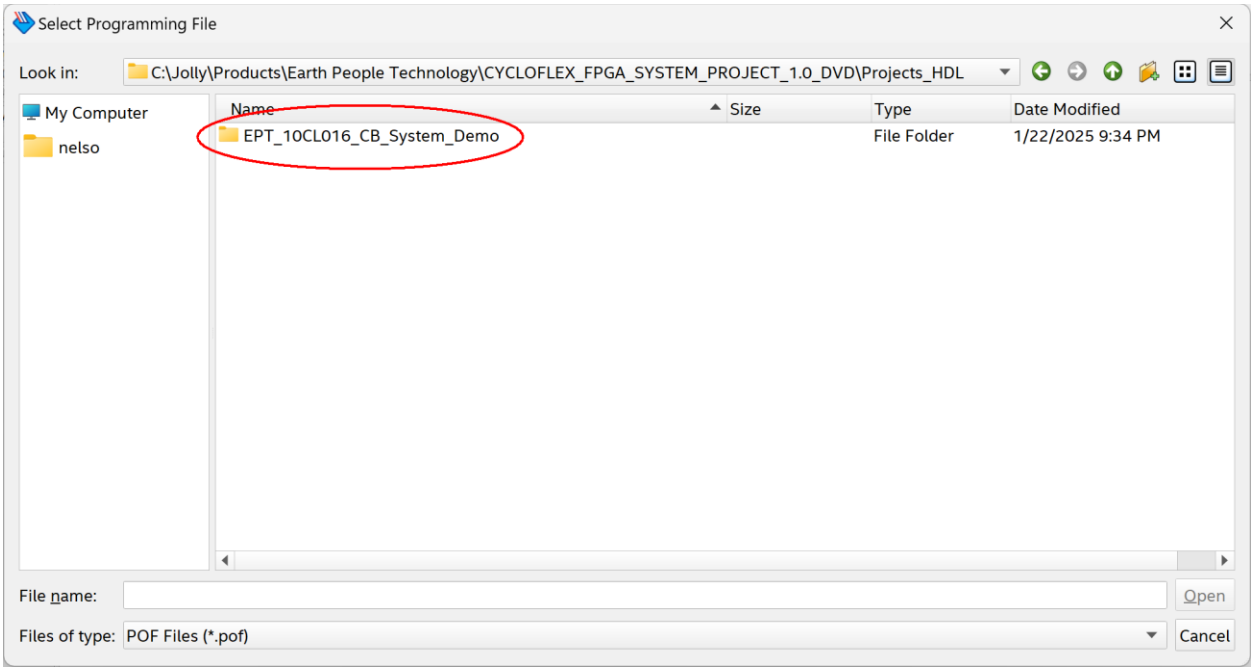
In the DVD project, navigate to the C:\ CycloFlex_FPGA_SYSTEM_PROJECT_x.x_DVD (The "x.x" is the latest available version of the project DVD).



