EARTHPEOPLE
T e c h n o l o g y

FPGA Development System User Manual

# DUEPROLOGIC

# FPGA  DEVELOPMENT SYSTEM
# User Manual

The DueProLogic (DPL) and its integrated development and distinctive runtime environment has been specifically designed for Electrical Engineering students, hobbyists, and entrepreneurs prototyping/developing/running projects involving logic, with the added opportunity, of readily mating with a widely used microprocessor board, the Arduino Due, and other ARM Cortex compatibles.  The combination of FPGA programmable logic and a microcontroller is unbeatable in an educational student learning setting and in many other projects where each can bring its strength.

The DPL FPGA development system provides a convenient, user-friendly work flow by connecting seamlessly with Altera's Quartus Prime 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 and a standard USB cable. The Active Host SDK provides a highly configurable bi-directional communications interface between Arduino and host. It connects transparently with the Active Transfer Library in the FPGA code. This Active Host/Active Transfer combination eliminates the complexity of designing a USB communication system. No scheduling USB transfers, USB driver interface or inf file changes are needed. The EPT FPGA development system is a unique combination of hardware and software.

**http://www.earthpeopletechnology.com/**

EARTHPEOPLE Technology

FPGA Development System User Manual

## Table of Contents

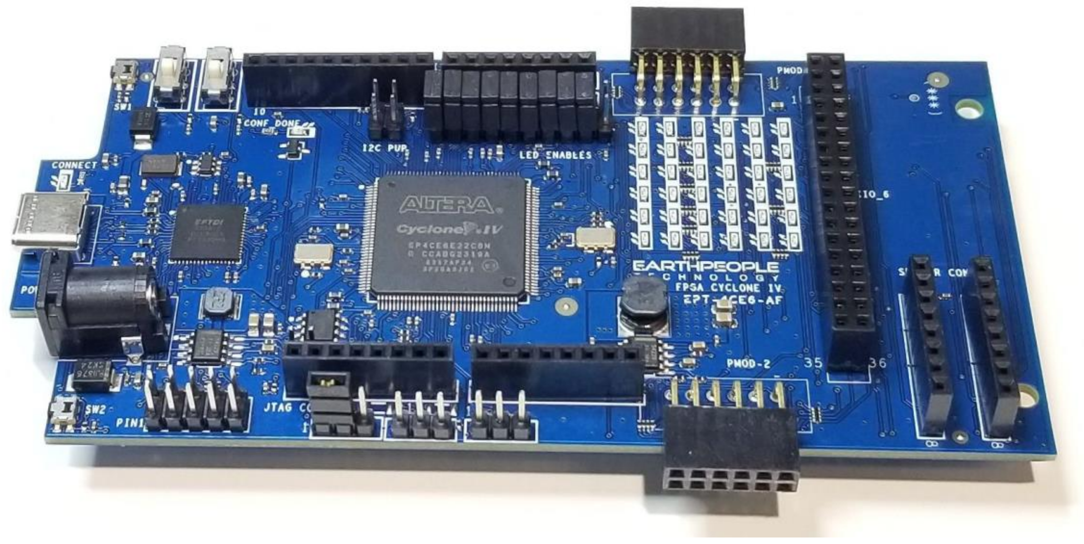
FPGA Development System User Manual

## 1 Introduction and General Description

The DueProLogic gives learners the opportunity to have a hands-on approach when learning logic, exploring different iterations of schematic/code designs and the operation of those circuits.

With the DPL's FPGA, projects can be easily be attempted which rely on asynchronous, exceedingly fast, and even multiple separate concurrent logic structures operating in parallel. Logic circuits are implemented within the FPGA at few-nanosecond gate speeds and highly parallel in operation, effectively a few hundred MHz. Programmable logic is today's technology for logic learners and implementers. The DPL allows the learner to be more productive and better focus on the underlying logic and integration with the non-logic aspects of non-trivial projects.

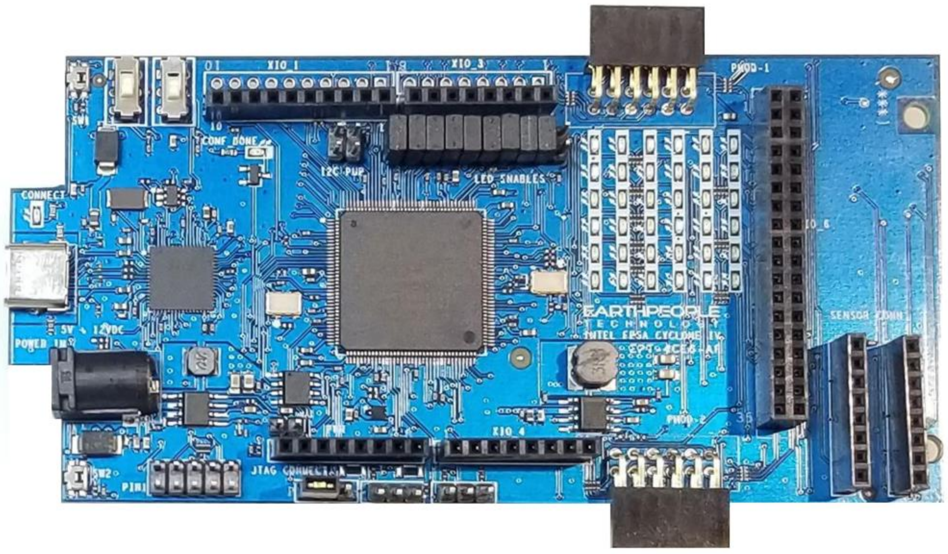
The Earth People Technology FPGA development system hardware consists of a High Speed (480 Mb/s) USB to Serial bus chip and an FPGA. The USB interface provides both Configuration of the FPGA and a Serial transfer path. The software consists of the Active Host SDK for the PC. The firmware includes the Active Transfer Library which is used in the FPGA to provide advanced functions for control and data transfer to/from the Arduino.



FPGA Development System User Manual

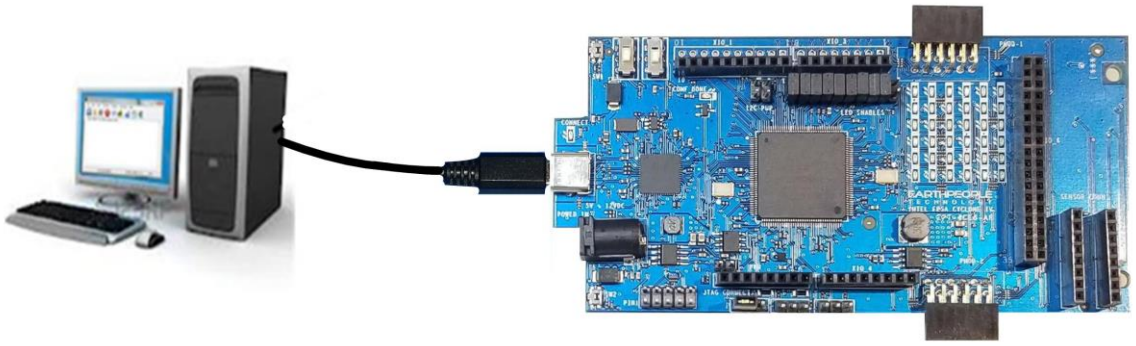
The DueProLogic FPGA Development System allows users to write HDL code (either Verilog or VHDL) that will implement any digital logic circuit. The user's HDL code is compiled and synthesized and packaged into a programming file. The programming file is programmed into the Configuration Flash using one channel of the USB to Serial chip, the FT2232H.The Active Host SDK contains COM Port functions which maintains device connection, polling, writes and includes a unique receive mechanism that automatically transfers data from DPL when data is ready. It also alerts the user code when the COM Port has stored the transfer and the data is available to the software GUI (graphical user interface). Users do not need to interface with the USB Host Driver or any Windows drivers. They need only to include the Active Host SDK in their projects. The Active Transfer Libraries must be included in the FPGA project to take advantage of the configurability of the Active Host SDK. All of the drivers, libraries, and project source code are available at www.earthpeopletechnology.com .

## 1.1 Test Driving the Active Host Test Application



The DueProLogic board comes pre-loaded with the EPT_Platform_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.

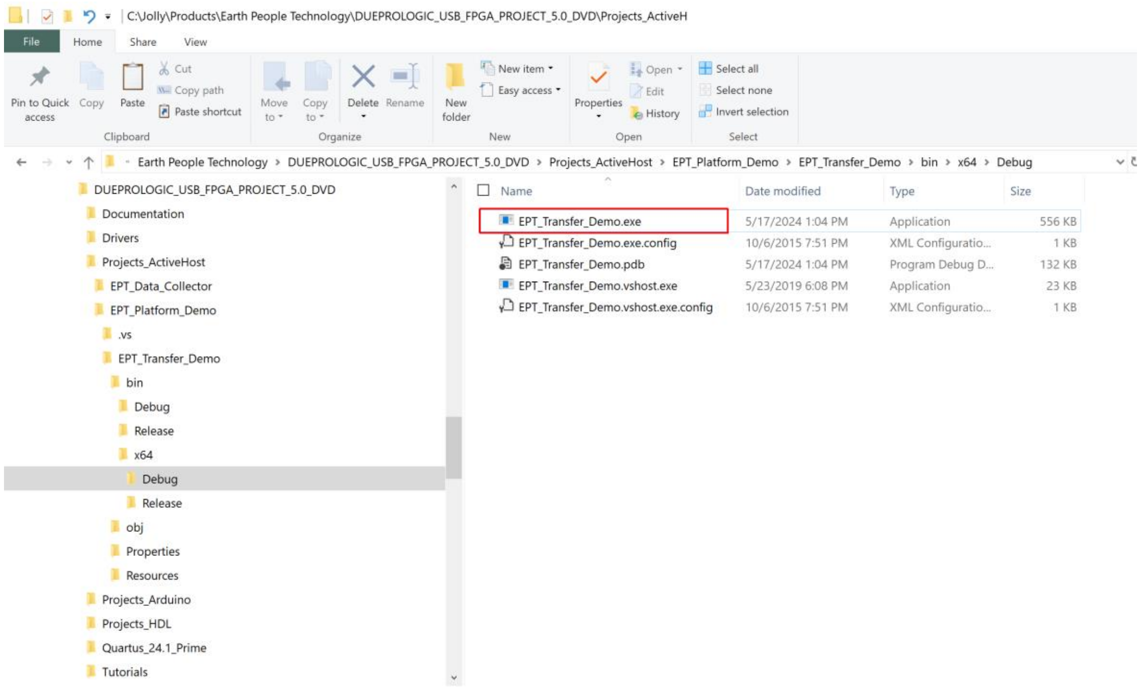
To test drive the application, connect the DPL to the Windows PC using a USB-C cable. Load the driver for the board. See the section “EPT Drivers” for instructions on loading the DPL driver. If the USB driver fails to load, the Windows OS will indicate that no driver was loaded for the device. In the 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.

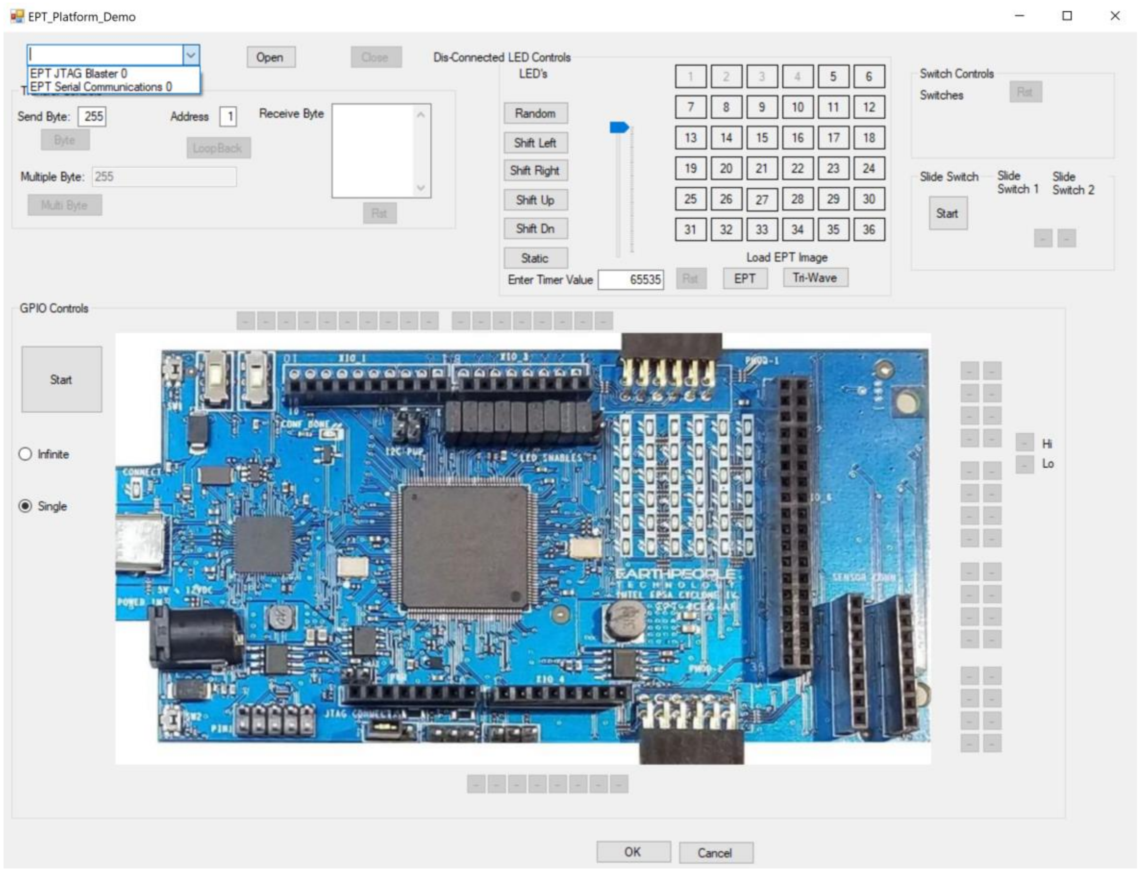
Browse to the Projects_ActiveHost\EPT_Platform_Demo\EPT_Transfer_Demo\bin\X64\Debug\

folder on the DUEPROLOGIC_USB_FPGA_PROJECT_x.x_DVD.

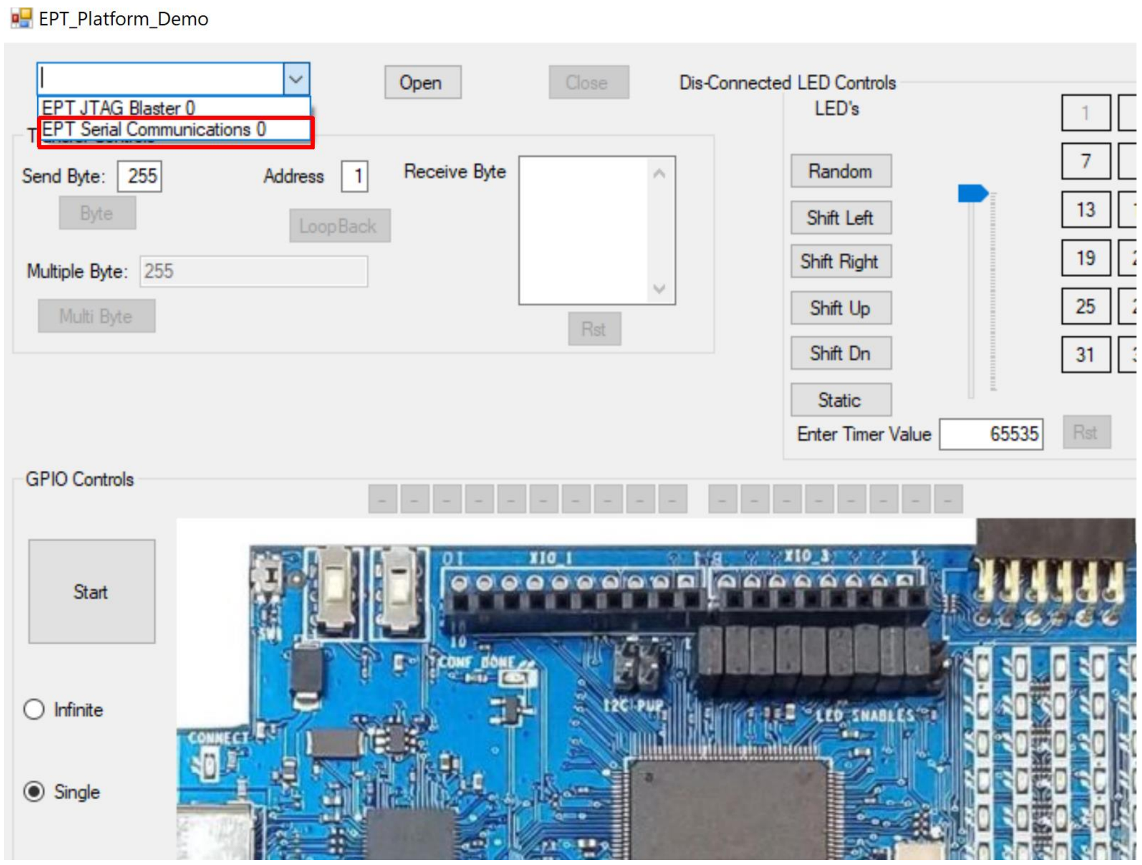


| Name | Date modified | Type | Size |
|---|---|---|---|
| EPT_Transfer_Demo.exe | 5/17/2024 1:04 PM | Application | 556 KB |
| EPT_Transfer_Demo.exe.config | 10/6/2015 7:51 PM | XML Configuratio... | 1 KB |
| EPT_Transfer_Demo.pdb | 5/17/2024 1:04 PM | Program Debug D... | 132 KB |
| EPT_Transfer_Demo.vshost.exe | 5/23/2019 6:08 PM | Application | 23 KB |
| EPT_Transfer_Demo.vshost.exe.config | 10/6/2015 7:51 PM | XML Configuratio... | 1 KB |

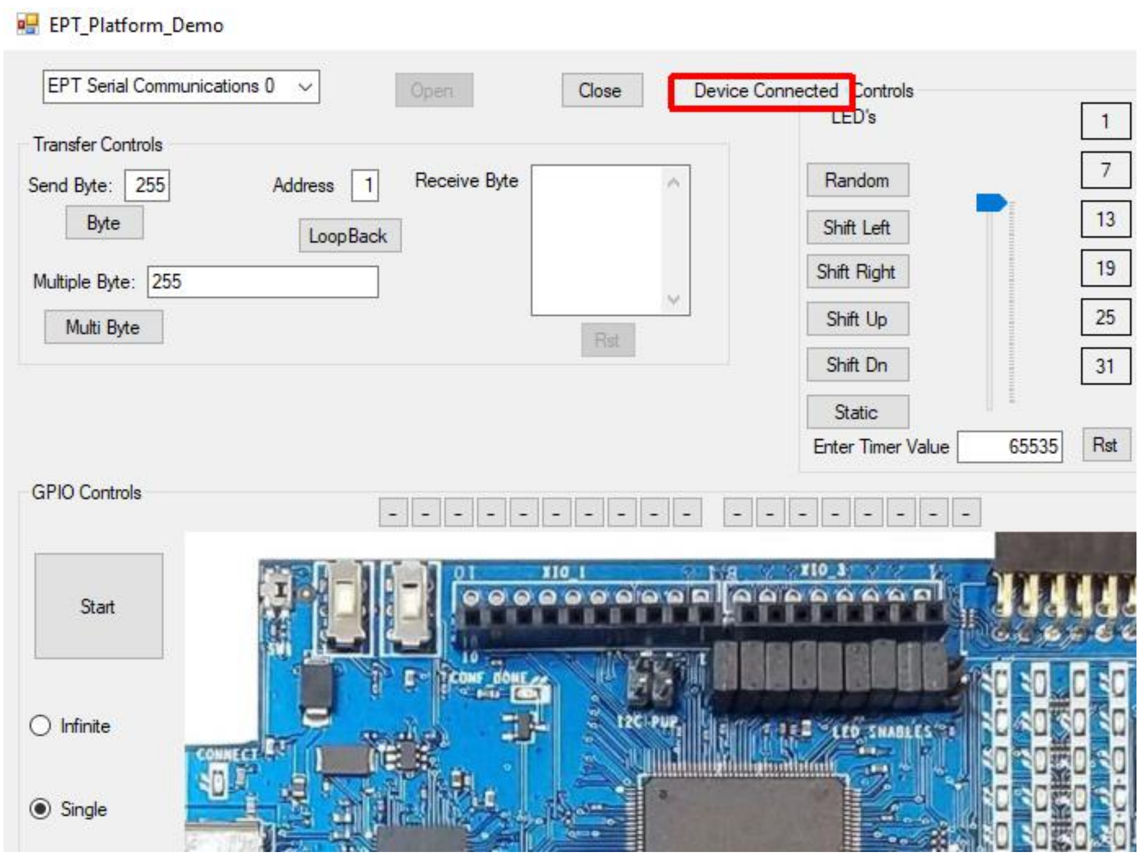
Double click on the EPT_Transfer_Demo.exe. The application should load with a Windows form.



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

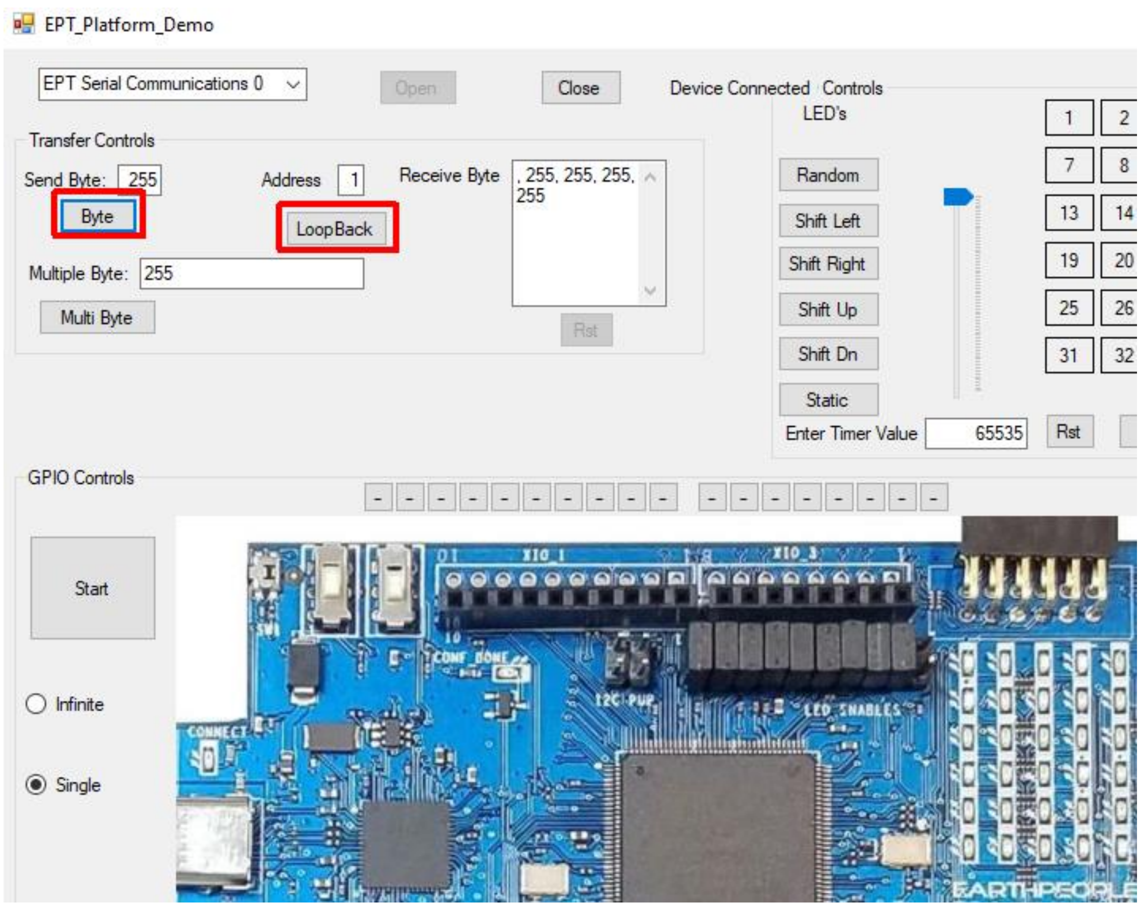


When the EPT Platform Demo software has properly accessed the board, the “Device Connected” label will appear.

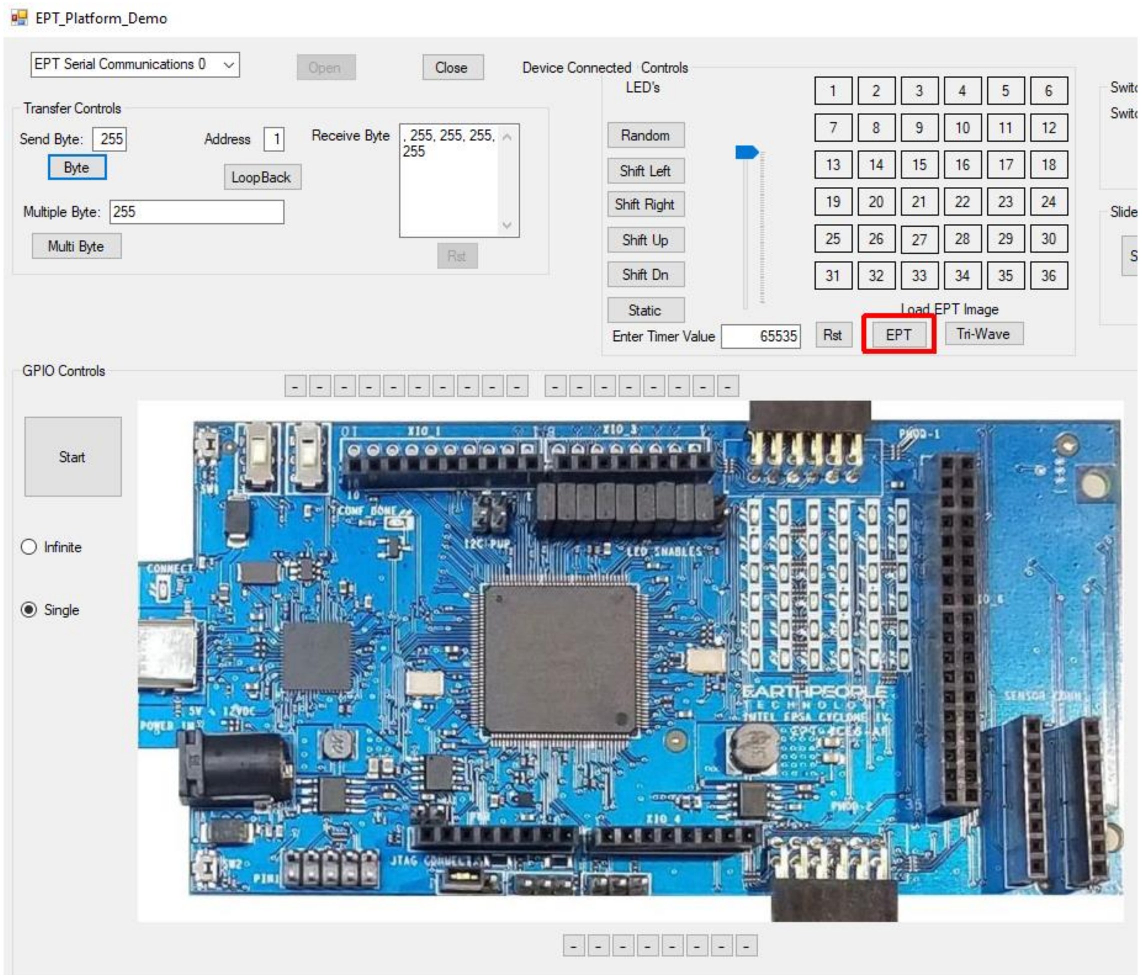


In the "Transfer Controls" Group, leave the Address set at 1 for the Transfer Controls Group.

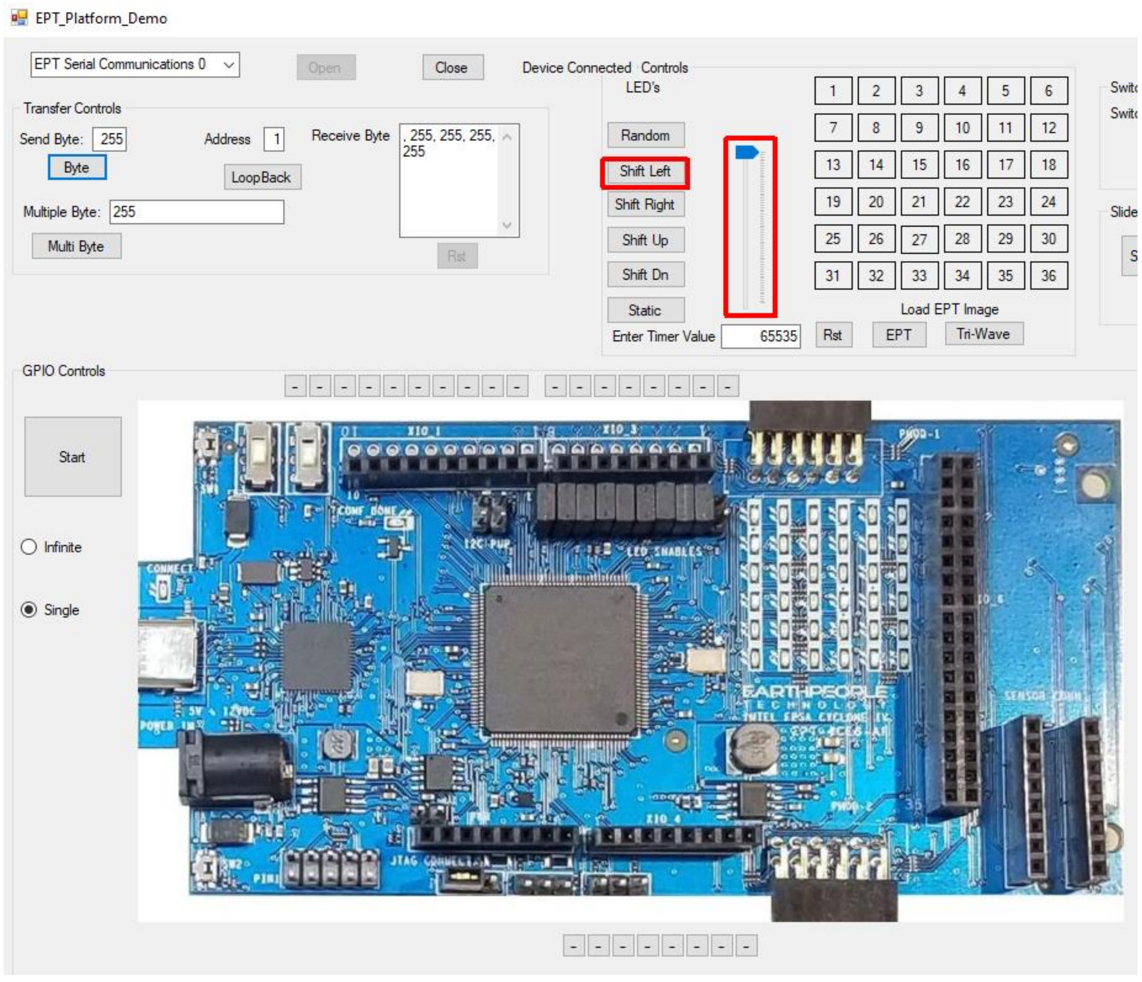
To exercise the Single Byte Transfer EndTerm, click the "LoopBack" button in the Transfer Controls group. Type in several numbers separated by a space and less 256 into the Multiple Byte textbox. Then hit the Multi Byte button. The numbers appear in the Receive Byte textbox.



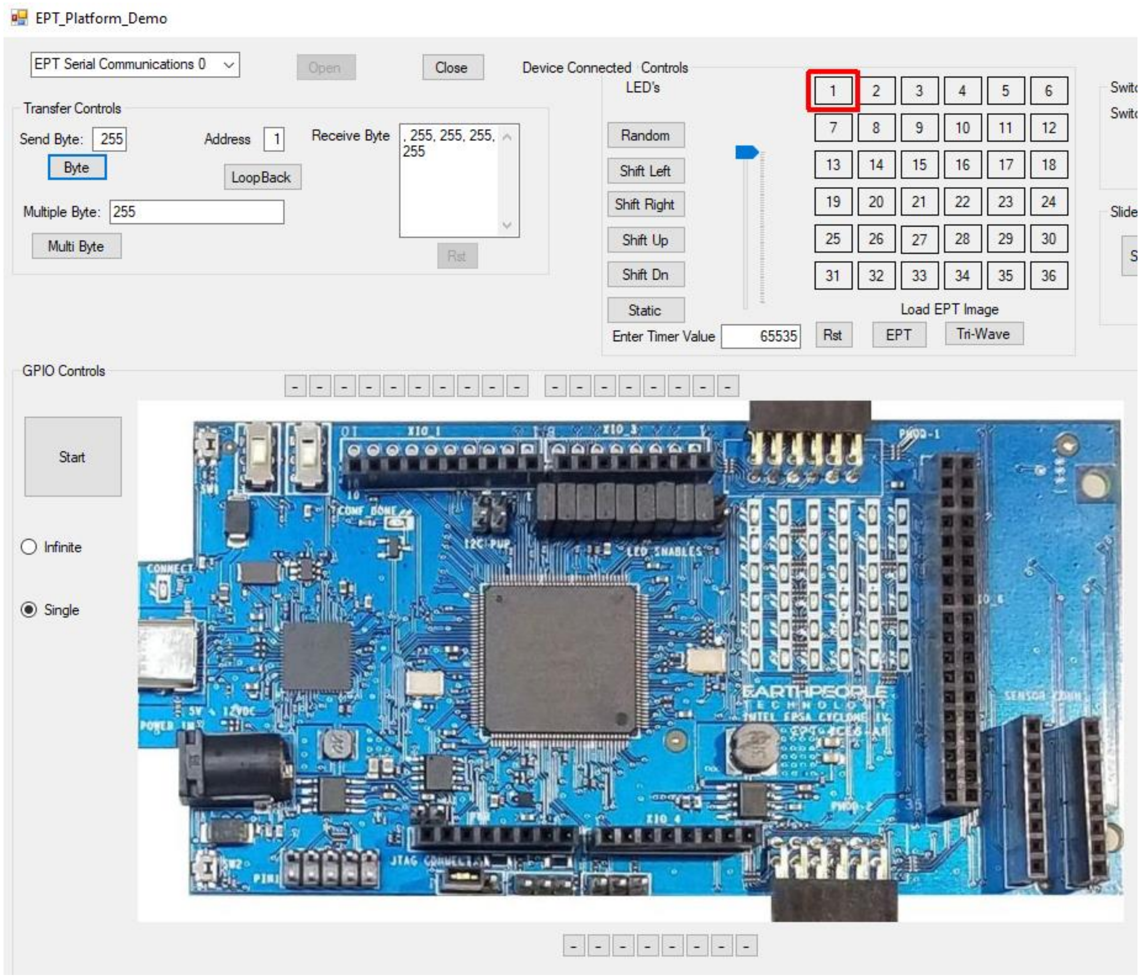
Under LED Controls, press the "EPT" button under the "Load EPT Image".



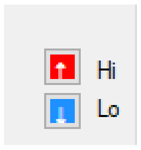
The characters “EPT” will be displayed on the 6x6 LED array. All the characters cannot be displayed at once, so the “Shift Left” or “Shift Right” button must be pressed to see all characters.

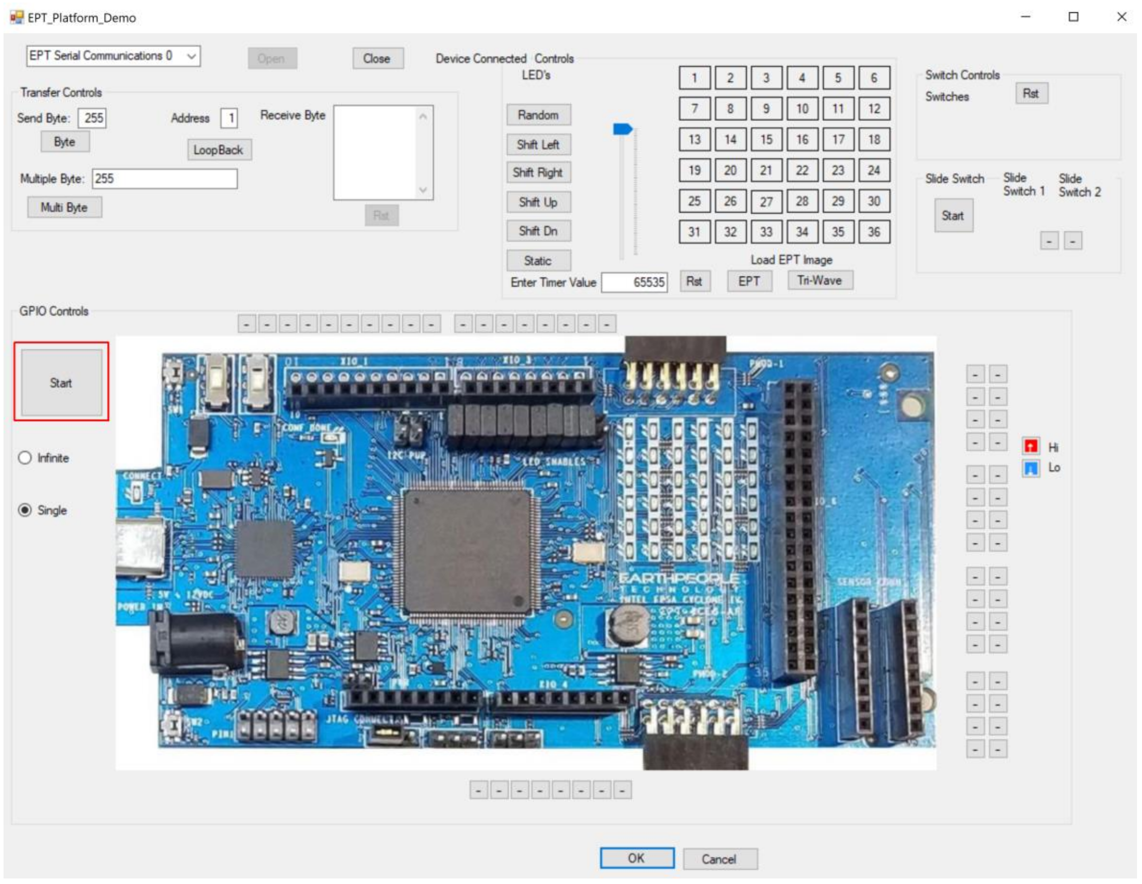

EPT_Platform_Demo
EPT Serial Communications 0
Open
Close
Device Connected
Controls
LED's
Transfer Controls
Send Byte: 255
Byte
Address 1
Receive Byte
, 255, 255, 255, 255
LoopBack
Multiple Byte: 255
Multi Byte
Rst
Random
Shift Left
Shift Right
Shift Up
Shift Dn
Static
Enter Timer Value 65535
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
Load EPT Image
Rst
EPT
Tri-Wave
GPIO Controls
Start
Infinite
Single

Next press any of the numbered buttons in the "LED Controls" group. This will toggle the corresponding LED in the 6x6 LED Array. It performs this operation as a Block Write. So, an entire LED frame will be transferred to the DueProLogic each time a button is pressed.

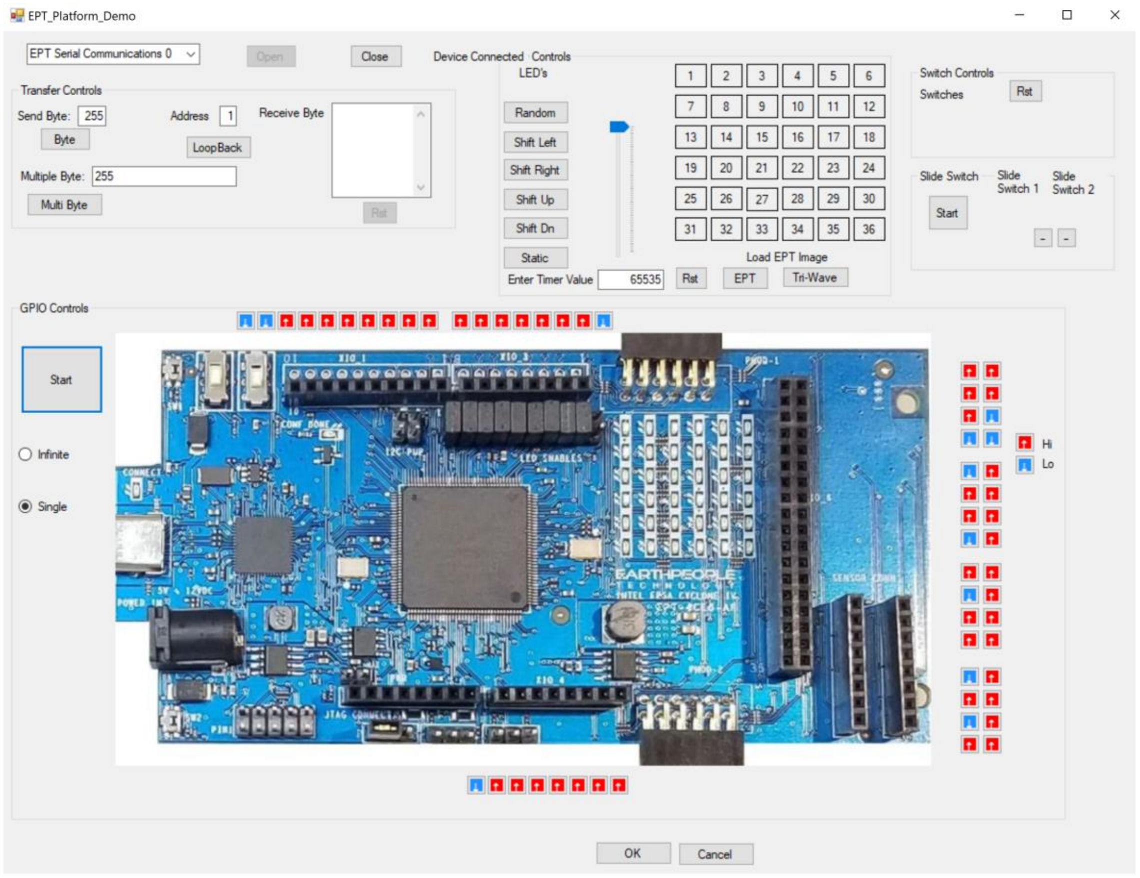

EPT_Platform_Demo
EPT Serial Communications 0
Open
Close
Device Connected
Controls
LED's
Transfer Controls
Send Byte: 255
Address 1
Receive Byte
, 255, 255, 255, 255
Byte
LoopBack
Multiple Byte: 255
Multi Byte
Rst
Random
Shift Left
Shift Right
Shift Up
Shift Dn
Static
Enter Timer Value 65535
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
31 32 33 34 35 36
Load EPT Image
Rst
EPT
Tri-Wave
GPIO Controls
Start
Infinite
Single

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





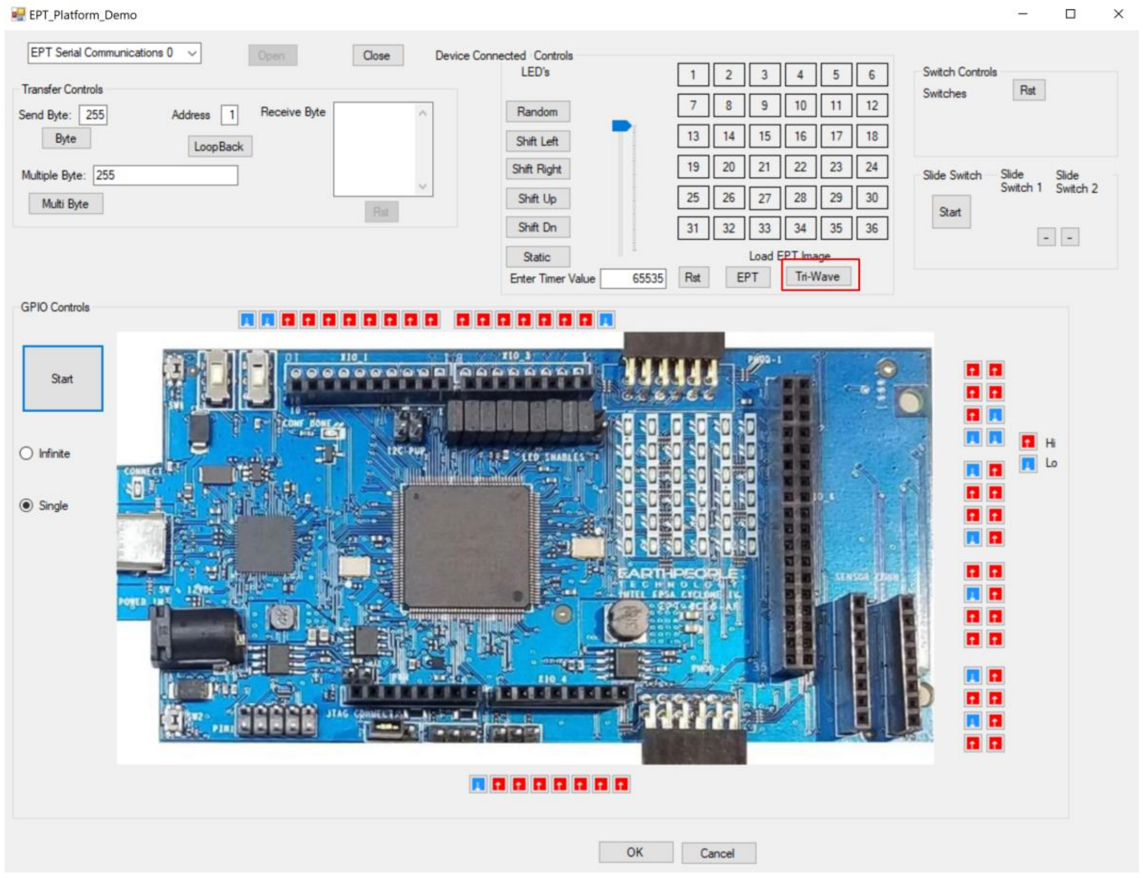
The results of each pin are displayed next to the image of the DueProLogic in separate buttons.

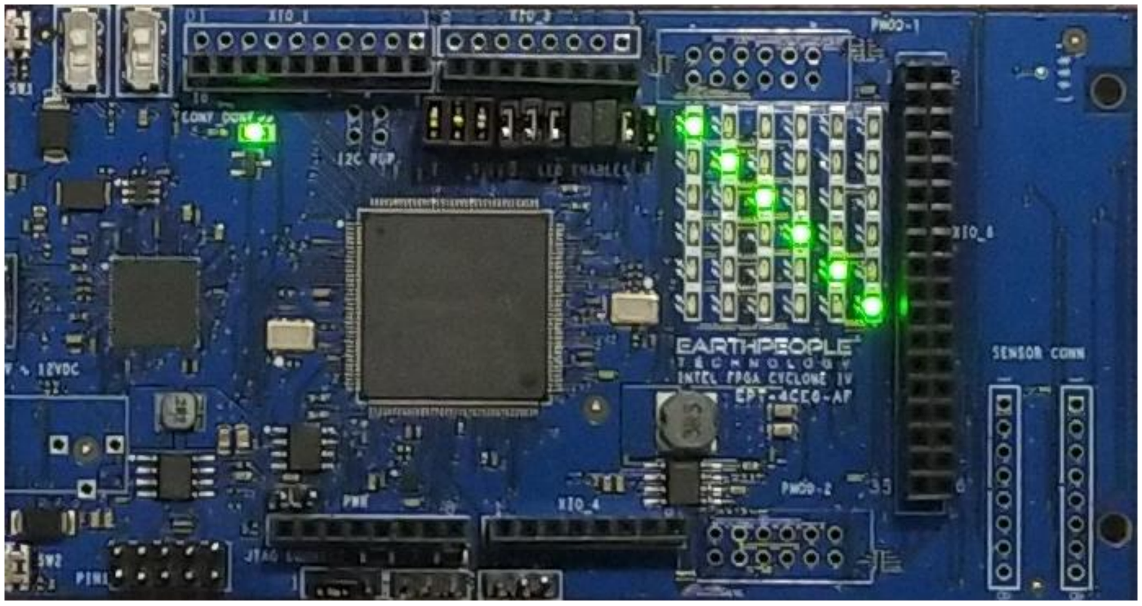


Each input at the board edge connector is floating, it has no solid connection to +3.3V or Ground. So, a mixture of Hi and Lo indications is normal.