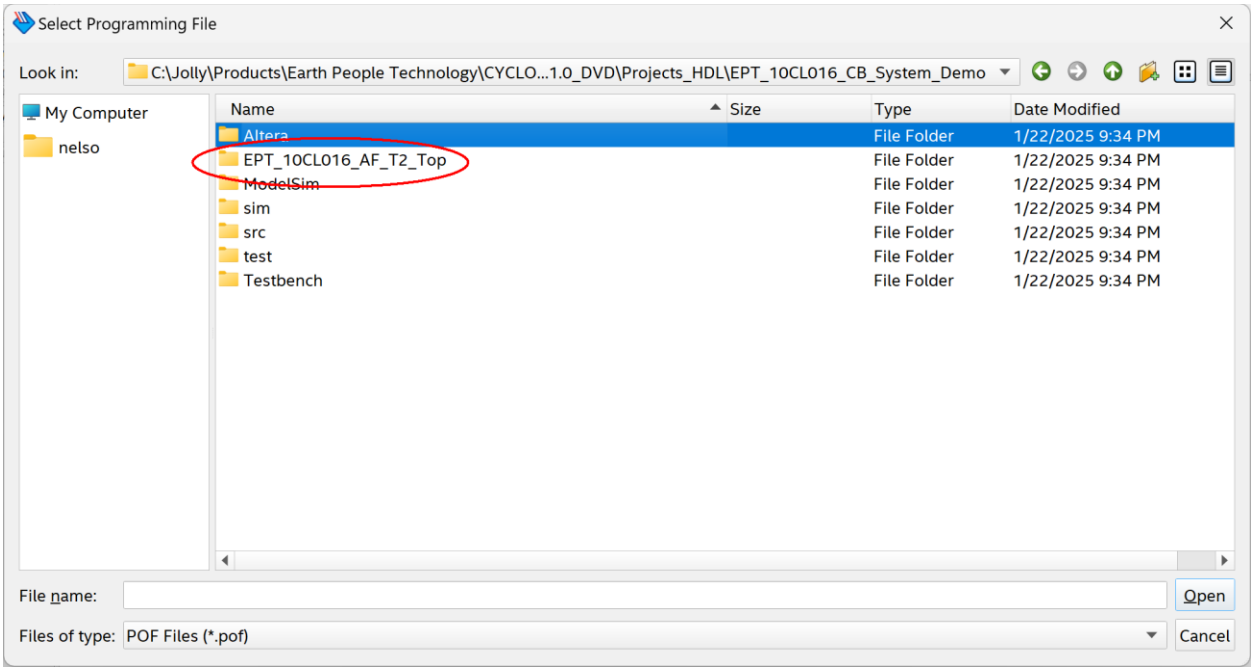
Double Click ont eh Projects_HDL folder



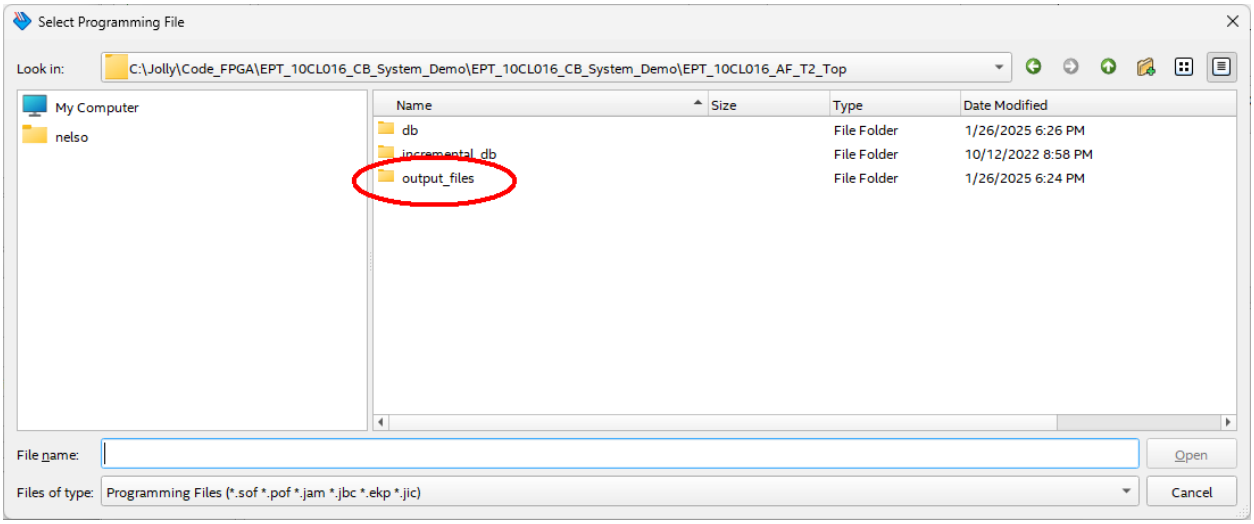
Double Click on the EPT_10CL016_CB_System_Demo Folder



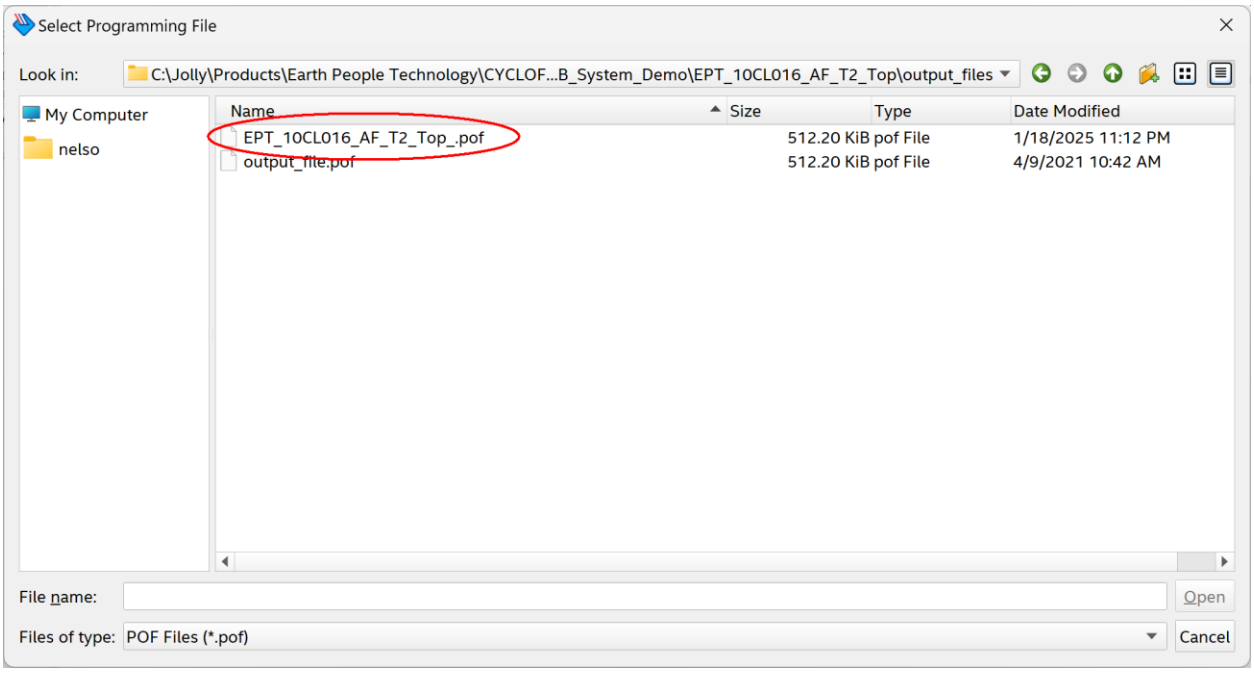
Double Click on the EPT_10CL016_AF_T2_Top Folder



Double Click on the output_files Folder

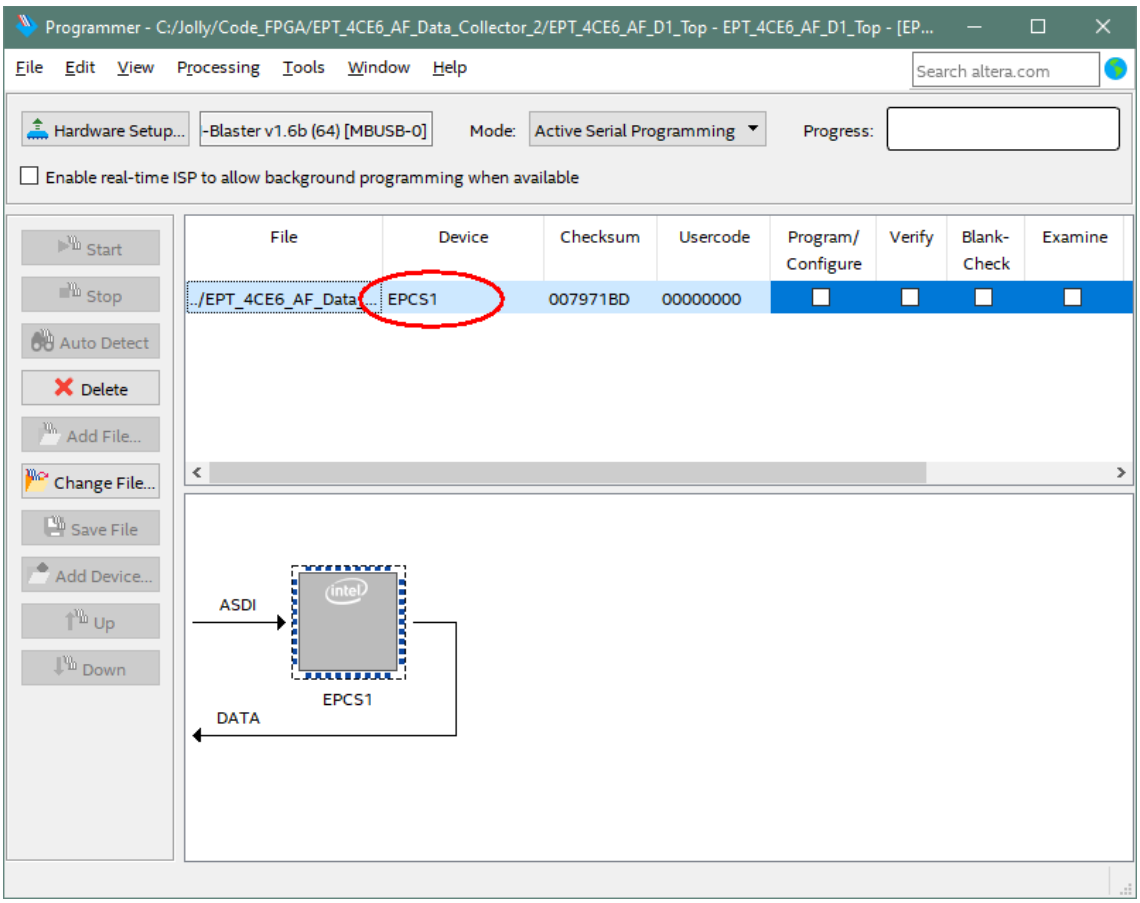


Double click on the “EPT_10CL016_AF_T2.pof” file.

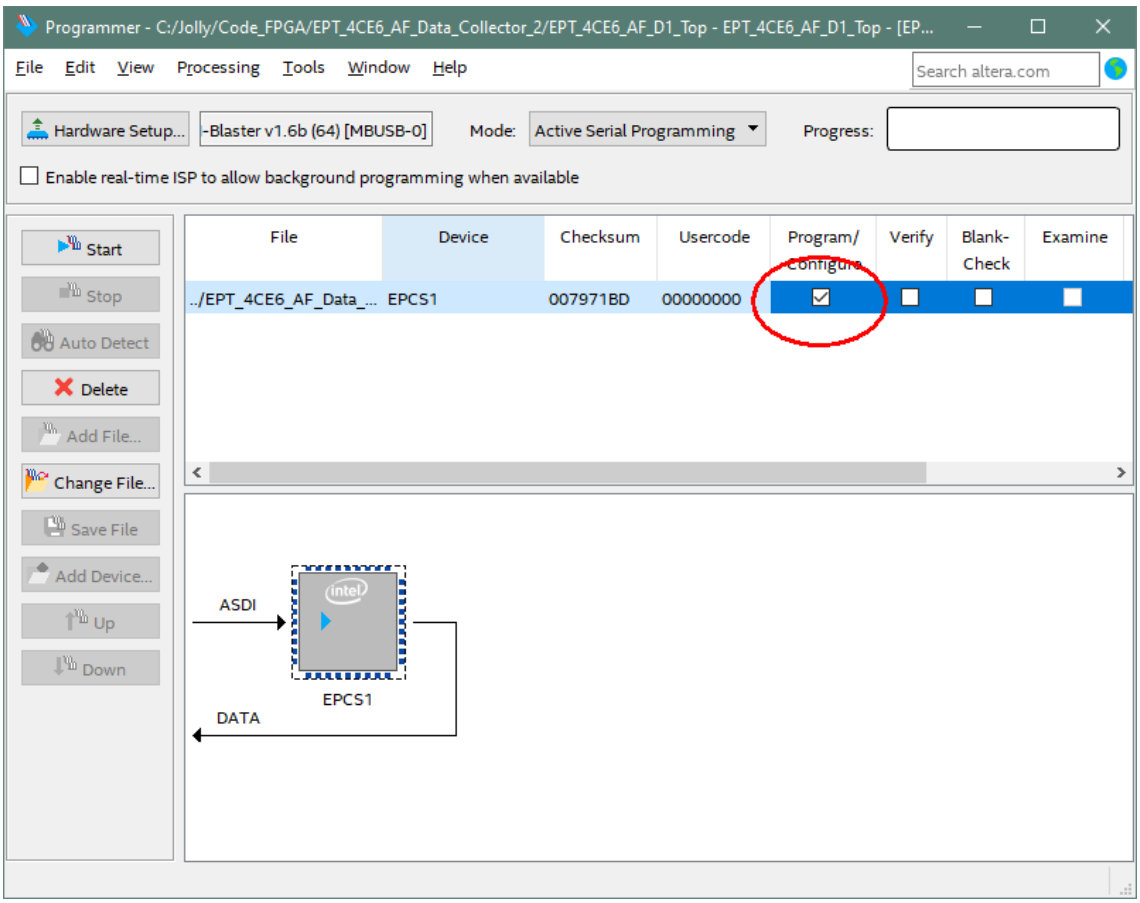


Double click on the “EPT_10CL016_AF_T2.pof” file. Click the Open button in the lower right corner.

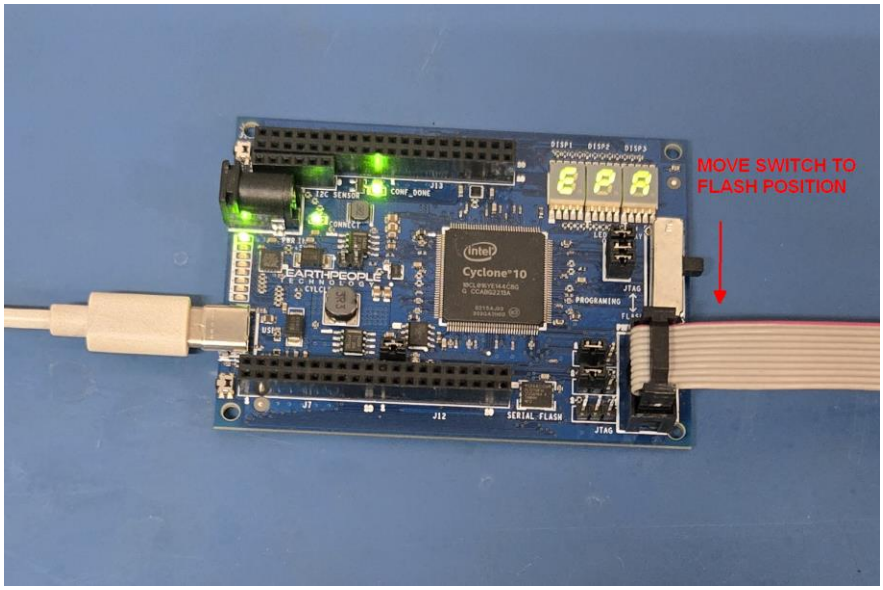
Select the EPCS1 under “Device”.