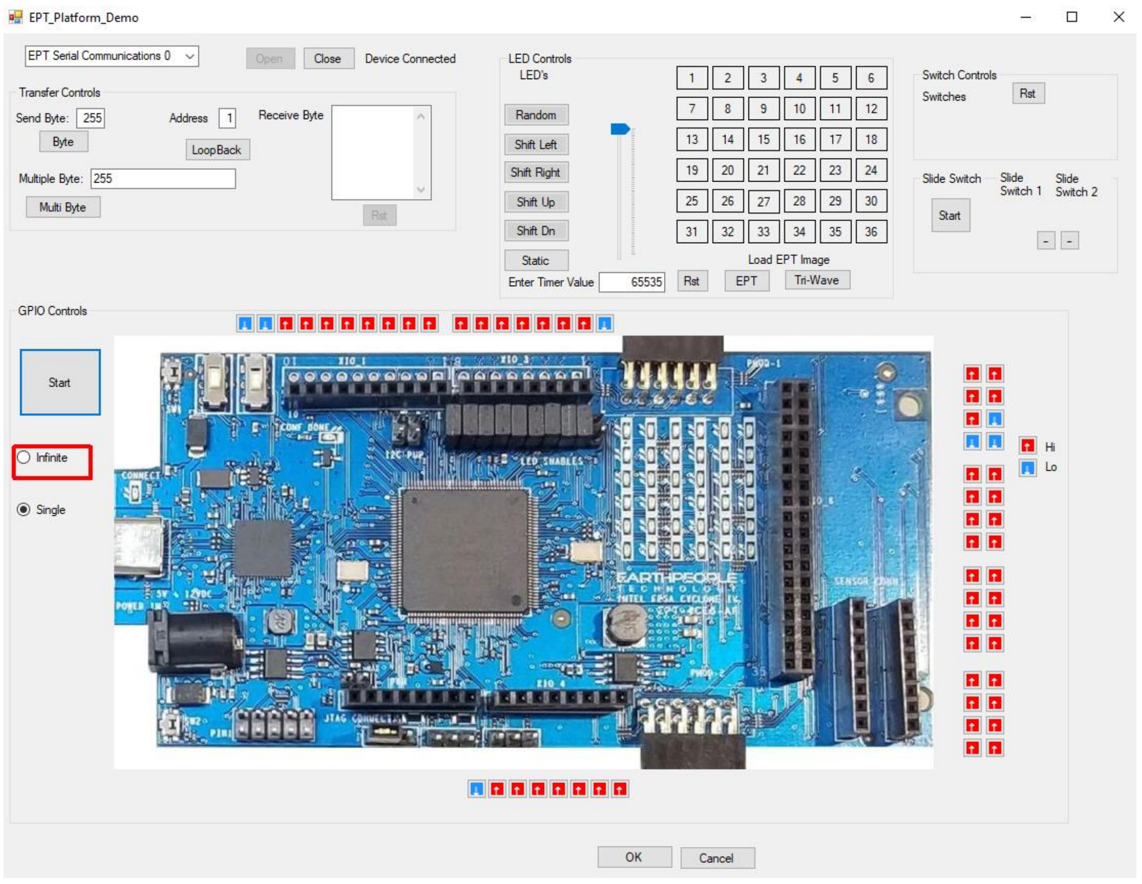
Next, click on the “Tri-Wave” button under the LED Controls Group.

FPGA Development System User Manual

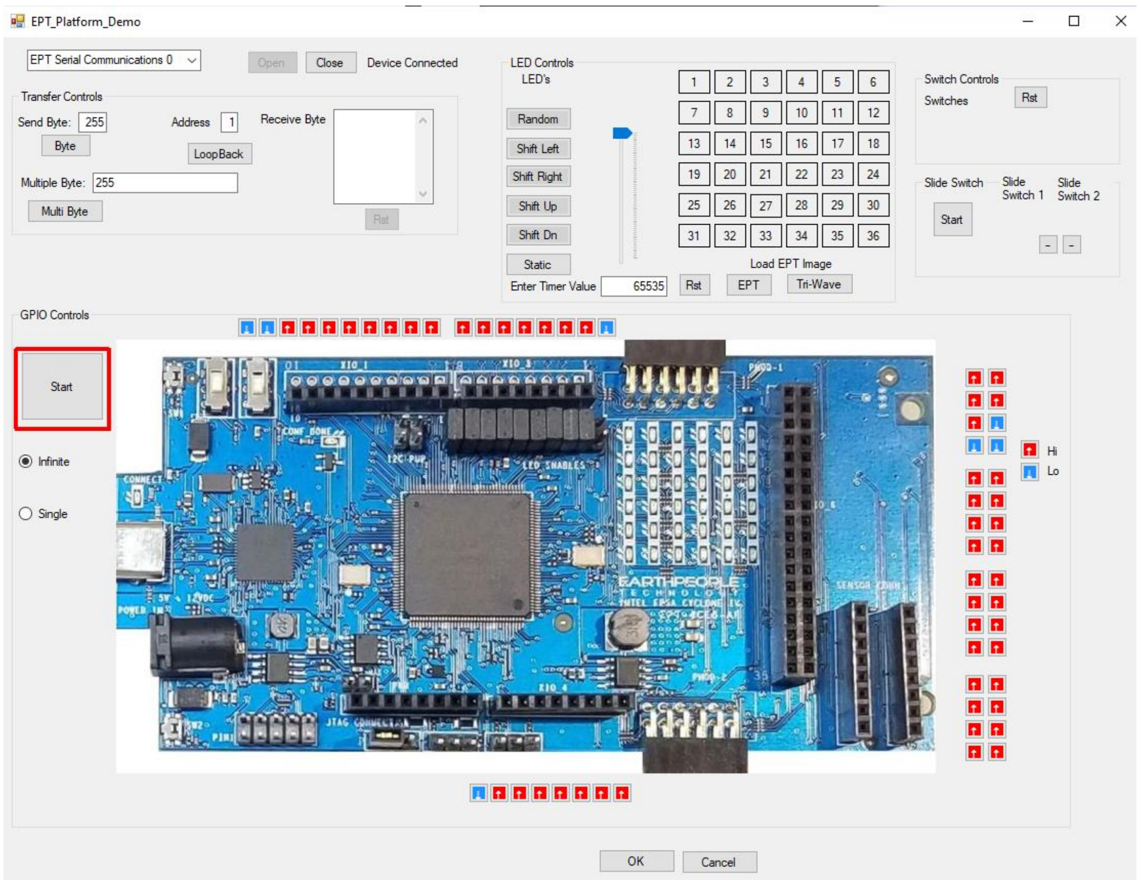




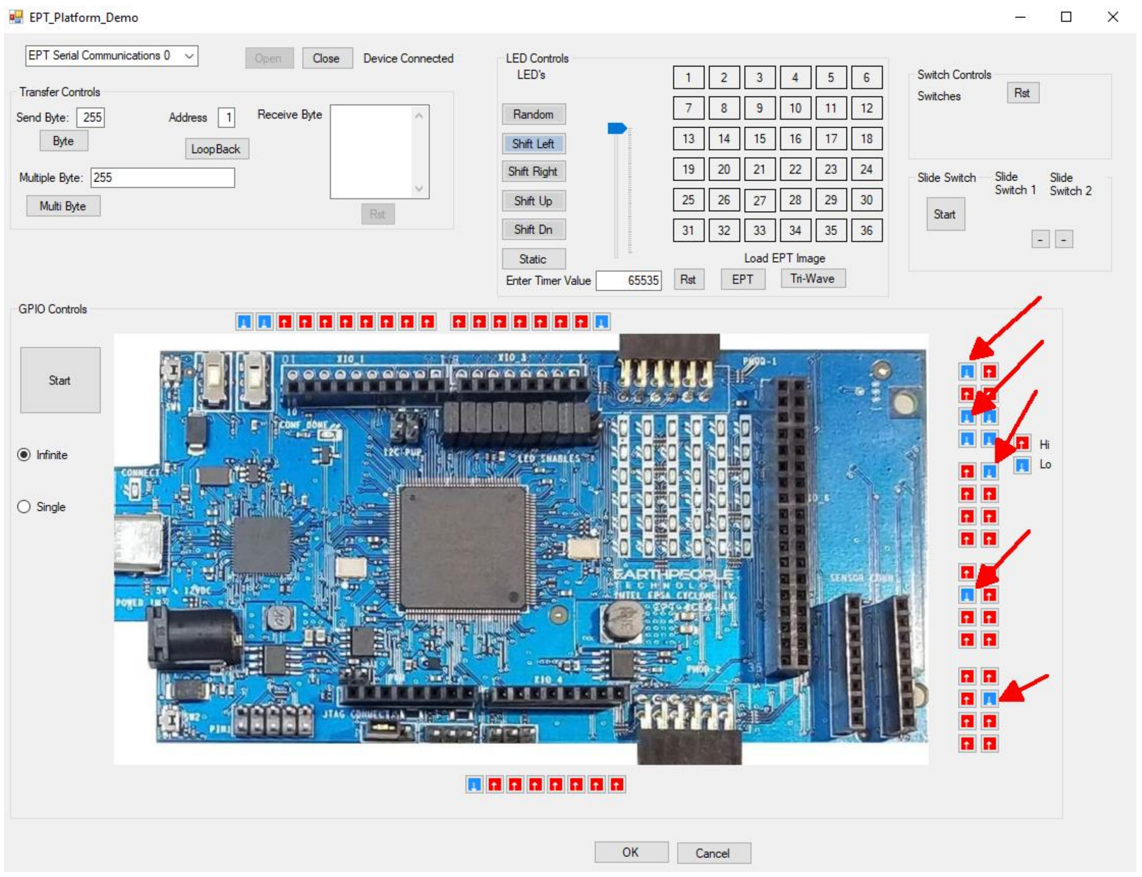
Then, click on the “Infinite” Radio button in the “GPIO Controls” Group Box.



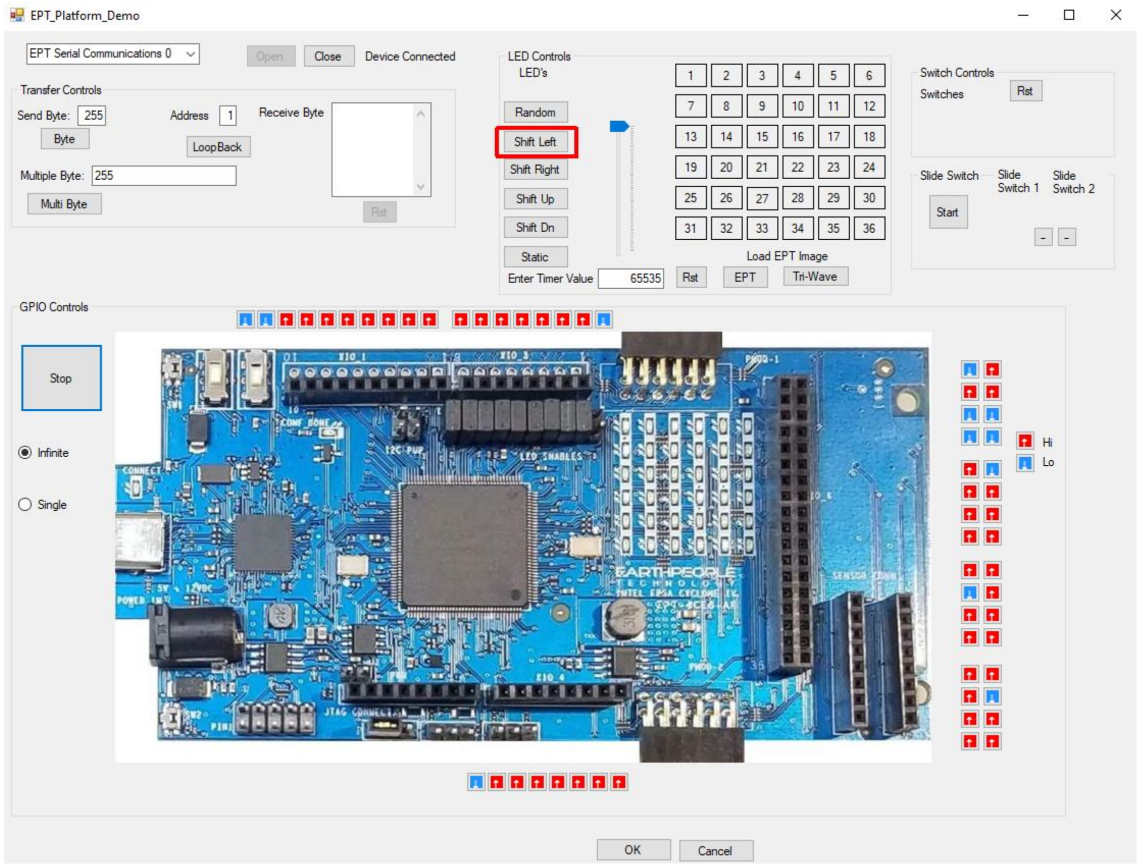
Then, click on the "Start" Button.



Notice that the GPIO's attached to the LEDS in the 6x6 LED Array are now Lo instead of Hi.

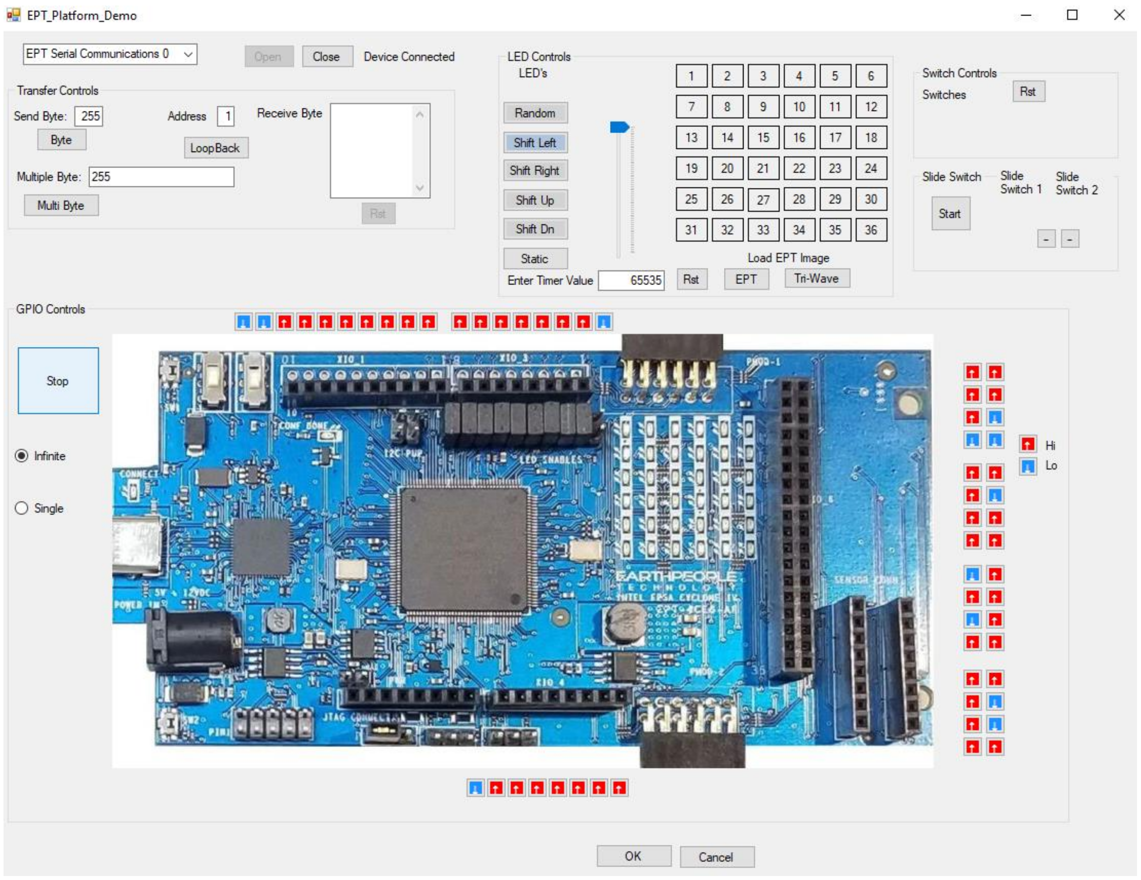


Click on the “Shift Left” button.



Notice the GPIO's connected to the LEDs will change as the image on the 6x6 LED image shifts.

.



EPT Serial Communications 0

Send Byte: 255

Address 1

Multiple Byte: 255

Enter Timer Value 65535

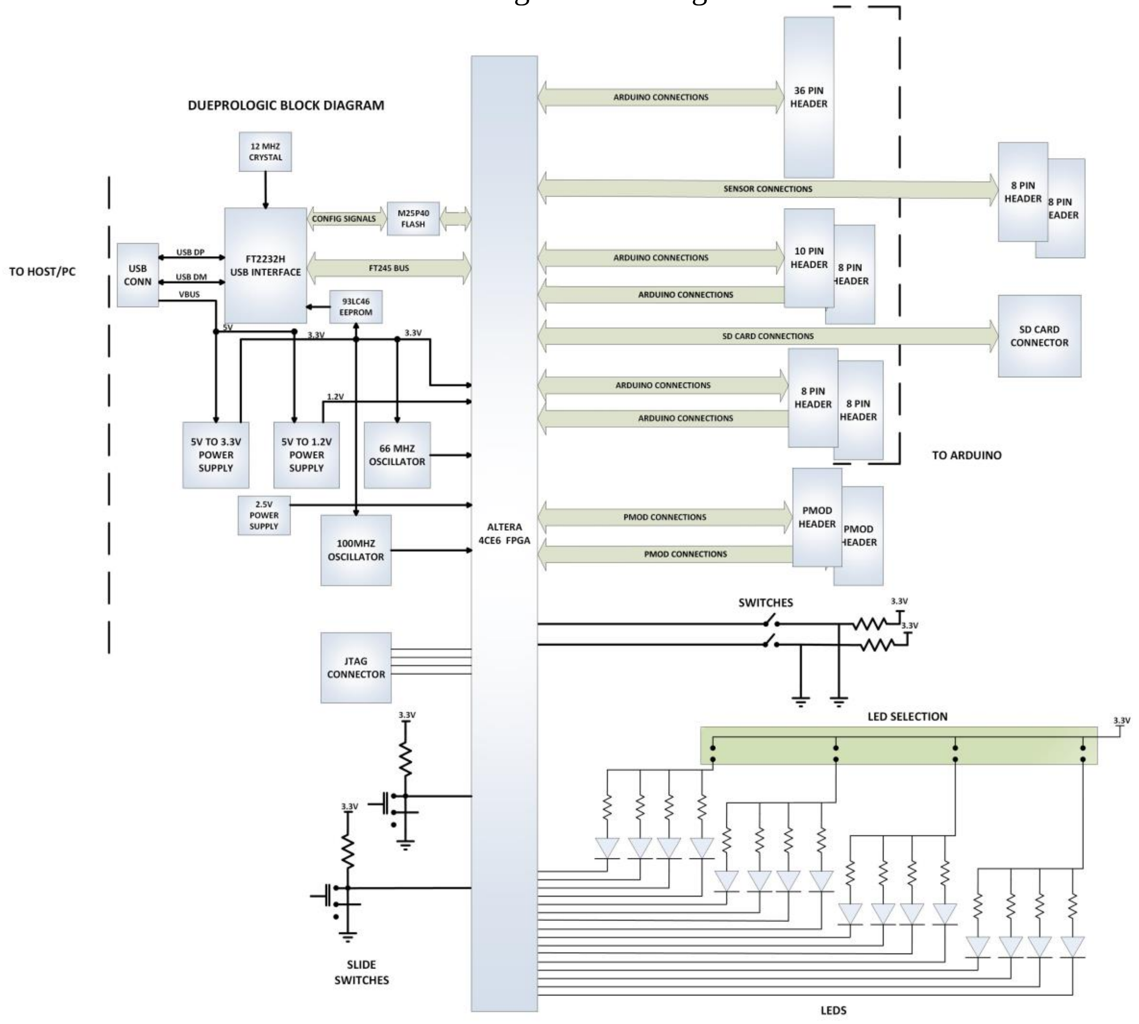
The user can also connect +3.3V or Ground using a jumper to any of the GPIO's in the board edge connector. Be sure NOT to connect +3.3V to a Ground pin or Ground to a +3.3V pin. Consult the Data Sheet for the location of the Inputs on the connectors.

## *1.2 Hardware Description*

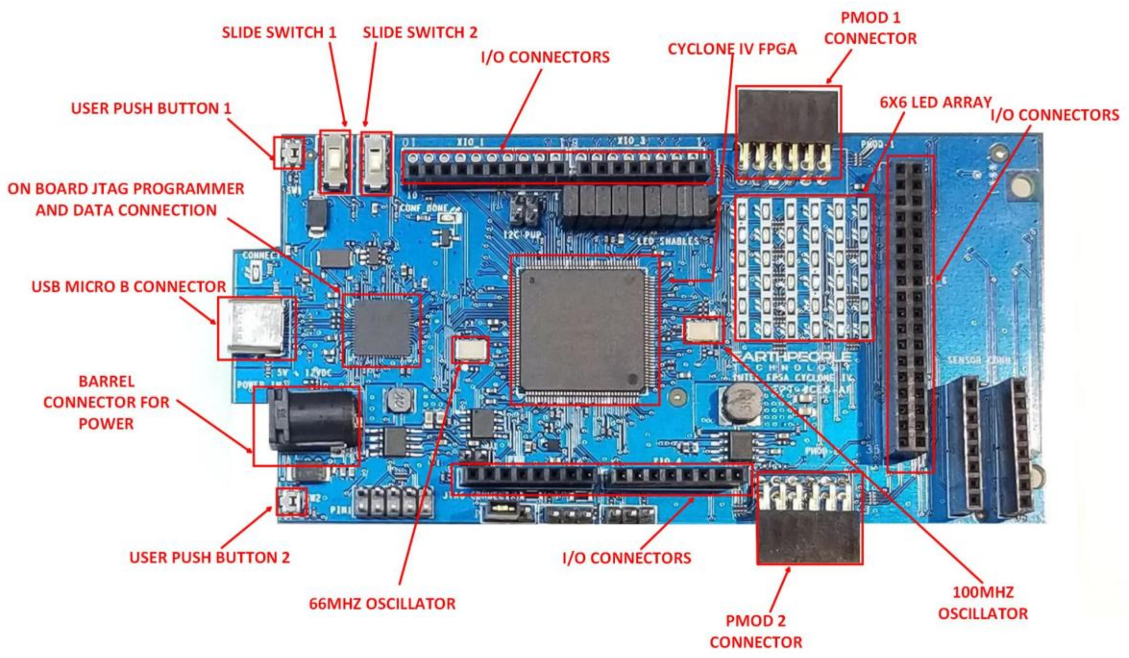
The DueProLogic board is equipped with an Altera EP4CE6E22C8 FPGA; which is programmed using the Altera Quartus Prime software. The FPGA has 6672 Logic Elements and 276480 Total RAM Bits. An on board 66 MHz oscillator is used by the EPT Active Transfer Library to provide serial transfer using the UART method. Fifty Four I/O's from the FPGA are attached to six separate user connectors. The user connectors are organized to fit the Due Arduino platform. There is a separate 36 pin dual row connector at the rear of the board arranged to mate with standard Bread boards. There are 36 green User LED’s, two Slide Switches and two Push Buttons that are controllable by the user code. The hardware features are as follows.

- Altera EP4CE6 FPGA with 6272 Logic Cells
- Dual Channel High Speed USB FT2232H
- 66 MHz oscillator for driving USB data transfers and users code
- 100MHz oscillator for scaling up/down for users needs
- Standard SD Card interface for memory expansion
- 54 user Input/Outputs (+3.3V only)
- 36 Green LED Array accessible by the user
- Two PCB switches accessible by the user
- Two Slide switches
- Two PMOD Connectors

DueProLogic Block Diagram



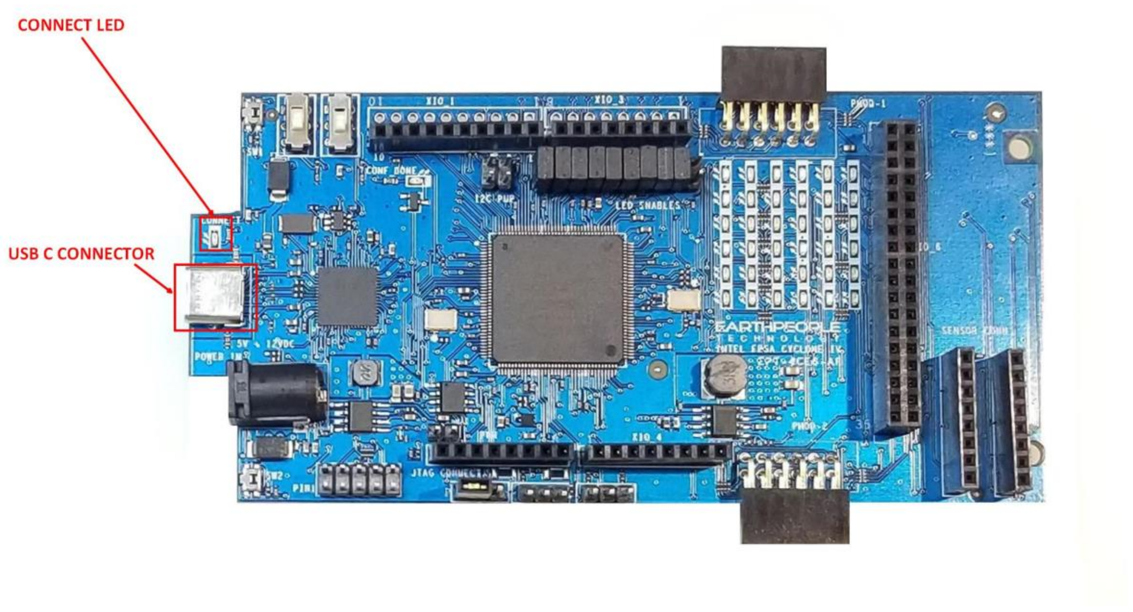
DueProLogic
Hardware



### 1.2.1 Host PC Connection

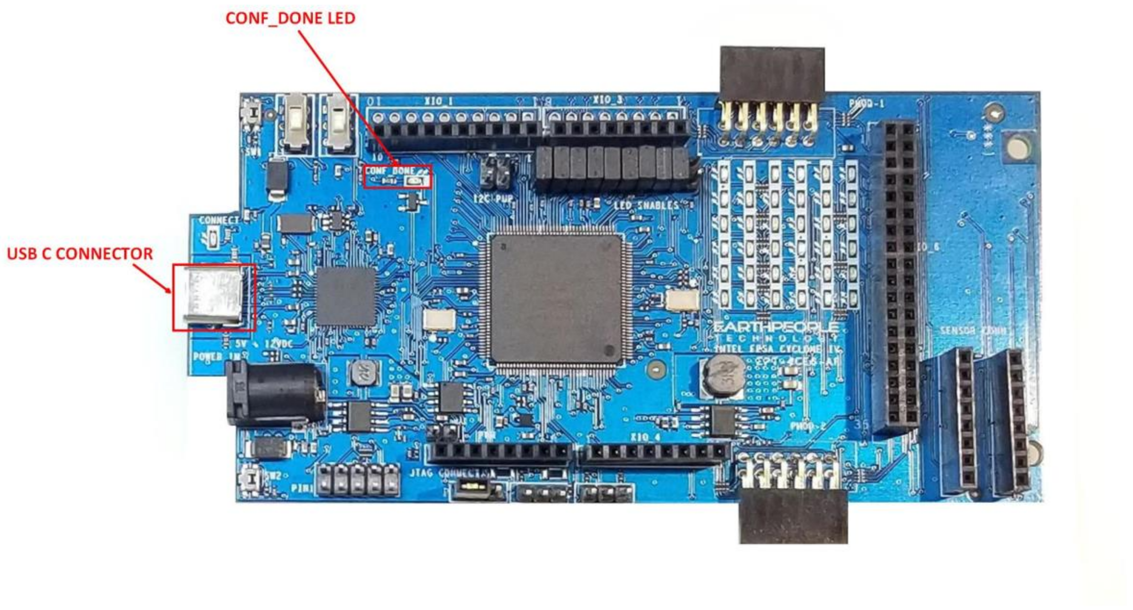
The DueProLogic includes an LED that signifies the connection of the board with the Host PC. The connect LED has the word "CONNECT" in silkscreen next to the LED. This LED will only light up once the Host PC has correctly enumerated the USB device (FT2232HQ chip). When this LED is lit up it can tell the user three things:

- Power has been applied to the DueProLogic via USB
- The FT2232HQ chip is working properly
- The Host PC has found the appropriate driver and will communicate with the DueProLogic

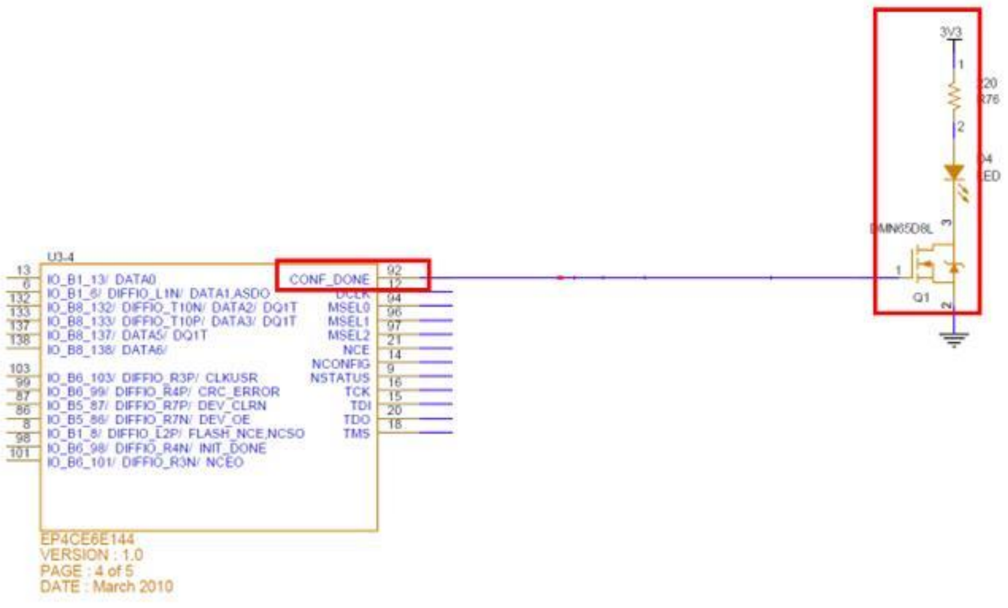


### 1.2.2 CONF DONE LED

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.

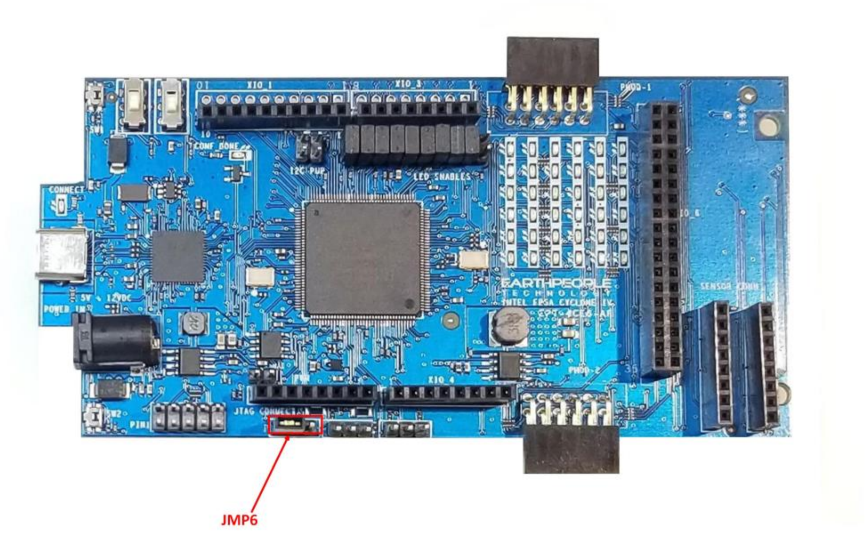

3V3
220
R76
D4
LED
DMN65D8L
Q1
U3-4
CONF_DONE
IO_B1_13/ DATA0
IO_B1_6/ DIFFIO_L1N/ DATA1,ASDO
IO_B8_132/ DIFFIO_T10N/ DATA2/ DQ1T
IO_B8_133/ DIFFIO_T10P/ DATA3/ DQ1T
IO_B8_137/ DATA5/ DQ1T
IO_B8_138/ DATA6/
IO_B6_103/ DIFFIO_R3P/ CLKUSR
IO_B6_99/ DIFFIO_R4P/ CRC_ERROR
IO_B5_87/ DIFFIO_R7P/ DEV_CLRN
IO_B5_86/ DIFFIO_R7N/ DEV_OE
IO_B1_8/ DIFFIO_L2P/ FLASH_NCE,NCSO
IO_B6_98/ DIFFIO_R4N/ INIT_DONE
IO_B6_101/ DIFFIO_R3N/ NCEO
DCLK
MSEL0
MSEL1
MSEL2
NCE
NCONFIG
NSTATUS
TCK
TDI
TDO
TMS
EP4CE6E144
VERSION : 1.0
PAGE : 4 of 5
DATE : March 2010

### 1.2.3 Jumper Settings

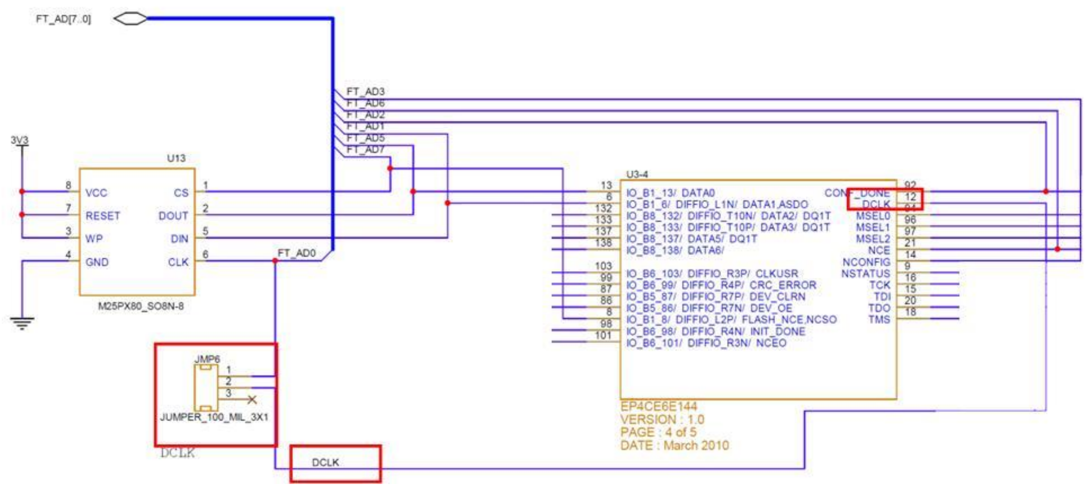
There are six jumper selections on the DPL. These jumper selections will set up the DPL for proper operation and also some optional settings.

#### 1.2.3.1 JMP6

The jumper setting JMP6 is used for proper operation of configuring the on board flash for the FPGA.



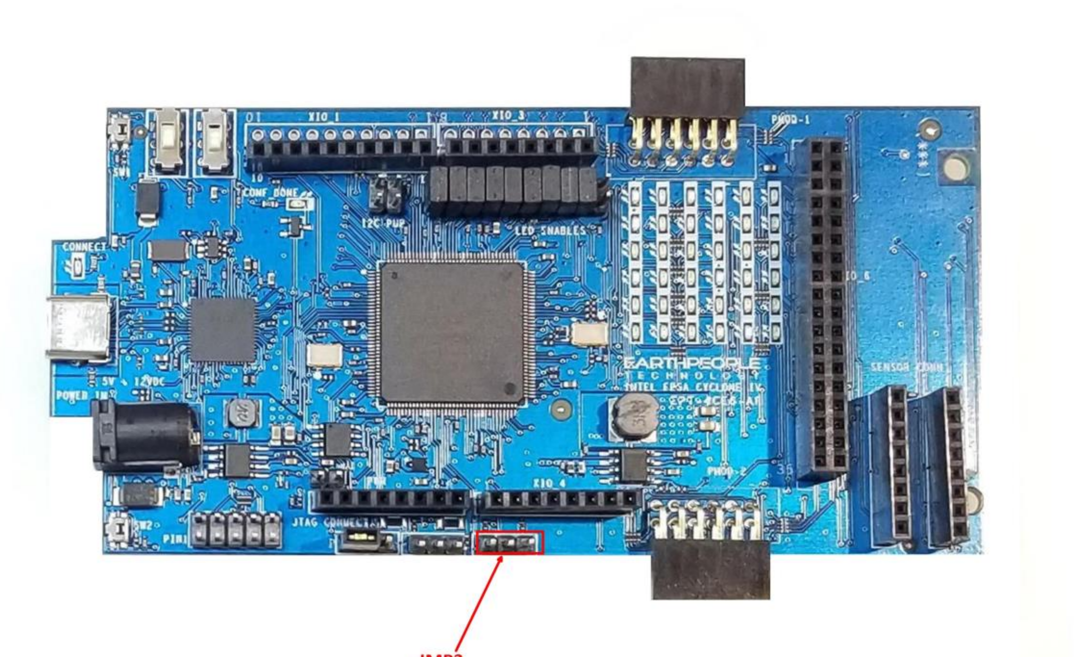
JMP6 is used to allow DCLK to pass from FPGA to the configuration flash. This jumper must have a shunt install on pins 1 and 2 for proper operation of the configuration flash and FPGA.


FT_AD[7..0]
FT_AD3
FT_AD6
FT_AD2
FT_AD1
FT_AD5
FT_AD7
3V3
U13
VCC
RESET
WP
GND
CS
DOUT
DIN
CLK
M25PX80_SO8N-8
FT_AD0
JMP6
JUMPER_100_MIL_3X1
DCLK
DCLK
U3-4
IO_B1_13/ DATA0
IO_B1_6/ DIFFIO_L1N/ DATA1,ASDO
IO_B8_132/ DIFFIO_T10N/ DATA2/ DQ1T
IO_B8_133/ DIFFIO_T10P/ DATA3/ DQ1T
IO_B8_137/ DATA5/ DQ1T
IO_B8_138/ DATA6/
IO_B6_103/ DIFFIO_R3P/ CLKUSR
IO_B6_99/ DIFFIO_R4P/ CRC_ERROR
IO_B5_87/ DIFFIO_R7P/ DEV_CLRN
IO_B5_86/ DIFFIO_R7N/ DEV_OE
IO_B1_8/ DIFFIO_L2P/ FLASH_NCE,NCSO
IO_B6_98/ DIFFIO_R4N/ INIT_DONE
IO_B6_101/ DIFFIO_R3N/ NCEO
CONF_DONE
DCLK
MSEL0
MSEL1
MSEL2
NCE
NCONFIG
NSTATUS
TCK
TDI
TDO
TMS
EP4CE6E144
VERSION : 1.0
PAGE : 4 of 5
DATE : March 2010

When using the JTAG only operation of the FPGA (using the *.sof file to program the FPGA), this jumper should be removed.

### 1.2.3.2 JMP3

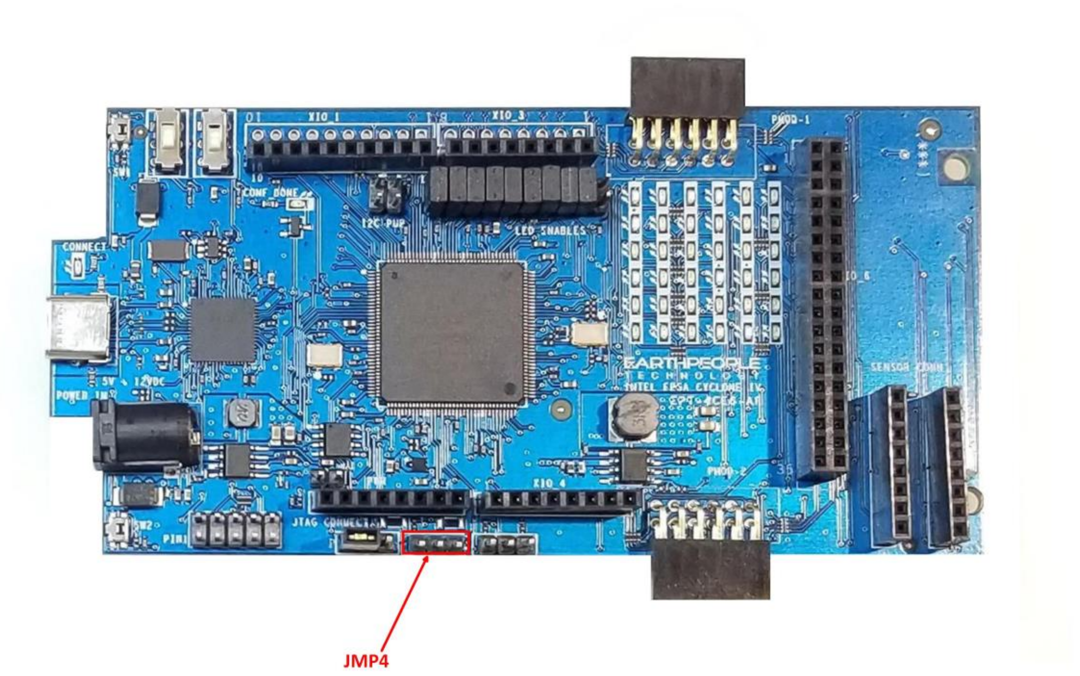
JMP3 provides a pullup/pulldown to the "NCONFIG" signal of the FPGA. When using the configuration flash to configure the FPGA,



Do Not Add a jumper. When programming the FPGA using JTAG, connect a jumper between pins 1 and 2 to add a pullup to the “NCONFIG” signal.

#### 1.2.3.3 JMP4

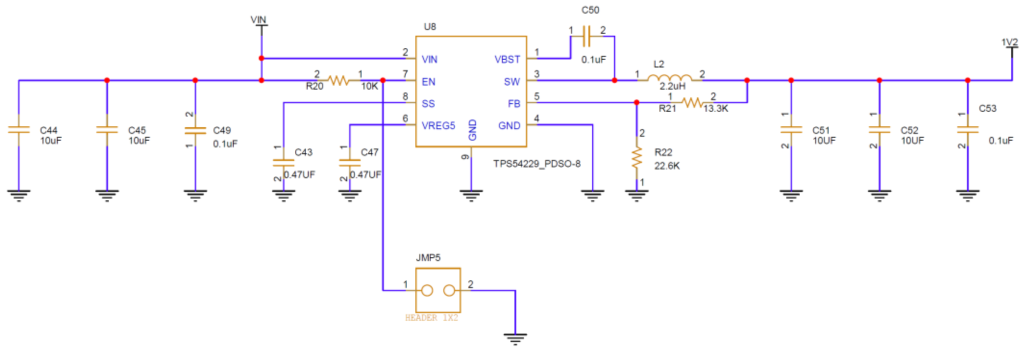
JMP4 provides a pullup/pulldown to the “CONF_DONE” signal of the FPGA. When using the configuration flash to configure the FPGA,

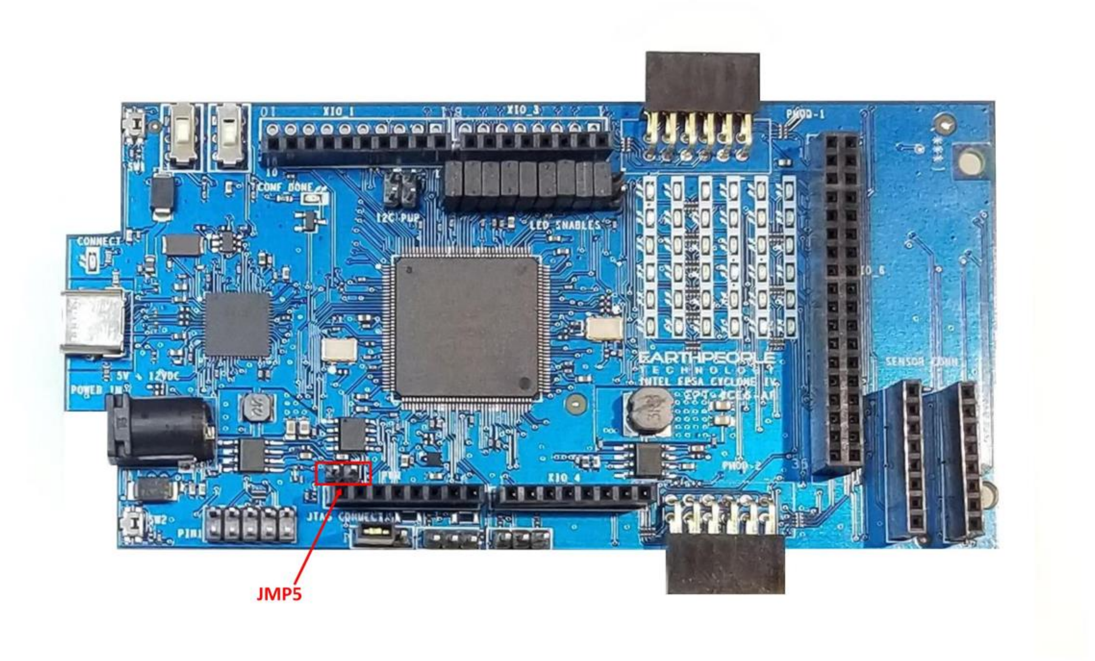


Do Not Add a jumper. When programming the FPGA using JTAG, connect a jumper between pins 1 and 2 to add a pullup to the "CONF_DONE" signal.

### 1.2.3.4 JMP5

JMP5 provides a pulldown on the EN signal of the 1.2V Power Supply.



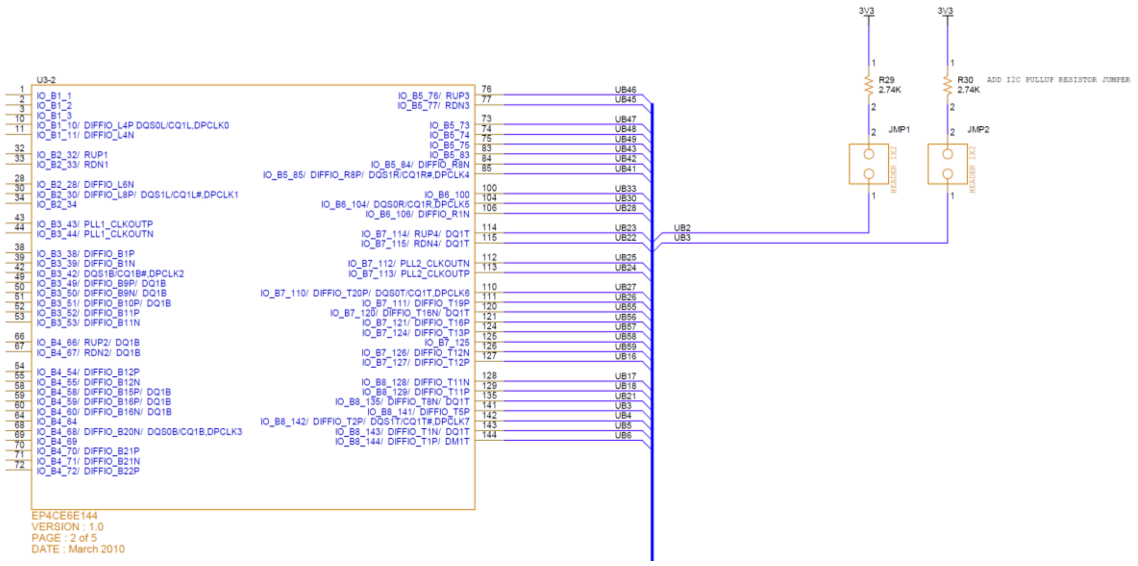


This jumper should remain unconnected for proper operation of the FPGA.

Only use this jumper if the configuration flash file has become corrupted and the only method to load a new configuration flash file is to disable the FPGA. For further instructions on the use of this jumper please email Earth People Technology directly.

### 1.2.3.5 JMP1 and JMP2

JMP1 and JMP2 are used to add pullups to the I2C communication signals.

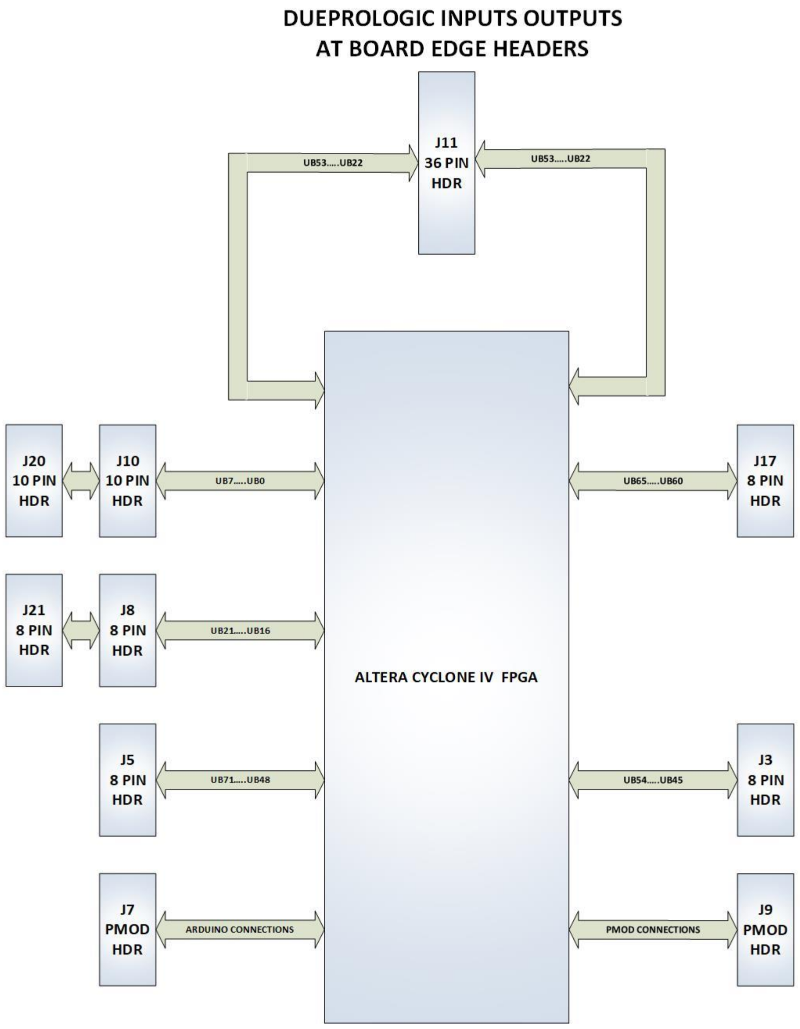


If signals UB2 and UB3 are needed for I2C signals SDA and SDC, connect a shunt to the jumpers to provide the 2.7K Ohm pullup resistors.

### 1.2.4 Inputs and Outputs

**Do not connect a 5Volt device to the DPL**

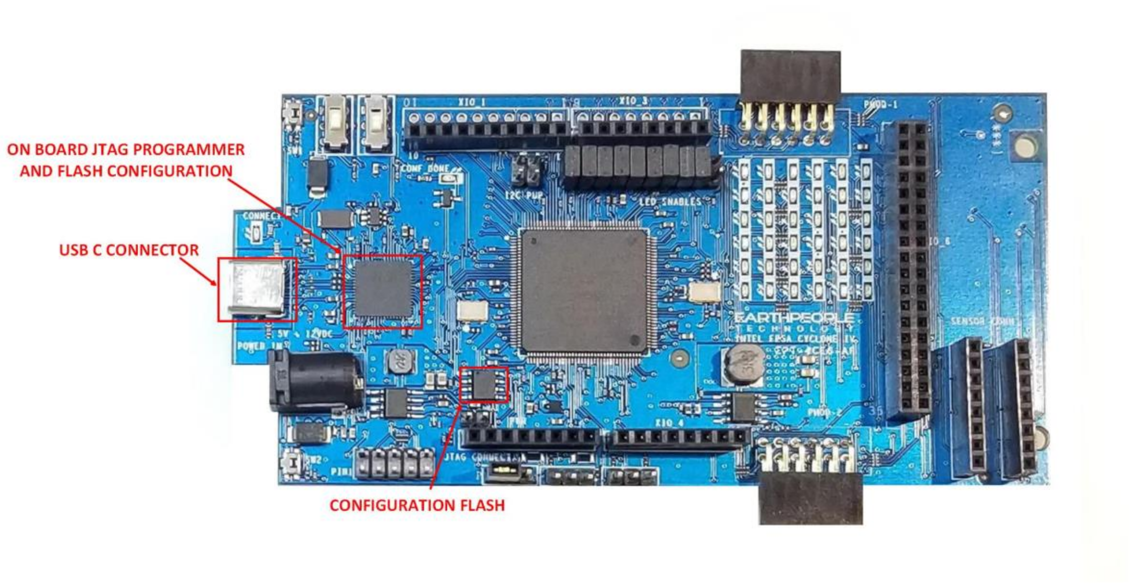
There are 54 Inputs/Outputs which are +3.3Volt only. The FPGA I/O's are organized as separate pins and connect to the Arduino connectors and separate connectors. Each I/O must be defined as input or output in the user code. Each pin of the connectors can behave as either input or output. Refer to the DueProLogic Data Sheet for exact pinout location.

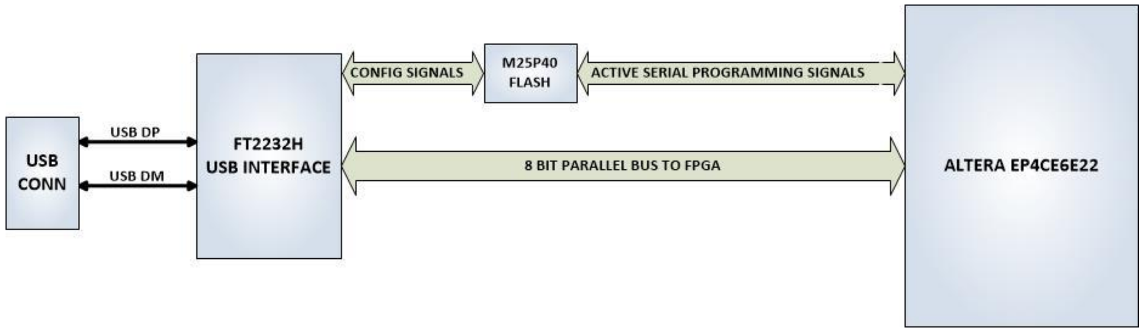


Each Header includes a Power and Ground connection which allows an easy way to power sensors or bread boards directly from the DPL.

### 1.2.5 FPGA Configuration

The Cyclone IV FPGA is configured for operation when the power is applied to the board. A dedicated Configuration Flash chip is included on the DueProLogic for the purpose of configuring the FPGA on power up. The DPL uses the second channel of the FT2232H chip as a dedicated Flash programming port. The Configuration Flash can be



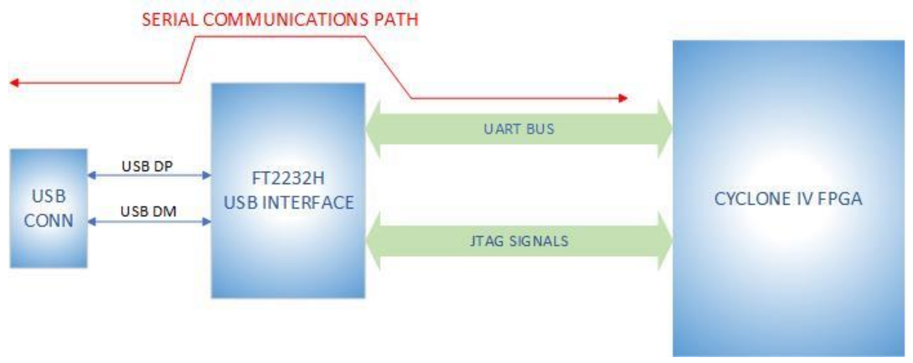


programmed directly from Quartus Prime by using the EPT-Blaster driver. Follow the instructions in the “EPT Drivers” section of this manual. Follow the instructions in the section “Setting up the Project and Compiling” to program the flash on the DPL.

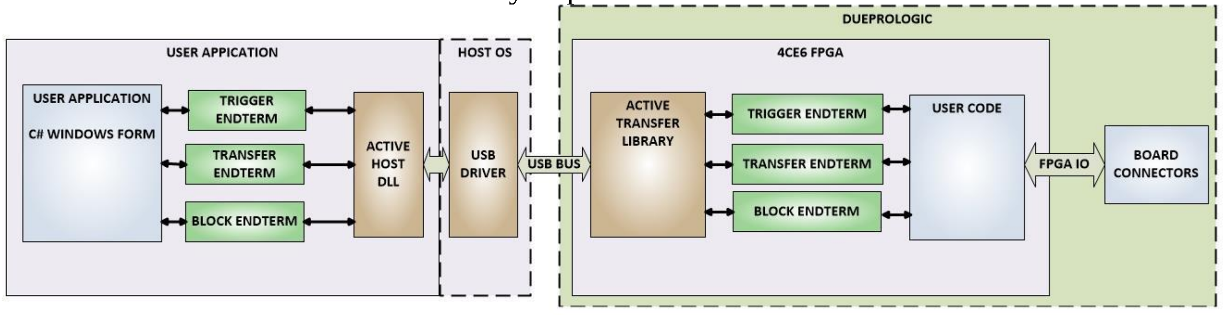
### 1.2.6 FT2232H Dual Channel USB to Serial Chip

The DueProLogic contains an FTDI 2232H dual channel high speed (480 Mb/s) USB to FIFO (first in-first out) integrated circuit to interface between the Host PC and the FPGA. The FT2232H provides a means of data conversion from USB to serial/ parallel data and serial/parallel to USB for data being sent from the FPGA to the PC. Channel A is configured as a Flash Configuration bus and Channel B is configured as a serial bus. FPGA Programming commands are transmitted via the channel A interface.  Channel B has one dual port 4Kbyte FIFO for transmission from Host PC to the FPGA, it also has one dual port 4Kbyte FIFO for receiving data from the FPGA to the Host PC.

The serial communications pathway is used to send and receive data to/from the Host PC. The pathway comprises the USB-C Connector, USB to Serial chip and the Cyclone IV FPGA. The USB driver from FTDI is required for the FT2232H chip. Please see the section “EPT Drivers” for instructions on driver installation.



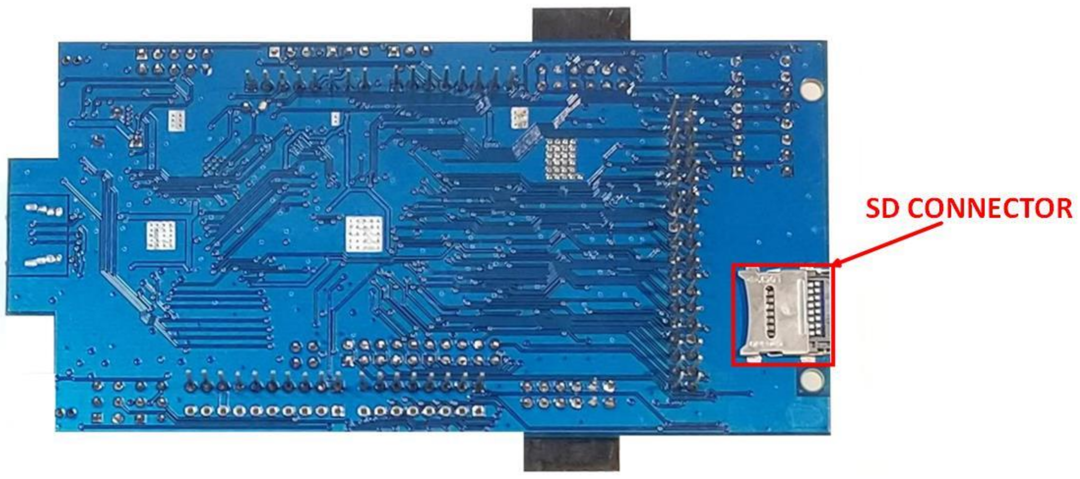
The serial communications pathway between host and FPGA is transparent to the user. The Active Host API allows users to easily implement a data read/write in software.

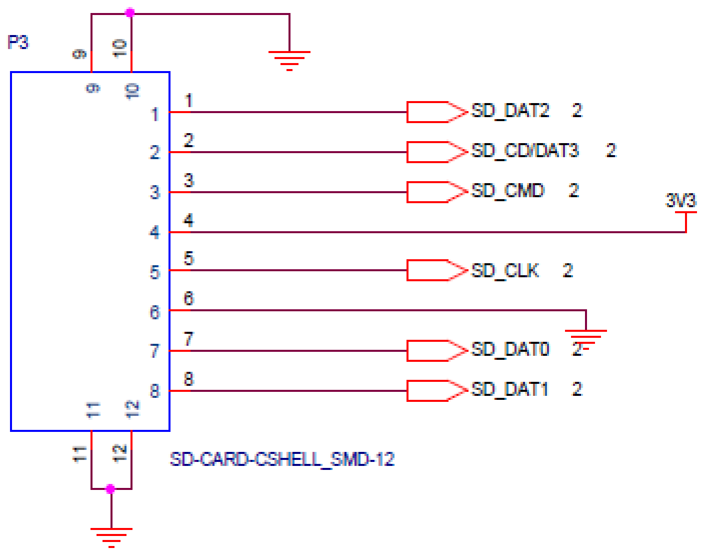


Please see the section “Active Host Application” on instructions for how to use the Active Host API to build an application for data read/write between Host and DPL.

### 1.2.7 SD Card Interface

The DPL includes a standard SD Card Interface. The SD connector is on the bottom of the DPL. This interface allows the user to add expansion memory or the standard SD interface.





SD Card Connector

### 1.2.8 SD Card Protocol

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

#### 1.2.8.1 References

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

#### 1.2.8.2 Compatibility

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

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

#### 1.2.8.3 Protocol

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

There are three types of signal:

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

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

#### 1.2.8.4 Physical layer

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

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

#### 1.2.8.5 Framing

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

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

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

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

#### 1.2.8.6 Commands and Responses

A command or response has the following format:

| Bit | # | Value | Name |
|---|---|---|---|
| 47 | 1 | 0 | Start bit |
| 46 | 1 | 1 for commands, 0 for responses | Transmitter bit |

| | | | |
|---|---|---|---|
| 45-40 | 6 | | Command ID |
| 39-8 | 32 | | Argument |
| 7-1 | 7 | | CRC |
| 0 | 1 | 1 | Stop bit |

The CRC is a 7 bit CRC with polynomial $x^7 + x^3 + 1$. A table driven form can be found in the Linux kernel under lib/crc7.c. Bitwise forms may be generated using pycrc with the parameters
`--width=7 --poly=9 --reflect-in=0 --reflect-out=0 --xor-out=0 --xor-in=0 --generate c --algorithm=bit-by-bit-fast`
such as

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

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

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

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

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

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

Some examples are:

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

Fields:

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

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

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

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

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

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

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

### 1.2.8.7 Data

Data has the following format:

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

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

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

PENDING: Add an example data message with CRC.

### 1.2.8.8 Handover

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

### 1.2.8.9 Initialisation

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

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

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

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

ready then repeat the CMD55/ACMD41 until it is

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

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

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

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

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

MMC cards are initialised using a similar but different method.

An example flow captured from Linux on a SC2440 is:

| Phase | Command | Response | Notes |
|---|---|---|---|
| CMD0 | 400000000095 | None | |
| CMD55 | 770000000065 | 370000012083 | |

| | | | |
|---|---|---|---|
| ACMD41 SEND_OP_COND | 69001000005F | 3F00FF8000FF | Card is busy |
| CMD55 | 770000000065 | 370000012083 | |
| ACMD41 | 69001000005F | 3F00FF8000FF | Card is still busy |
| CMD55 | 770000000065 | 370000012083 | |
| ACMD41 SEND_OP_COND | 69001000005F | 3F80FF8000FF | Card is ready |
| CMD2 ALL_SEND_CID | 42000000004D | 3F1D4144534420202010A0400BC10088ADFF | |
| CMD3 SEND_RELATIVE_ADDR | 43000100007F | 03B368050019 | RCA of 0xB368 |

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

#### 1.2.8.10 Reading

Once initialised reading from a card is straight forward.

To read a single page:

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

To read consecutive pages:

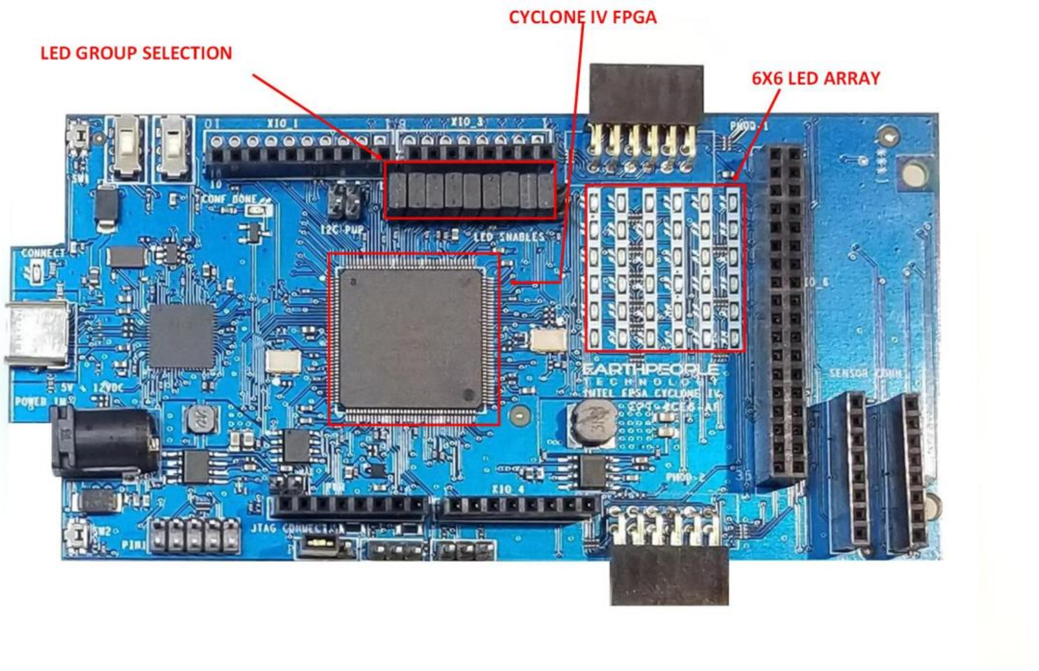
- Send CMD18 READ_MULTIPLE_BLOCK with the starting offset as the argument
- Receive as many blocks as you want on the DAT lines
- Send CMD12 STOP_TRANSMISSION once the done

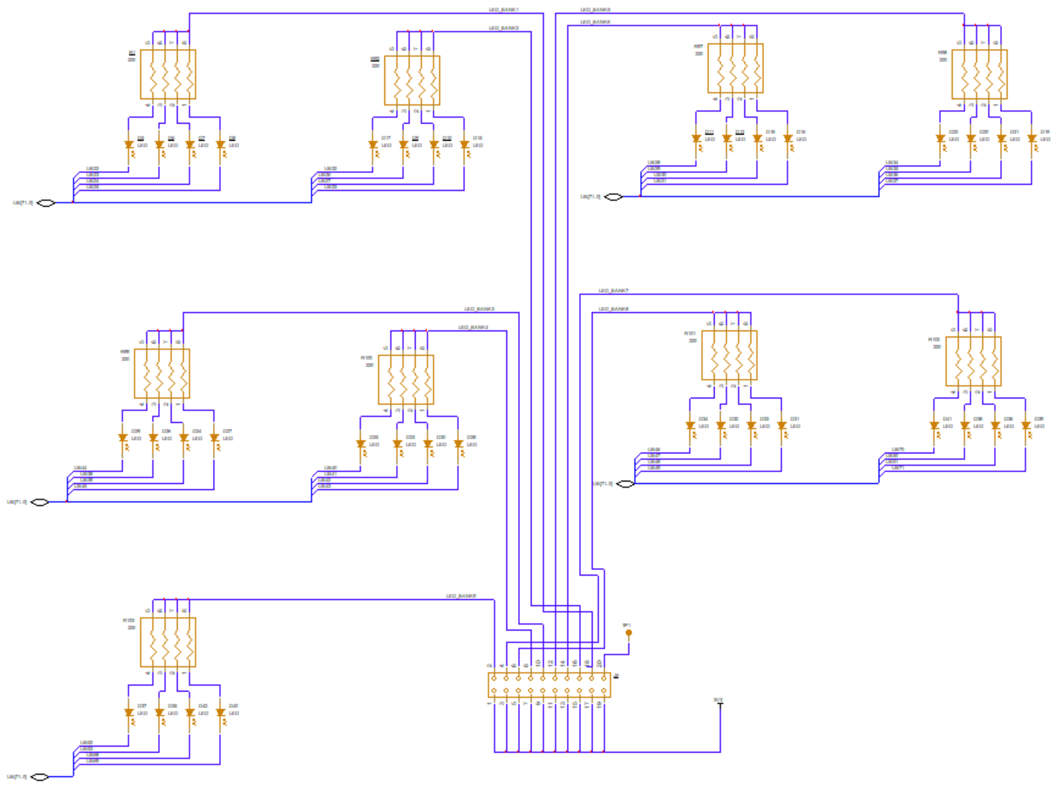
Note that there will be a response to these commands and the response may be interleaved with the data. See Figure 3-3: (Multiple) Block Read Operation for more information.

On SDHC cards the offset is in terms of 512 byte blocks. On SD cards the offset is in bytes and the number of bytes received depends onCMD16 SET_BLOCKLEN

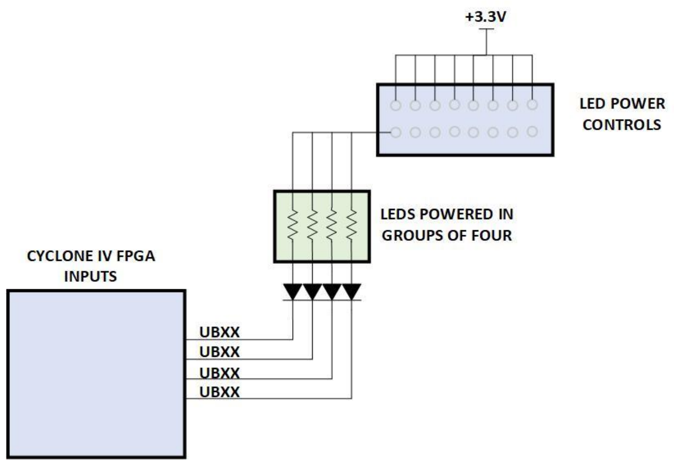
### 1.2.9 LEDs

The DueProLogic includes a 6x6 Green LED array. Each LED is sinked to an individual pin on the FPGA. Each LED is current limited to 6mA. The total current consumed for all 36 LEDs is 216mAs. The FPGA can easily sink this current. So, individually sinking all 36 LEDs makes easy control for User Code. The DueProLogic also contains a method to turn on/off the LEDs in four unit blocks. A jumper is used to control the state of each LED block.



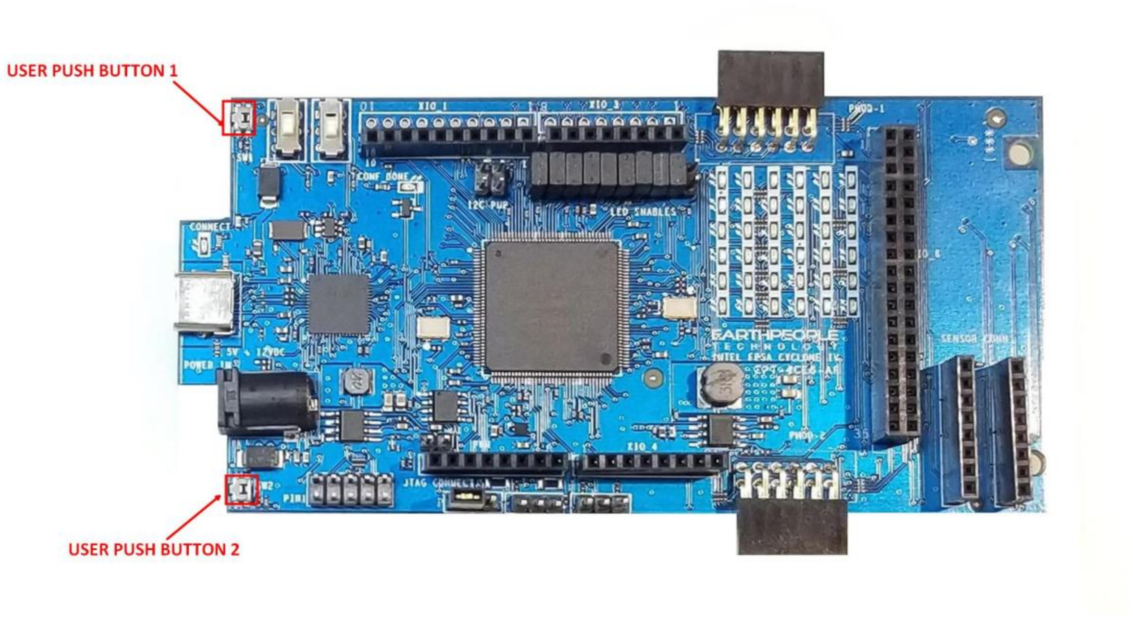


LED Power Control allows a group of four LEDs to be powered by one jumper. When the jumper is connected, the LEDs are powered by +3.3V. When disconnected, the LEDs will not light up and the IO pin in the appropriate Board Edge Connector can be used without an LED.

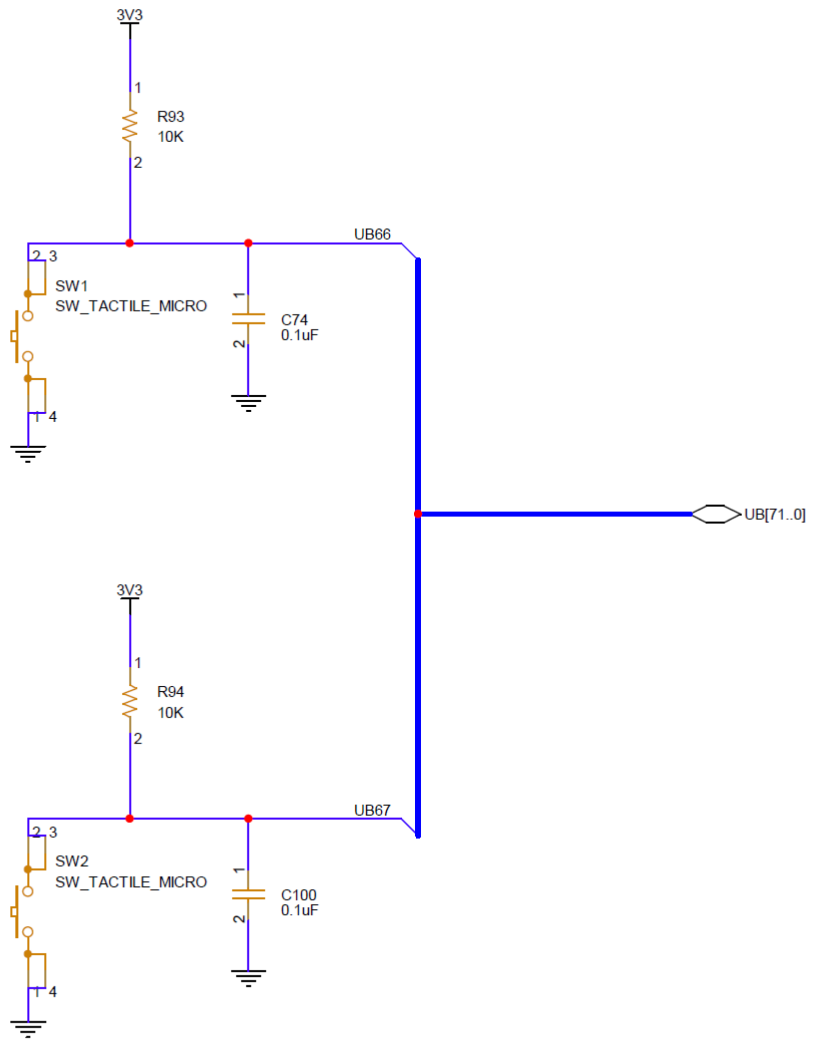


### 1.2.10 Pushbuttons

The DueProLogic includes two pushbuttons that can be used in a way the user would like. They are both attached to separate debounce circuits. These pushbuttons were designed to use as little PCB are as possible. The switch is engaged by the plunger which is directed off the PCB.



The switches will momentarily provide a ground connection to the appropriate pin on the FPGA. When Idle, the signals are pulled high through a 10K Ohm resistor.



Push Button Switches

| Component | Net Name | Pin on Cyclone IV |
|---|---|---|
| SW1 | UB66 | 91 |
| SW2 | UB67 | 90 |

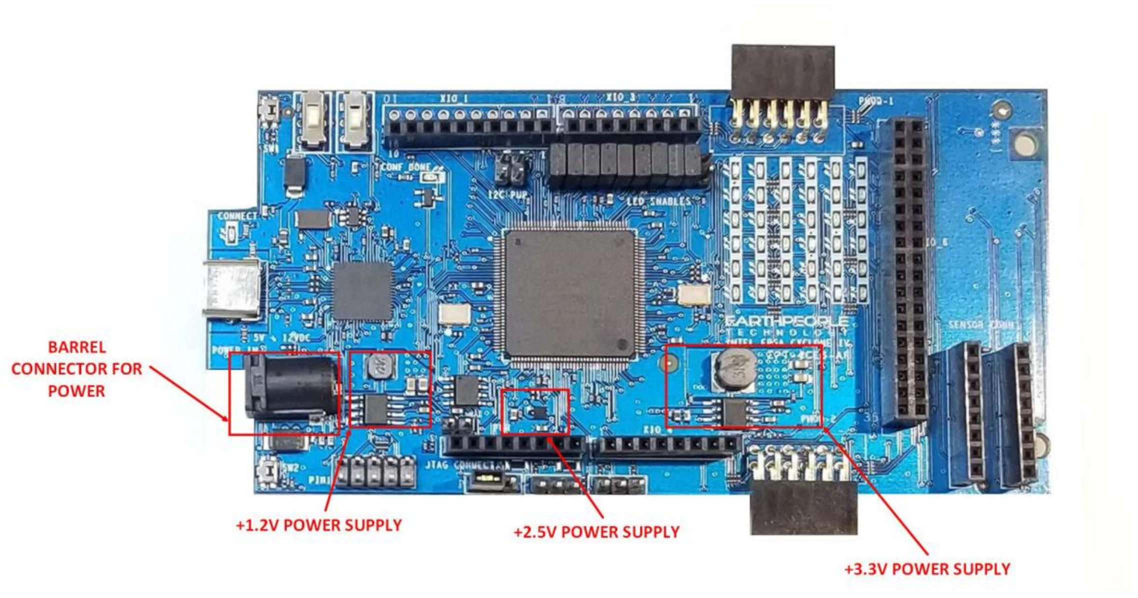
### 1.2.11 Power Options

There are two power options for the DueProLogic

- USB-C Connection to a PC (+5VDC)
- Barrel Connector (+5VDC to +12VDC)

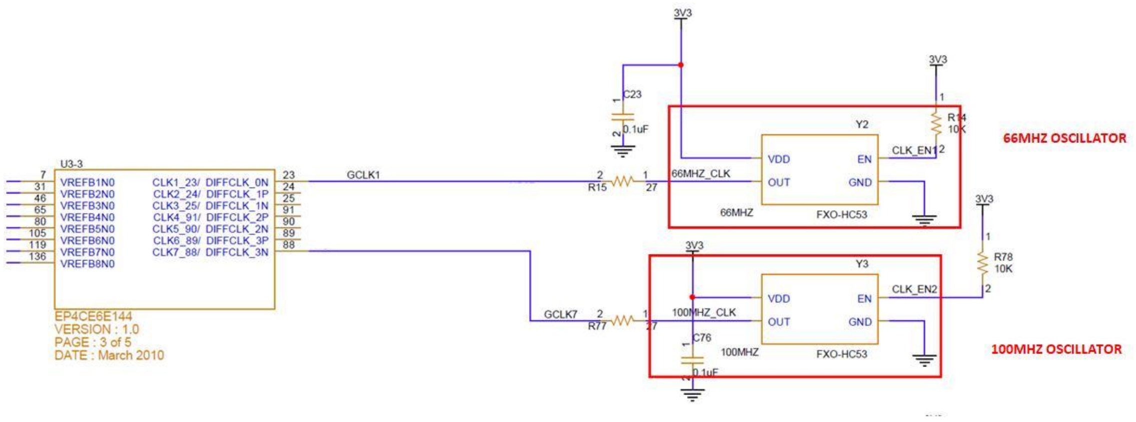
The USB Micro B Connector is a the standard connector specified by the USB SIG.

The Barrel Connector is the located on the front of the board. It has 2.0mm inner diameter and 5.5mm outer diameter. The inner male connection is the positive connection and the outer connection is the ground. This connection can accept between +5VDC and +12VDC power. However, great care must be used when using this connection. This power is applied to several pins on the I/O headers. Consult the Data Sheet for these connections.



### 1.2.12 Clock Domains

There are two clock domains external to the Cyclone IV FPGA, 66 MHz and 100 MHz. The 66 MHz oscillator is a +3.3VDC device that provides a high speed clock to the FPGA. It is a CMOS device that provides a stable 66 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.



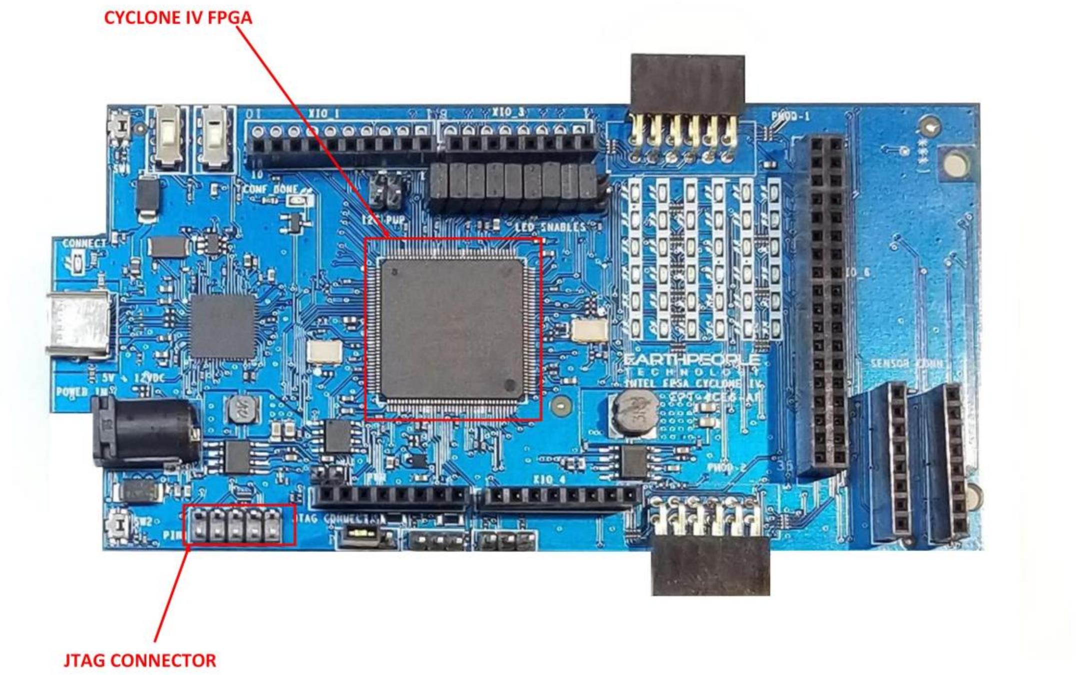
The 100MHz oscillator is a +3.3VDC device that provides a stable source for high speed clocking. This oscillator can be used for higher speed functions such as fast communications, memories and control.

Both oscillators have an enable signal that are individually controlled by the FPGA. The oscillators are operational when the enable signal is high. The oscillators enable signals each have 10K pullups to keep them on at all times.

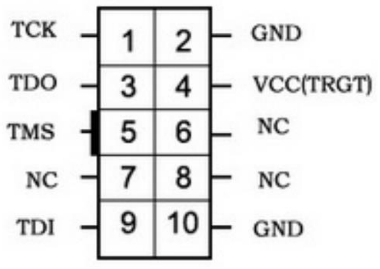
| **Component** | **Signal Name** | **FPGA Pin** |
|---|---|---|
| Y2 (66MHz Oscillator) | 66MHZ_CLK | 23 |
| Y3 (100MHz Oscillator) | 100MHZ_CLK | 88 |

### 1.2.13 JTAG Header

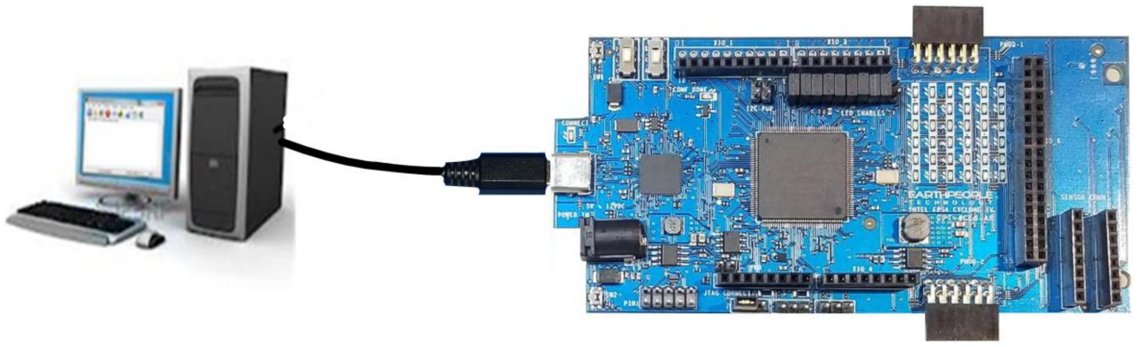
The DueProLogic can be programmed by writing to the on board configuration flash or by directly accessing the FPGA JTAG connection.



Connector J13 provides access to the FPGA JTAG connections. The pinout follows the USB Blaster arrangement which is standard on all Intel/Altera devices. Be sure to follow the pin 1 orientation.
This connector uses the standard Altera Blaster connector pinout.



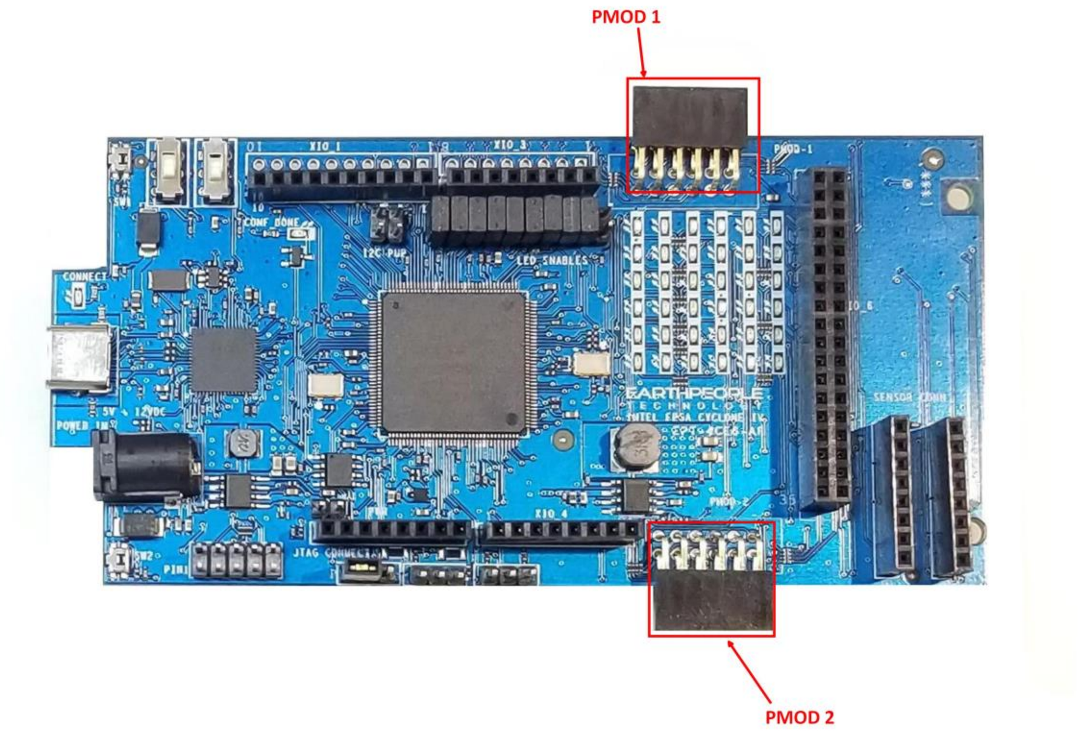
The VCC(TRGT) is set to +3.3V on the DueProLogic. There are no jumper settings to make in order to program the Cyclone IV FPGA. Just connect a compatible Blaster to the connector and the PC, then use the Quartus software to program the FPGA.



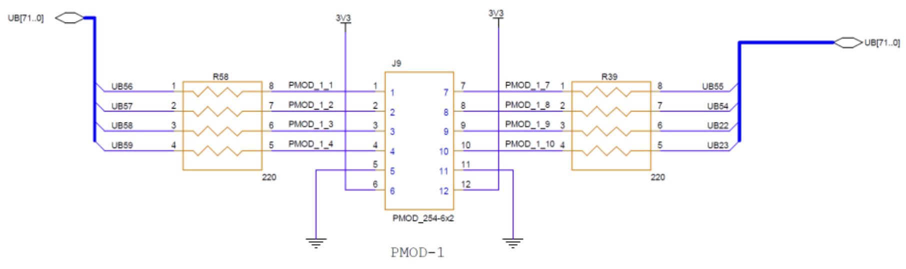
Connect an Intel/Altera compatible programmer to J13, and the FPGA can be programmed directly from Quartus Prime Lite.

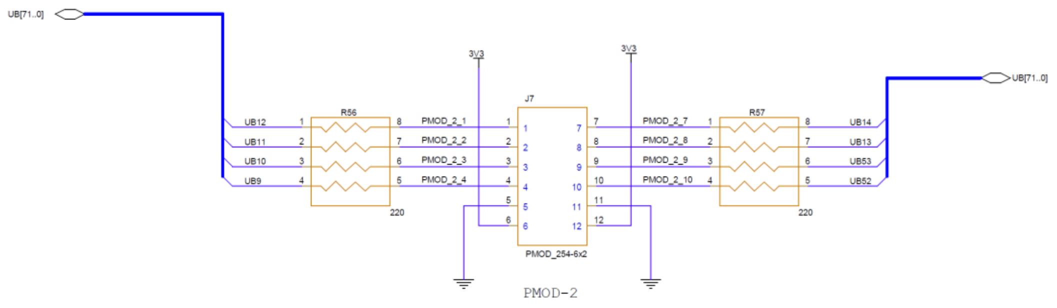
### 1.2.14 PMOD Connectors

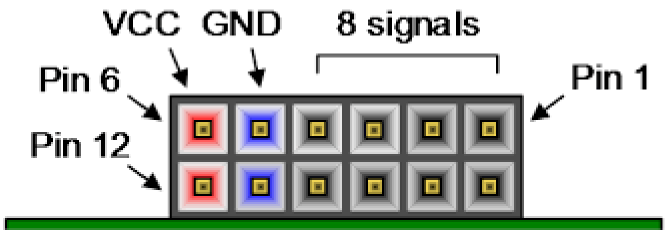
The DueProLogic includes two PMOD Connectors. These two connectors are located towards the rear of the board.



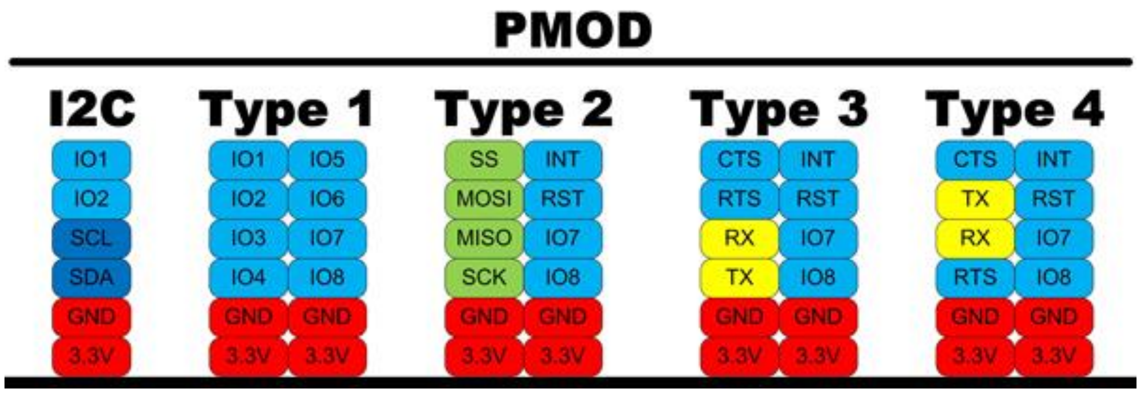
The PMOD pinouts follows the standard pinout. Pin 1 is located in the upper right when facing the connector.







VCC is +3.3V and the Inputs and Outputs of the 8 signals are +3.3V only. The eight I/O's are connected directly to FPGA pins and can be designated as any communications standard.

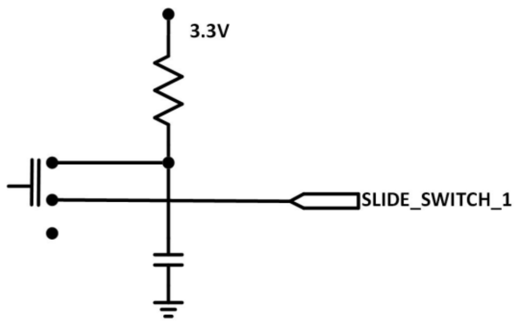


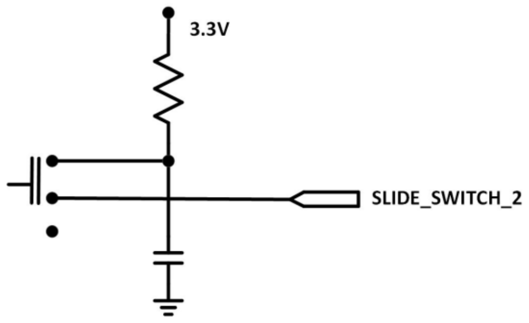
The PMOD have the following connections to the Cyclone IV chip:

| PMOD Pin Number | Signal Name | Cyclone IV Pin Number |
|---|---|---|
| 1-1 | UB56 | 121 |
| 1-2 | UB57 | 124 |
| 1-3 | UB58 | 125 |
| 1-4 | UB59 | 126 |
| 1-7 | UB55 | 120 |
| 1-8 | UB54 | 119 |
| 1-9 | UB22 | 115 |
| 1-10 | UB23 | 114 |
| 2-1 | UB12 | 67 |
| 2-2 | UB11 | 66 |
| 2-3 | UB10 | 53 |
| 2-4 | UB9 | 52 |
| 2-7 | UB14 | 55 |
| 2-8 | UB13 | 54 |
| 2-9 | UB53 | 69 |
| 2-10 | UB52 | 70 |

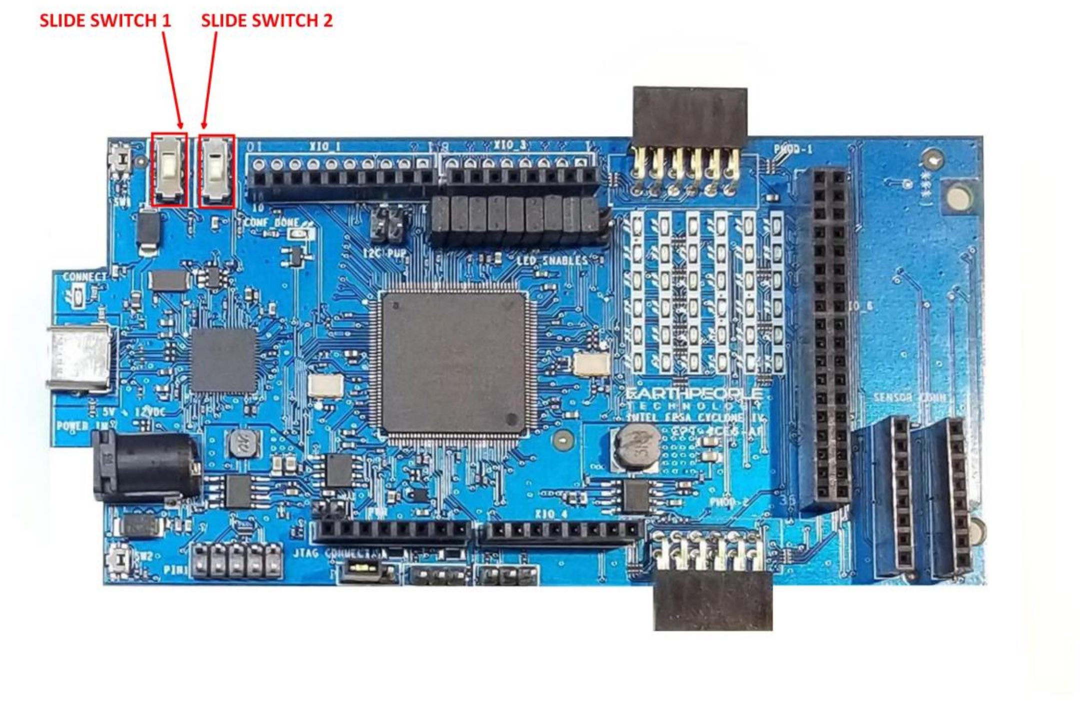
### 1.2.15 Slide Switches

The DueProLogic includes two slide switches. Both are full contact switches. They include a 1uF cap to ground to debounce both switches.





The slide switches provide a connection to either +3.3V in position 1or Ground in position 2.



| Component | Net Name | Pin on Cyclone IV |
|---|---|---|
| SW3 | SLIDE_SWITCH_1 | 91 |
| SW4 | SLIDE_SWITCH_2 | 90 |

## 1.3 *FPGA Active Host Development*

The DueProLogic comes complete with step by step instructions on building an entire communications system from FPGA to Windows Host. Using the EPT Active Host SDK provides an easy to use programming interface. The tools required are the free Visual Studio IDE and the free Quartus Prime.

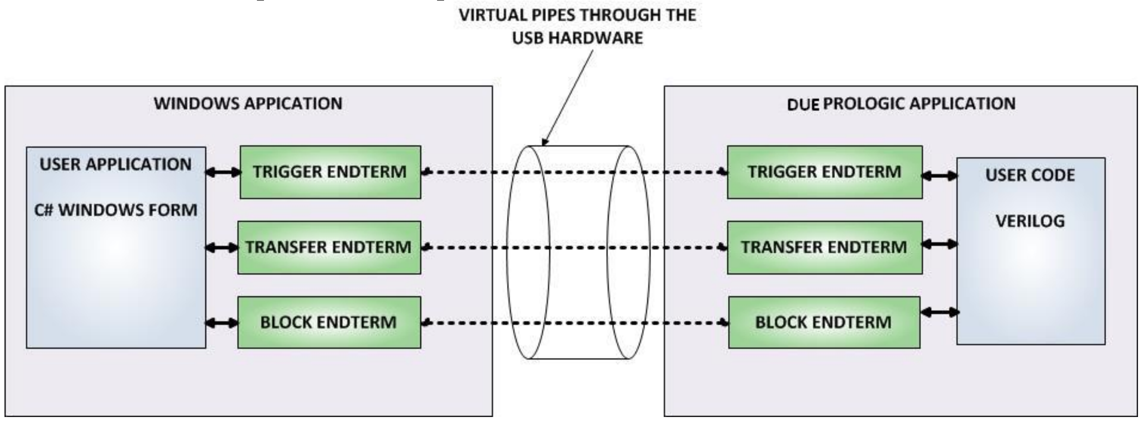
The DueProLogic also comes with instructions on how to build a communications systems between the Arduino Due and the DPL. The DVD included in the kit has all source files, compiled projects and user manuals to assist the user in using the software. In order to assemble a full communications system between the PC and the DPL, the following software items are required:

- Active Host SDK – This library provides the communication mechanism  on the PC
- Active Transfer library – This library is included and synthesized into the users code on the FPGA

These two software components provide a communication mechanism by talking to each other over an item called “EndTerms”. EndTerms is the name given to the virtual “pipes” provided by the above libraries. The next two sections provides a brief introduction to EndTerms. Following sections will provide greater detail on using EndTerms.