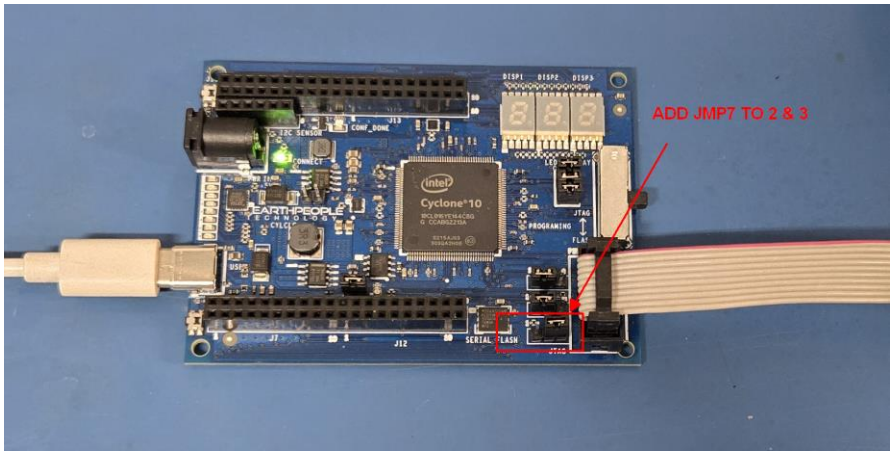
Next, selet the checkbox under the “Program/Configure” of the Programmer Tool.


Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
-Blaster v1.6b (64) [MBUSB-0]
Mode: Active Serial Programming
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
../EPT_4CE6_AF_Data_...
EPCS1
007971BD
00000000
ASDI
EPCS1
DATA

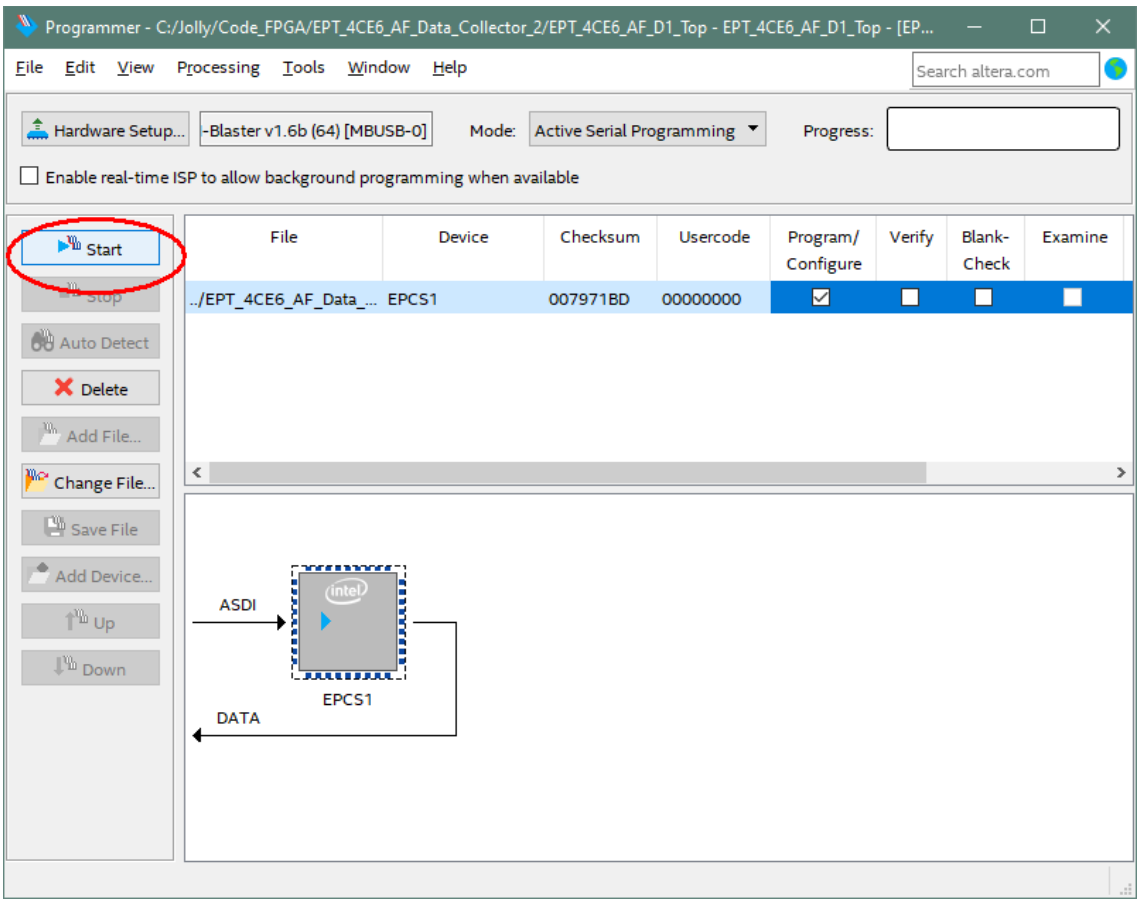
Move the JTAG switch to the “Flash” position