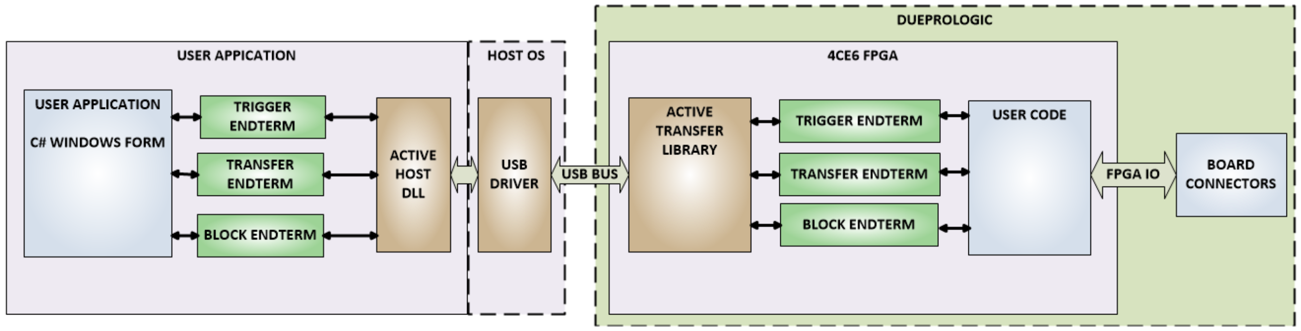
## *1.4 Active Host EndTerms*

The Active Host SDK is provided as a set of functions, including the COM Port functionality, which easily interfaces to application software written in C#, C++ or C. It runs on the PC and provides transparent connection

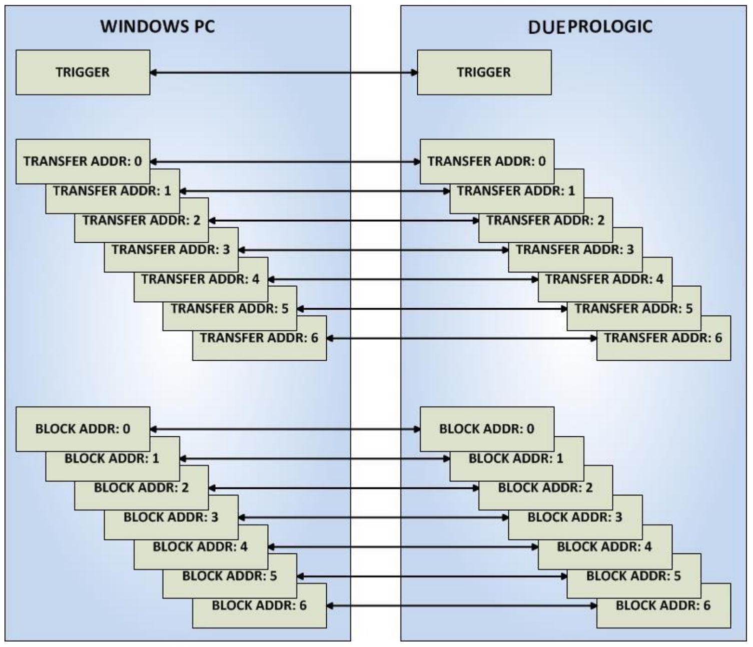


from PC/Windows application code through the USB driver to the user FPGA code. The user code connects to “Endterms” in the Active Host SDK. These Host “Endterms” have complementary HDL “Endterms” in the Active Transfer Library. Users have seamless bi-directional communications at their disposal in the form of:

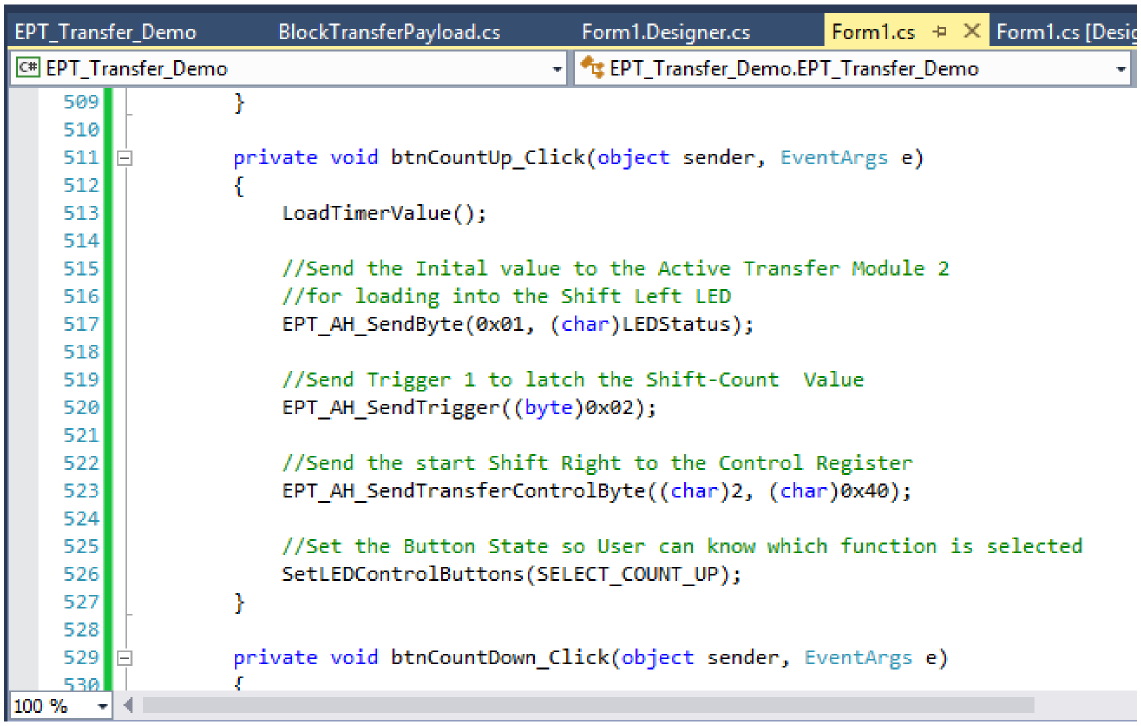
- Trigger Endterm
- Transfer Endterm
- Block Endterm



User code writes to the Endterms as function calls. Just include the address of the individual module (there are eight individually addressable modules of each Endterm).

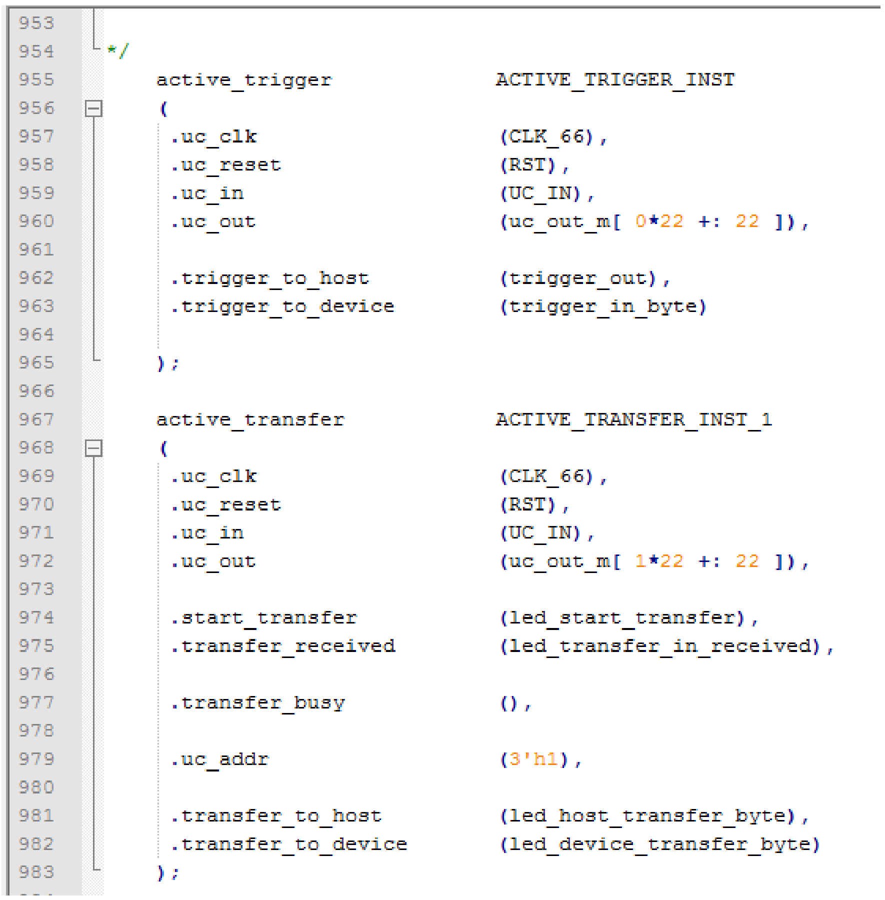


Immediately after writing to the selected Endterm, the value is received at the HDL Endterm in the FPGA.



```
        }

        private void btnCountUp_Click(object sender, EventArgs e)
        {
            LoadTimerValue();

            //Send the Inital value to the Active Transfer Module 2
            //for loading into the Shift Left LED
            EPT_AH_SendByte(0x01, (char)LEDStatus);

            //Send Trigger 1 to latch the Shift-Count  Value
            EPT_AH_SendTrigger((byte)0x02);

            //Send the start Shift Right to the Control Register
            EPT_AH_SendTransferControlByte((char)2, (char)0x40);

            //Set the Button State so User can know which function is selected
            SetLEDControlButtons(SELECT_COUNT_UP);
        }

        private void btnCountDown_Click(object sender, EventArgs e)
        {
```

The above is a code sample from a C# Windows Form. You can see functions that write a byte to the FPGA (EPT_AH_SendByte(0x01, (char)LEDStatus)) and write a trigger bit to the FPGA (EPT_AH_SendTrigger((byte)0x02).



```
*/
    active_trigger              ACTIVE_TRIGGER_INST
    (
     .uc_clk                    (CLK_66),
     .uc_reset                  (RST),
     .uc_in                     (UC_IN),
     .uc_out                    (uc_out_m[ 0*22 +: 22 ]),

     .trigger_to_host           (trigger_out),
     .trigger_to_device         (trigger_in_byte)

    );

    active_transfer             ACTIVE_TRANSFER_INST_1
    (
     .uc_clk                    (CLK_66),
     .uc_reset                  (RST),
     .uc_in                     (UC_IN),
     .uc_out                    (uc_out_m[ 1*22 +: 22 ]),

     .start_transfer            (led_start_transfer),
     .transfer_received         (led_transfer_in_received),

     .transfer_busy             (),

     .uc_addr                   (3'h1),

     .transfer_to_host          (led_host_transfer_byte),
     .transfer_to_device        (led_device_transfer_byte)
    );
```

The above code is the interface Verilog which resides in the FPGA. When the C# Windows form sends a byte or trigger, the signals in the FPGA code react and allow the user code to receive the byte and trigger and perform some function with the information. In the case of this example, the LEDs will change state.

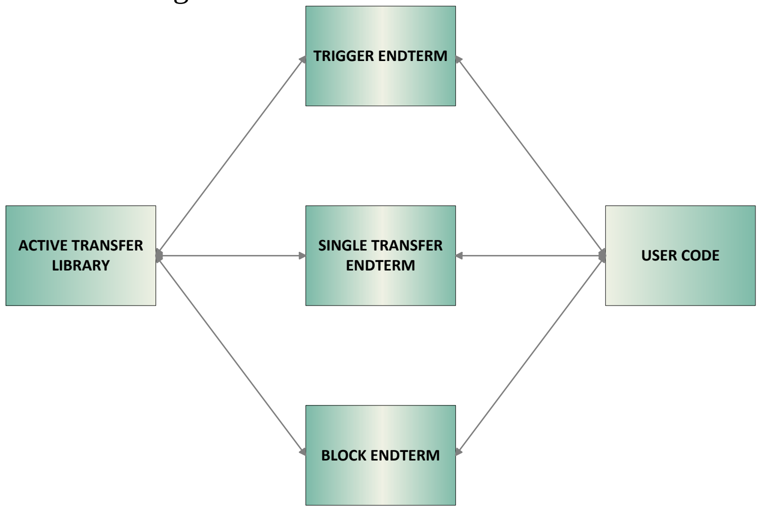
Receiving data from the FPGA is made simple by Active Host. Active Host transfers data from the FPGA as soon as it is available. It stores the transferred data into circular buffer. When the transfer is complete, Active Host invokes a callback function which is registered in the users application. This callback function provides a mechanism to transparently receive data from the FPGA. The user application does not need to schedule a read from the USB or call any blocking threads.

### 1.5 Active Transfer EndTerms

The Active Transfer Library is a portfolio of HDL modules that provides an easy to use yet powerful USB transfer mechanism. The user HDL code communicates with EndTerms in the form of modules. These EndTerm modules are commensurate with the Active Host EndTerms. There are three types of EndTerms in the Active Transfer Library:

- Trigger Endterm
- Transfer Endterm
- Block Endterm

They each have a simple interface that the user HDL code can use to send or receive data across the USB. Writing to an EndTerm will cause the data to immediately arrive



at the commensurate EndTerm in the Active Host/user application. The transfer through the USB is transparent. User HDL code doesn't need to set up Endpoints or respond to Host initiated data requests. The whole process is easy yet powerful.

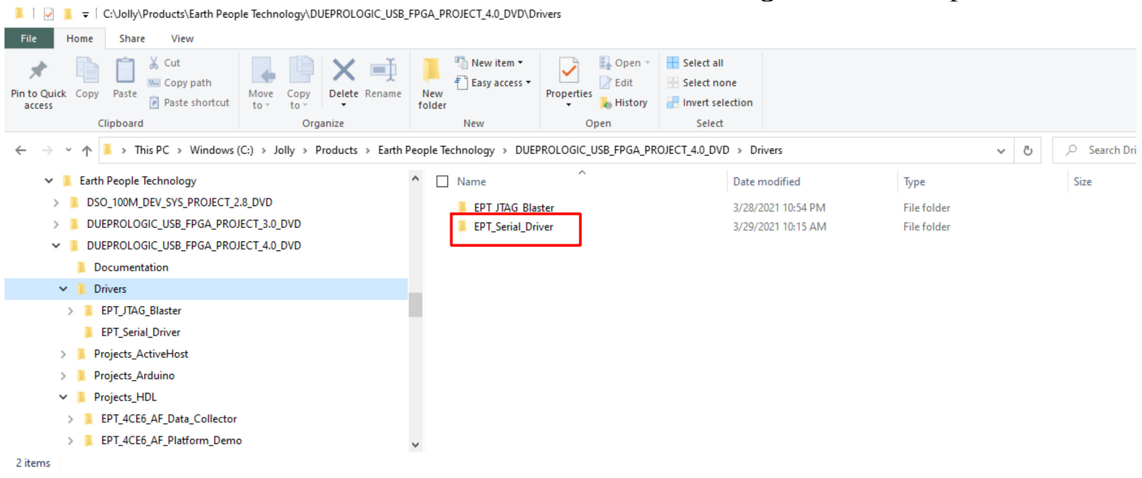
# 2 EPT Drivers

The DueProLogic 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.

## 2.1 USB Driver

The DueProLogic uses an FTDI FT2232H USB to Serial chip. This chip provides the USB interface to the PC and the serial/FIFO interface to the FPGA. The FT2232H 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 2.12.28 driver is easily accomplished by double clicking the CDM21228_Setup.exe.

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

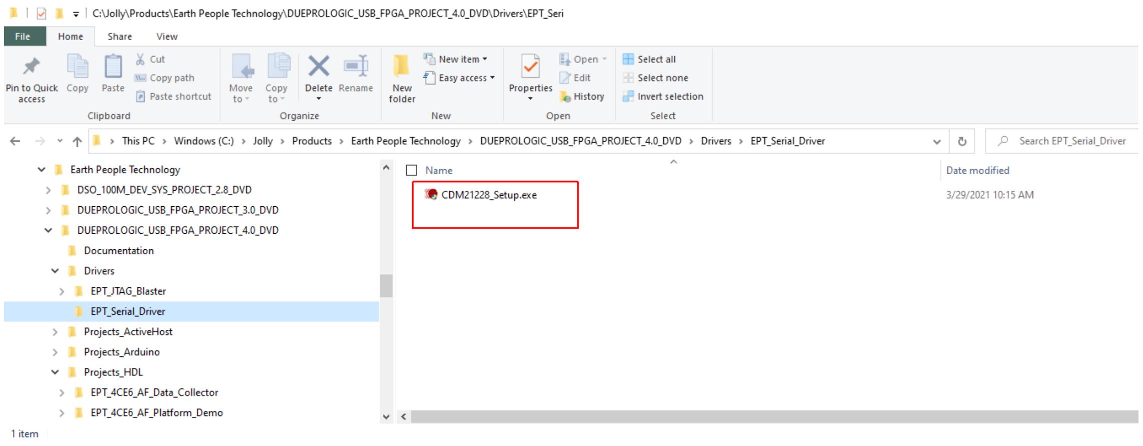


C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD\Drivers

This PC > Windows (C:) > Jolly > Products > Earth People Technology > DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD > Drivers

| Name | Date modified | Type | Size |
|---|---|---|---|
| EPT_JTAG_Blaster | 3/28/2021 10:54 PM | File folder | |
| EPT_Serial_Driver | 3/29/2021 10:15 AM | File folder | |

2 items



C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD\Drivers\EPT_Seri

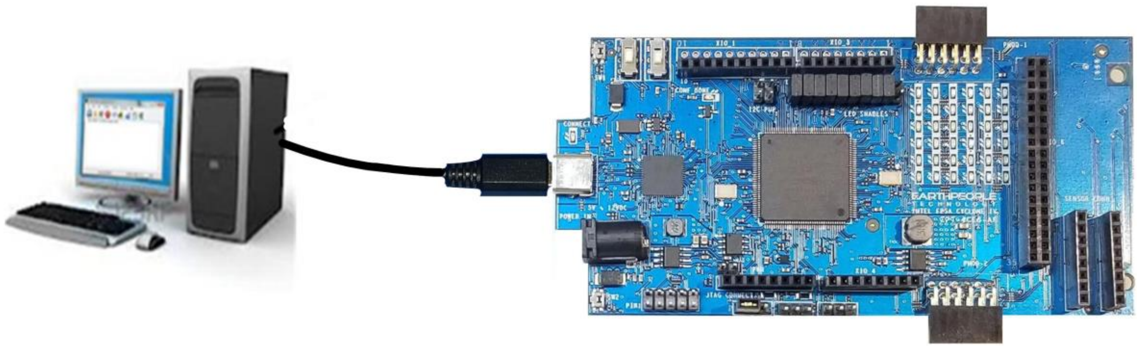
This PC > Windows (C:) > Jolly > Products > Earth People Technology > DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD > Drivers > EPT_Serial_Driver

| Name | Date modified |
|---|---|
| CDM21228_Setup.exe | 3/29/2021 10:15 AM |

1 item

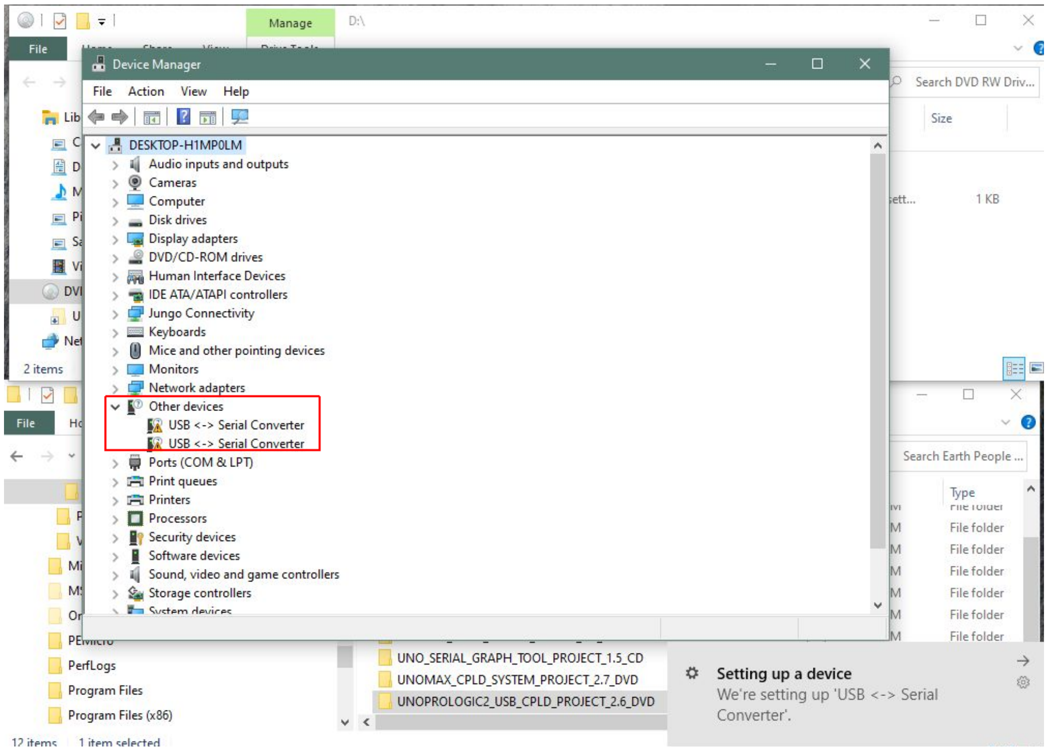
Double click on the *.exe file and select the default settings when the software tool queries the user.

Plug in the DueProLogic into an available USB port.



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

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



Device Manager

File Action View Help

- DESKTOP-H1MP0LM
  - Audio inputs and outputs
  - Cameras
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Jungo Connectivity
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Other devices
    - USB <-> Serial Converter
    - USB <-> Serial Converter
  - Ports (COM & LPT)
  - Print queues
  - Printers
  - Processors
  - Security devices
  - Software devices
  - Sound, video and game controllers
  - Storage controllers
  - System devices

UNO_SERIAL_GRAPH_TOOL_PROJECT_1.5_CD
UNOMAX_CPLD_SYSTEM_PROJECT_2.7_DVD
UNOPROLOGIC2_USB_CPLD_PROJECT_2.6_DVD

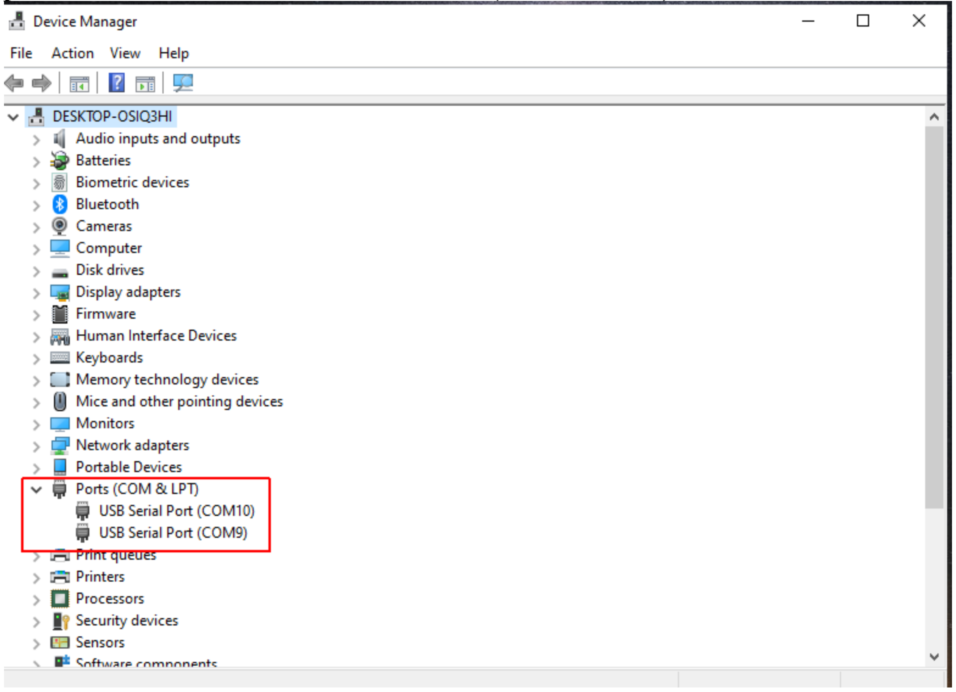
Setting up a device
We're setting up 'USB <-> Serial Converter'.

Right click on the “USB <-> Serial Converter” icon that failed to load. Then select the “Load driver from Known Location”. Navigate to the DUEPROLOGIC_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 DPL. The DPL 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 DueProLogic can be used for programming and USB data transfers.

### 2.2 JTAG DLL Insert to Quartus Prime Lite

The JTAG DLL Insert to Quartus Prime Lite allows the Programmer Tool under Quartus to recognize the DueProLogic. The DueProLogic can then be selected and perform programming of the FPGA. The file, jtag_hw_mbftdi_blaster.dll must be placed into the folder that hosts the jtag_server for Quartus.

#### 2.2.1 Installing Quartus

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

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

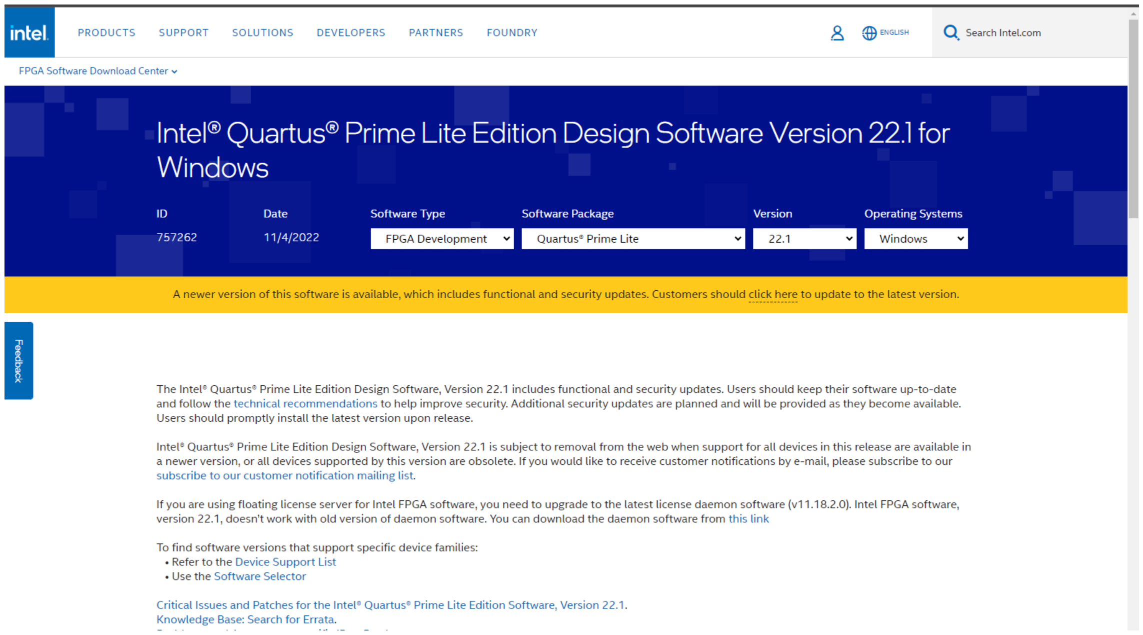
#### 2.2.2 Downloading Quartus

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.



Intel® Quartus® Prime Lite Edition Design Software Version 22.1 for Windows

| ID | Date | Software Type | Software Package | Version | Operating Systems |
|---|---|---|---|---|---|
| 757262 | 11/4/2022 | FPGA Development | Quartus® Prime Lite | 22.1 | Windows |

A newer version of this software is available, which includes functional and security updates. Customers should click here to update to the latest version.

The Intel® Quartus® Prime Lite Edition Design Software, Version 22.1 includes functional and security updates. Users should keep their software up-to-date and follow the technical recommendations to help improve security. Additional security updates are planned and will be provided as they become available. Users should promptly install the latest version upon release.

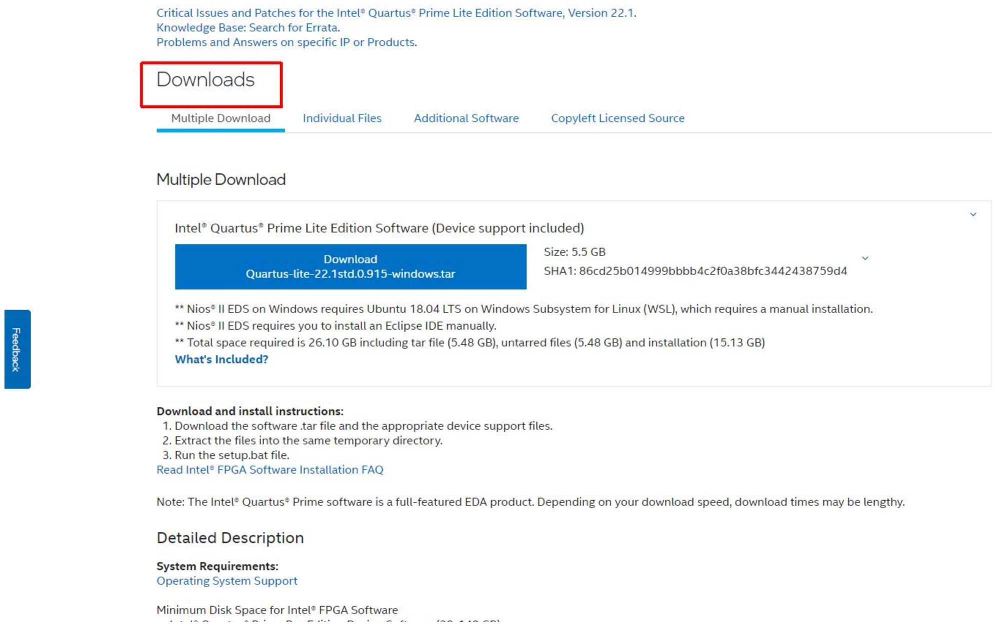
Intel® Quartus® Prime Lite Edition Design Software, Version 22.1 is subject to removal from the web when support for all devices in this release are available in a newer version, or all devices supported by this version are obsolete. If you would like to receive customer notifications by e-mail, please subscribe to our subscribe to our customer notification mailing list.

If you are using floating license server for Intel FPGA software, you need to upgrade to the latest license daemon software (v11.18.2.0). Intel FPGA software, version 22.1, doesn't work with old version of daemon software. You can download the daemon software from this link

To find software versions that support specific device families:
- Refer to the Device Support List
- Use the Software Selector

Critical Issues and Patches for the Intel® Quartus® Prime Lite Edition Software, Version 22.1.
Knowledge Base: Search for Errata.

Scroll down the page to the “Downloads” section.



Critical Issues and Patches for the Intel® Quartus® Prime Lite Edition Software, Version 22.1.
Knowledge Base: Search for Errata.
Problems and Answers on specific IP or Products.

Downloads

Multiple Download | Individual Files | Additional Software | Copyleft Licensed Source

Multiple Download

Intel® Quartus® Prime Lite Edition Software (Device support included)

Download
Quartus-lite-22.1std.0.915-windows.tar

Size: 5.5 GB
SHA1: 86cd25b014999bbbb4c2f0a38bfc3442438759d4

** Nios® II EDS on Windows requires Ubuntu 18.04 LTS on Windows Subsystem for Linux (WSL), which requires a manual installation.
** Nios® II EDS requires you to install an Eclipse IDE manually.
** Total space required is 26.10 GB including tar file (5.48 GB), untarred files (5.48 GB) and installation (15.13 GB)
**What's Included?**

**Download and install instructions:**
1. Download the software .tar file and the appropriate device support files.
2. Extract the files into the same temporary directory.
3. Run the setup.bat file.

Read Intel® FPGA Software Installation FAQ

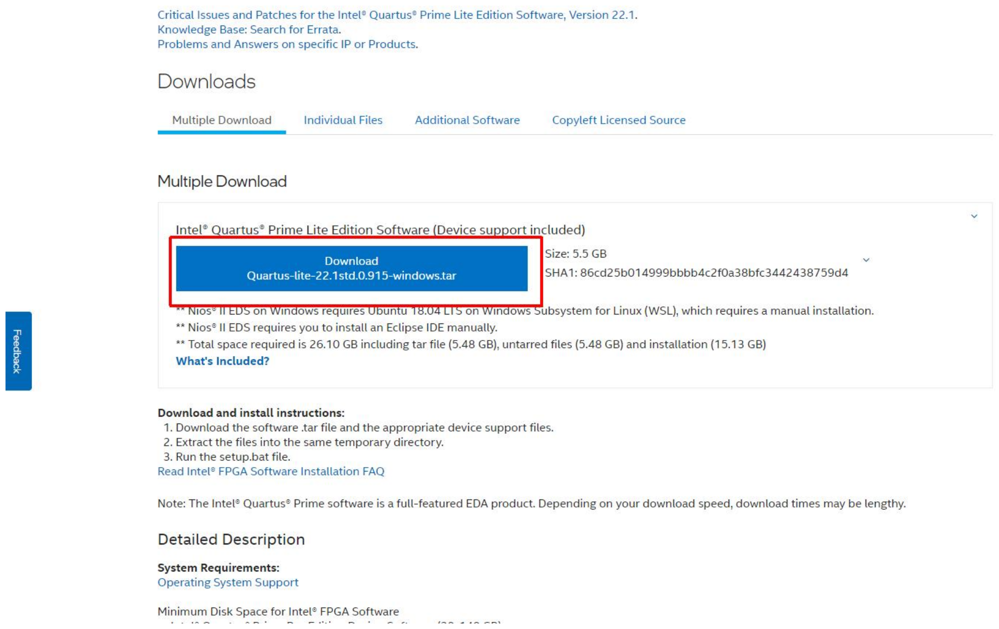
Note: The Intel® Quartus® Prime software is a full-featured EDA product. Depending on your download speed, download times may be lengthy.

Detailed Description

**System Requirements:**
Operating System Support

Minimum Disk Space for Intel® FPGA Software

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


Critical Issues and Patches for the Intel® Quartus® Prime Lite Edition Software, Version 22.1.
Knowledge Base: Search for Errata.
Problems and Answers on specific IP or Products.

Downloads

Multiple Download | Individual Files | Additional Software | Copyleft Licensed Source

Multiple Download

Intel® Quartus® Prime Lite Edition Software (Device support included)

Download
Quartus-lite-22.1std.0.915-windows.tar

Size: 5.5 GB
SHA1: 86cd25b014999bbbb4c2f0a38bfc3442438759d4

** Nios® II EDS on Windows requires Ubuntu 18.04 LTS on Windows Subsystem for Linux (WSL), which requires a manual installation.
** Nios® II EDS requires you to install an Eclipse IDE manually.
** Total space required is 26.10 GB including tar file (5.48 GB), untarred files (5.48 GB) and installation (15.13 GB)
What's Included?

Download and install instructions:
1. Download the software .tar file and the appropriate device support files.
2. Extract the files into the same temporary directory.
3. Run the setup.bat file.
Read Intel® FPGA Software Installation FAQ

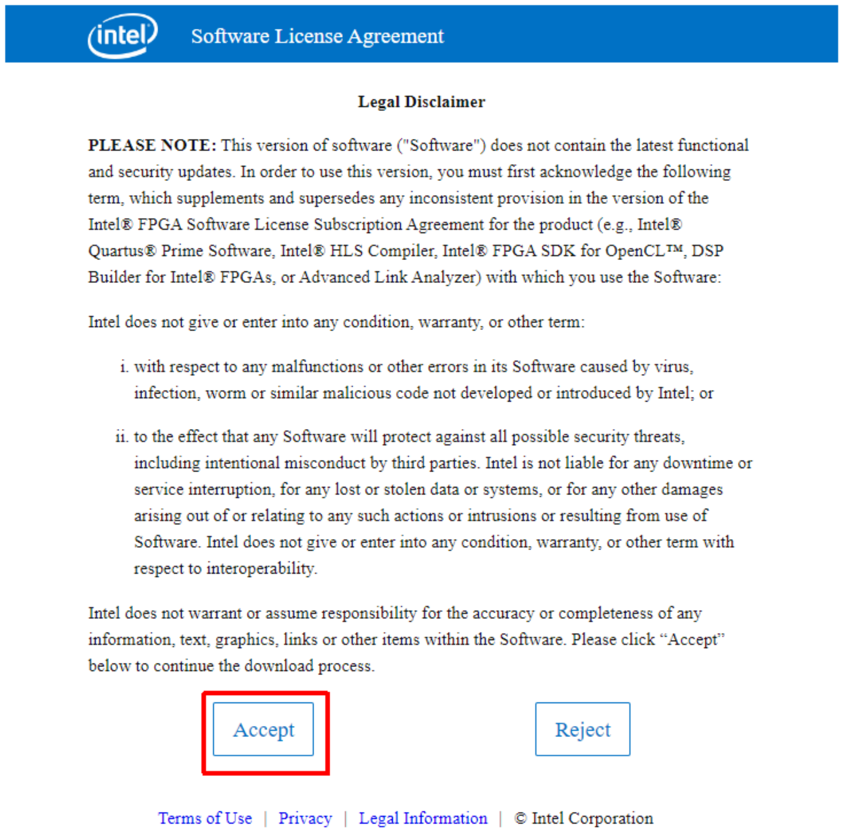
Note: The Intel® Quartus® Prime software is a full-featured EDA product. Depending on your download speed, download times may be lengthy.

Detailed Description

System Requirements:
Operating System Support

Minimum Disk Space for Intel® FPGA Software

Click through the Legal Stuff.



**Legal Disclaimer**

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

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

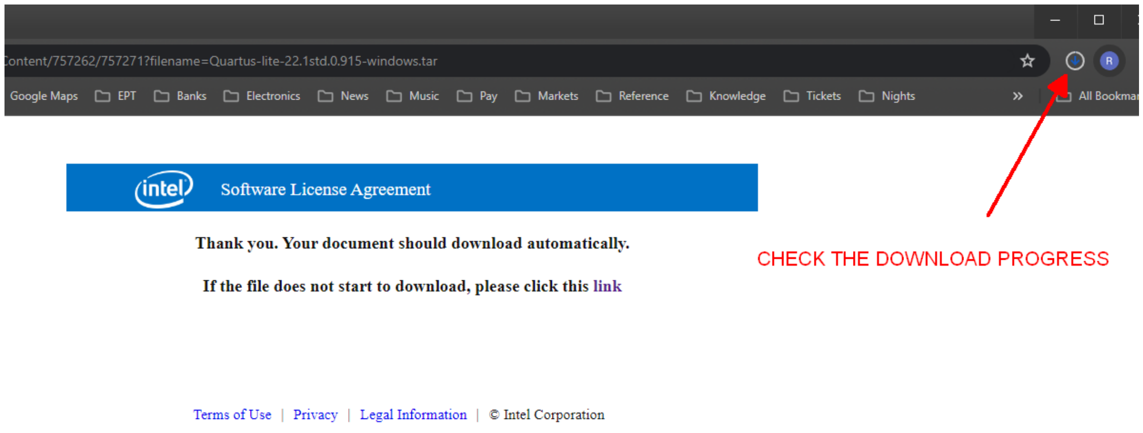
i. with respect to any malfunctions or other errors in its Software caused by virus, infection, worm or similar malicious code not developed or introduced by Intel; or

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

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

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

This will start the download.


ontent/757262/757271?filename=Quartus-lite-22.1std.0.915-windows.tar

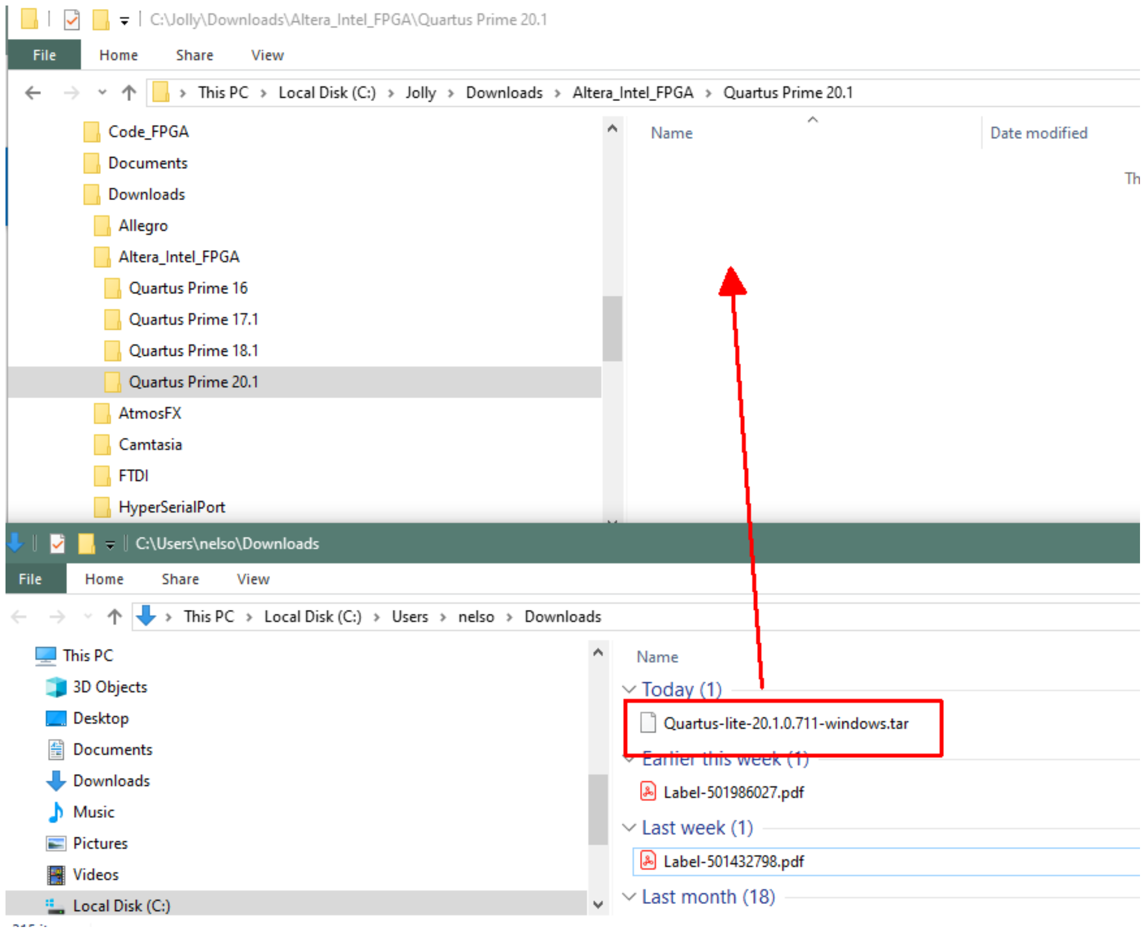
Google Maps
EPT
Banks
Electronics
News
Music
Pay
Markets
Reference
Knowledge
Tickets
Nights
All Bookmar
Software License Agreement

Thank you. Your document should download automatically.

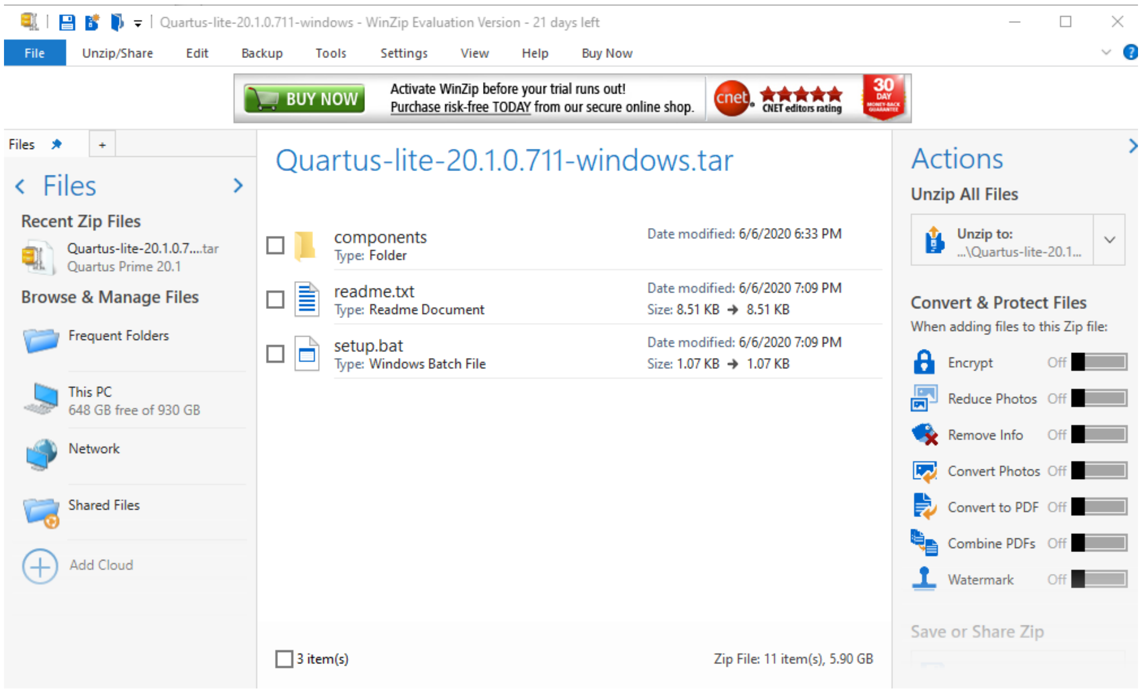
If the file does not start to download, please click this link

CHECK THE DOWNLOAD PROGRESS
Terms of Use | Privacy | Legal Information | © Intel Corporation

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.



Quartus-lite-20.1.0.711-windows.tar

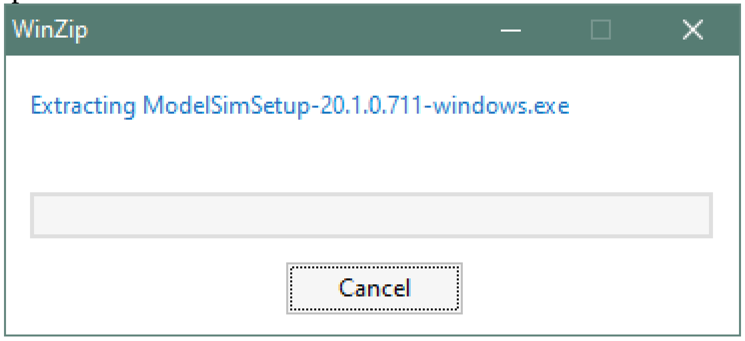
components
Type: Folder
Date modified: 6/6/2020 6:33 PM

readme.txt
Type: Readme Document
Date modified: 6/6/2020 7:09 PM
Size: 8.51 KB → 8.51 KB

setup.bat
Type: Windows Batch File
Date modified: 6/6/2020 7:09 PM
Size: 1.07 KB → 1.07 KB

3 item(s)
Zip File: 11 item(s), 5.90 GB

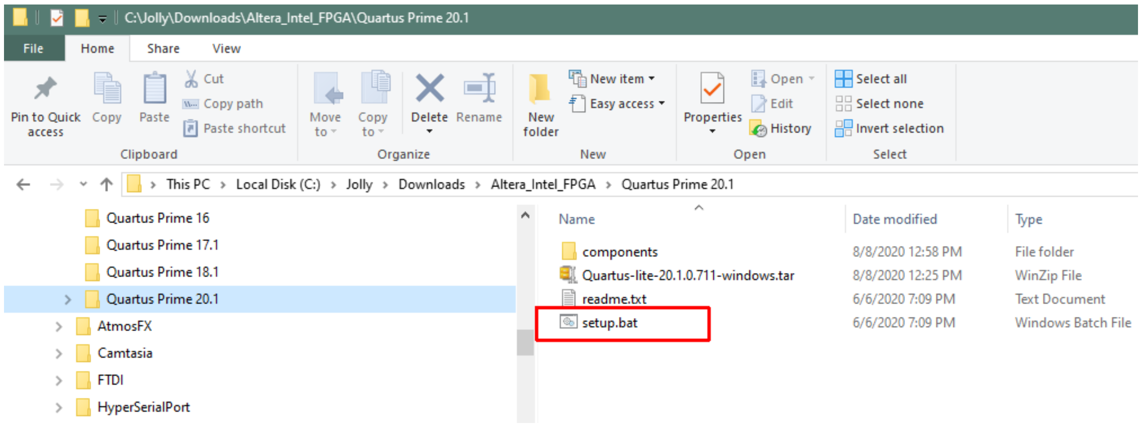
The tool will unpack all files.



Extracting ModelSimSetup-20.1.0.711-windows.exe

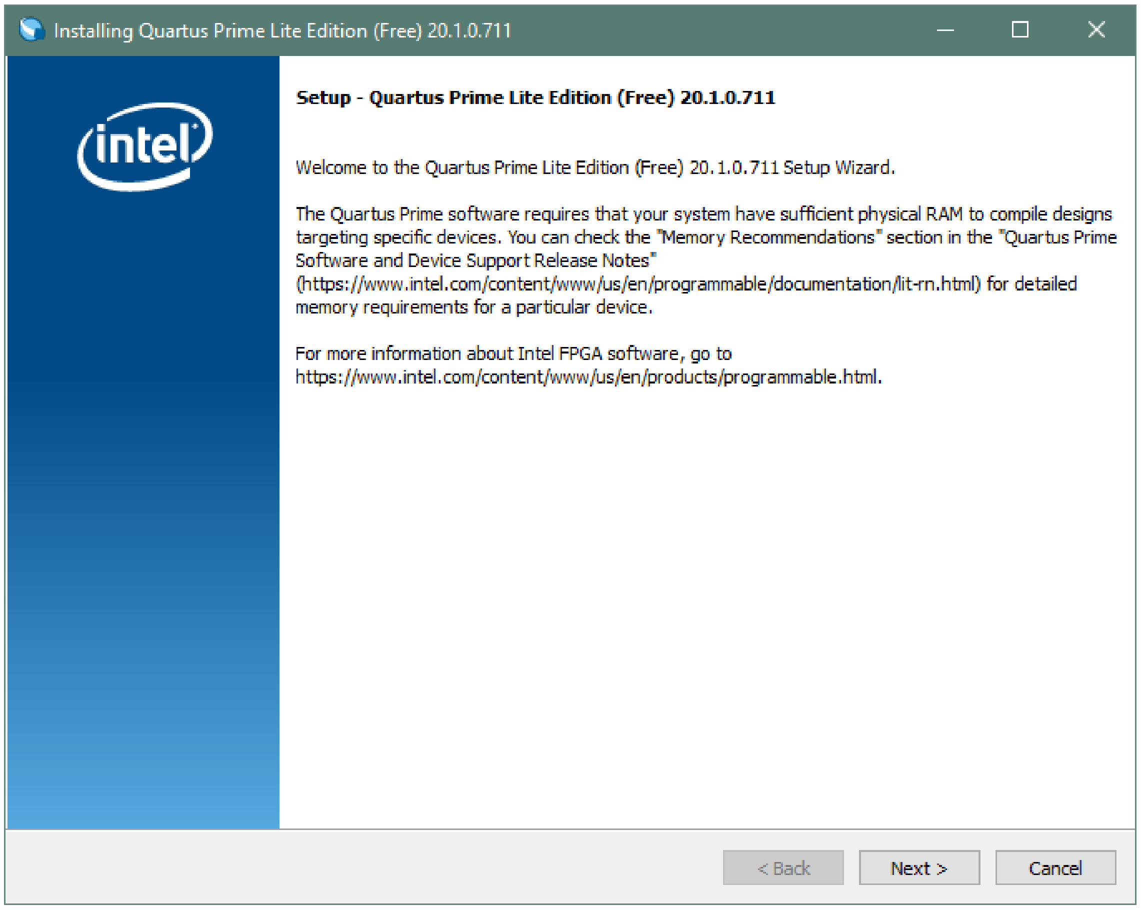
## 2.2.3 Quartus Installer

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



| Name | Date modified | Type |
|---|---|---|
| components | 8/8/2020 12:58 PM | File folder |
| Quartus-lite-20.1.0.711-windows.tar | 8/8/2020 12:25 PM | WinZip File |
| readme.txt | 6/6/2020 7:09 PM | Text Document |
| setup.bat | 6/6/2020 7:09 PM | Windows Batch File |

Click “Next” on the Introduction Window.



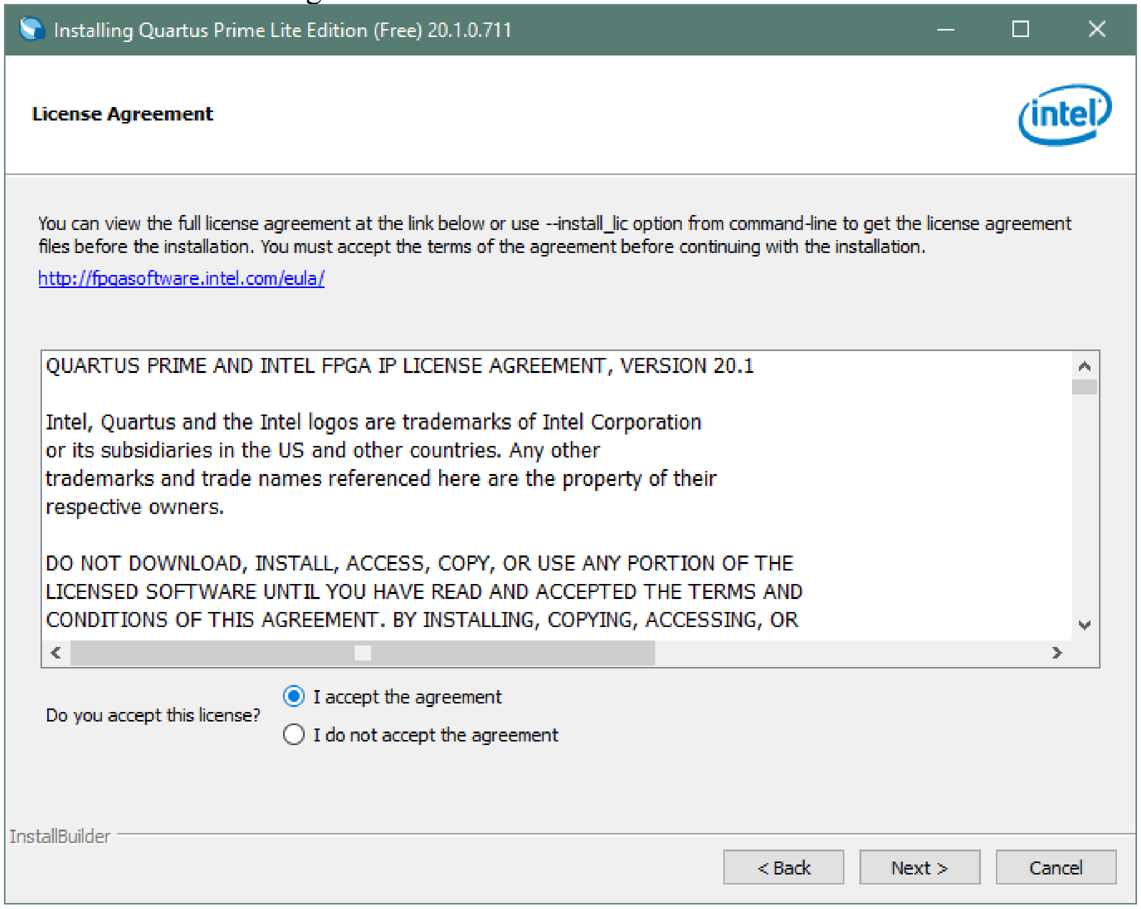
**Setup - Quartus Prime Lite Edition (Free) 20.1.0.711**

Welcome to the Quartus Prime Lite Edition (Free) 20.1.0.711 Setup Wizard.

The Quartus Prime software requires that your system have sufficient physical RAM to compile designs targeting specific devices. You can check the "Memory Recommendations" section in the "Quartus Prime Software and Device Support Release Notes" (https://www.intel.com/content/www/us/en/programmable/documentation/lit-rn.html) for detailed memory requirements for a particular device.

For more information about Intel FPGA software, go to https://www.intel.com/content/www/us/en/products/programmable.html.

Click the checkbox to agree to the license terms. Then click "Next".



Installing Quartus Prime Lite Edition (Free) 20.1.0.711

**License Agreement**

You can view the full license agreement at the link below or use --install_lic option from command-line to get the license agreement files before the installation. You must accept the terms of the agreement before continuing with the installation.

http://fpgasoftware.intel.com/eula/

QUARTUS PRIME AND INTEL FPGA IP LICENSE AGREEMENT, VERSION 20.1

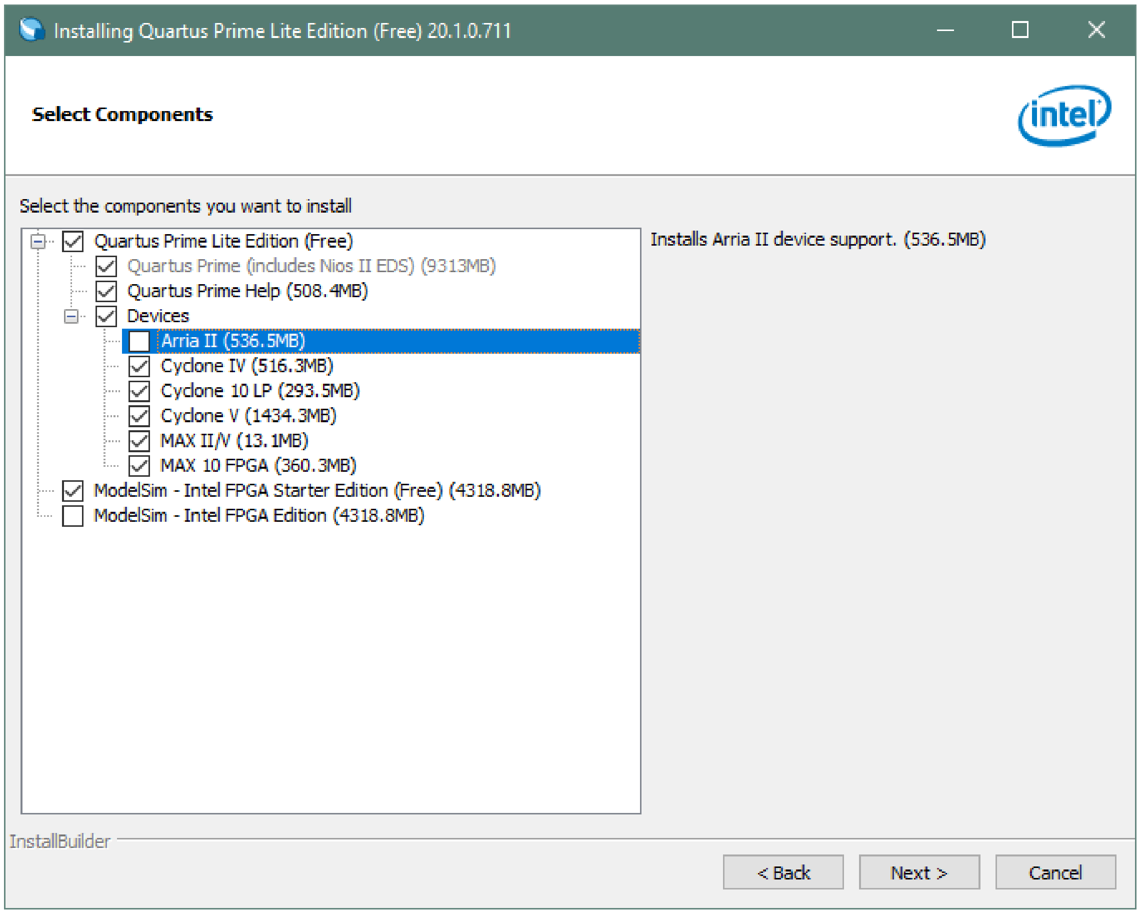
Intel, Quartus and the Intel logos are trademarks of Intel Corporation
or its subsidiaries in the US and other countries. Any other
trademarks and trade names referenced here are the property of their
respective owners.

DO NOT DOWNLOAD, INSTALL, ACCESS, COPY, OR USE ANY PORTION OF THE
LICENSED SOFTWARE UNTIL YOU HAVE READ AND ACCEPTED THE TERMS AND
CONDITIONS OF THIS AGREEMENT. BY INSTALLING, COPYING, ACCESSING, OR

Do you accept this license? [✓ I accept the agreement] I do not accept the agreement

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).


Installing Quartus Prime Lite Edition (Free) 20.1.0.711

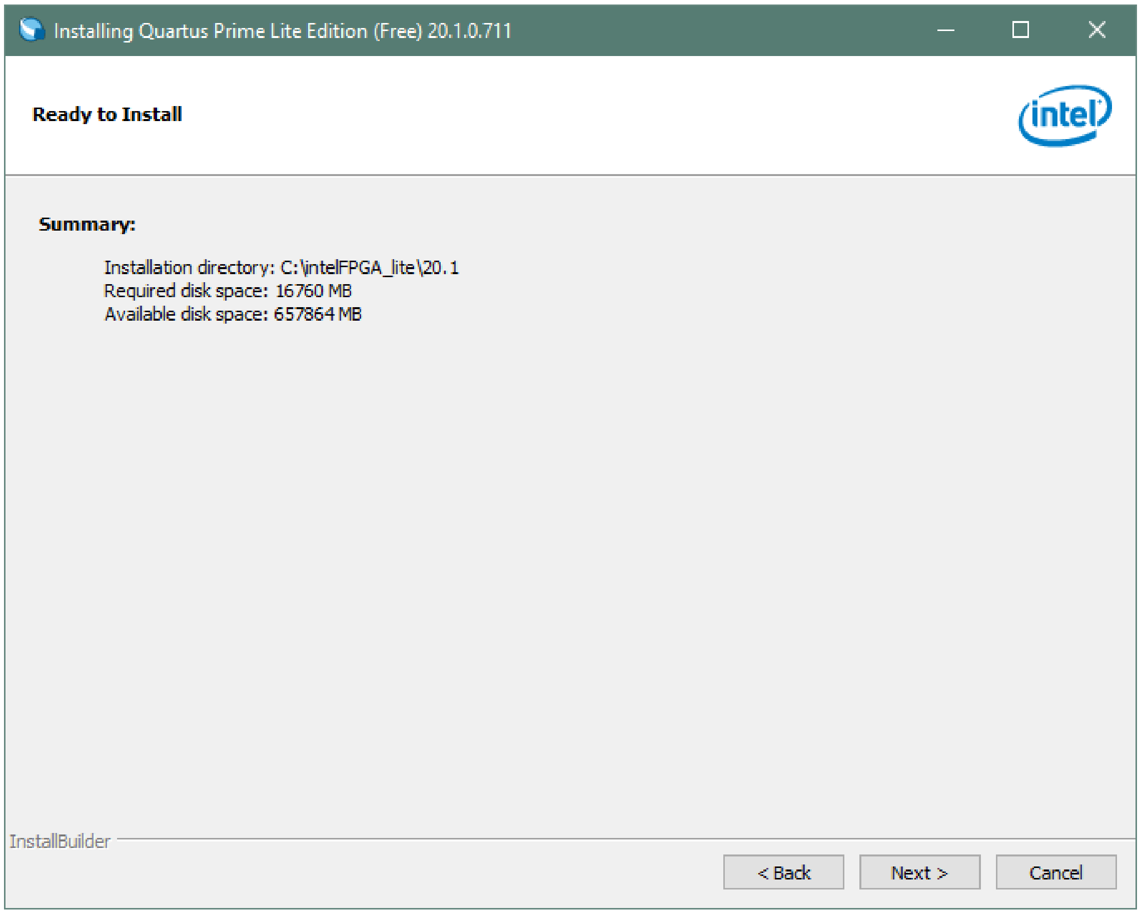
**Select Components**

Select the components you want to install

- [✓] Quartus Prime Lite Edition (Free)
  - [✓] Quartus Prime (includes Nios II EDS) (9313MB)
  - [✓] Quartus Prime Help (508.4MB)
  - [✓] Devices
    - [ ] Arria II (536.5MB)
    - [✓] Cyclone IV (516.3MB)
    - [✓] Cyclone 10 LP (293.5MB)
    - [✓] Cyclone V (1434.3MB)
    - [✓] MAX II/V (13.1MB)
    - [✓] MAX 10 FPGA (360.3MB)
- [✓] ModelSim - Intel FPGA Starter Edition (Free) (4318.8MB)
- [ ] ModelSim - Intel FPGA Edition (4318.8MB)

Installs Arria II device support. (536.5MB)

Click “Next” to accept the defaults

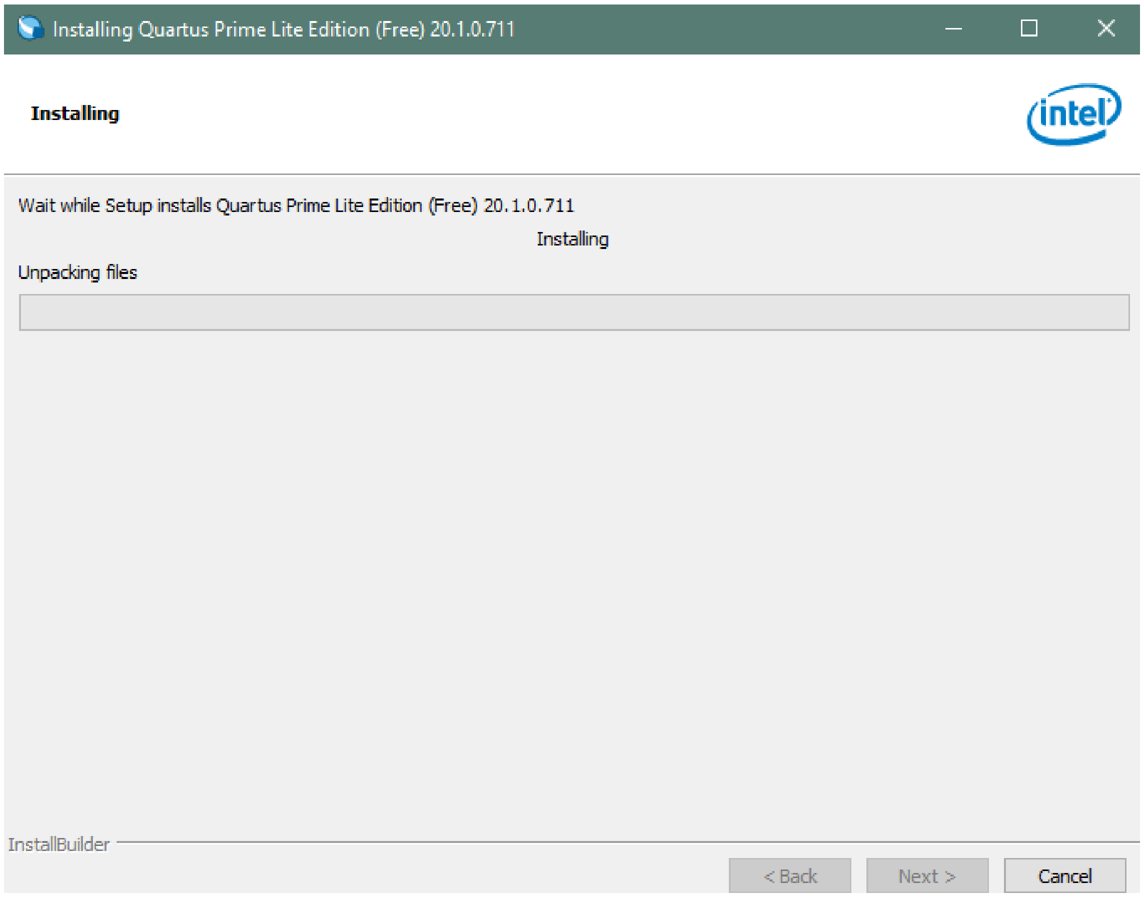

Installing Quartus Prime Lite Edition (Free) 20.1.0.711

**Ready to Install**

**Summary:**

Installation directory: C:\intelFPGA_lite\20.1
Required disk space: 16760 MB
Available disk space: 657864 MB

Click "Next" to accept the defaults


Installing Quartus Prime Lite Edition (Free) 20.1.0.711

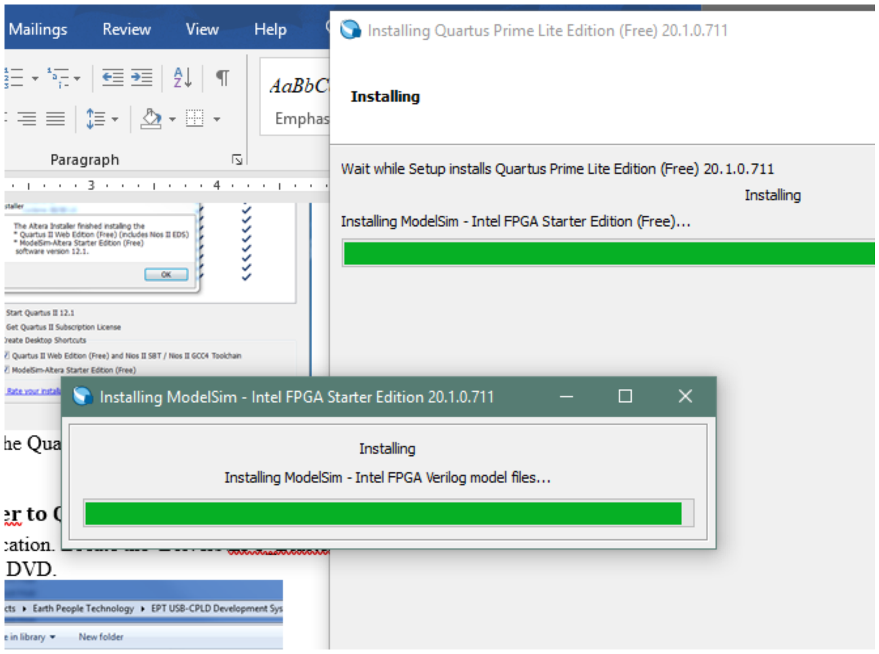
**Installing**

Wait while Setup installs Quartus Prime Lite Edition (Free) 20.1.0.711

Installing

Unpacking files

Wait for the installation to complete.


Installing Quartus Prime Lite Edition (Free) 20.1.0.711

Installing

Wait while Setup installs Quartus Prime Lite Edition (Free) 20.1.0.711

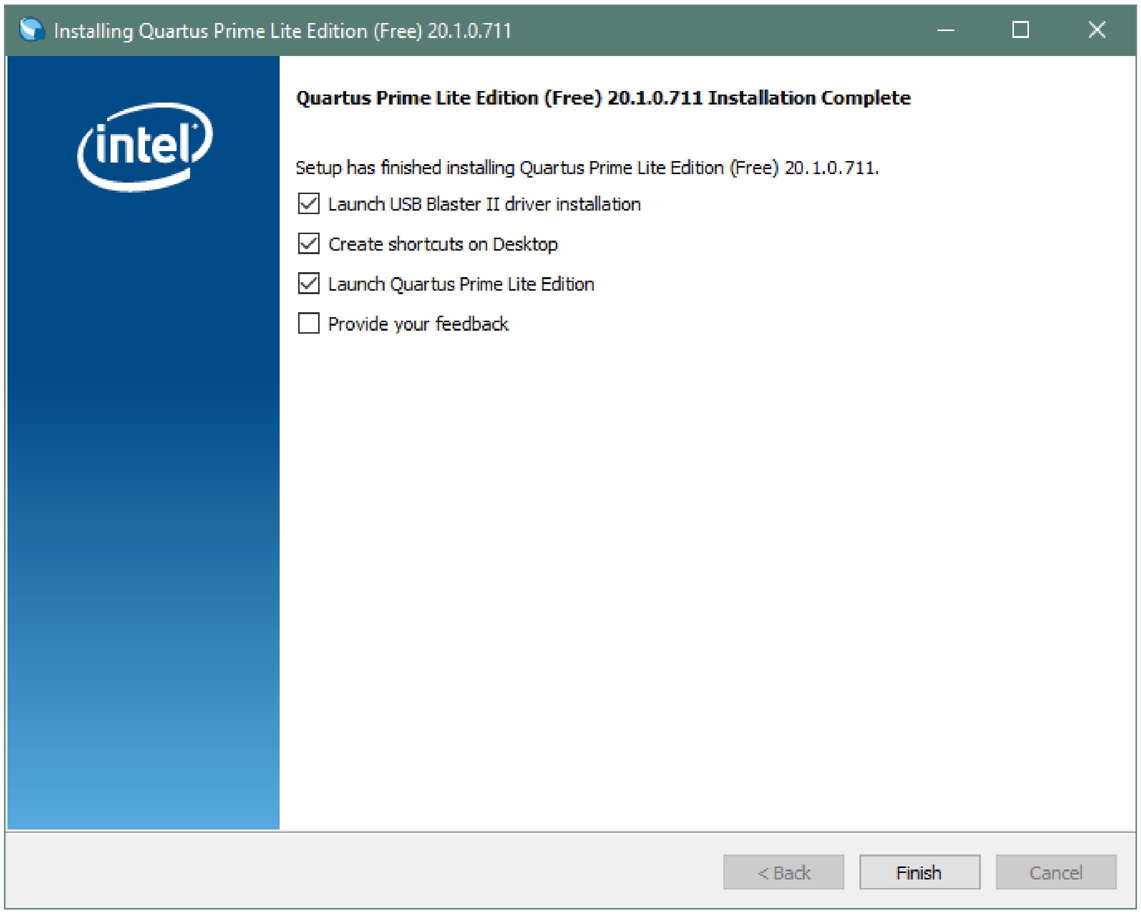
Installing

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...



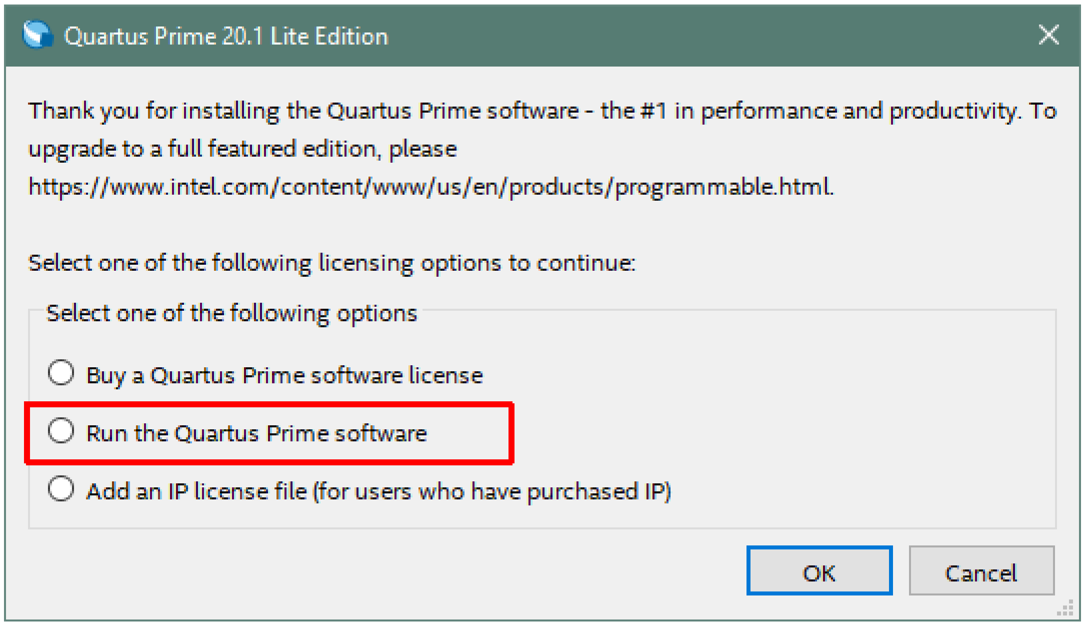
Installing Quartus Prime Lite Edition (Free) 20.1.0.711

**Quartus Prime Lite Edition (Free) 20.1.0.711 Installation Complete**

Setup has finished installing Quartus Prime Lite Edition (Free) 20.1.0.711.

- [✓] Launch USB Blaster II driver installation
- [✓] Create shortcuts on Desktop
- [✓] Launch Quartus Prime Lite Edition
- [ ] Provide your feedback

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


Quartus Prime 20.1 Lite Edition

Thank you for installing the Quartus Prime software - the #1 in performance and productivity. To upgrade to a full featured edition, please https://www.intel.com/content/www/us/en/products/programmable.html.

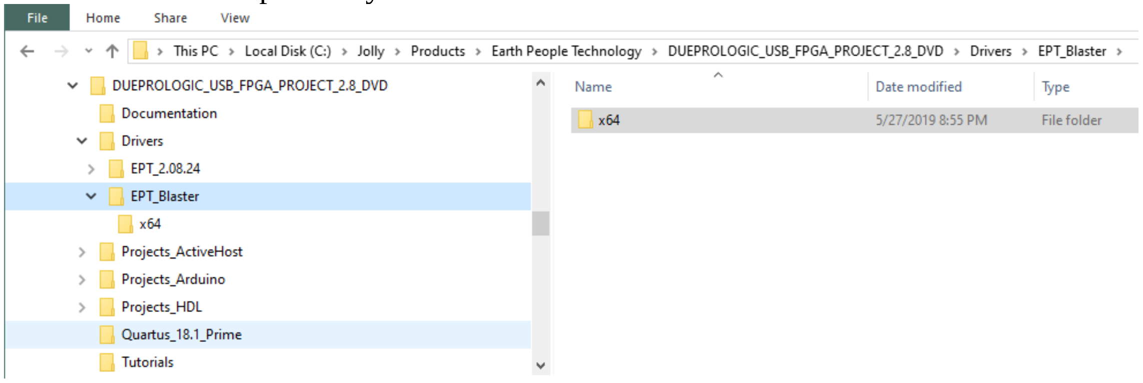
Select one of the following licensing options to continue:

Select one of the following options

- Buy a Quartus Prime software license
- Run the Quartus Prime software
- Add an IP license file (for users who have purchased IP)

### 2.2.4 Adding the EPT_Blaster to Quartus Prime

Close out the Quartus Prime application. Locate the \Drivers\EPT_Blaster folder on the EPT  FPGA Development System DVD.

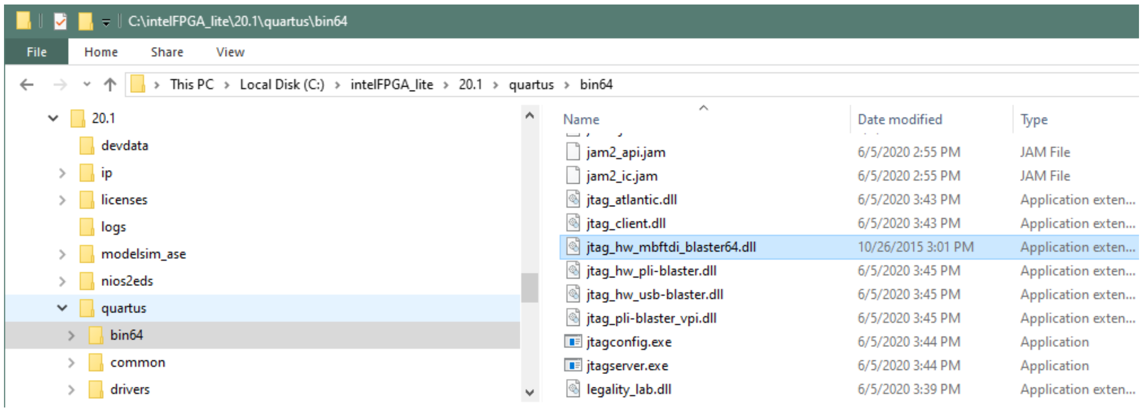

This PC > Local Disk (C:) > Jolly > Products > Earth People Technology > DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD > Drivers > EPT_Blaster >

| Name | Date modified | Type |
|---|---|---|
| x64 | 5/27/2019 8:55 PM | File folder |

Follow these directions:

1. Open the C:\EPT FPGA Development System DVD\Drivers\EPT_Blaster\x64 folder.
2. Select the file “jtag_hw_mbftdi_blaster.dll” and copy it.
3. Browse over to C:\intelFPGA_lite\xx.x\quartus\bin64.
4. Right click in the folder and select Paste

5. Click Ok.
6. Open the Quartus Prime application.



C:\intelFPGA_lite\20.1\quartus\bin64

File Home Share View

This PC > Local Disk (C:) > intelFPGA_lite > 20.1 > quartus > bin64

| Name | Date modified | Type |
|---|---|---|
| jam2_api.jam | 6/5/2020 2:55 PM | JAM File |
| jam2_ic.jam | 6/5/2020 2:55 PM | JAM File |
| jtag_atlantic.dll | 6/5/2020 3:43 PM | Application exten... |
| jtag_client.dll | 6/5/2020 3:43 PM | Application exten... |
| jtag_hw_mbftdi_blaster64.dll | 10/26/2015 3:01 PM | Application exten... |
| jtag_hw_pli-blaster.dll | 6/5/2020 3:45 PM | Application exten... |
| jtag_hw_usb-blaster.dll | 6/5/2020 3:45 PM | Application exten... |
| jtag_pli-blaster_vpi.dll | 6/5/2020 3:45 PM | Application exten... |
| jtagconfig.exe | 6/5/2020 3:44 PM | Application |
| jtagserver.exe | 6/5/2020 3:44 PM | Application |
| legality_lab.dll | 6/5/2020 3:39 PM | Application exten... |

The DLL is installed and the JTAG server should recognize it. Go to the section "Programming the FPGA" of this manual for testing of the programming. If the driver is not found in the Programmer Tool->Hardware Setup box, contact Earth People Technology for support. There are three methods to contact EPT for support:

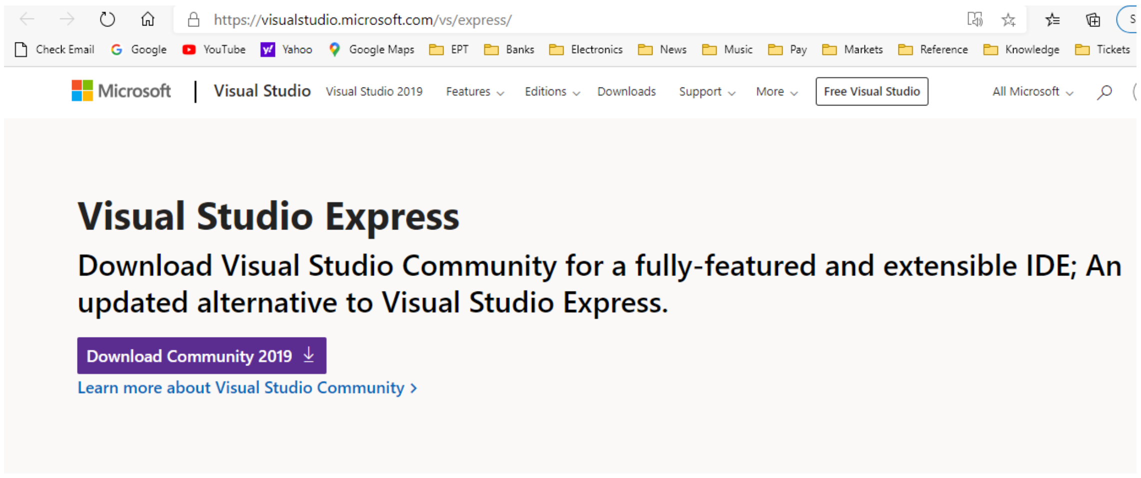
https://www.earthpeopletechnology.com->Forums
support@earthpeopletechnology.com
sales@earthpeopletechnology.com

## 2.3 Active Host Application SDK

Download the latest version of Microsoft Visual C# Express environment from Microsoft. It's a free download.

https://visualstudio.microsoft.com/vs/express/

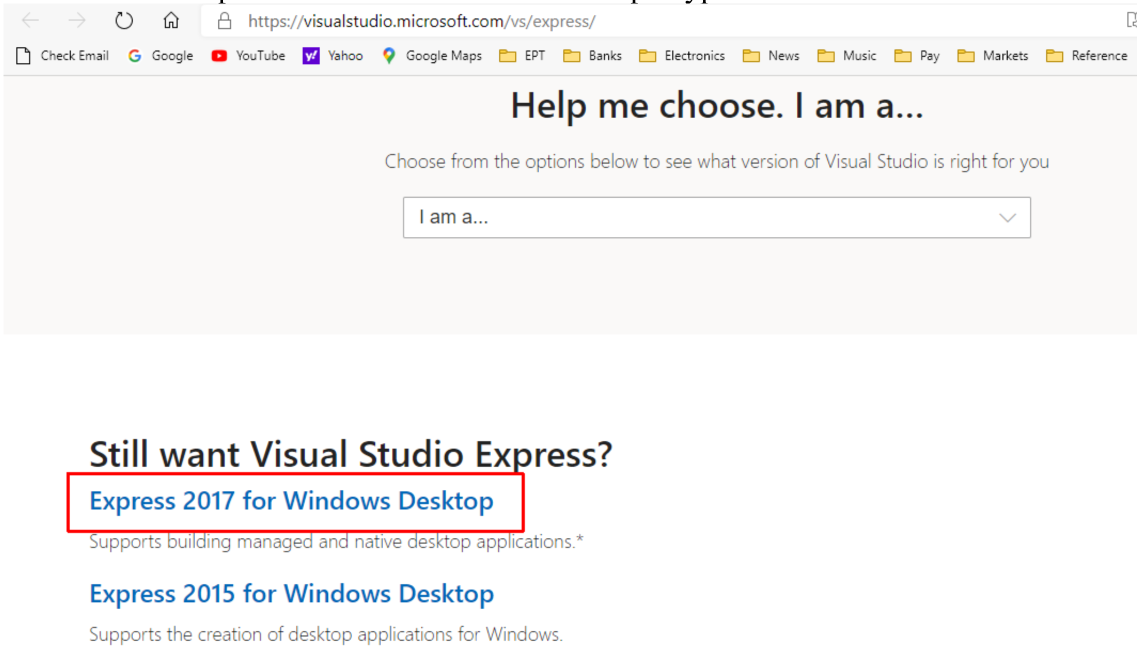
Go to the website and click on the "+" icon next to the Visual C# Express.



Visual Studio Express

Download Visual Studio Community for a fully-featured and extensible IDE; An updated alternative to Visual Studio Express.

Click on the "Express 20xx for Windows Desktop" hypertext.



Help me choose. I am a...

Choose from the options below to see what version of Visual Studio is right for you

Still want Visual Studio Express?

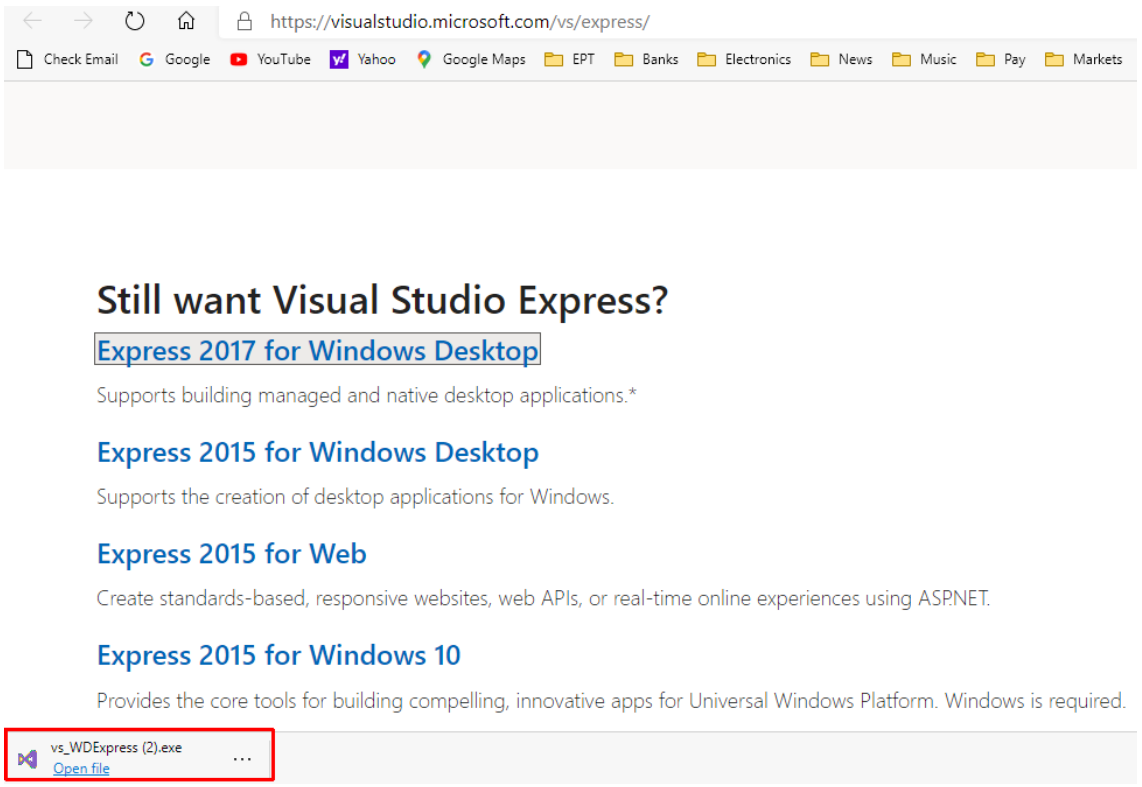
Express 2017 for Windows Desktop

Supports building managed and native desktop applications.*

Express 2015 for Windows Desktop

Supports the creation of desktop applications for Windows.

The download manager file will download the "WDExpress.exe" file.


https://visualstudio.microsoft.com/vs/express/

Still want Visual Studio Express?

Express 2017 for Windows Desktop

Supports building managed and native desktop applications.*

Express 2015 for Windows Desktop

Supports the creation of desktop applications for Windows.

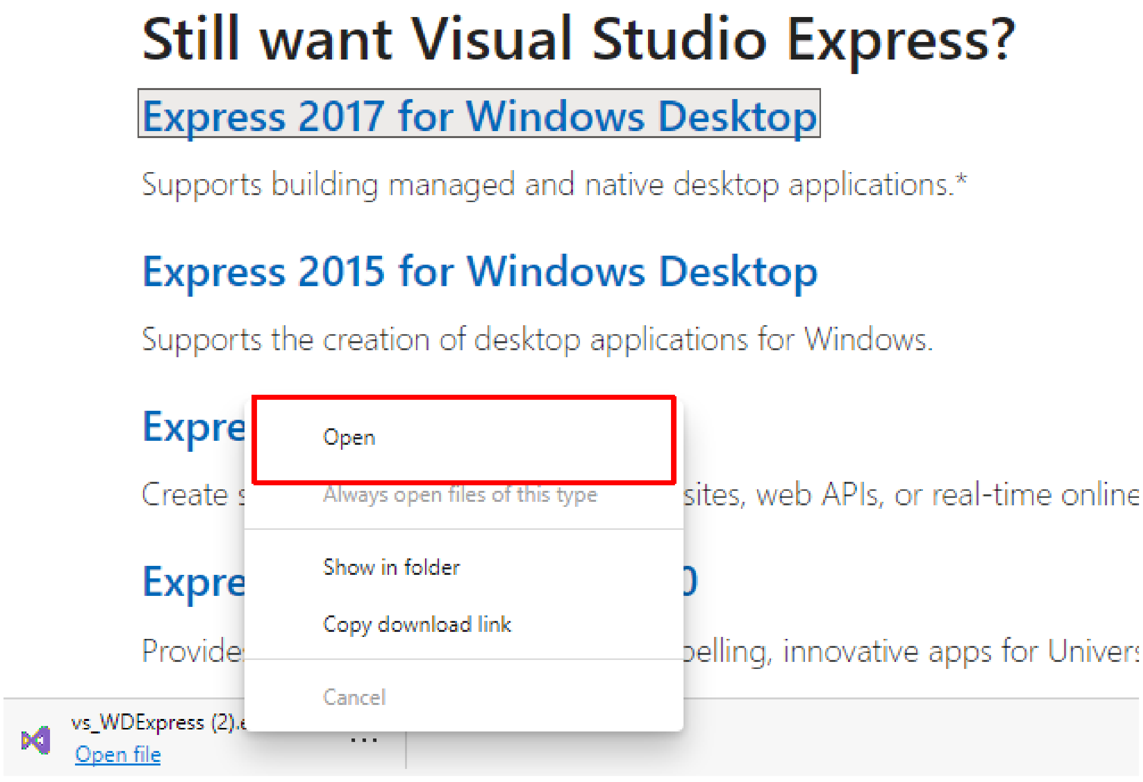
Express 2015 for Web

Create standards-based, responsive websites, web APIs, or real-time online experiences using ASP.NET.

Express 2015 for Windows 10

Provides the core tools for building compelling, innovative apps for Universal Windows Platform. Windows is required.

Right click on the WDExpress.exe.



Still want Visual Studio Express?

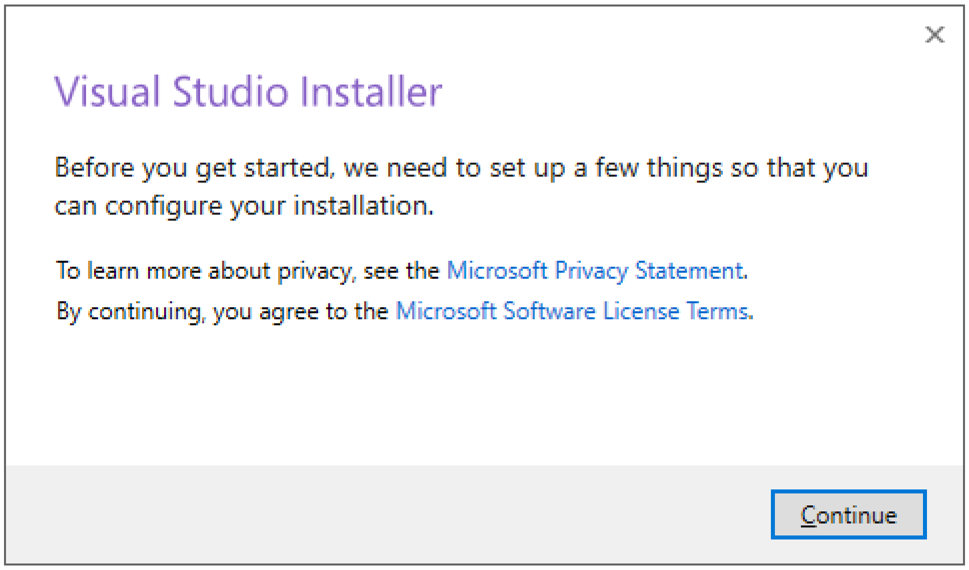
Express 2017 for Windows Desktop

Supports building managed and native desktop applications.*

Express 2015 for Windows Desktop

Supports the creation of desktop applications for Windows.

Click the “Continue” button.



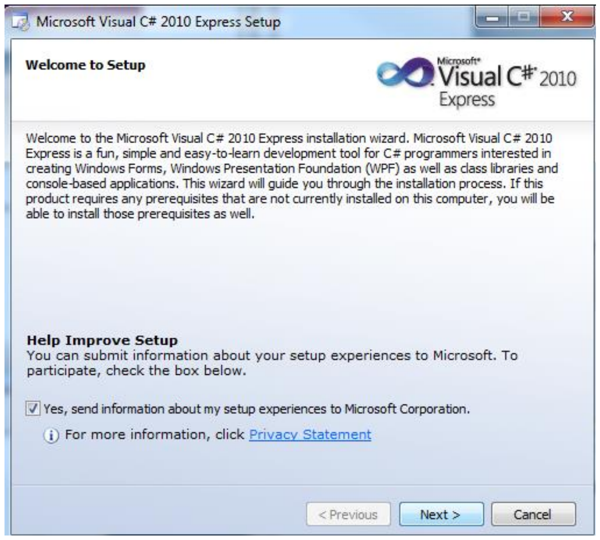
Visual Studio Installer

Before you get started, we need to set up a few things so that you can configure your installation.

To learn more about privacy, see the Microsoft Privacy Statement.

By continuing, you agree to the Microsoft Software License Terms.

Next, follow the on screen windows and accept the default answers.


Microsoft Visual C# 2010 Express Setup

Welcome to Setup

Welcome to the Microsoft Visual C# 2010 Express installation wizard. Microsoft Visual C# 2010 Express is a fun, simple and easy-to-learn development tool for C# programmers interested in creating Windows Forms, Windows Presentation Foundation (WPF) as well as class libraries and console-based applications. This wizard will guide you through the installation process. If this product requires any prerequisites that are not currently installed on this computer, you will be able to install those prerequisites as well.

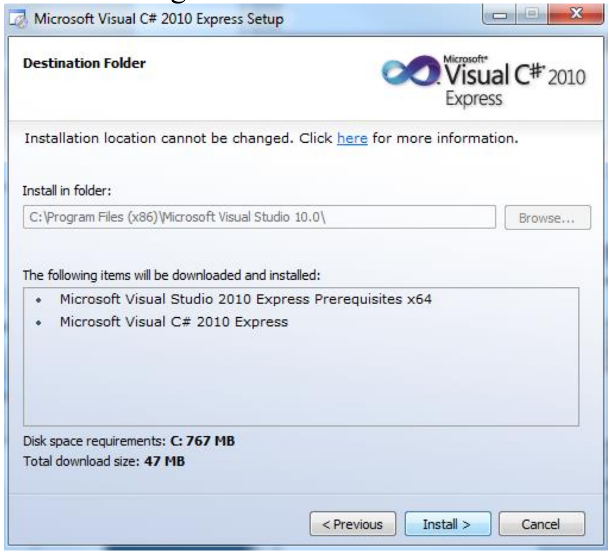
Help Improve Setup

You can submit information about your setup experiences to Microsoft. To participate, check the box below.

[✓] Yes, send information about my setup experiences to Microsoft Corporation.

For more information, click Privacy Statement

Click “Next”, accept the license agreement. Click “Next”.


Microsoft Visual C# 2010 Express Setup

Destination Folder

Installation location cannot be changed. Click here for more information.

Install in folder:

C:\Program Files (x86)\Microsoft Visual Studio 10.0\

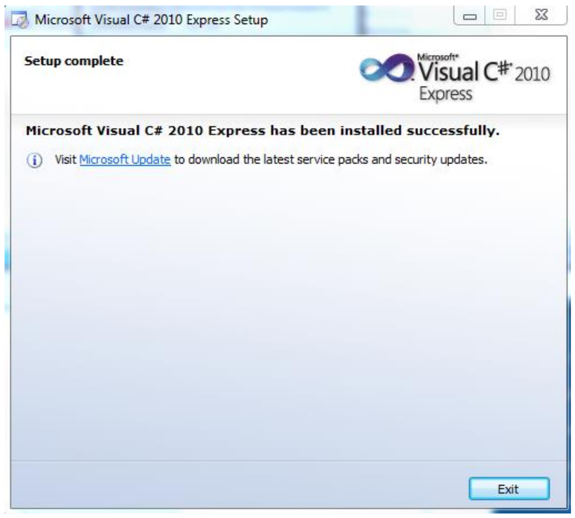
The following items will be downloaded and installed:

- Microsoft Visual Studio 2010 Express Prerequisites x64
- Microsoft Visual C# 2010 Express

Disk space requirements: **C: 767 MB**

Total download size: **47 MB**

Visual C# 2010 Express will install. This may take up to twenty minutes depending on your internet connection.



Microsoft Visual C# 2010 Express Setup

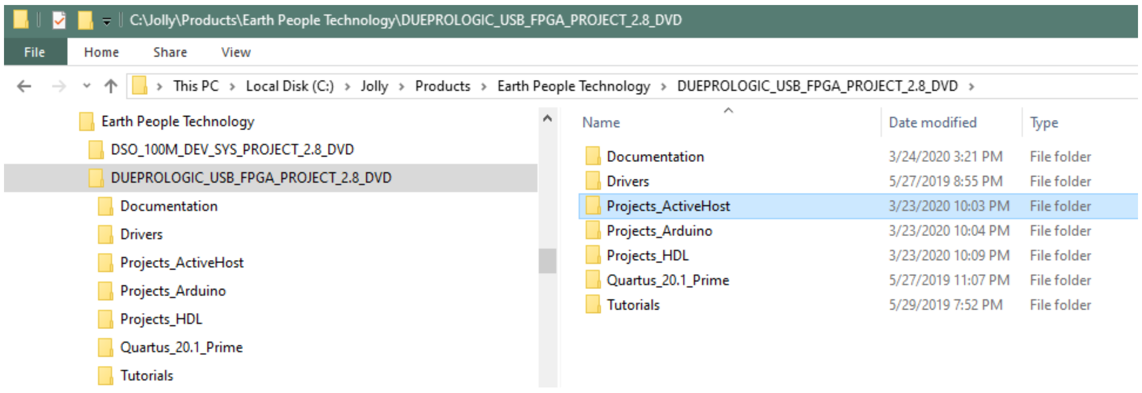
Setup complete

Microsoft Visual C# 2010 Express has been installed successfully.

Visit Microsoft Update to download the latest service packs and security updates.

The installed successfully window will be displayed when Visual C# Express is ready to use.

The Active Host Application Software will allow the user to create a custom applications on the PC using the EndTerms to perform Triggers and Data Transfer to/from the DueProLogic. The methods and parameters of the Active Host library are explained in the Active Host Application section. Locate the \Projects_ActiveHost folders on the EPT FPGA Development System DVD.