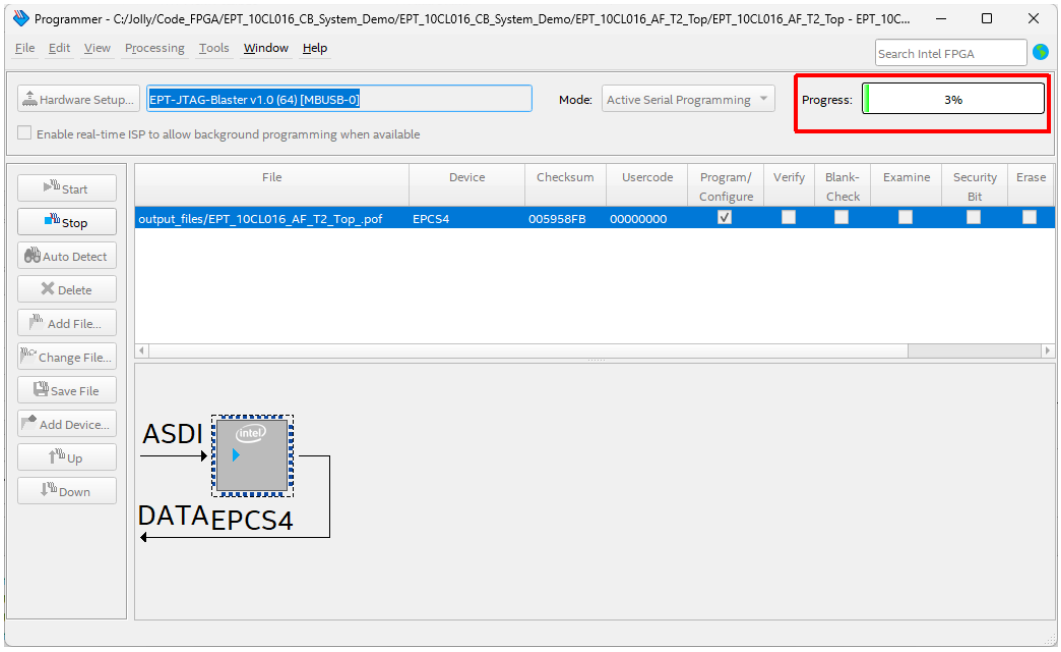
Add JMP7 to Pins 2& 3.



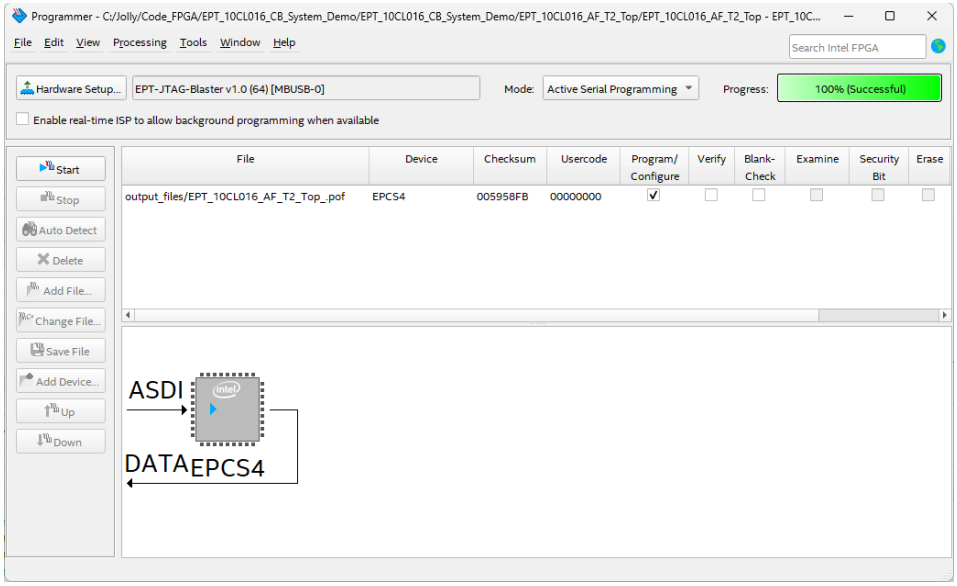
Click on the Start button to to start programming the FPGA. The Progress bar will indicate the progress of programming.



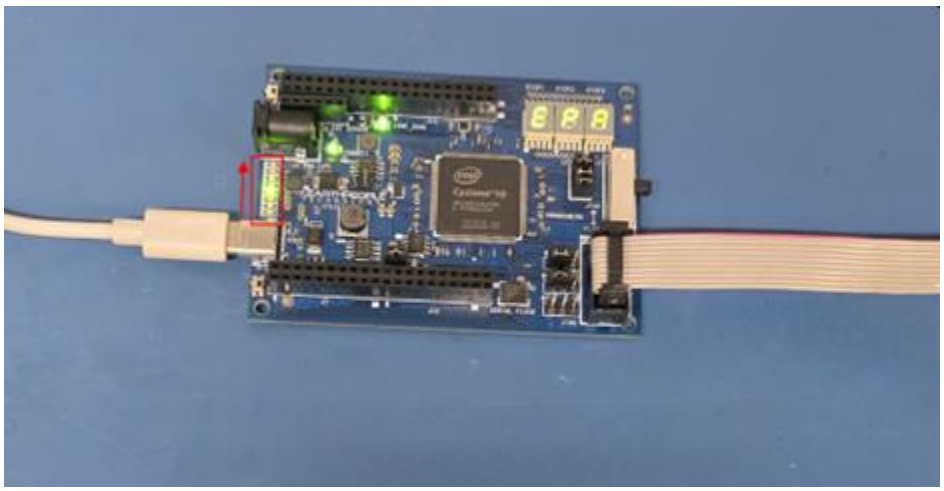
The programming of the CycloFlex will start and you can check the progress in the “Progress” Bar.



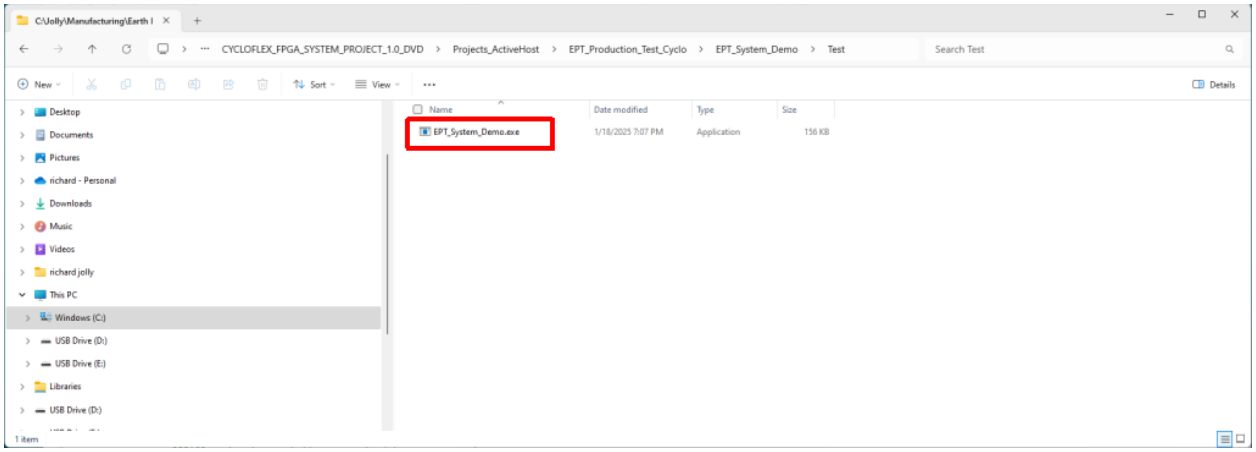
When the programming is complete, the Progress bar will indicate success.



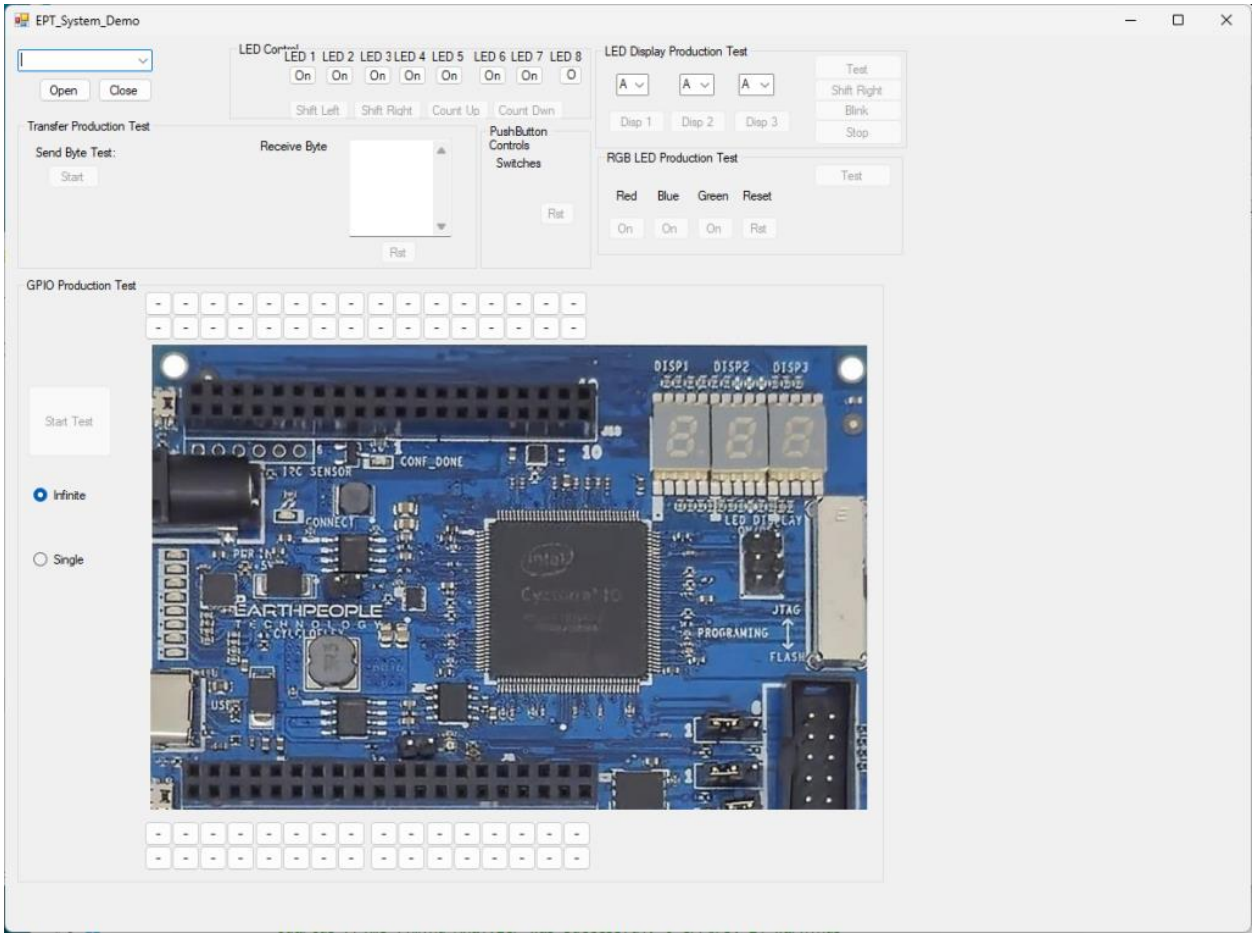
Once the Quartus Programmer has completed, move the JTAG Switch into the “JTAG” Position.



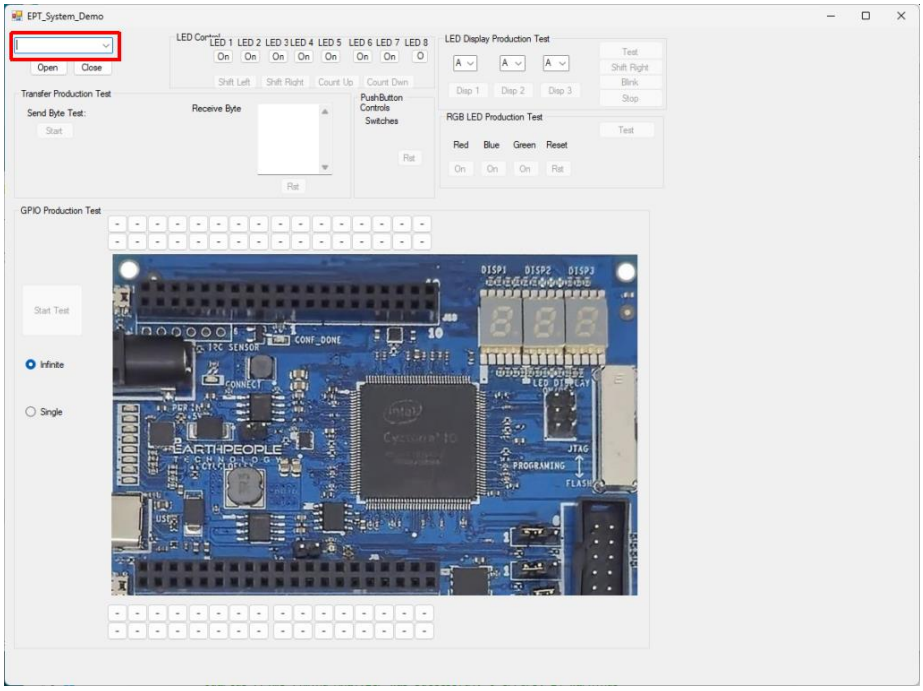
Go back to the EPT CycloFlex Production Test Software folder. Double click on the "EPT_System_Demo.exe" file