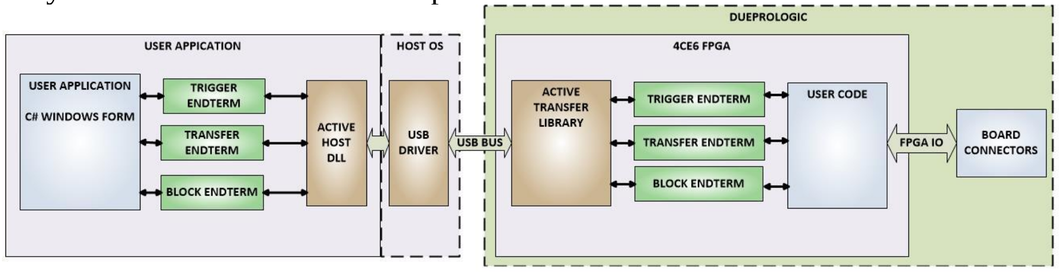
C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD

File
Home
Share
View
This PC › Local Disk (C:) › Jolly › Products › Earth People Technology › DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD ›
Earth People Technology
DSO_100M_DEV_SYS_PROJECT_2.8_DVD
DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD
Documentation
Drivers
Projects_ActiveHost
Projects_Arduino
Projects_HDL
Quartus_20.1_Prime
Tutorials

| Name | Date modified | Type |
|---|---|---|
| Documentation | 3/24/2020 3:21 PM | File folder |
| Drivers | 5/27/2019 8:55 PM | File folder |
| Projects_ActiveHost | 3/23/2020 10:03 PM | File folder |
| Projects_Arduino | 3/23/2020 10:04 PM | File folder |
| Projects_HDL | 3/23/2020 10:09 PM | File folder |
| Quartus_20.1_Prime | 5/27/2019 11:07 PM | File folder |
| Tutorials | 5/29/2019 7:52 PM | File folder |

## 3 FPGA Active Transfer Library

The Active Transfer Library is an HDL library designed to transfer data to and from the DueProLogic via High Speed (480 MB/s) USB. It is a set of pre-compiled HDL files that the user will add to their project before building it. The description of what the library does and how to use its components are described in this manual.



## 3.1 EPT Active Transfer System Overview

The Active Transfer System components consist of the following:

- active_serial_library.v
- ft_245_state_machine.v
- endpoint_registers.vqm
- active_trigger.v
- active_transfer.v
- active_block.v

The Active_Serial_Library provides the communication to the USB hardware. While separate Input and Output buses provide bi-directional communications with the plug in modules. See the figure below for an overview of the EPT Active_Transfer system.

**EPT Active Transfer Library Overview**

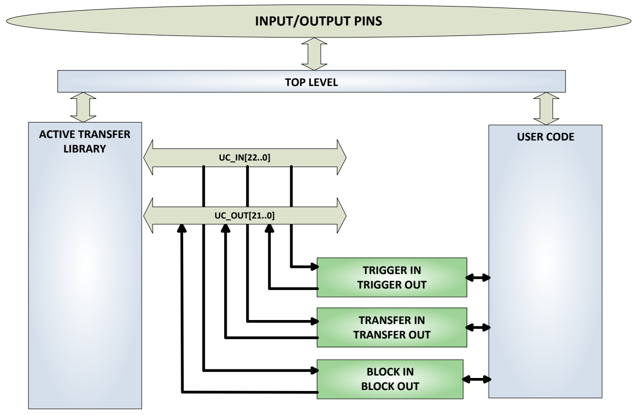
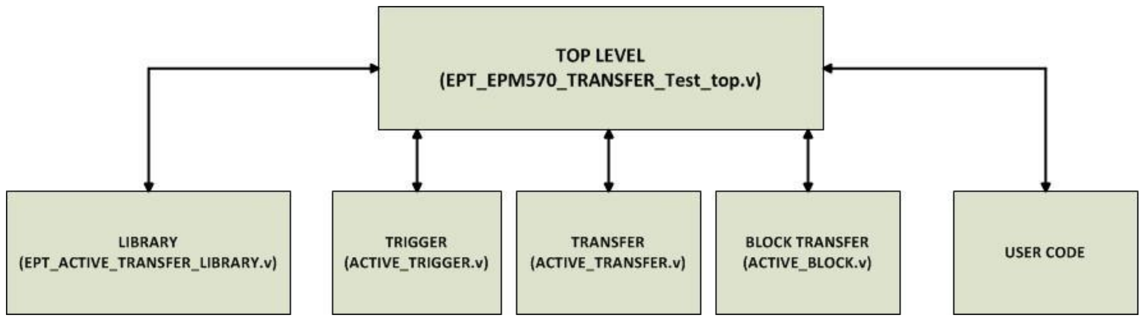


Figure 6 shows how the modules of the EPT Active Transfer Library attach to the overall user project. The EPT Active_Transfer_Library.vqm, Active_Trigger.vqm, Active_Transfer.vqm and Active_Block.vqm  modules are instantiated in the top level of the user project. The User_Code module is also instantiated in the top level. The Active_Transfer modules communicate with the User_Code through module parameters. Each module is a bi-directional component that facilitates data transfer from PC to FPGA. The user code can send a transfer to the Host, and the Host can send a transfer to the user code. This provides significant control for both data transfers and signaling from the user code to PC. The Triggers are used to send momentary signals that can turn on (or off) functions in user code or PC. The Active Transfer is used to send a single byte. And the Active Block is used to send a block of data. The Active_Transfer and Active_Block modules have addressing built into them. This means the user can declare up to 8 individual instantiations of Active_Transfer or Active_Block, and send/receive data to each module separately.

## *3.2 Active Transfer Library*

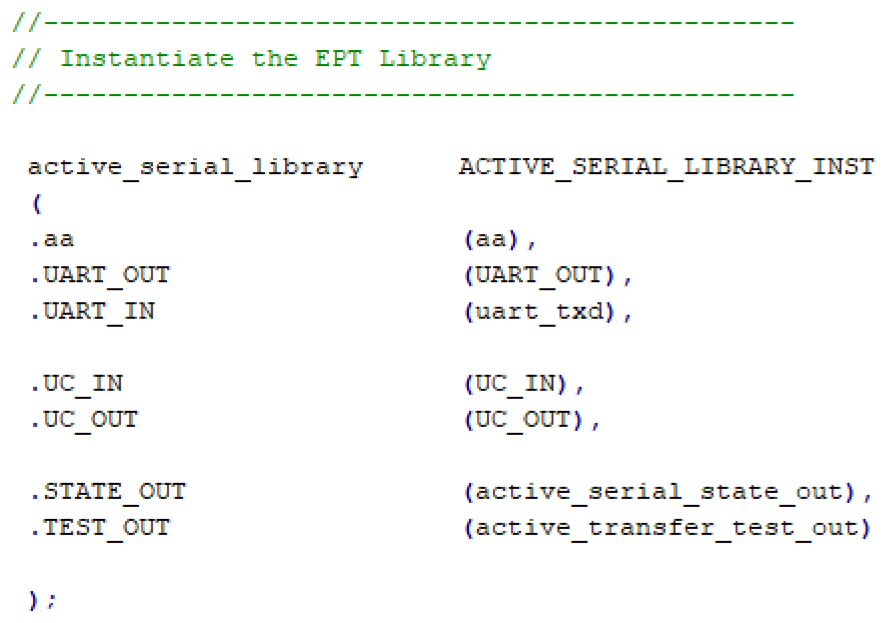
The Active Transfer Library contains the command, control, and data transfer mechanism that allows users to quickly build powerful communication schemes in the FPGA. Coupled with the Active Host application on the PC, this tools allows users to focus on creating programmable logic applications and not have to become distracted by USB Host drivers and timing issues. The Active Transfer Library is pre-compiled file that the user will include in the project files.



```
//##########################################################################
//#
//# Copyright   Earth People Technology Inc. 2021
//#
//#
//# File Name:  EPT_4CE6_AF_D1_Top.v
//# Author:     Earth People Technology
//# Date:       March 14, 2021
//# Revision:   A
//#
//# Development: EPT Platform Demonstration Project
//# Application: Altera Cyclone IV FPGA
//# Description: This file contains verilog code which will allow access
//#              to Active Transfer Library.
//#
//#
//#
//#************************************************************************
//#
//# Revision History:
//#         DATE        VERSION     DETAILS
//#         3/14/21     1           Created
//#
//#
//#
//##########################################################################
```

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

module EPT_4CE6_AF_D1_Top (


    input  wire                  CLK_66MHZ,
    input  wire                  RST,

    //UART COMMAND BUS
    output wire                  UART_OUT, //To external
    input wire                   UART_IN, //From external


    output wire    [7:0]         XIO_1,        //XIO -- UB
    input  wire    [5:0]         XIO_2,        //XIO -- UB
    output wire    [5:0]         XIO_3,        //XIO -- UB
    input  wire    [5:0]         XIO_4,        //XIO -- UB
    output wire    [5:0]         XIO_5,        //XIO -- UB
    output wire    [31:0]        XIO_6,        //XIO -- UB
    output wire    [3:0]         XIO_7,        //XIO -- UB

    input wire                   PB_SWITCH_1,
    input wire                   PB_SWITCH_2


    );
```

The interface from the library to the user code is two uni directional buses, UC_IN[22:0] and UC_OUT[20:0]. The UC_IN[22:0] bus is an output bus (from the library, input bus to the Active Modules) that is used channel data, address, length and control information to the Active Modules. The UC_OUT[21:0] bus is an input bus (to the library, output bus from the Active Modules) that is used to communicate data, address, length, and control information to the Active Modules.

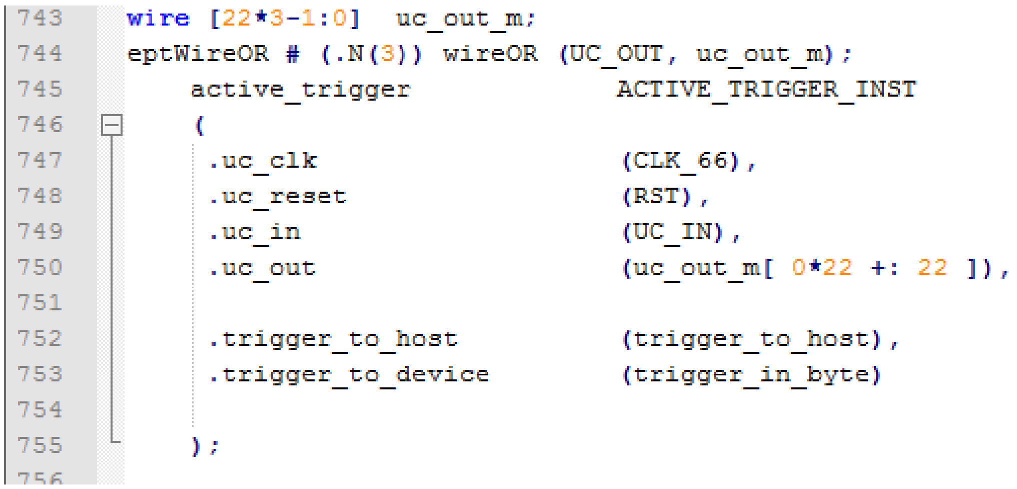


```
//-----------------------------------------------
// Instantiate the EPT Library
//-----------------------------------------------

 active_serial_library      ACTIVE_SERIAL_LIBRARY_INST
 (
 .aa                        (aa),
 .UART_OUT                  (UART_OUT),
 .UART_IN                   (uart_txd),

 .UC_IN                     (UC_IN),
 .UC_OUT                    (UC_OUT),

 .STATE_OUT                 (active_serial_state_out),
 .TEST_OUT                  (active_transfer_test_out)

 );
```

The UART signals are used to channel data, and control signals to the USB interface chip. These signals are connected directly to input and output pins of the FPGA.

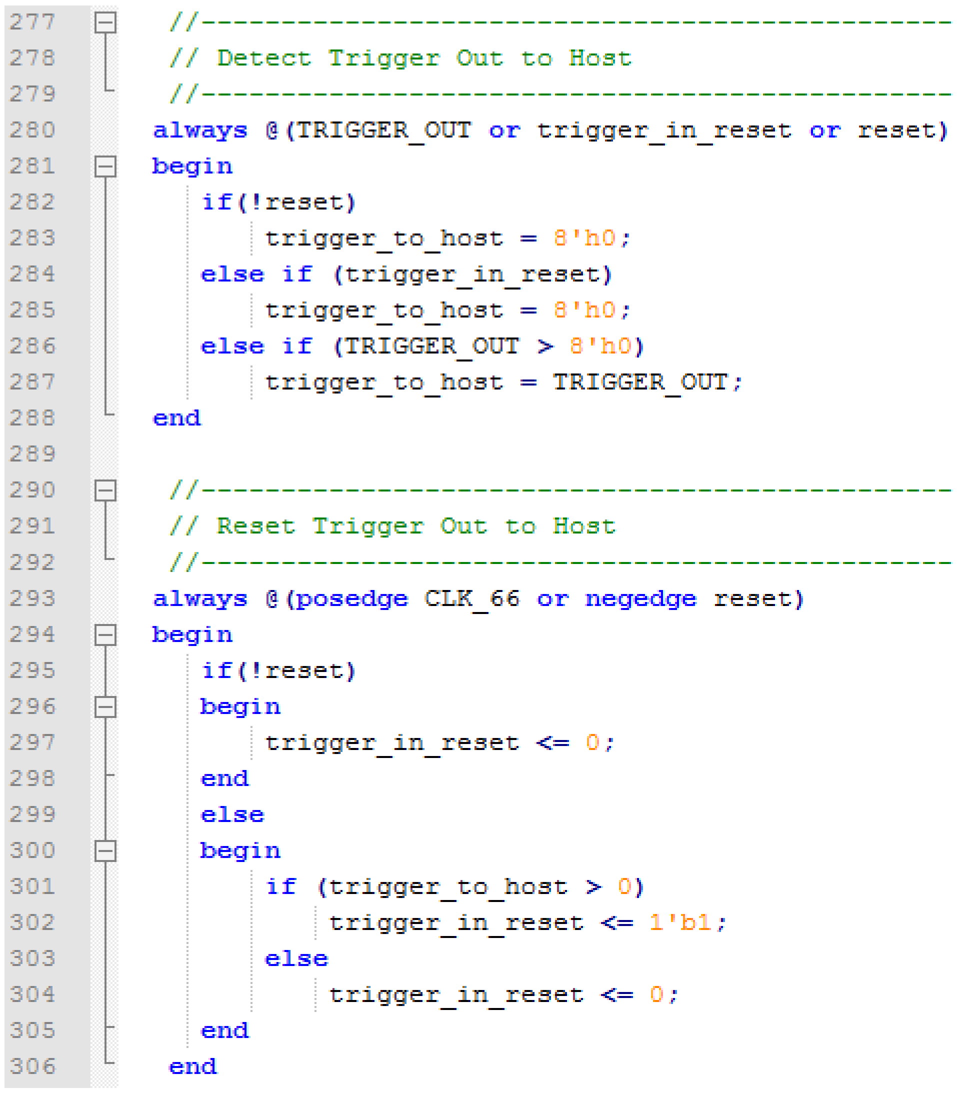
### 3.2.1 Active Trigger EndTerm

The Active Trigger has eight individual self resetting, active high, signals. These signals are used to send a momentary turn on/off command to Host/User code. The Active Trigger is not addressable so the module will be instantiated only once in the top level.



```
wire [22*3-1:0]  uc_out_m;
eptWireOR # (.N(3)) wireOR (UC_OUT, uc_out_m);
    active_trigger               ACTIVE_TRIGGER_INST
    (
     .uc_clk                     (CLK_66),
     .uc_reset                   (RST),
     .uc_in                      (UC_IN),
     .uc_out                     (uc_out_m[ 0*22 +: 22 ]),

     .trigger_to_host            (trigger_to_host),
     .trigger_to_device          (trigger_in_byte)

    );
```

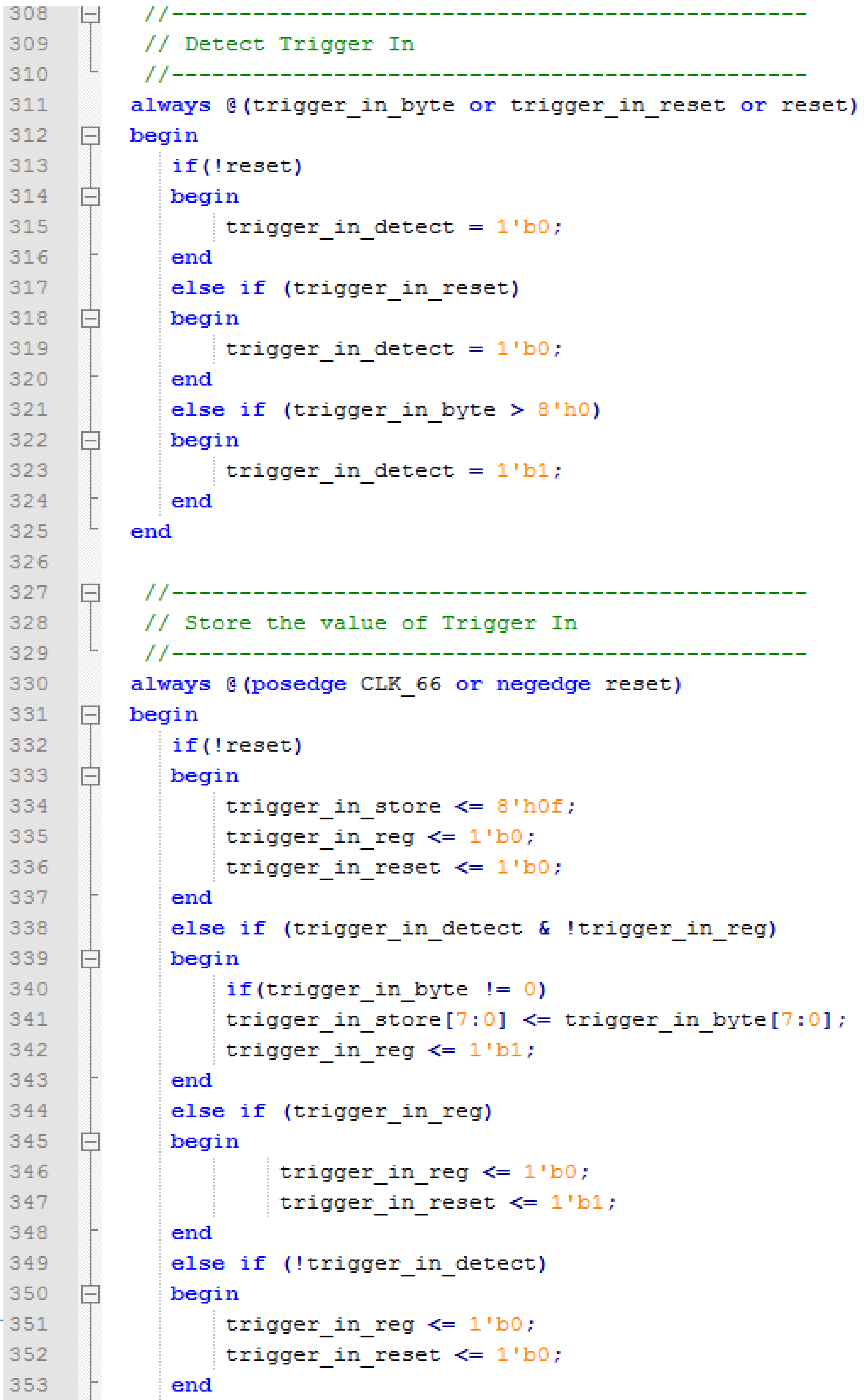
To send a trigger, decide which bit (or multiple bits) of the eight bits you want to send the trigger on. Then, set that bit (or bits) high. The Active Transfer Library will send a high on that trigger bit for one clock cycle (66 MHz), then reset itself to zero. The bit can stay high on the user code and does not need to be reset to zero. However, if the user sends another trigger using the trigger byte, then any bit that is set high will cause a trigger to occur on the Host side.



```
 //---------------------------------------------
 // Detect Trigger Out to Host
 //---------------------------------------------
always @(TRIGGER_OUT or trigger_in_reset or reset)
begin
   if(!reset)
       trigger_to_host = 8'h0;
   else if (trigger_in_reset)
       trigger_to_host = 8'h0;
   else if (TRIGGER_OUT > 8'h0)
       trigger_to_host = TRIGGER_OUT;
end

 //---------------------------------------------
 // Reset Trigger Out to Host
 //---------------------------------------------
always @(posedge CLK_66 or negedge reset)
begin
   if(!reset)
   begin
       trigger_in_reset <= 0;
   end
   else
   begin
       if (trigger_to_host > 0)
           trigger_in_reset <= 1'b1;
       else
           trigger_in_reset <= 0;
   end
 end
```

So, care should be used if the user code uses byte masks to send triggers. It is best to set only the trigger bits needed for a given time when sending triggers.

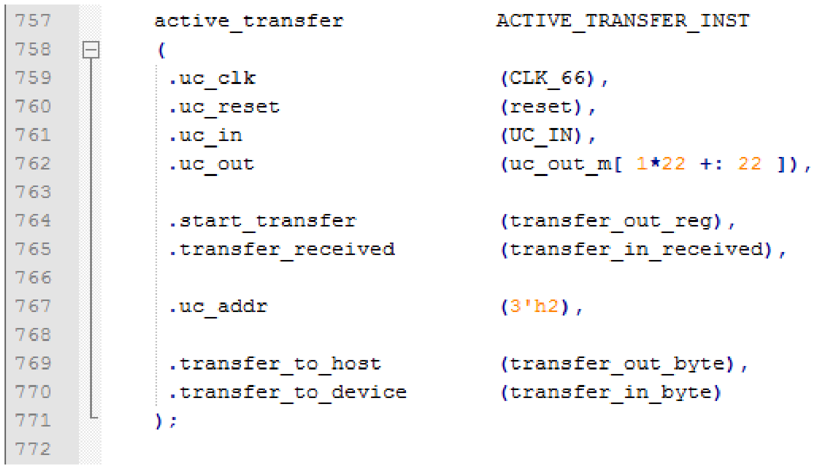
The user code must be setup to receive triggers from the Host. This can be done by using an asynchronous always block. Whenever a change occurs on a particular trigger bit (or bits), a conditional branch can detect if the trigger bit is for that block of code. Then, execute some code based on that trigger.



```
 //-----------------------------------------------
 // Detect Trigger In
 //-----------------------------------------------
always @(trigger_in_byte or trigger_in_reset or reset)
begin
   if(!reset)
   begin
       trigger_in_detect = 1'b0;
   end
   else if (trigger_in_reset)
   begin
       trigger_in_detect = 1'b0;
   end
   else if (trigger_in_byte > 8'h0)
   begin
       trigger_in_detect = 1'b1;
   end
end

 //-----------------------------------------------
 // Store the value of Trigger In
 //-----------------------------------------------
always @(posedge CLK_66 or negedge reset)
begin
   if(!reset)
   begin
       trigger_in_store <= 8'h0f;
       trigger_in_reg <= 1'b0;
       trigger_in_reset <= 1'b0;
   end
   else if (trigger_in_detect & !trigger_in_reg)
   begin
       if(trigger_in_byte != 0)
       trigger_in_store[7:0] <= trigger_in_byte[7:0];
       trigger_in_reg <= 1'b1;
   end
   else if (trigger_in_reg)
   begin
           trigger_in_reg <= 1'b0;
           trigger_in_reset <= 1'b1;
   end
   else if (!trigger_in_detect)
   begin
       trigger_in_reg <= 1'b0;
       trigger_in_reset <= 1'b0;
   end
```

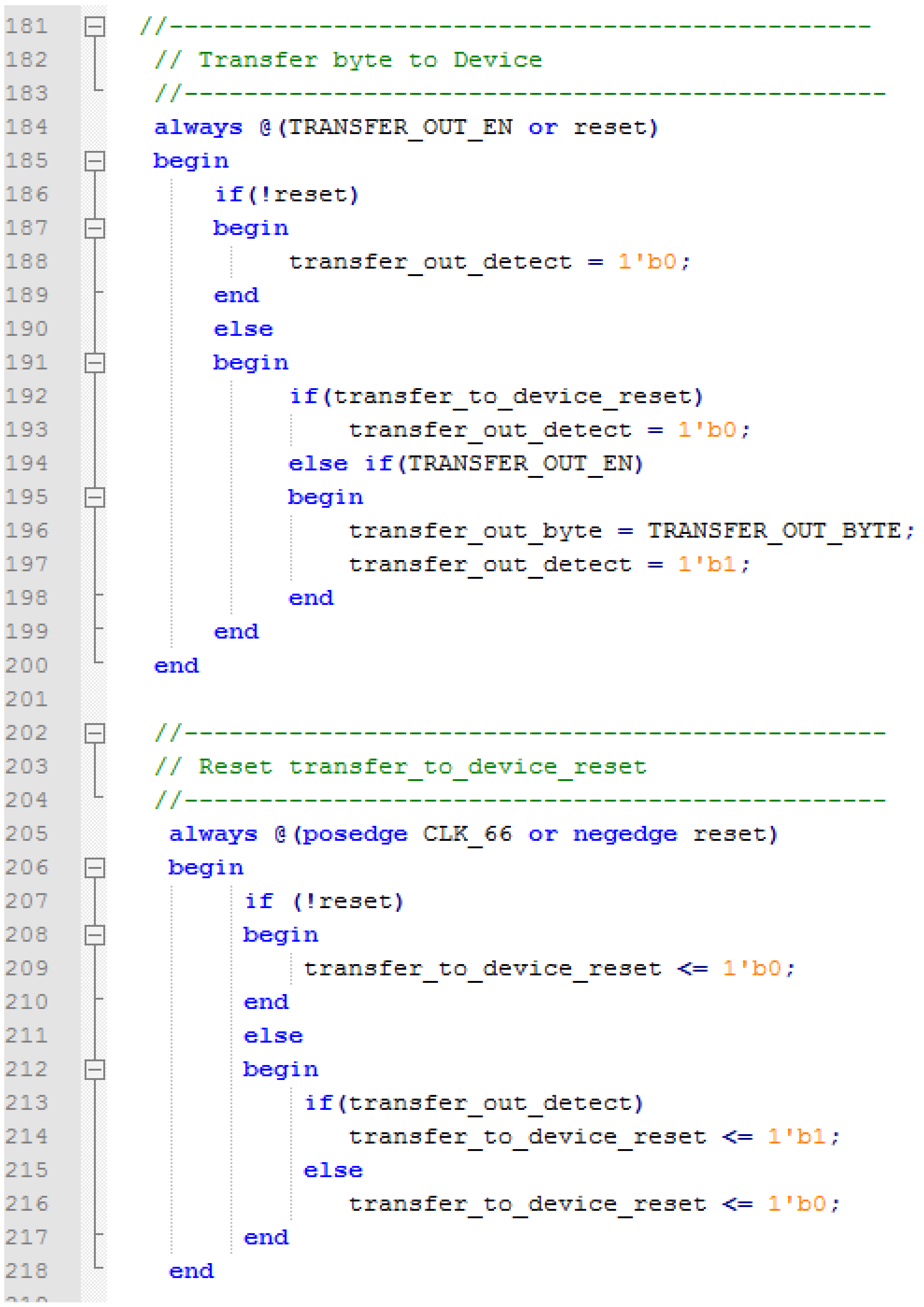
### 3.2.2 Active Transfer EndTerm

The Active Transfer module is used to send or receive a byte to/from the Host. This is useful when the user's microcontroller needs to send a byte from a measurement to the Host for display or processing. The Active Transfer module is addressable, so up to eight individual modules can be instantiated and separately addressed.



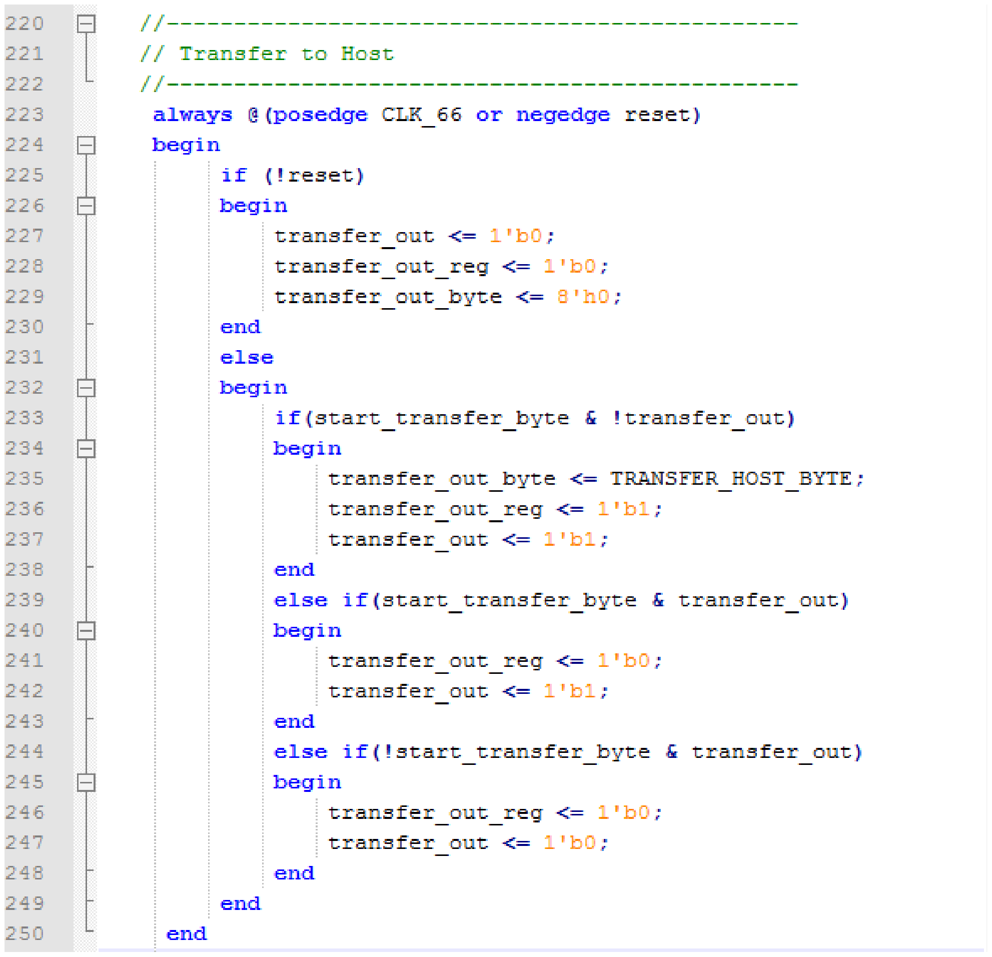
```
active_transfer                ACTIVE_TRANSFER_INST
(
 .uc_clk                       (CLK_66),
 .uc_reset                     (reset),
 .uc_in                        (UC_IN),
 .uc_out                       (uc_out_m[ 1*22 +: 22 ]),

 .start_transfer               (transfer_out_reg),
 .transfer_received            (transfer_in_received),

 .uc_addr                      (3'h2),

 .transfer_to_host             (transfer_out_byte),
 .transfer_to_device           (transfer_in_byte)
);
```

To send a byte to the Host, select the appropriate address that corresponds to an address on Host side. Place the byte in the "transfer_to_host" parameter, then strobe the "start_transfer" bit. Setting the "start_transfer" bit to high will send one byte from the "transfer_to_host" byte to the Host on the next clock high signal (66 MHz). The "start_transfer" bit can stay high for the duration of the operation of the device, the Active Transfer module will not send another byte. In order to send another byte, the user must cycle the "start_transfer" bit to low for a minimum of one clock cycle (66 MHz). After the "start_transfer" bit has been cycled low, the rising edge of the bit will cause the byte on the "transfer_to_host" parameter to transfer to the host.



```
//-----------------------------------------------
 // Transfer byte to Device
 //-----------------------------------------------
 always @(TRANSFER_OUT_EN or reset)
 begin
     if(!reset)
     begin
          transfer_out_detect = 1'b0;
     end
     else
     begin
          if(transfer_to_device_reset)
              transfer_out_detect = 1'b0;
          else if(TRANSFER_OUT_EN)
          begin
              transfer_out_byte = TRANSFER_OUT_BYTE;
              transfer_out_detect = 1'b1;
          end
     end
 end

 //-----------------------------------------------
 // Reset transfer_to_device_reset
 //-----------------------------------------------
  always @(posedge CLK_66 or negedge reset)
  begin
       if (!reset)
       begin
           transfer_to_device_reset <= 1'b0;
       end
       else
       begin
           if(transfer_out_detect)
              transfer_to_device_reset <= 1'b1;
           else
              transfer_to_device_reset <= 1'b0;
       end
  end
```

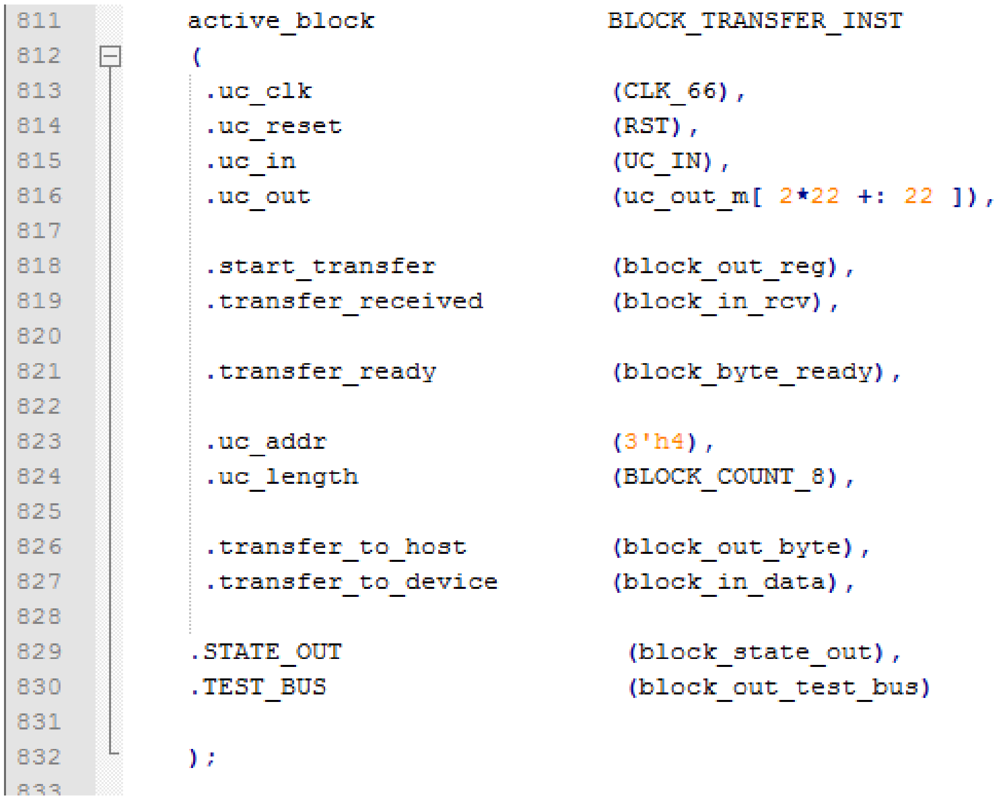
To receive a byte, the Active Host will send a byte using it's SDK. The user code must monitor the transfer_received port. The transfer_received port will assert high for one clock cycle (66 MHz) when a byte is ready for reading on the transfer_to_device port. User code should use an asynchronous always block to detect when the transfer_received port is asserted. Upon assertion, the user code should read the byte from the transfer_to_device port into a local register.



```
//-----------------------------------------------
// Transfer to Host
//-----------------------------------------------
 always @(posedge CLK_66 or negedge reset)
 begin
      if (!reset)
      begin
          transfer_out <= 1'b0;
          transfer_out_reg <= 1'b0;
          transfer_out_byte <= 8'h0;
      end
      else
      begin
          if(start_transfer_byte & !transfer_out)
          begin
              transfer_out_byte <= TRANSFER_HOST_BYTE;
              transfer_out_reg <= 1'b1;
              transfer_out <= 1'b1;
          end
          else if(start_transfer_byte & transfer_out)
          begin
              transfer_out_reg <= 1'b0;
              transfer_out <= 1'b1;
          end
          else if(!start_transfer_byte & transfer_out)
          begin
              transfer_out_reg <= 1'b0;
              transfer_out <= 1'b0;
          end
      end
  end
```

## 3.2.3 Active Block EndTerm

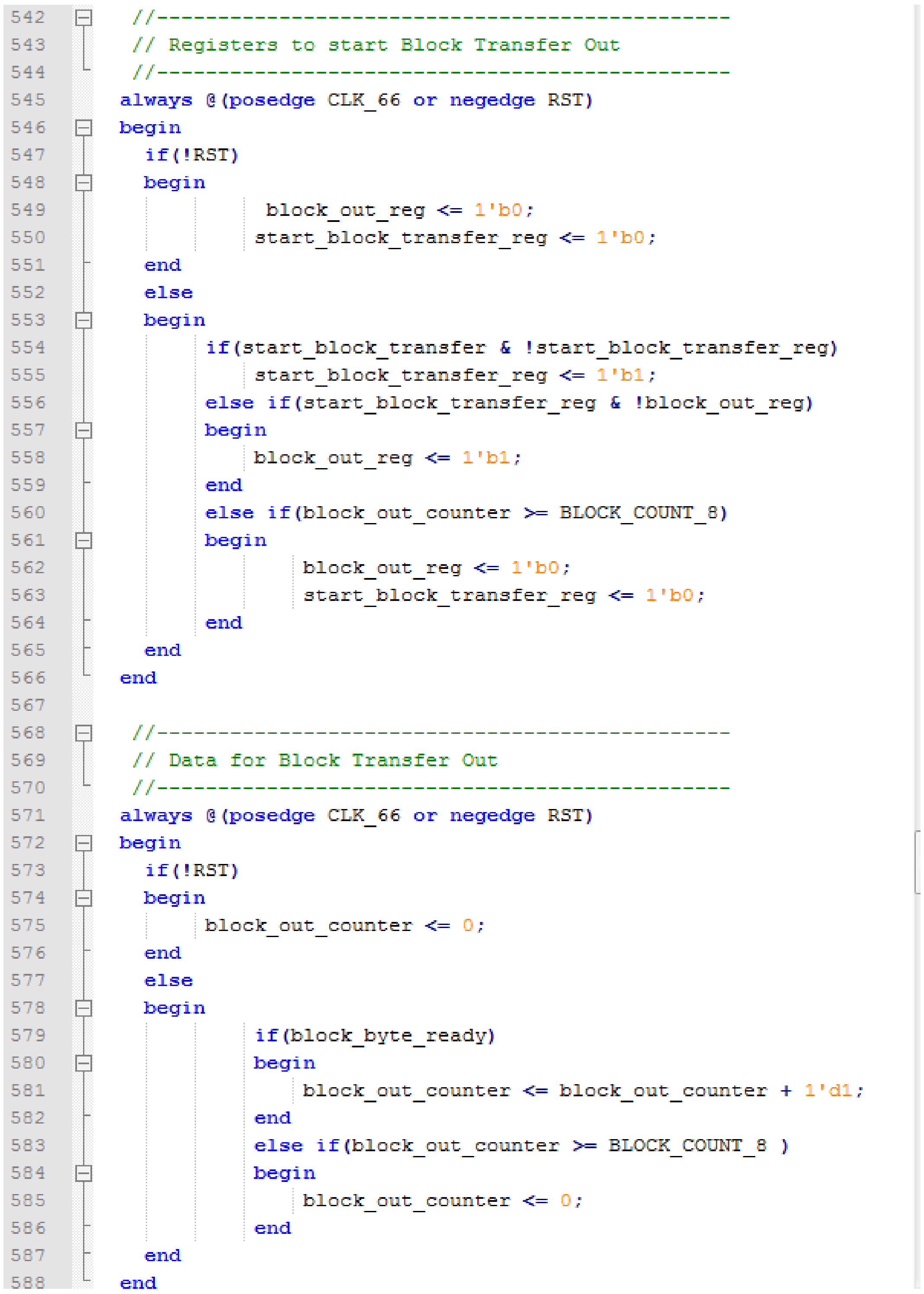
The Active Block module is designed to transfer blocks of data between Host and User Code and vice versa. This allows buffers of data to be transferred  with a minimal amount of code. The Active Block module is addressable, so up to eight individual modules can be instantiated and separately addressed. The length of the block to be transferred must also be specified in the uc_length port.



```
active_block                BLOCK_TRANSFER_INST
(
 .uc_clk                     (CLK_66),
 .uc_reset                   (RST),
 .uc_in                      (UC_IN),
 .uc_out                     (uc_out_m[ 2*22 +: 22 ]),

 .start_transfer             (block_out_reg),
 .transfer_received          (block_in_rcv),

 .transfer_ready             (block_byte_ready),

 .uc_addr                    (3'h4),
 .uc_length                  (BLOCK_COUNT_8),

 .transfer_to_host           (block_out_byte),
 .transfer_to_device         (block_in_data),

.STATE_OUT                    (block_state_out),
.TEST_BUS                     (block_out_test_bus)

);
```

To send a block, it’s best to have buffer filled in a previous transaction, Then assert the start_transfer bit. This method is opposed to collecting and processing data bytes after the start_transfer bit has been asserted and data is being sent to the Host.

Once the buffer to send is filled with the requisite amount of data, the address and buffer length should be written to the uc_addr and uc_length ports. Set the start_transfer bit high, the user code should monitor the transfer_ready port. At the rising edge of the transfer_ready port, the byte at transfer_to_host  port is transferred to the USB chip. Once this occurs, the user code should copy the next byte in the buffer to transfer_to_host port. On the next rising edge of transfer-ready, the byte at transfer_to_host will be transferred to theUSB chip. This process continues until the number of bytes desicribed by the uc_length have been transferred into the USB chip.



```
 //-----------------------------------------------
 // Registers to start Block Transfer Out
 //-----------------------------------------------
always @(posedge CLK_66 or negedge RST)
begin
  if(!RST)
  begin
          block_out_reg <= 1'b0;
          start_block_transfer_reg <= 1'b0;
  end
  else
  begin
      if(start_block_transfer & !start_block_transfer_reg)
          start_block_transfer_reg <= 1'b1;
      else if(start_block_transfer_reg & !block_out_reg)
      begin
          block_out_reg <= 1'b1;
      end
      else if(block_out_counter >= BLOCK_COUNT_8)
      begin
              block_out_reg <= 1'b0;
              start_block_transfer_reg <= 1'b0;
      end
  end
end

 //-----------------------------------------------
 // Data for Block Transfer Out
 //-----------------------------------------------
always @(posedge CLK_66 or negedge RST)
begin
  if(!RST)
  begin
      block_out_counter <= 0;
  end
  else
  begin
          if(block_byte_ready)
          begin
              block_out_counter <= block_out_counter + 1'd1;
          end
          else if(block_out_counter >= BLOCK_COUNT_8 )
          begin
              block_out_counter <= 0;
          end
  end
end
```

To receive a buffer from the Host, the user code should monitor the transfer_received port for assertion. When the bit is asserted, the next rising edge of transfer_ready will indicate that the byte at transfer_to_device is ready for the user code to read.

[Add code snippet showing Active Block Module bytes received by the user code]

## *3.3 Timing Diagram for Active Transfer EndTerms*

The Active Transfer Library uses the 66 MHz clock to organize the transfers to Host and transfer to Device. The timing of the transfers depends on this clock and the specifications of the USB chip. Users should use the timing diagrams to ensure proper operation of user code in data transfer.

### 3.3.1 Active Trigger EndTerm Timing

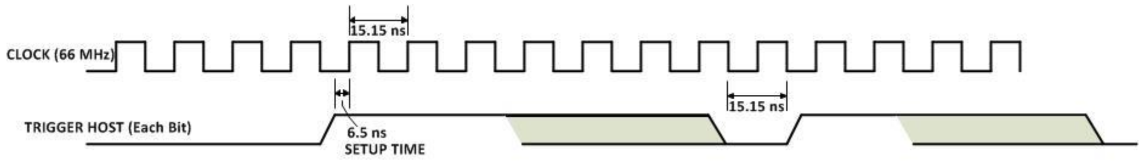


Figure xx Active Trigger to Host Timing

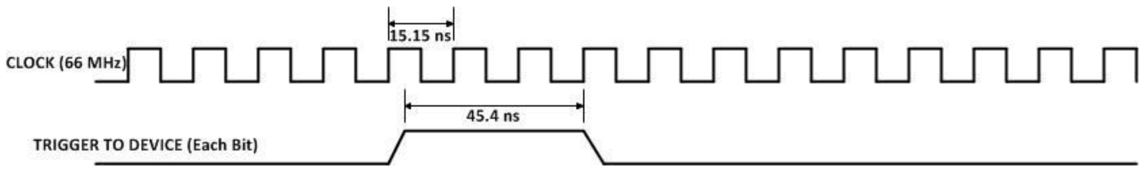


Figure xx Active Trigger to Device Timing

### 3.3.2 Active Transfer EndTerm Timing

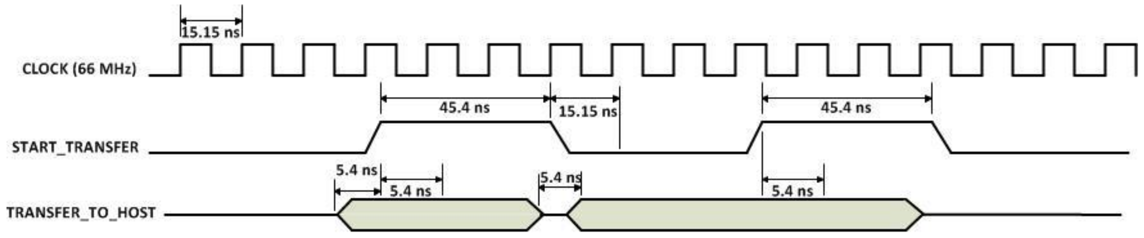


Figure xx Active Transfer To Host Timing

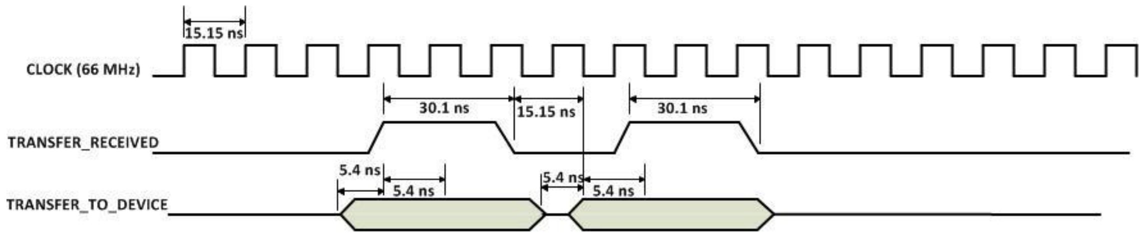


Figure xx Active Transfer To Device Timing

### 3.3.3 Active Block EndTerm Timing

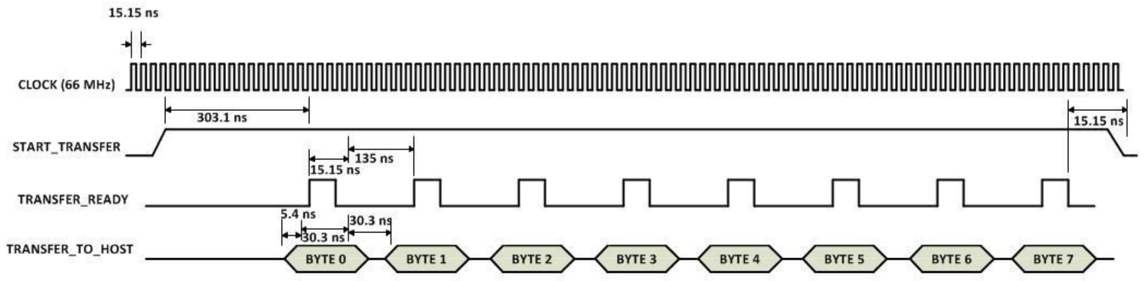


Figure xx Active Block To Host Timing

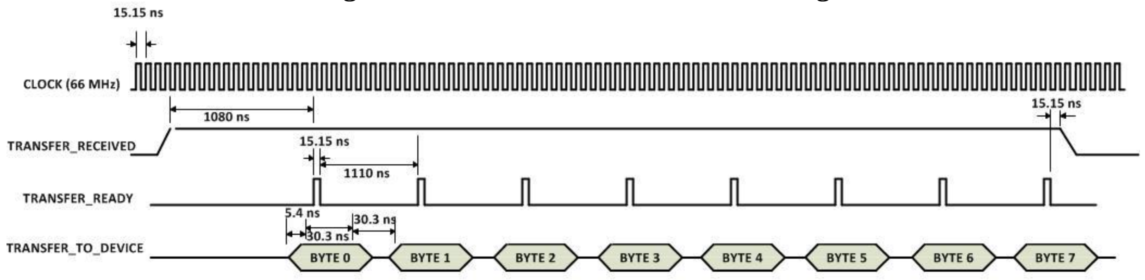
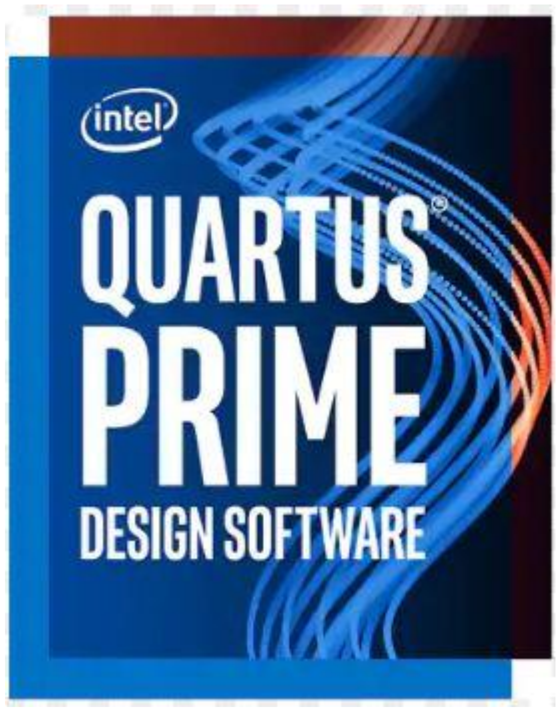


Figure xx Active Block To Device Timing

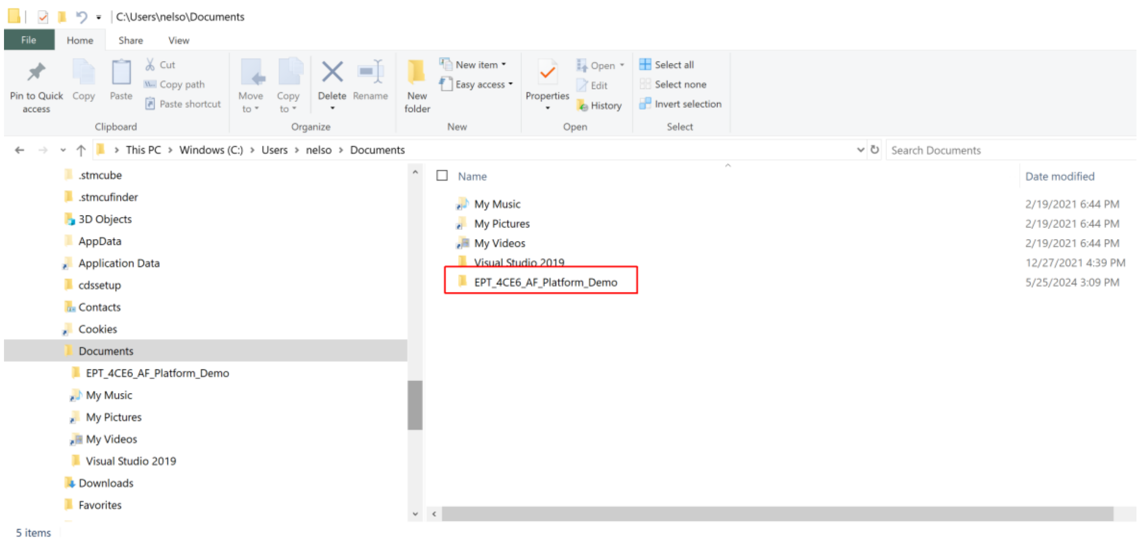
# 4 Compiling, Synthesizing, and Programming FPGA



The FPGA on the EPT-4CE6-AF-D2 can be programmed with the Active Transfer Library and custom HDL code created by the user. Programming the FPGA requires the use of the Quartus Prime software and a standard USB cable. There are no extra parts to buy, just plug in the USB cable. Once the user HDL code is written according to the syntax rules of the language (Verilog and VHDL) it can be compiled and synthesized using the Quartus Prime 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.

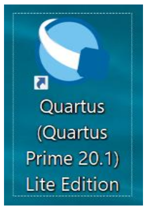
## *4.1 Setting up the Project and Compiling*

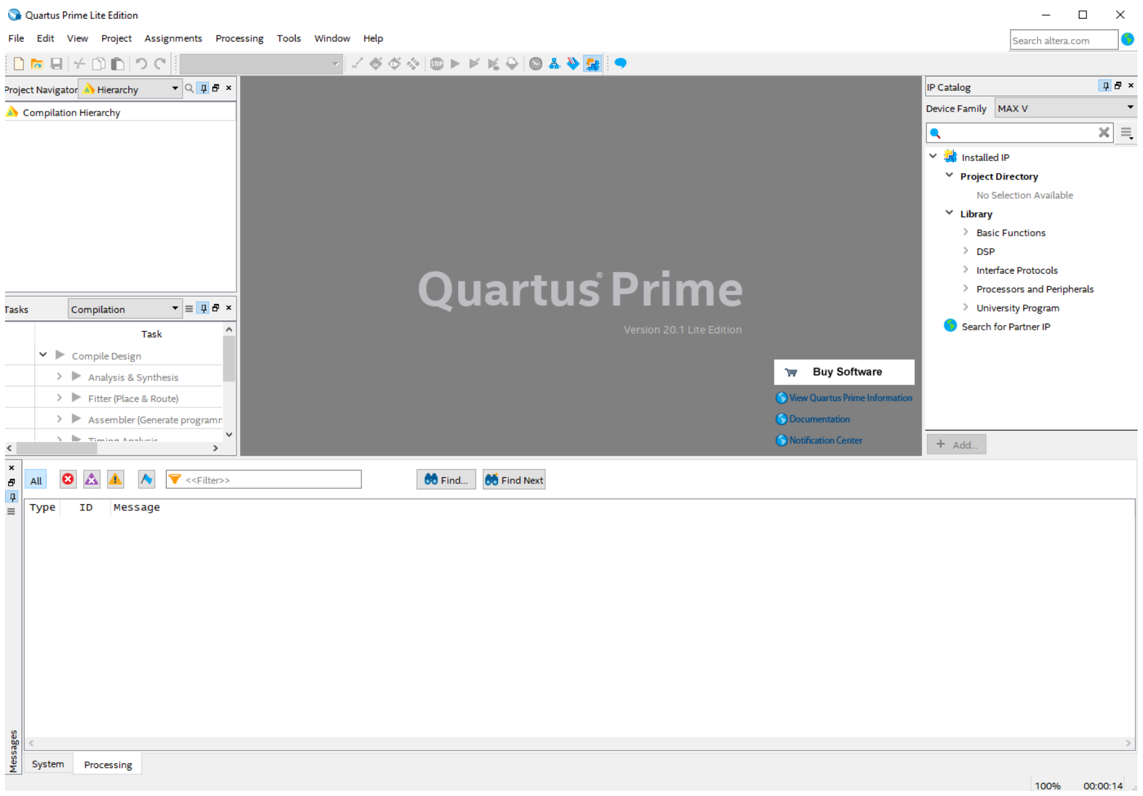
Once the HDL code (Verilog or VHDL) is written and verified using a simulator, a project can be created using Quartus Prime. Writing the HDL code and simulating it will be covered in later sections. Bring up Quartus Prime, then use Windows Explorer to browse to C:\Users\nelso\Documents\ to create a new directory called: “EPT_4CE6_AF_Platform_Demo”.



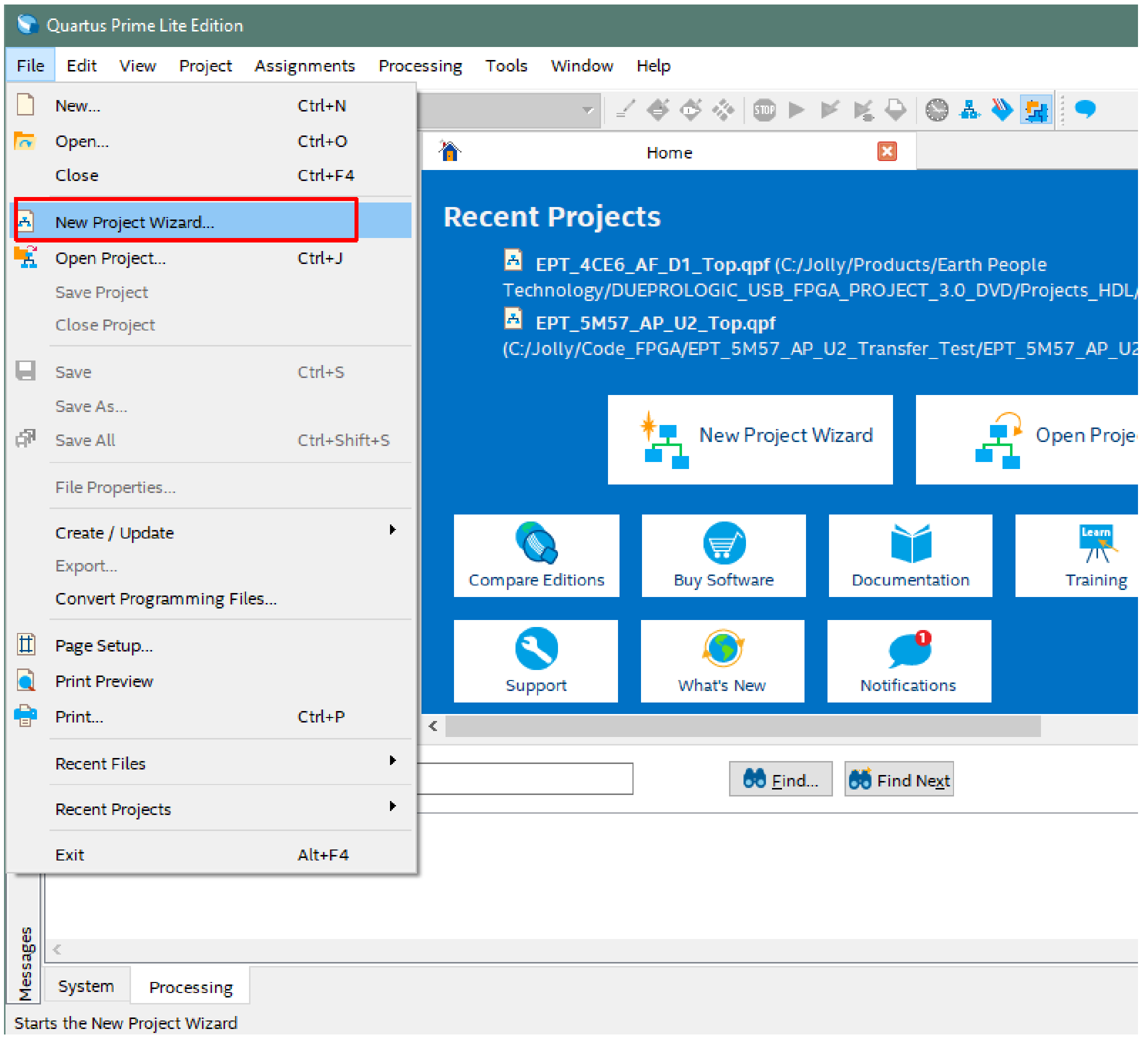
| Name | Date modified |
|---|---|
| My Music | 2/19/2021 6:44 PM |
| My Pictures | 2/19/2021 6:44 PM |
| My Videos | 2/19/2021 6:44 PM |
| Visual Studio 2019 | 12/27/2021 4:39 PM |
| EPT_4CE6_AF_Platform_Demo | 5/25/2024 3:09 PM |

Open Quartus Prime by clicking on the icon.





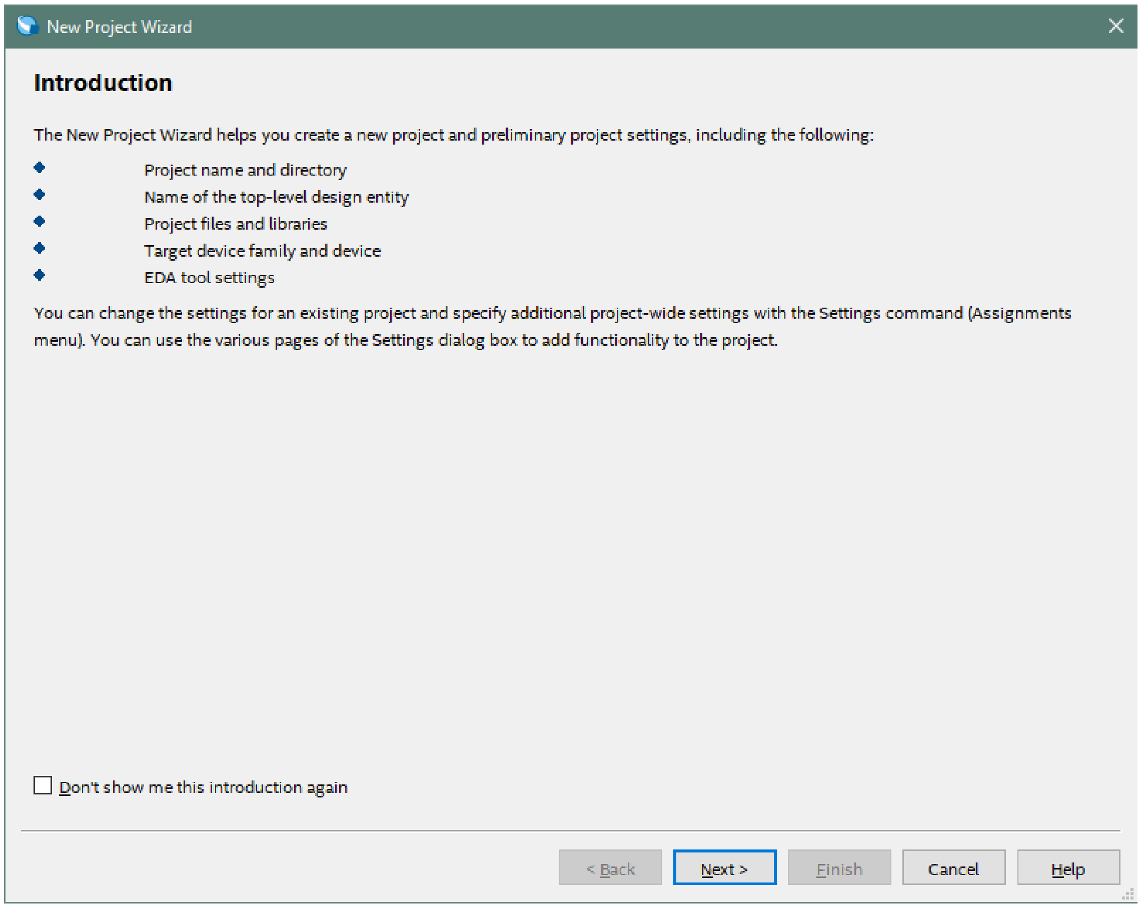
Under Quartus, Select File->New Project Wizard. The Wizard will walk you through setting up files and directories for your project.



At the Top-Level Entity page, browse to the

C:\Users\nelso\Documents\ EPT_4CE6_AF_Platform_Demo

directory to store your project. Type in a name for your project "EPT_4CE6_AF_D1_Top".

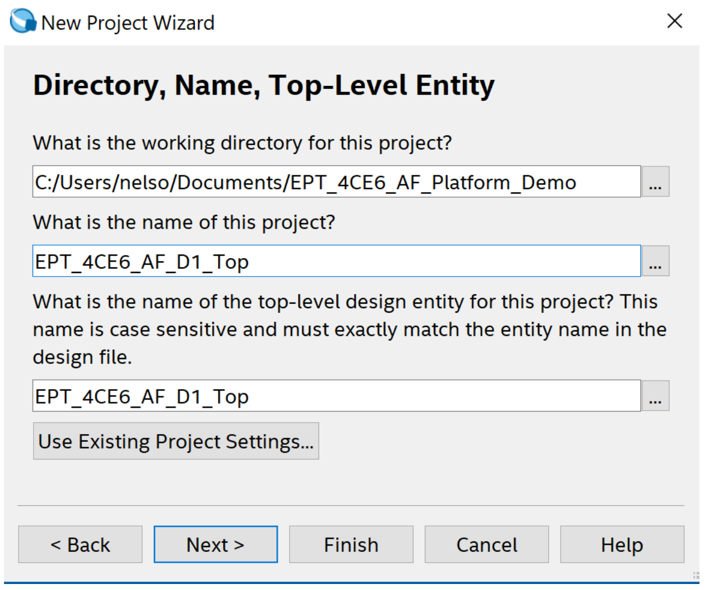


Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/Users/nelso/Documents/EPT_4CE6_AF_Platform_Demo

What is the name of this project?

EPT_4CE6_AF_D1_Top

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

EPT_4CE6_AF_D1_Top

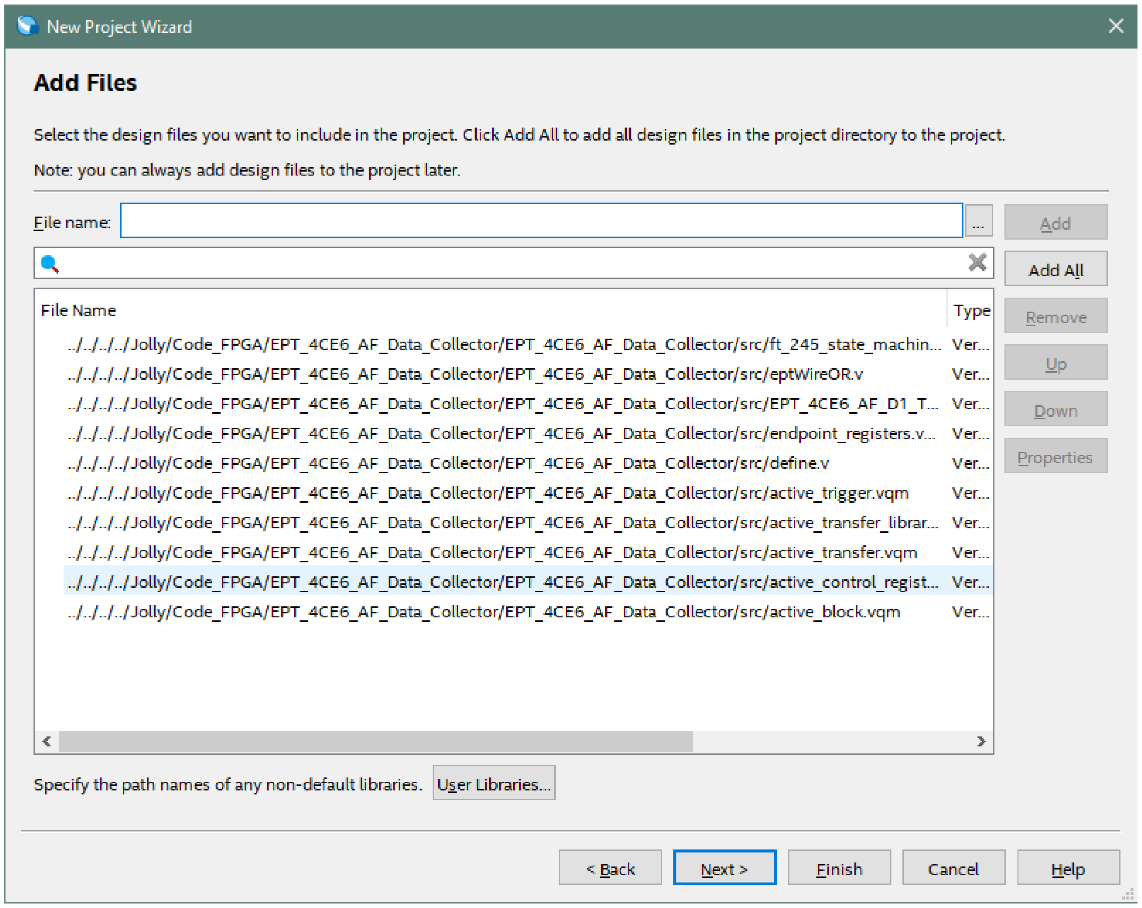
Select Next. At the Add Files window: Browse to the

Projects_HDL\EPT_4CE6_AF_Platform_Demo\src

folder of the EPT FPGA Development System DVD. Copy the files from the \src directory.

- active_block.v
- active_transfer.v
- active_trigger.v
- active_transfer_uart.v
- active_serial_library.v
- define.v
- ft_245_state_machine.v
- endpoint_registers.v
- eptWireOr.v
- flipflop.v
- image_block_in.v
- image_ept_logo.v
- image_face_1.v
- image_tri_wave.v
- led_update.v

- lfsr.v
- mux.v
- read_control_logic.v
- serial_clock.v
- sync_fifo.v
- uart_receiver.v
- uart_top.v
- uart_transmitter.v
- write_control_logic.v
- EPT_4CE6_AF_D1_Top.v



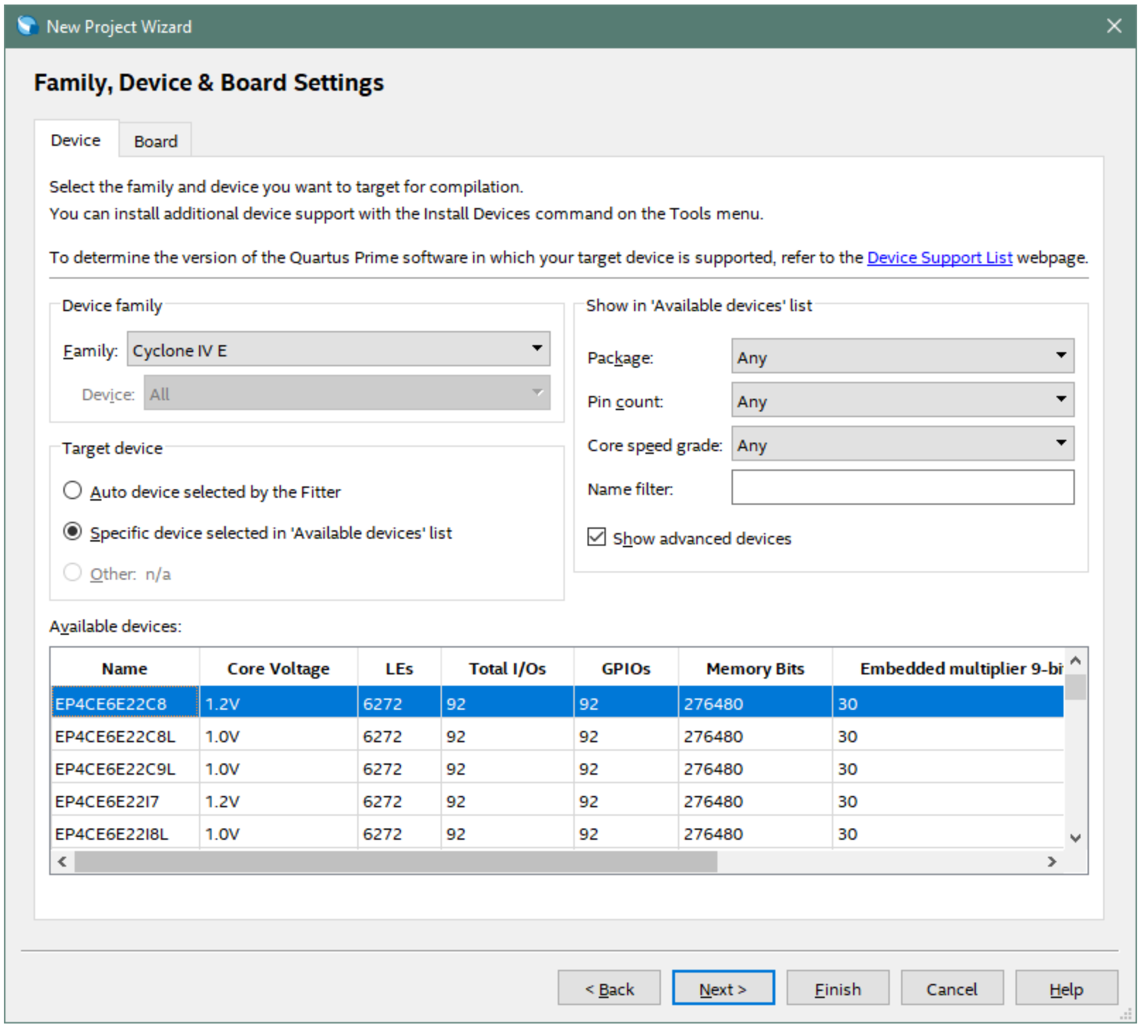
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

Note: you can always add design files to the project later.

| File Name | Type |
|---|---|
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/ft_245_state_machin... | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/eptWireOR.v | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/EPT_4CE6_AF_D1_T... | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/endpoint_registers.v... | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/define.v | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/active_trigger.vqm | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/active_transfer_librar... | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/active_transfer.vqm | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/active_control_regist... | Ver... |
| ../../../../Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector/EPT_4CE6_AF_Data_Collector/src/active_block.vqm | Ver... |

Specify the path names of any non-default libraries.

Select Next, at the Device Family group, select Cyclone IV for Family. In the Available Devices group, browse down to EP4CE6E22C8 for Name.



New Project Wizard

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: Cyclone IV E

Device: All

Target device

Auto device selected by the Fitter [✓ Specific device selected in 'Available devices' list] Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

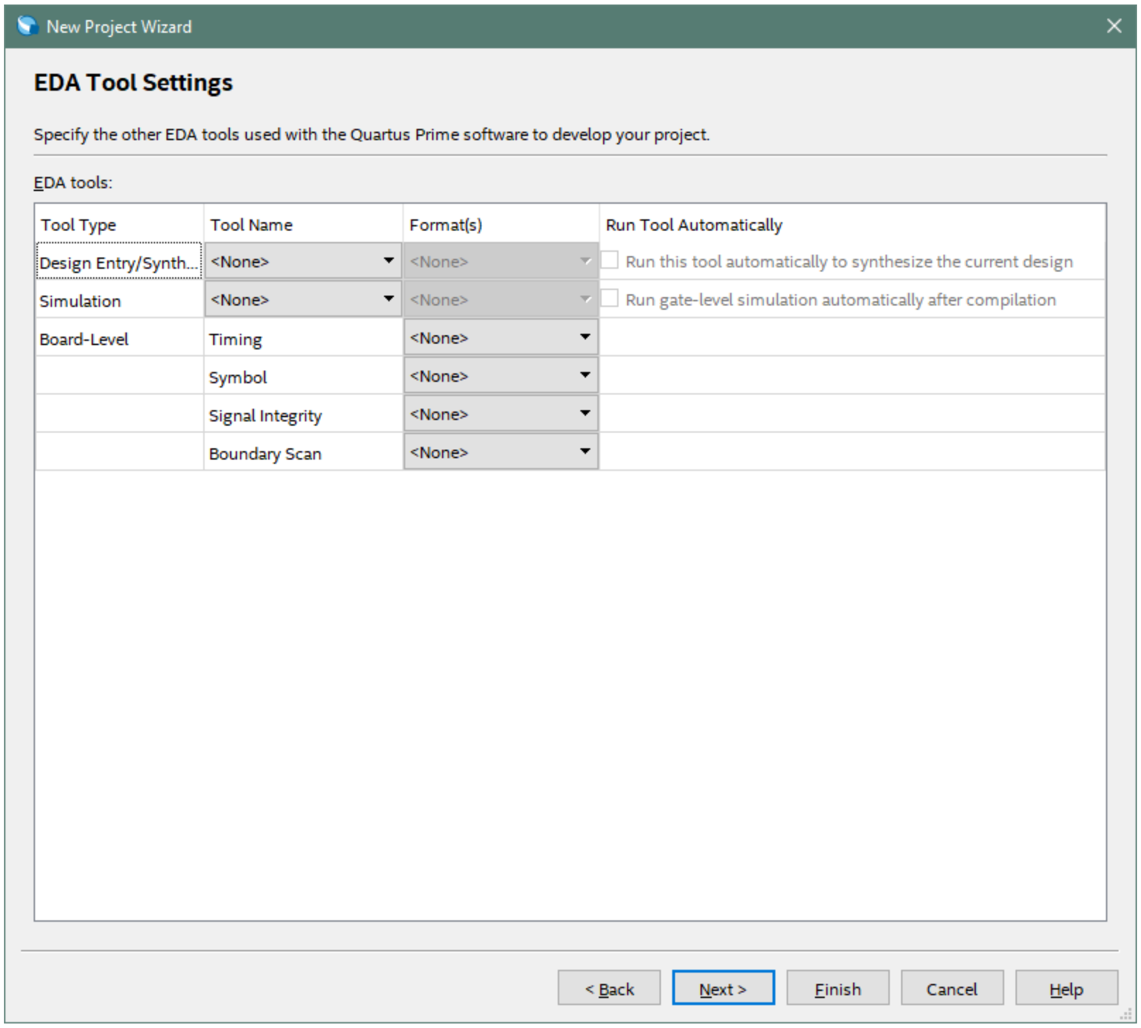
Name filter:

[✓] Show advanced devices

Available devices:

| Name | Core Voltage | LEs | Total I/Os | GPIOs | Memory Bits | Embedded multiplier 9-bi |
|---|---|---|---|---|---|---|
| EP4CE6E22C8 | 1.2V | 6272 | 92 | 92 | 276480 | 30 |
| EP4CE6E22C8L | 1.0V | 6272 | 92 | 92 | 276480 | 30 |
| EP4CE6E22C9L | 1.0V | 6272 | 92 | 92 | 276480 | 30 |
| EP4CE6E22I7 | 1.2V | 6272 | 92 | 92 | 276480 | 30 |
| EP4CE6E22I8L | 1.0V | 6272 | 92 | 92 | 276480 | 30 |

Select Next, leave defaults for the EDA Tool Settings.



New Project Wizard

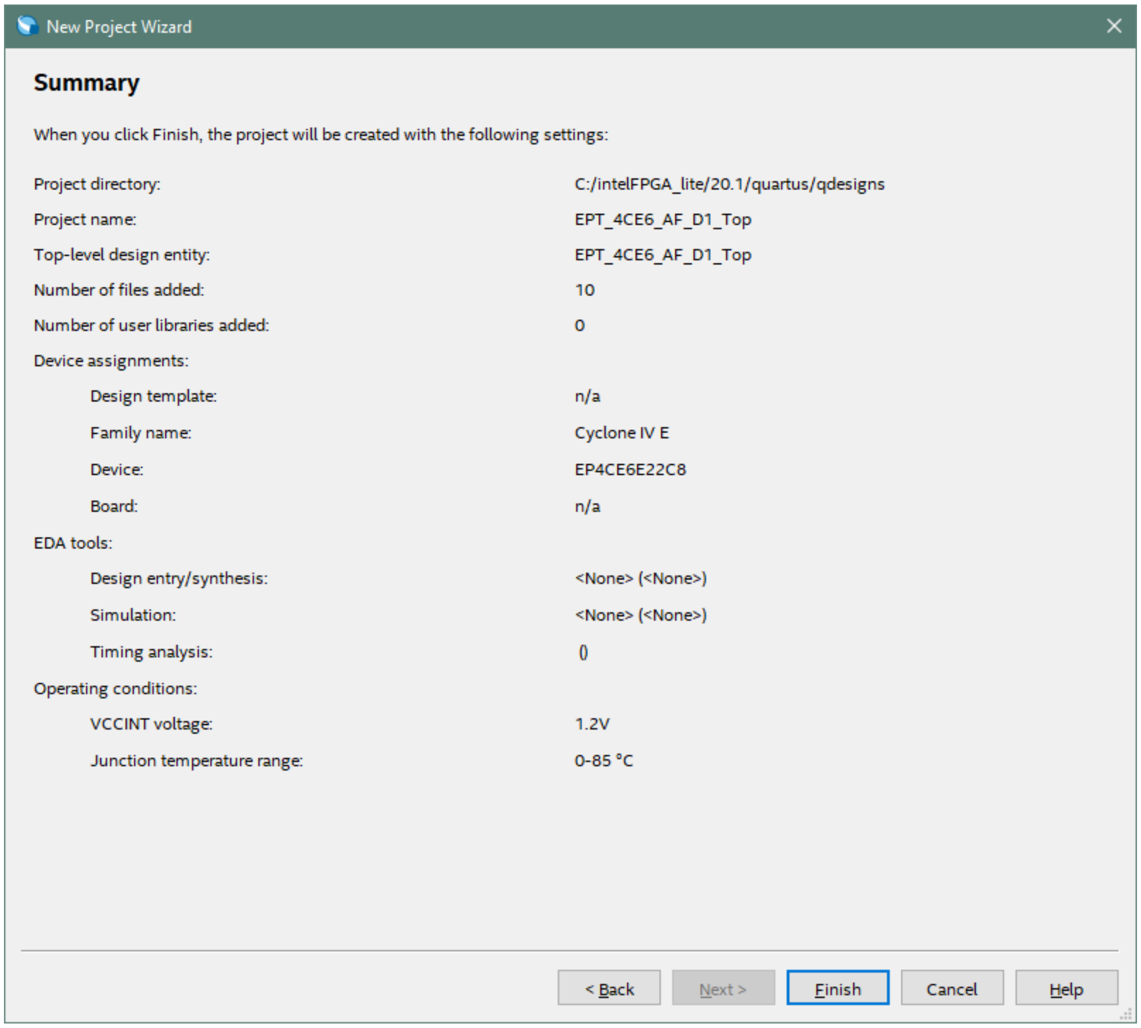
**EDA Tool Settings**

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

| Tool Type | Tool Name | Format(s) | Run Tool Automatically |
|---|---|---|---|
| Design Entry/Synth... | <None> | <None> | ☐ Run this tool automatically to synthesize the current design |
| Simulation | <None> | <None> | ☐ Run gate-level simulation automatically after compilation |
| Board-Level | Timing | <None> | |
| | Symbol | <None> | |
| | Signal Integrity | <None> | |
| | Boundary Scan | <None> | |

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



**Summary**

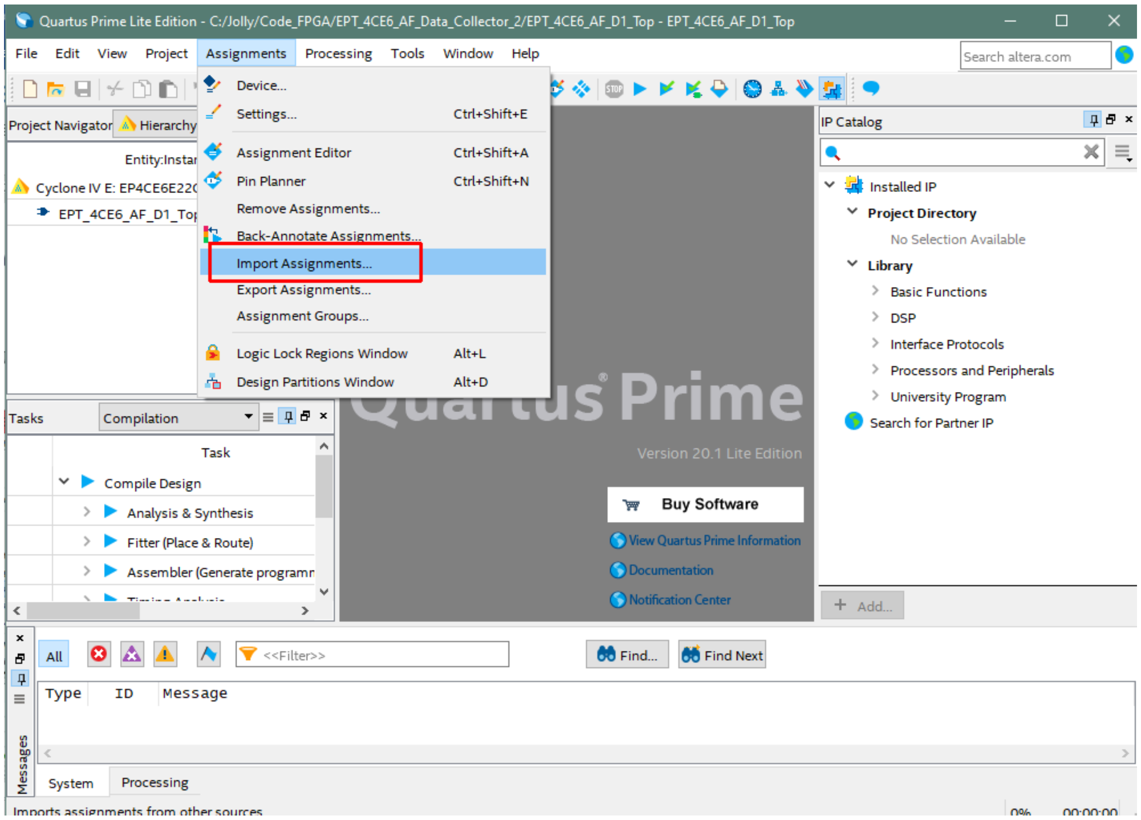
When you click Finish, the project will be created with the following settings:

| | |
|---|---|
| Project directory: | C:/intelFPGA_lite/20.1/quartus/qdesigns |
| Project name: | EPT_4CE6_AF_D1_Top |
| Top-level design entity: | EPT_4CE6_AF_D1_Top |
| Number of files added: | 10 |
| Number of user libraries added: | 0 |
| Device assignments: | |
| Design template: | n/a |
| Family name: | Cyclone IV E |
| Device: | EP4CE6E22C8 |
| Board: | n/a |
| EDA tools: | |
| Design entry/synthesis: | <None> (<None>) |
| Simulation: | <None> (<None>) |
| Timing analysis: | 0 |
| Operating conditions: | |
| VCCINT voltage: | 1.2V |
| Junction temperature range: | 0-85 °C |

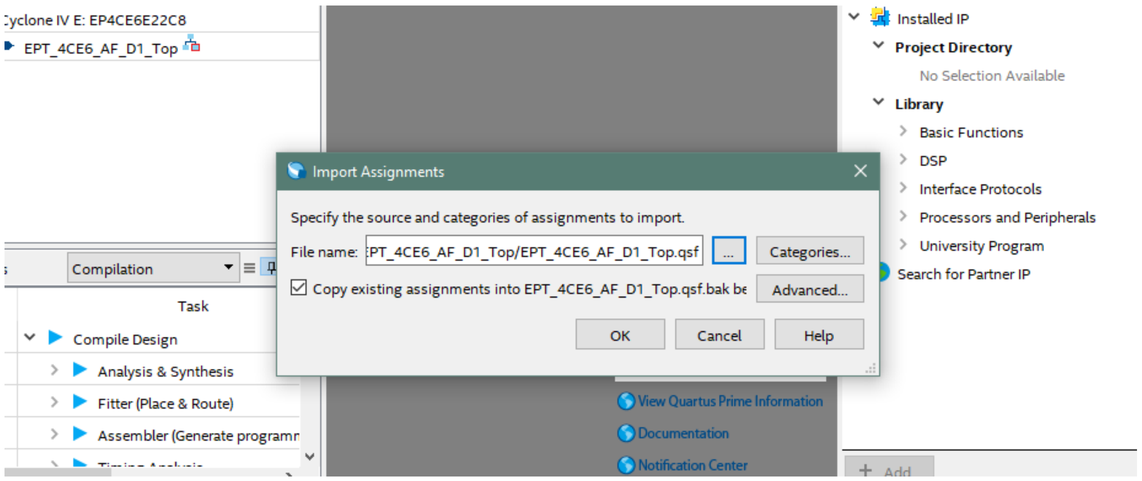
Next, we will select the pins and synthesize the project.

### 4.1.1 Selecting Pins and Synthesizing

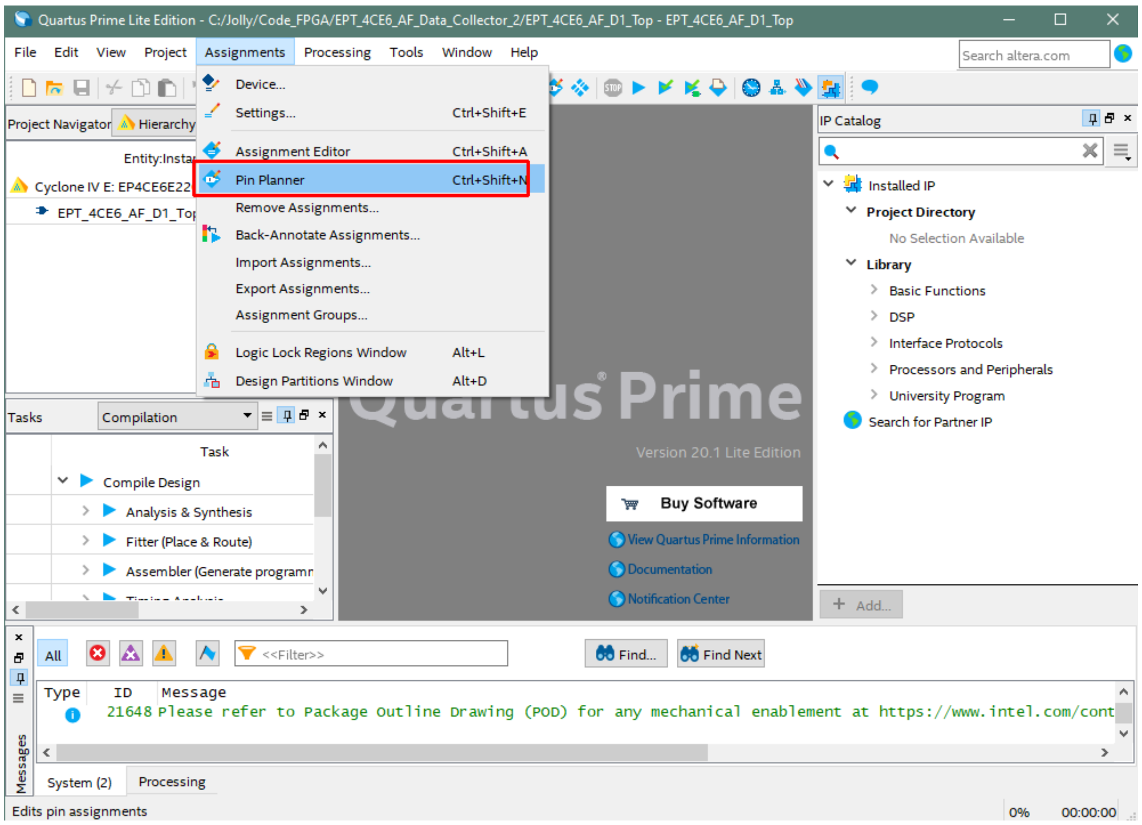
With the project created, we need to assign pins to the project. The signals defined in the top level file (in this case: EPT_4CE6_AF_D1_Top.v) will connect directly to pins on the FPGA. The Pin Planner Tool from Quartus Prime will add the pins and check to verify that our pin selections do not violate 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_Transfer_Demo \ EPT_4CE6_AF_D1_TOP folder of the EPT FPGA Development System DVD. Select the “EPT_4CE6_AF_D1_Top.qsf” file.

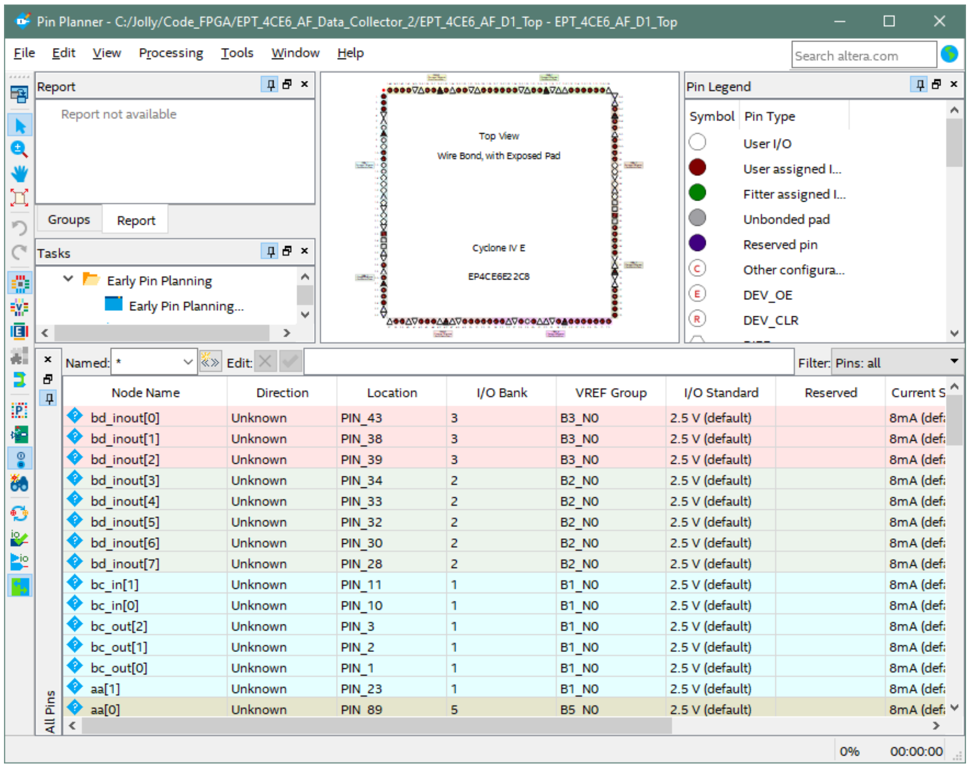

Import Assignments
Specify the source and categories of assignments to import.
File name: :PT_4CE6_AF_D1_Top/EPT_4CE6_AF_D1_Top.qsf
Categories...
Copy existing assignments into EPT_4CE6_AF_D1_Top.qsf.bak be
Advanced...
OK
Cancel
Help

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