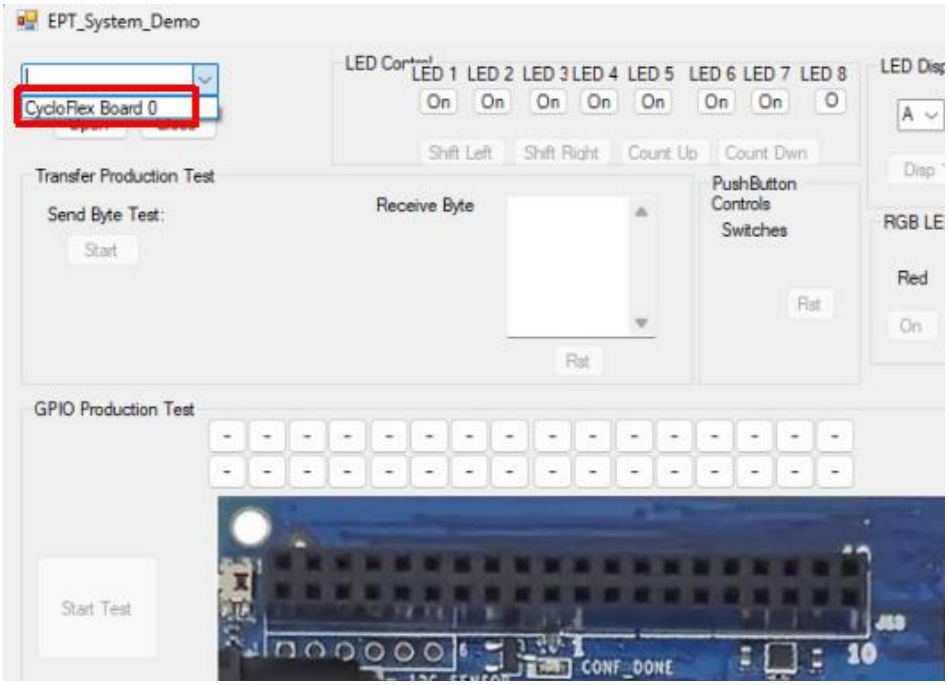
The CycloFlex Production Test software will open



Click on the Drop Down Box below the “EPT_System_Demo” text.

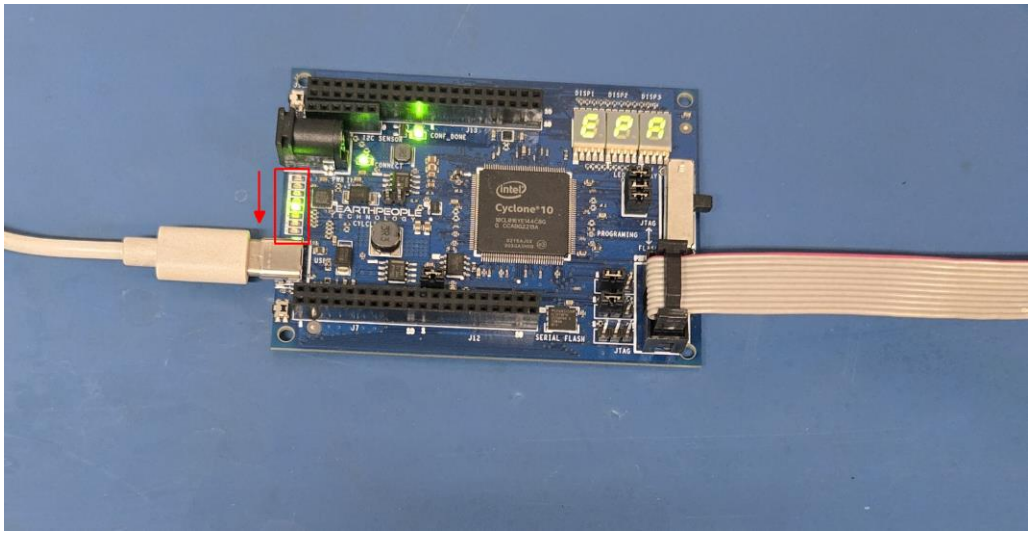


Select the “CycloFlex Board 0”



Click the “Open” button

Ensure that the “Heartbeat” function is displayed on the Green User LEDs (repeating sweep of the LEDs)



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

8.3 Creating a *.pof File for Configuration Flash

In the previous section, it was outlined how to program the Configuration Flash on the CycloFlex using Quartus Prime Lite. The instructions used the *.pof file that was created by EPT for the EPT_System_Demo project. The *.pof must be created for each project the user creates. This section provides instructions on how to create a *.pof file for your project.

Generation of *.pof file from *.sof file for Altera FPGA

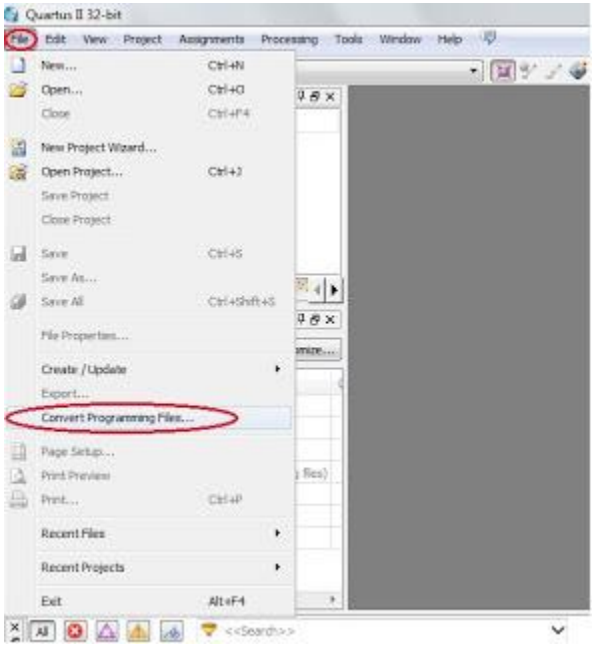
What is *.sof & *.pof file?

*.sof is SRAM Object File & *.pof is Programming Object File. Both files are used to configure FPGA.
FPGA contains SRAM. It holds the design configuration. To configure that memory *.sof file is

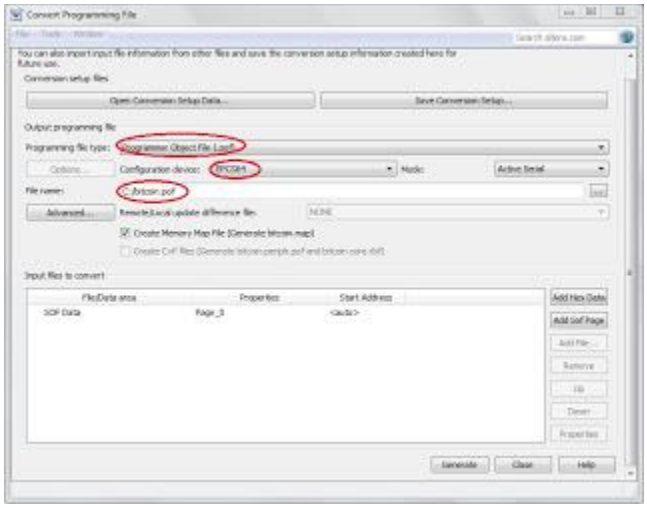
used. SRAM is volatile memory, during every power cycle it is necessary to reprogram FPGA. FPGA can be configured automatically using external non volatile memory. These memories can be Compact Flash Memory cards or dedicated high density flash memory.
*.pof file is used to configure such non volatile memories. FPGA has capability to access these memories and its internal logic can configure SRAM from external memory. So no need to configure FPGA manually.

Step 1:
Run Quartus Prime Lite

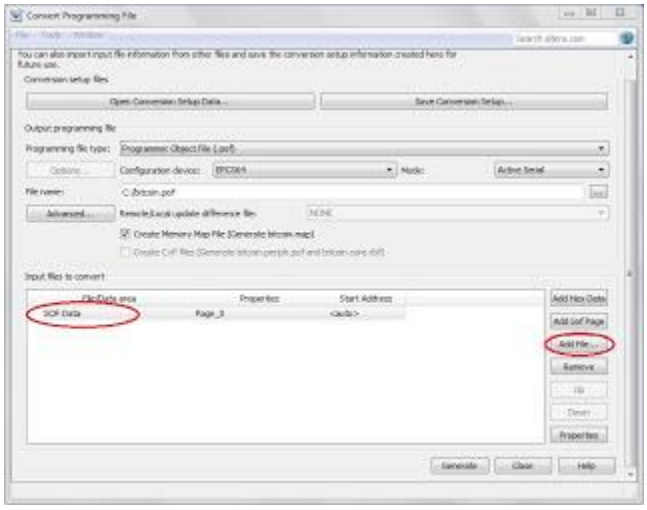
Step 2:
Go to "File Menu-> Convert Programming Files"



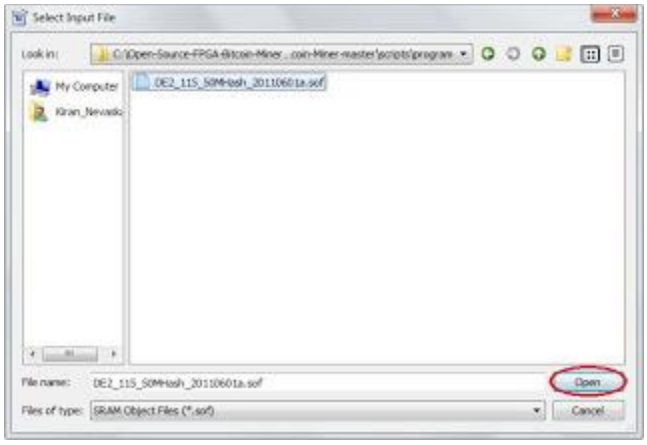
Step 3:
In Convert Programming File window select
a) Programming File Type as *.pof
b) Configuration device (in my case EPCS64)
c) Select path and file name for output file



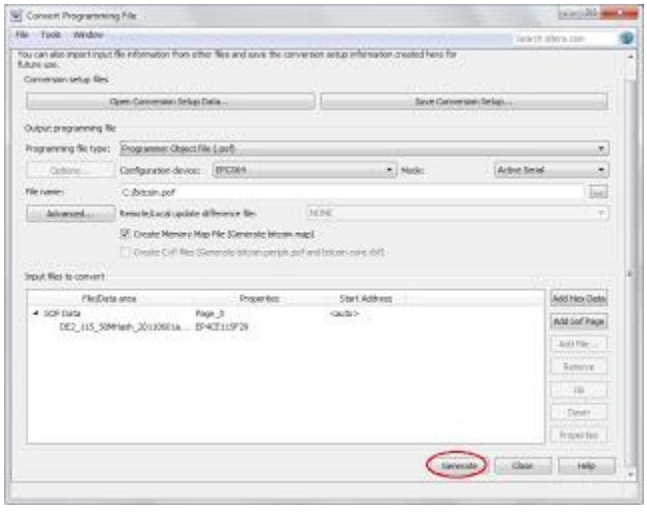
Step 4:
Now click on SOF Data as shown in below image and click on Add File



Step 5:
Browse .sof file using Select Input File browser and click on Open



Step 6:
Now click on Generate button in Convert Programming File window



Step 7:
If everything is OK then QuartusII will show message of success

Step 8:
Programming .pof file to Serial Configuration Device
a) Connect FPGA board to PC using USB Cable (Make sure you have connected to USB Blaster)
b) Select Serial Configuration Memory using slide switch(Prog Mode) on FPGA board and power on the FPGA board
c) Go to "Tools Menu->QuartusII Programmer" in QuartusII
d) Select "Mode" as "Active Serial Programming"
e) Click on Add file button and browse generated .pof file
f) Now Click on Start button. Programming may take couple of minutes. Time depends on size of the file
g) After successful operation power of the board and select "Run Mode" using slide switch instead of "Prog Mode". After powering on the board and you will find FPGA is working.
Sometimes it happens that QuartusII Programmer fails to detect Byte Blaster (USB JTAG). In this case check whether drivers are installed on not. You may locate drivers in "X:/altera/tool_version/quartus/drivers/usb-blaster-ii". X is the installation directory. In my case tool_version was 13.0.
Then Click on "Hardware Setup" button in QuartusII Programmer. In Available Hardware items you will find USB Blaster. Double click on USB Blaster and close the window. If you select USB Blaster properly the you will find name of USB Blaster next to the Hardware Setup Button in QuartusII Programmer.

Now you can program external configuration memory like EPCS64 using QuartusII Programmer with generated .pof file

CycloFlex Development System User Manual