```
Type  ID    Message
      21648 Please refer to Package Outline Drawing (POD) for any mechanical enablement at https://www.intel.com/cont
```

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.

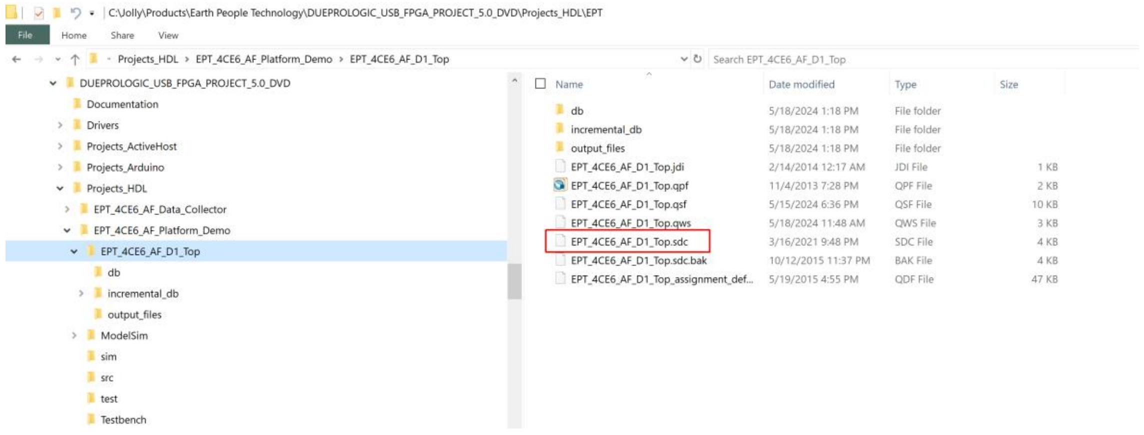


| Node Name | Direction | Location | I/O Bank | VREF Group | I/O Standard | Reserved | Current S |
|---|---|---|---|---|---|---|---|
| bd_inout[0] | Unknown | PIN_43 | 3 | B3_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[1] | Unknown | PIN_38 | 3 | B3_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[2] | Unknown | PIN_39 | 3 | B3_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[3] | Unknown | PIN_34 | 2 | B2_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[4] | Unknown | PIN_33 | 2 | B2_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[5] | Unknown | PIN_32 | 2 | B2_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[6] | Unknown | PIN_30 | 2 | B2_N0 | 2.5 V (default) | | 8mA (def: |
| bd_inout[7] | Unknown | PIN_28 | 2 | B2_N0 | 2.5 V (default) | | 8mA (def: |
| bc_in[1] | Unknown | PIN_11 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| bc_in[0] | Unknown | PIN_10 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| bc_out[2] | Unknown | PIN_3 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| bc_out[1] | Unknown | PIN_2 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| bc_out[0] | Unknown | PIN_1 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| aa[1] | Unknown | PIN_23 | 1 | B1_N0 | 2.5 V (default) | | 8mA (def: |
| aa[0] | Unknown | PIN_89 | 5 | B5_N0 | 2.5 V (default) | | 8mA (def: |

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-4CE6-AF-D1_TOP folder of the EPT FPGA Development System DVD.  Select the “EPT_4CE6_AF_D1_Top.sdc” file.


C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_5.0_DVD\Projects_HDL\EPT
File
Home
Share
View
Projects_HDL > EPT_4CE6_AF_Platform_Demo > EPT_4CE6_AF_D1_Top
Search EPT_4CE6_AF_D1_Top

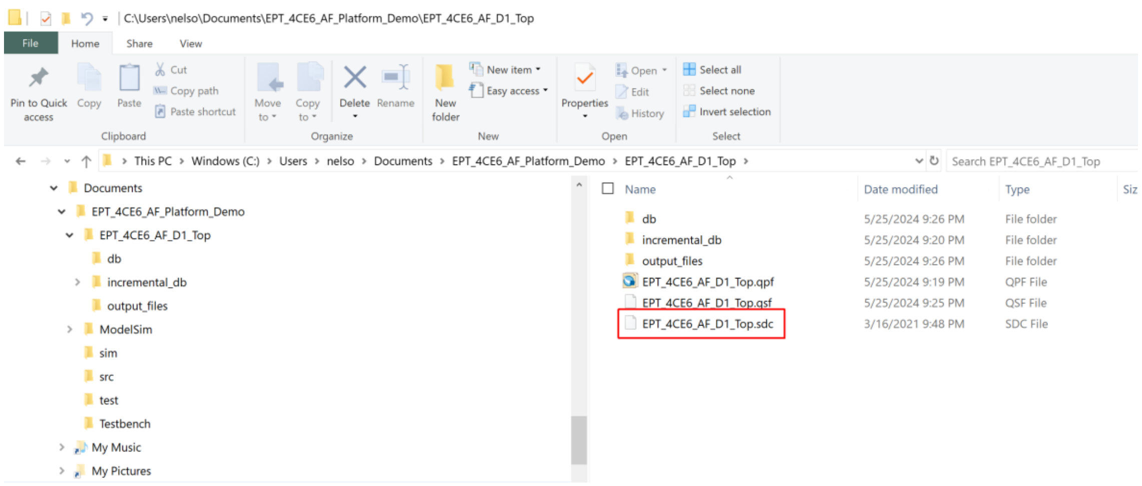
DUEPROLOGIC_USB_FPGA_PROJECT_5.0_DVD
- Documentation
- Drivers
- Projects_ActiveHost
- Projects_Arduino
- Projects_HDL
  - EPT_4CE6_AF_Data_Collector
  - EPT_4CE6_AF_Platform_Demo
    - EPT_4CE6_AF_D1_Top
      - db
      - incremental_db
      - output_files
    - ModelSim
    - sim
    - src
    - test
    - Testbench

| Name | Date modified | Type | Size |
|---|---|---|---|
| db | 5/18/2024 1:18 PM | File folder | |
| incremental_db | 5/18/2024 1:18 PM | File folder | |
| output_files | 5/18/2024 1:18 PM | File folder | |
| EPT_4CE6_AF_D1_Top.jdi | 2/14/2014 12:17 AM | JDI File | 1 KB |
| EPT_4CE6_AF_D1_Top.qpf | 11/4/2013 7:28 PM | QPF File | 2 KB |
| EPT_4CE6_AF_D1_Top.qsf | 5/15/2024 6:36 PM | QSF File | 10 KB |
| EPT_4CE6_AF_D1_Top.qws | 5/18/2024 11:48 AM | QWS File | 3 KB |
| EPT_4CE6_AF_D1_Top.sdc | 3/16/2021 9:48 PM | SDC File | 4 KB |
| EPT_4CE6_AF_D1_Top.sdc.bak | 10/12/2015 11:37 PM | BAK File | 4 KB |
| EPT_4CE6_AF_D1_Top_assignment_def... | 5/19/2015 4:55 PM | QDF File | 47 KB |

Copy the file and browse to

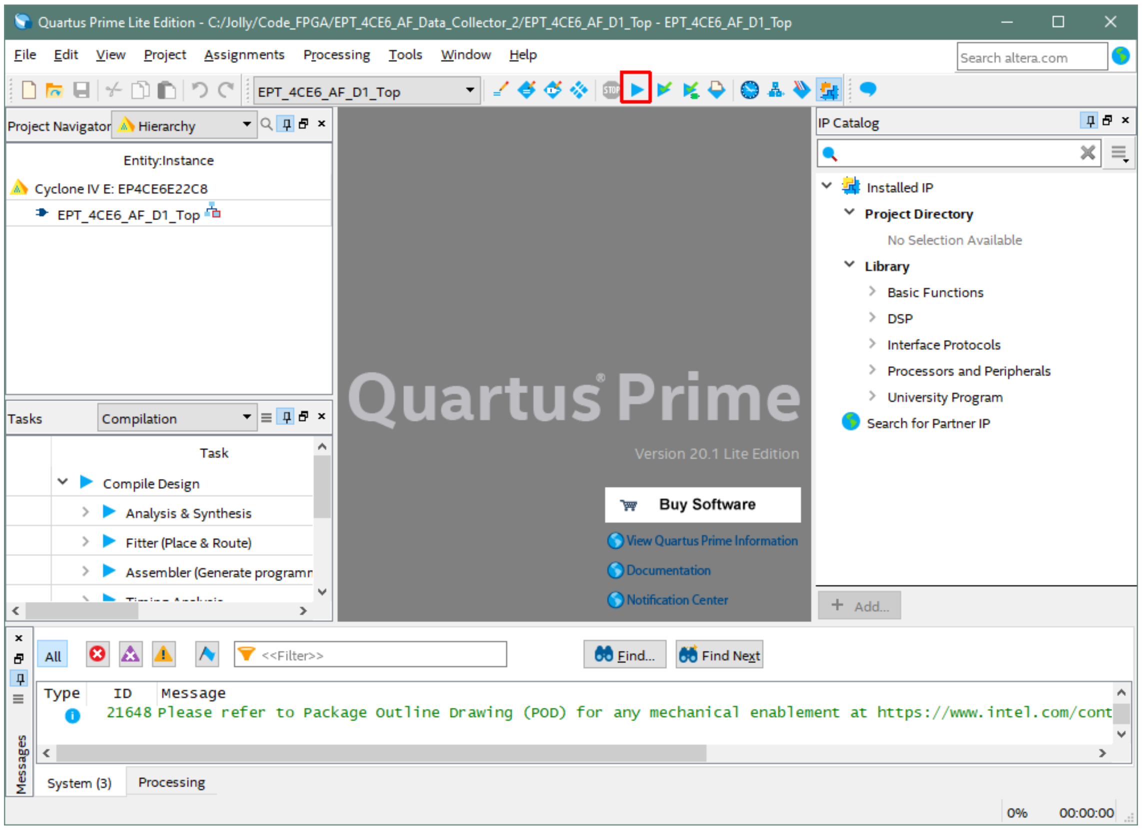
C:\Users\nelso\Documents\EPT_4CE6_AF_Platform_Demo \ EPT_4CE6_AF_D1_Top

directory. Paste the file.


C:\Users\nelso\Documents\EPT_4CE6_AF_Platform_Demo\EPT_4CE6_AF_D1_Top
File Home Share View
This PC > Windows (C:) > Users > nelso > Documents > EPT_4CE6_AF_Platform_Demo > EPT_4CE6_AF_D1_Top

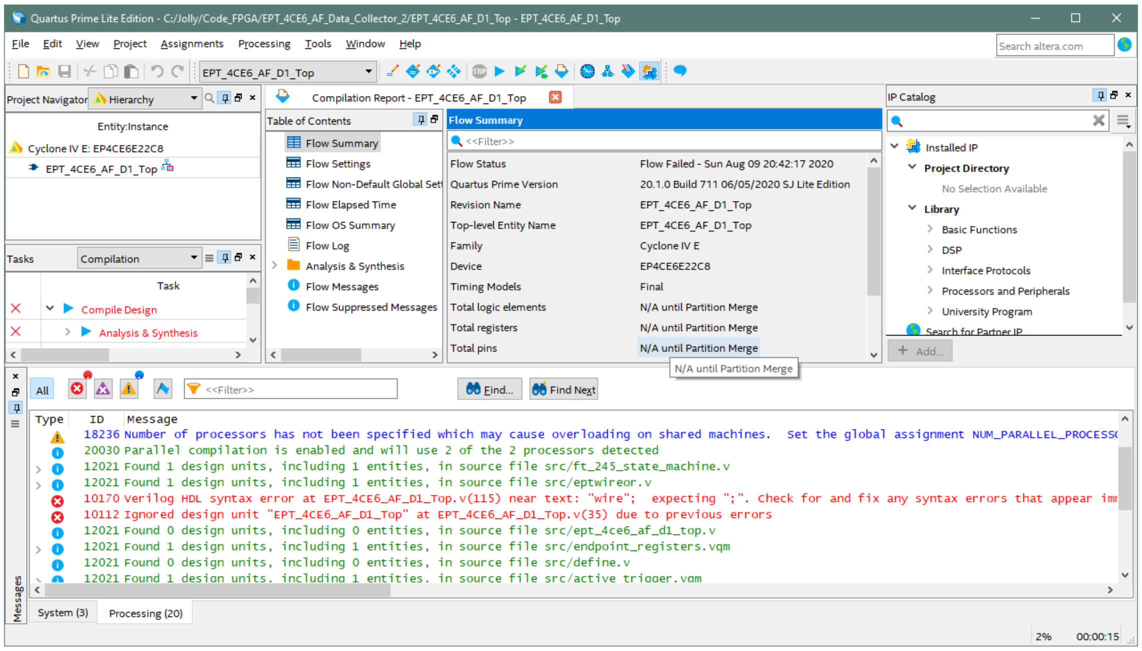
| Name | Date modified | Type |
|---|---|---|
| db | 5/25/2024 9:26 PM | File folder |
| incremental_db | 5/25/2024 9:20 PM | File folder |
| output_files | 5/25/2024 9:26 PM | File folder |
| EPT_4CE6_AF_D1_Top.qpf | 5/25/2024 9:19 PM | QPF File |
| EPT_4CE6_AF_D1_Top.qsf | 5/25/2024 9:25 PM | QSF File |
| EPT_4CE6_AF_D1_Top.sdc | 3/16/2021 9:48 PM | SDC File |

Select the Start Compilation button.


Quartus Prime Lite Edition - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top
File Edit View Project Assignments Processing Tools Window Help
EPT_4CE6_AF_D1_Top
Project Navigator Hierarchy
Cyclone IV E: EP4CE6E22C8
EPT_4CE6_AF_D1_Top
Tasks Compilation
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programm
Quartus Prime
Version 20.1 Lite Edition
Buy Software
IP Catalog
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Processors and Peripherals
University Program
Search for Partner IP

```
Type  ID    Message
      21648 Please refer to Package Outline Drawing (POD) for any mechanical enablement at https://www.intel.com/cont
```

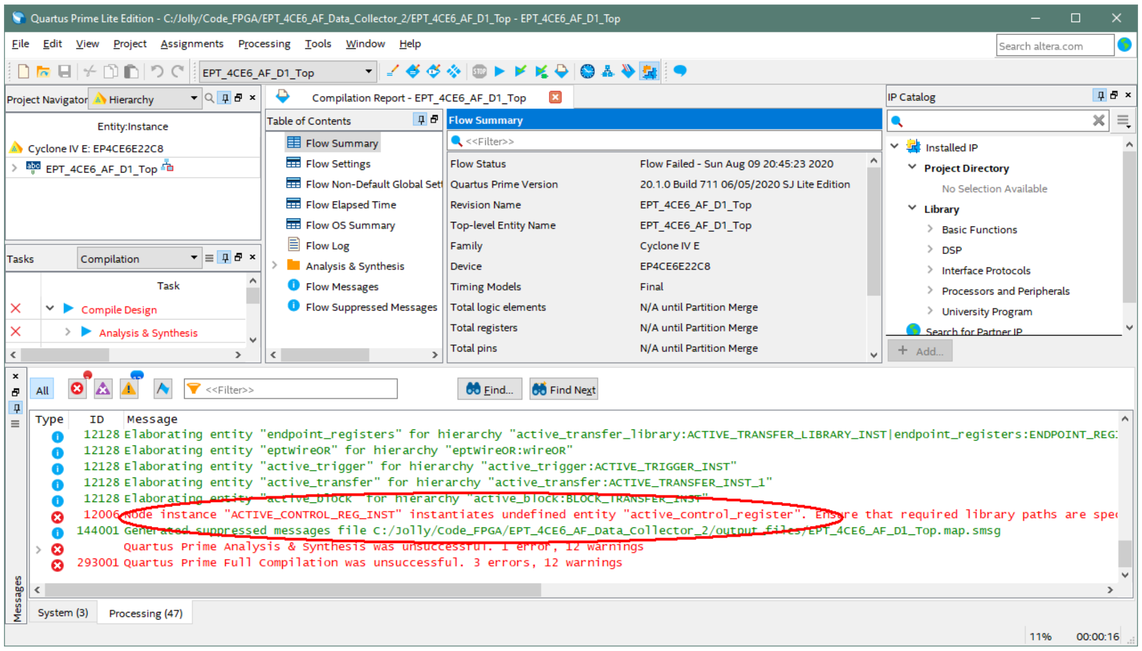
If you forget to include a file or some other error you should expect to see a screen similar to this:



| | |
|---|---|
| Flow Status | Flow Failed - Sun Aug 09 20:42:17 2020 |
| Quartus Prime Version | 20.1.0 Build 711 06/05/2020 SJ Lite Edition |
| Revision Name | EPT_4CE6_AF_D1_Top |
| Top-level Entity Name | EPT_4CE6_AF_D1_Top |
| Family | Cyclone IV E |
| Device | EP4CE6E22C8 |
| Timing Models | Final |
| Total logic elements | N/A until Partition Merge |
| Total registers | N/A until Partition Merge |
| Total pins | N/A until Partition Merge |

```
Type  ID    Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESS
      20030 Parallel compilation is enabled and will use 2 of the 2 processors detected
      12021 Found 1 design units, including 1 entities, in source file src/ft_245_state_machine.v
      12021 Found 1 design units, including 1 entities, in source file src/eptwireor.v
      10170 Verilog HDL syntax error at EPT_4CE6_AF_D1_Top.v(115) near text: "wire";  expecting ";". Check for and fix any syntax errors that appear im
      10112 Ignored design unit "EPT_4CE6_AF_D1_Top" at EPT_4CE6_AF_D1_Top.v(35) due to previous errors
      12021 Found 0 design units, including 0 entities, in source file src/ept_4ce6_af_d1_top.v
      12021 Found 1 design units, including 1 entities, in source file src/endpoint_registers.vqm
      12021 Found 0 design units, including 0 entities, in source file src/define.v
      12021 Found 1 design units, including 1 entities, in source file src/active_trigger.vqm
```

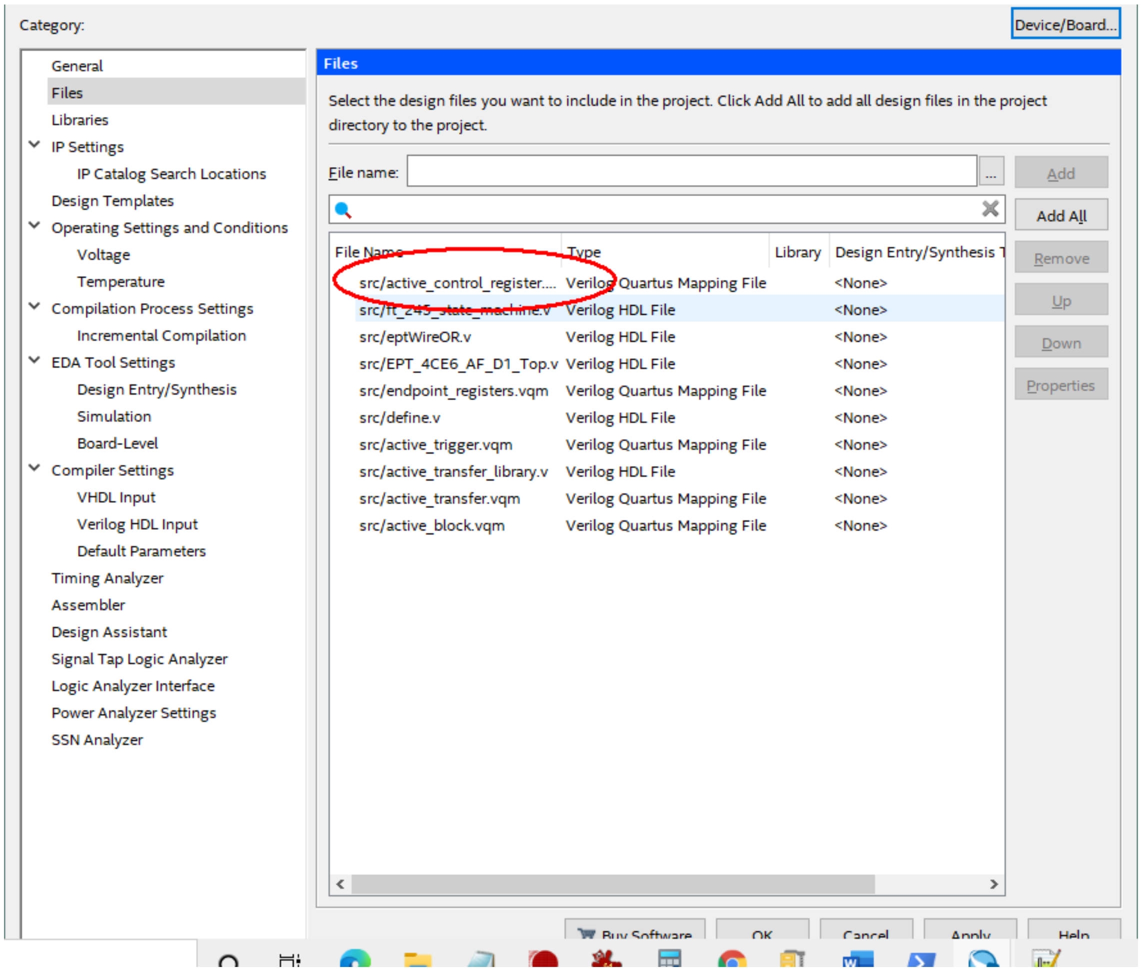
Click Ok, the select the “Error” tab to see the error.



| Flow Summary | |
|---|---|
| Flow Status | Flow Failed - Sun Aug 09 20:45:23 2020 |
| Quartus Prime Version | 20.1.0 Build 711 06/05/2020 SJ Lite Edition |
| Revision Name | EPT_4CE6_AF_D1_Top |
| Top-level Entity Name | EPT_4CE6_AF_D1_Top |
| Family | Cyclone IV E |
| Device | EP4CE6E22C8 |
| Timing Models | Final |
| Total logic elements | N/A until Partition Merge |
| Total registers | N/A until Partition Merge |
| Total pins | N/A until Partition Merge |

```
Type   ID    Message
      12128 Elaborating entity "endpoint_registers" for hierarchy "active_transfer_library:ACTIVE_TRANSFER_LIBRARY_INST|endpoint_registers:ENDPOINT_REG:
      12128 Elaborating entity "eptWireOR" for hierarchy "eptWireOR:wireOR"
      12128 Elaborating entity "active_trigger" for hierarchy "active_trigger:ACTIVE_TRIGGER_INST"
      12128 Elaborating entity "active_transfer" for hierarchy "active_transfer:ACTIVE_TRANSFER_INST_1"
      12128 Elaborating entity "active_block" for hierarchy "active_block:BLOCK_TRANSFER_INST"
      12006 Node instance "ACTIVE_CONTROL_REG_INST" instantiates undefined entity "active_control_register". Ensure that required library paths are spec
     144001 Generated suppressed messages file C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/output_files/EPT_4CE6_AF_D1_Top.map.smsg
            Quartus Prime Analysis & Synthesis was unsuccessful. 1 error, 12 warnings
     293001 Quartus Prime Full Compilation was unsuccessful. 3 errors, 12 warnings
```

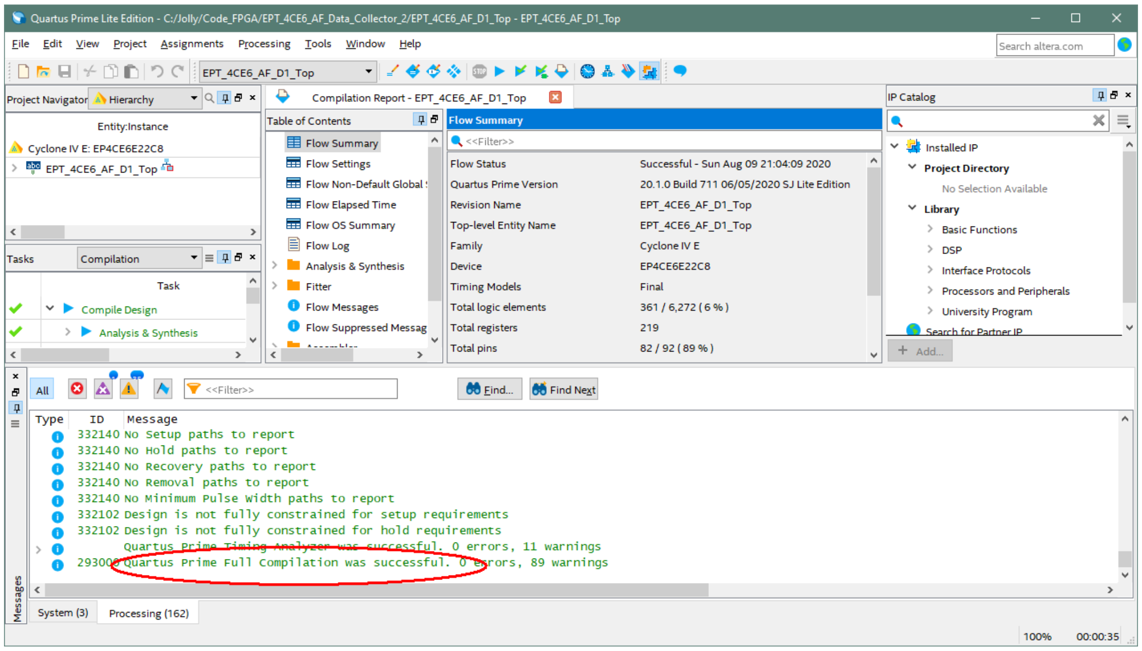
The error in this case is the missing file "active_control_register". Click on the Assignment menu, then select Settings, then select Files. Add the "active_control_register.v" file from the database.



Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

| File Name | Type | Library | Design Entry/Synthesis T |
|---|---|---|---|
| src/active_control_register.... | Verilog Quartus Mapping File | | <None> |
| src/ft_245_state_machine.v | Verilog HDL File | | <None> |
| src/eptWireOR.v | Verilog HDL File | | <None> |
| src/EPT_4CE6_AF_D1_Top.v | Verilog HDL File | | <None> |
| src/endpoint_registers.vqm | Verilog Quartus Mapping File | | <None> |
| src/define.v | Verilog HDL File | | <None> |
| src/active_trigger.vqm | Verilog Quartus Mapping File | | <None> |
| src/active_transfer_library.v | Verilog HDL File | | <None> |
| src/active_transfer.vqm | Verilog Quartus Mapping File | | <None> |
| src/active_block.vqm | Verilog Quartus Mapping File | | <None> |

Click Ok then re-run the Compile process. After successful completion, the screen should look like the following:


Quartus Prime Lite Edition - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top

Compilation Report - EPT_4CE6_AF_D1_Top

Flow Summary

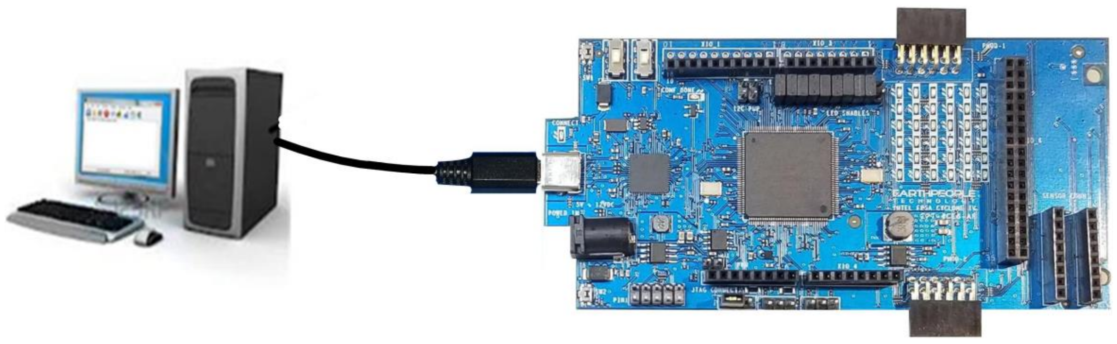
| | |
|---|---|
| Flow Status | Successful - Sun Aug 09 21:04:09 2020 |
| Quartus Prime Version | 20.1.0 Build 711 06/05/2020 SJ Lite Edition |
| Revision Name | EPT_4CE6_AF_D1_Top |
| Top-level Entity Name | EPT_4CE6_AF_D1_Top |
| Family | Cyclone IV E |
| Device | EP4CE6E22C8 |
| Timing Models | Final |
| Total logic elements | 361 / 6,272 ( 6 % ) |
| Total registers | 219 |
| Total pins | 82 / 92 ( 89 % ) |

```
Type   ID     Message
       332140 No Setup paths to report
       332140 No Hold paths to report
       332140 No Recovery paths to report
       332140 No Removal paths to report
       332140 No Minimum Pulse Width paths to report
       332102 Design is not fully constrained for setup requirements
       332102 Design is not fully constrained for hold requirements
              Quartus Prime Timing Analyzer was successful. 0 errors, 11 warnings
       293000 Quartus Prime Full Compilation was successful. 0 errors, 89 warnings
```

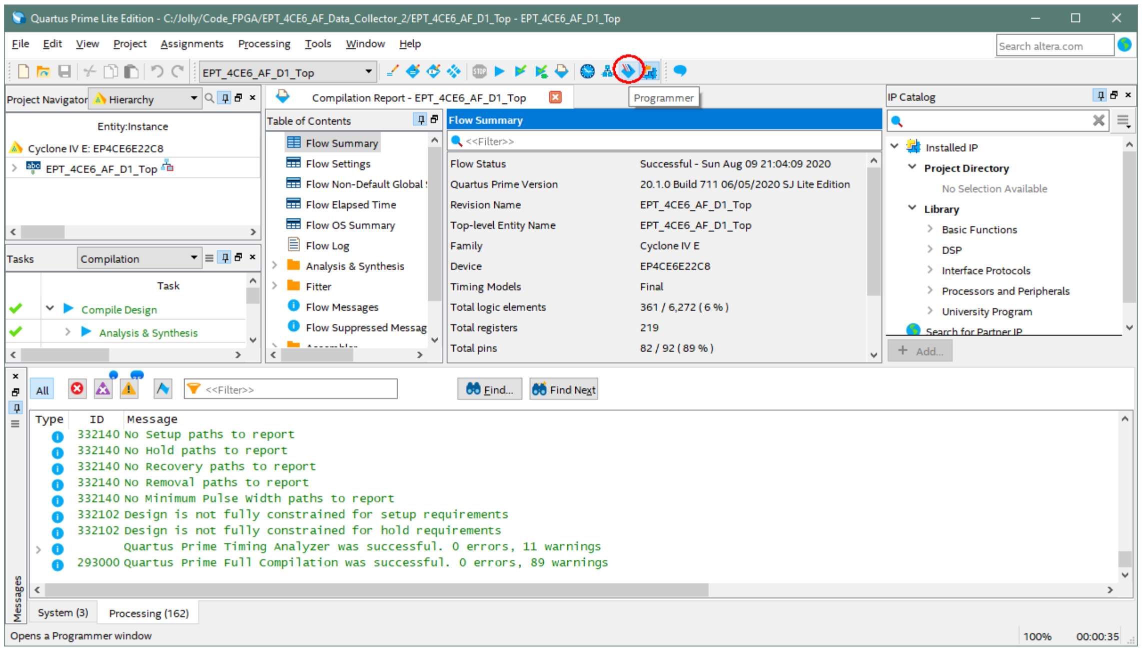
At this point the project has been successfully compiled, synthesized and a programming file has been produce. See the next section on how to program the FPGA.

### 4.1.2 Configuring the FPGA

Configuring the FPGA is quick and easy. All that is required is a standard USB-C cable and the EPT_Blaster Driver DLL. Connect the DueProLogic to the PC, open up Quartus Prime, open the programmer tool, and click the Start button. To program the DPL Configuration Flash, follow the steps to install the USB Driver and the JTAG Driver Insert for Quartus Prime.



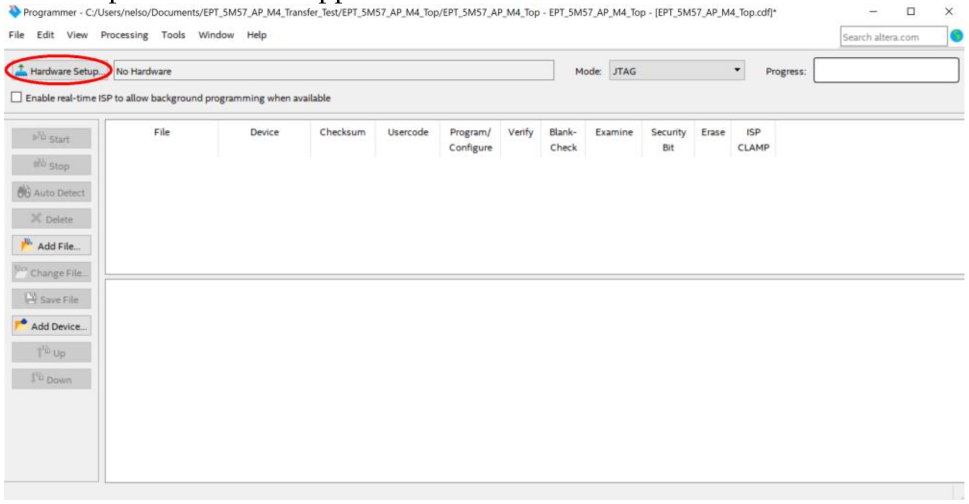
If the project created in the previous sections is not open, open it. Click on the Programmer button.



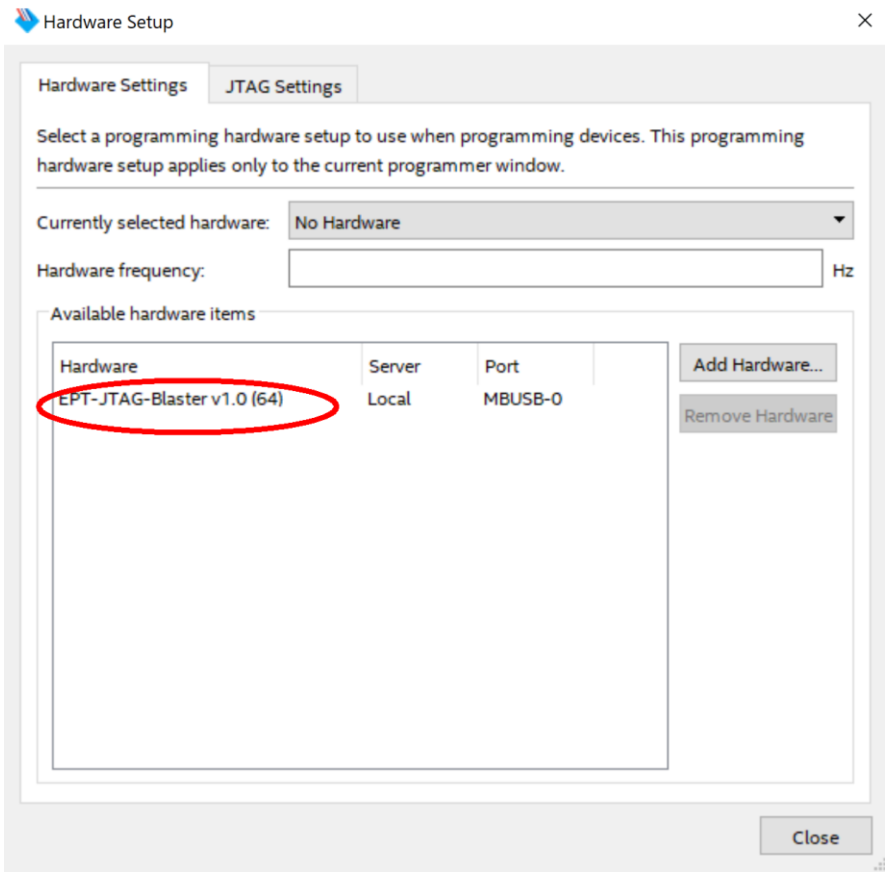
| Flow Summary | |
|---|---|
| Flow Status | Successful - Sun Aug 09 21:04:09 2020 |
| Quartus Prime Version | 20.1.0 Build 711 06/05/2020 SJ Lite Edition |
| Revision Name | EPT_4CE6_AF_D1_Top |
| Top-level Entity Name | EPT_4CE6_AF_D1_Top |
| Family | Cyclone IV E |
| Device | EP4CE6E22C8 |
| Timing Models | Final |
| Total logic elements | 361 / 6,272 ( 6 % ) |
| Total registers | 219 |
| Total pins | 82 / 92 ( 89 % ) |

```
Type  ID     Message
      332140 No Setup paths to report
      332140 No Hold paths to report
      332140 No Recovery paths to report
      332140 No Removal paths to report
      332140 No Minimum Pulse Width paths to report
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
             Quartus Prime Timing Analyzer was successful. 0 errors, 11 warnings
      293000 Quartus Prime Full Compilation was successful. 0 errors, 89 warnings
```

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


Programmer - C:/Users/nelso/Documents/EPT_5M57_AP_M4_Transfer_Test/EPT_5M57_AP_M4_Top/EPT_5M57_AP_M4_Top - EPT_5M57_AP_M4_Top - [EPT_5M57_AP_M4_Top.cdf]*
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
No Hardware
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

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



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

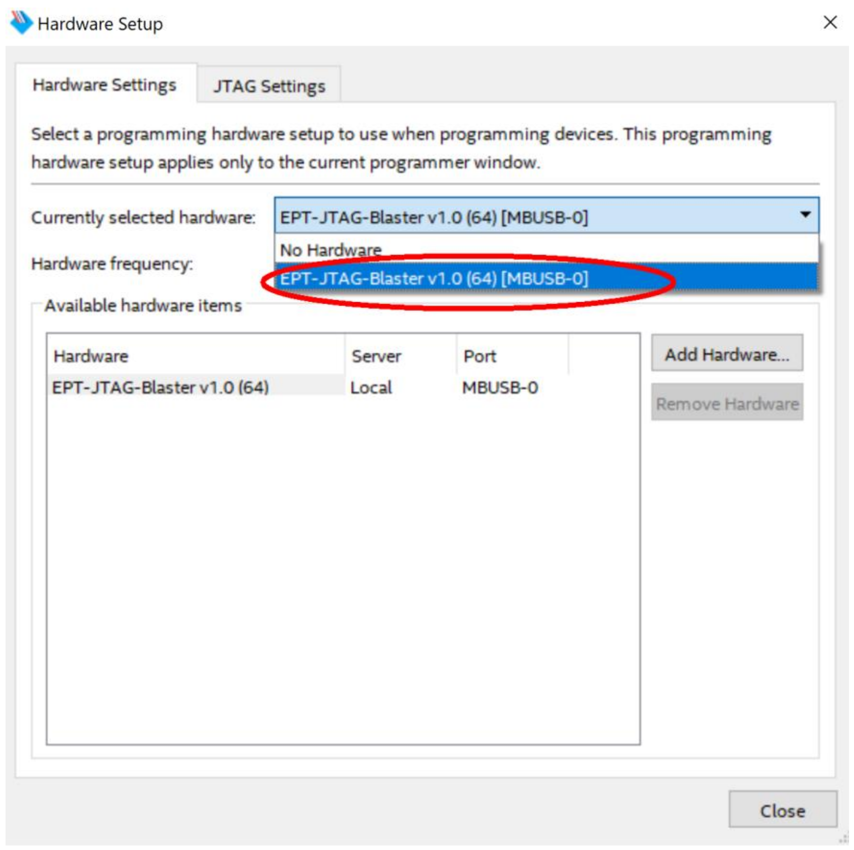
Currently selected hardware: No Hardware

Hardware frequency: Hz

Available hardware items

| Hardware | Server | Port |
|---|---|---|
| EPT-JTAG-Blaster v1.0 (64) | Local | MBUSB-0 |

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



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]

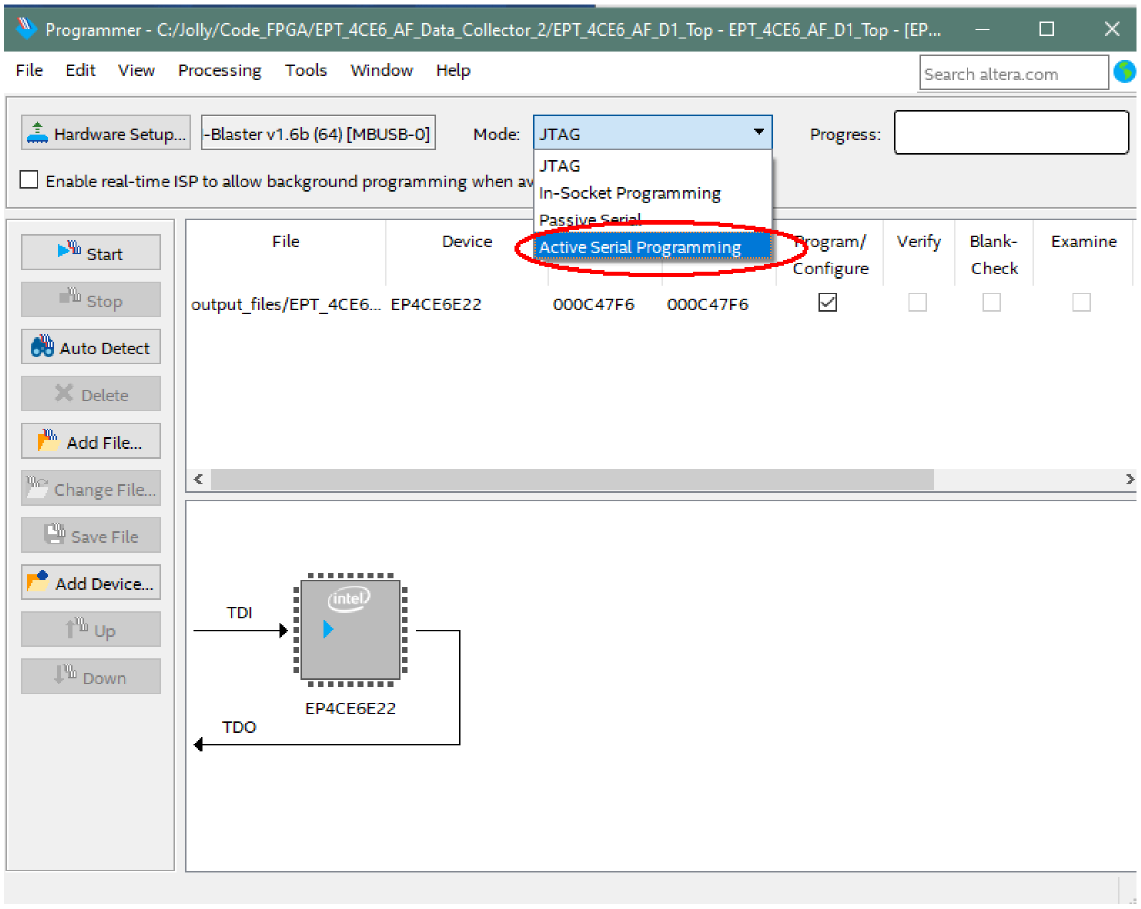
No Hardware
EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]

Hardware frequency:

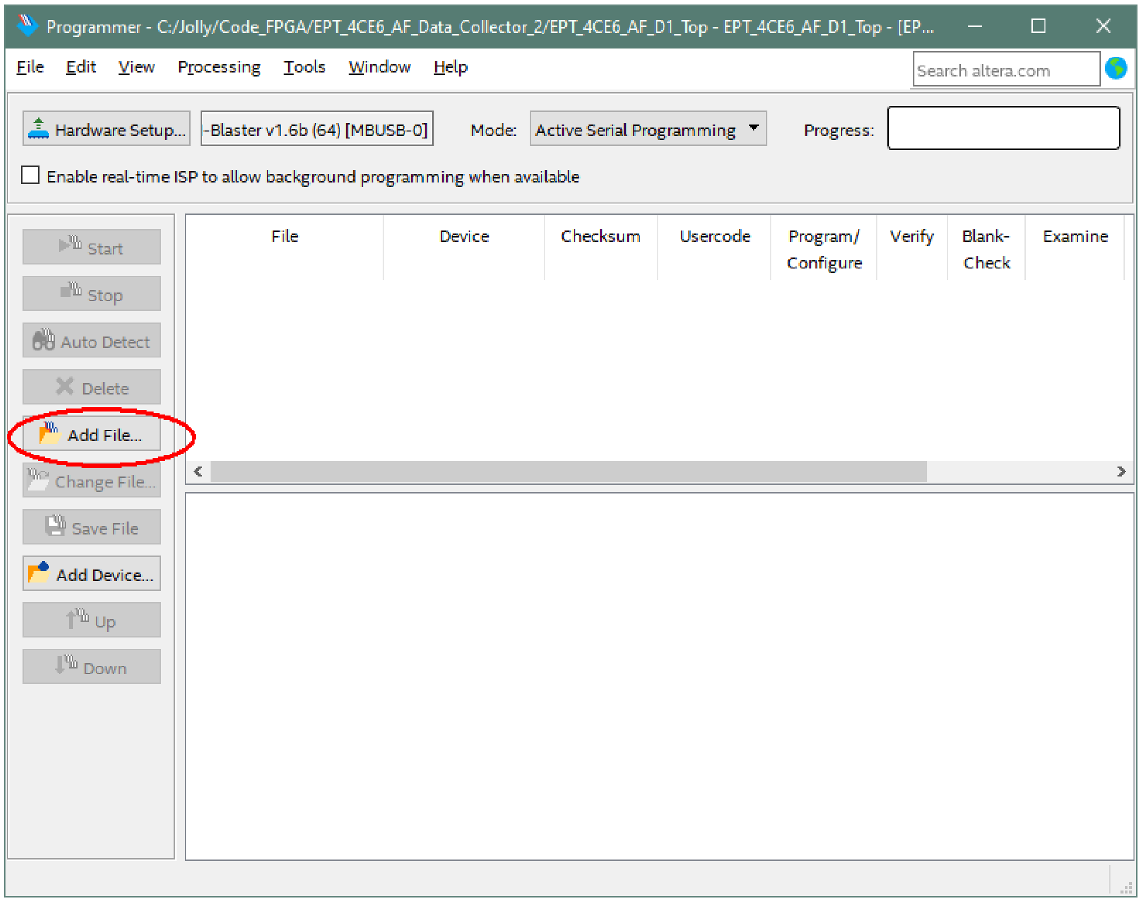
Available hardware items

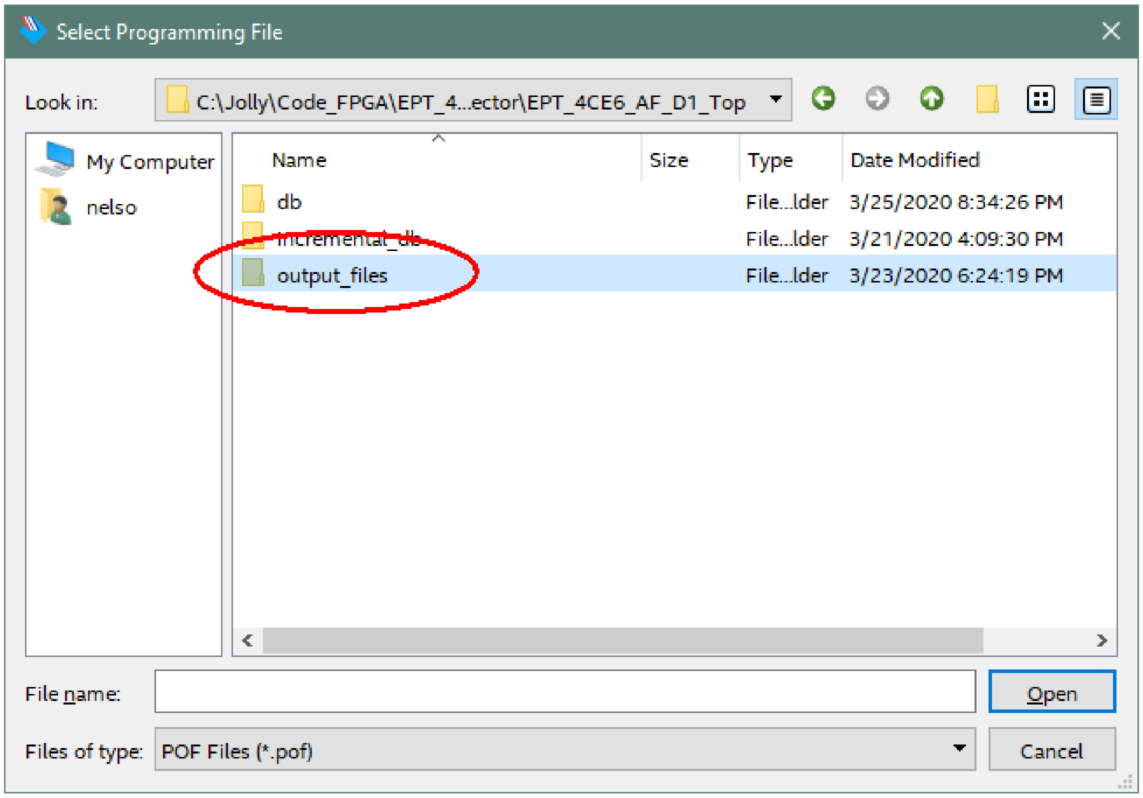
| Hardware | Server | Port |
|---|---|---|
| EPT-JTAG-Blaster v1.0 (64) | Local | MBUSB-0 |

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


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: JTAG
Progress:
JTAG
In-Socket Programming
Passive Serial
Active Serial Programming
Enable real-time ISP to allow background programming when av
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File
Device
Program/ Configure
Verify
Blank-Check
Examine
output_files/EPT_4CE6... EP4CE6E22 000C47F6 000C47F6
TDI
TDO
EP4CE6E22

Click on the “Add File” button

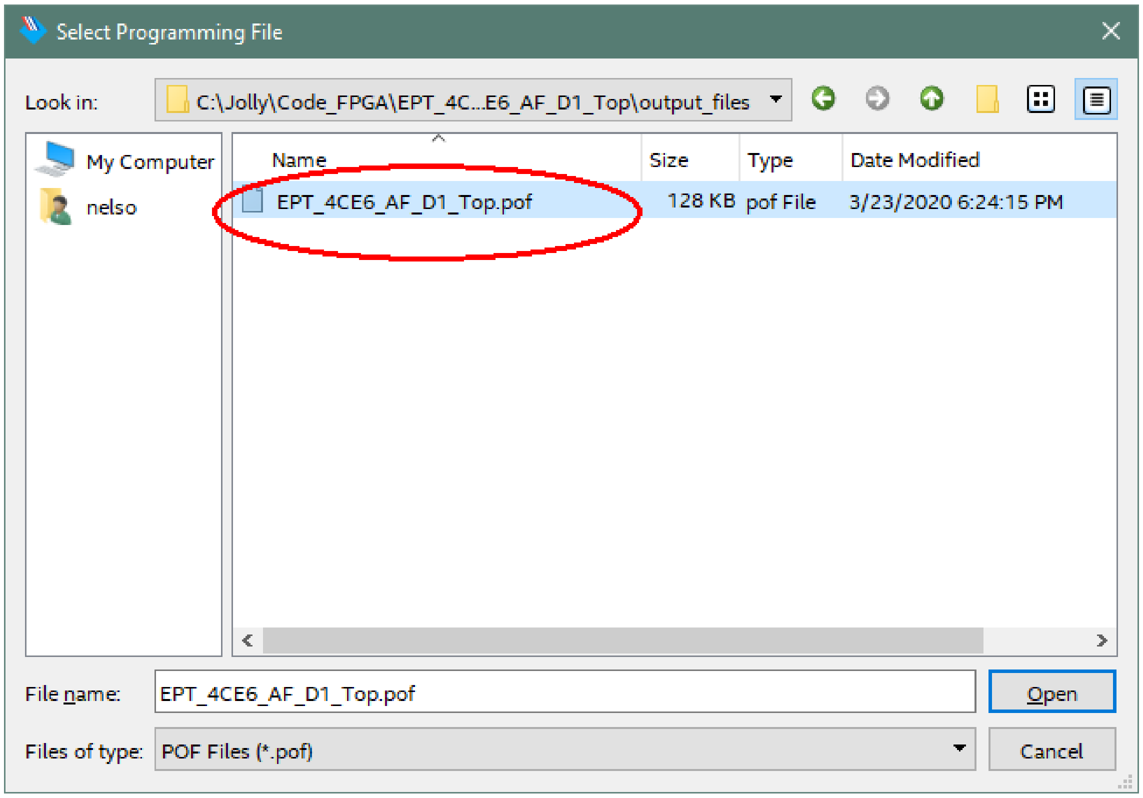

Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
-Blaster v1.6b (64) [MBUSB-0]
Mode: Active Serial Programming
Progress:
Enable real-time ISP to allow background 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



| Name | Size | Type | Date Modified |
|---|---|---|---|
| db | | File...lder | 3/25/2020 8:34:26 PM |
| incremental_db | | File...lder | 3/21/2020 4:09:30 PM |
| output_files | | File...lder | 3/23/2020 6:24:19 PM |

Files of type: POF Files (*.pof)

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


Select Programming File

Look in: C:\Jolly\Code_FPGA\EPT_4C...E6_AF_D1_Top\output_files

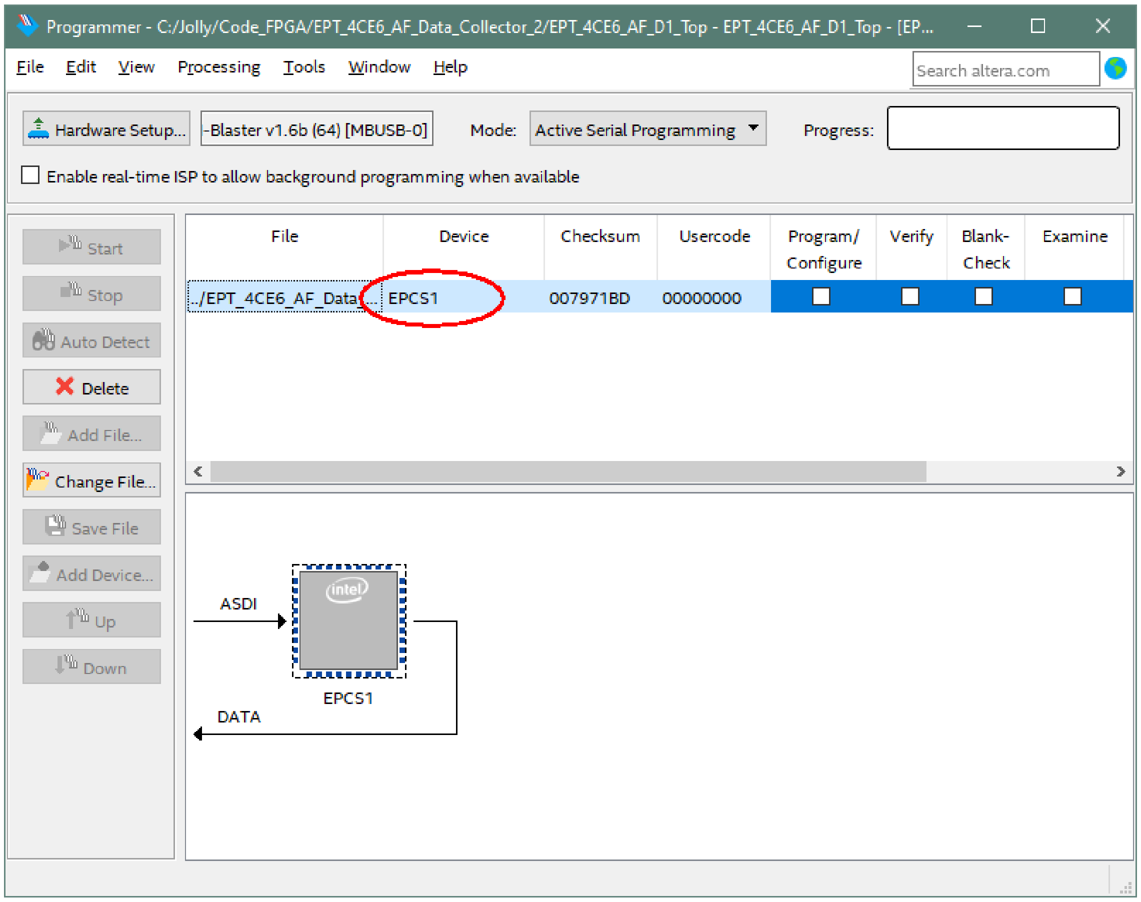
| Name | Size | Type | Date Modified |
|---|---|---|---|
| EPT_4CE6_AF_D1_Top.pof | 128 KB | pof File | 3/23/2020 6:24:15 PM |

File name: EPT_4CE6_AF_D1_Top.pof

Files of type: POF Files (*.pof)

Double click on the “EPT_4CE6_D1_Top.pof” file. Click the Open button in the lower right corner.

Select the EPCS1 under “Device”.



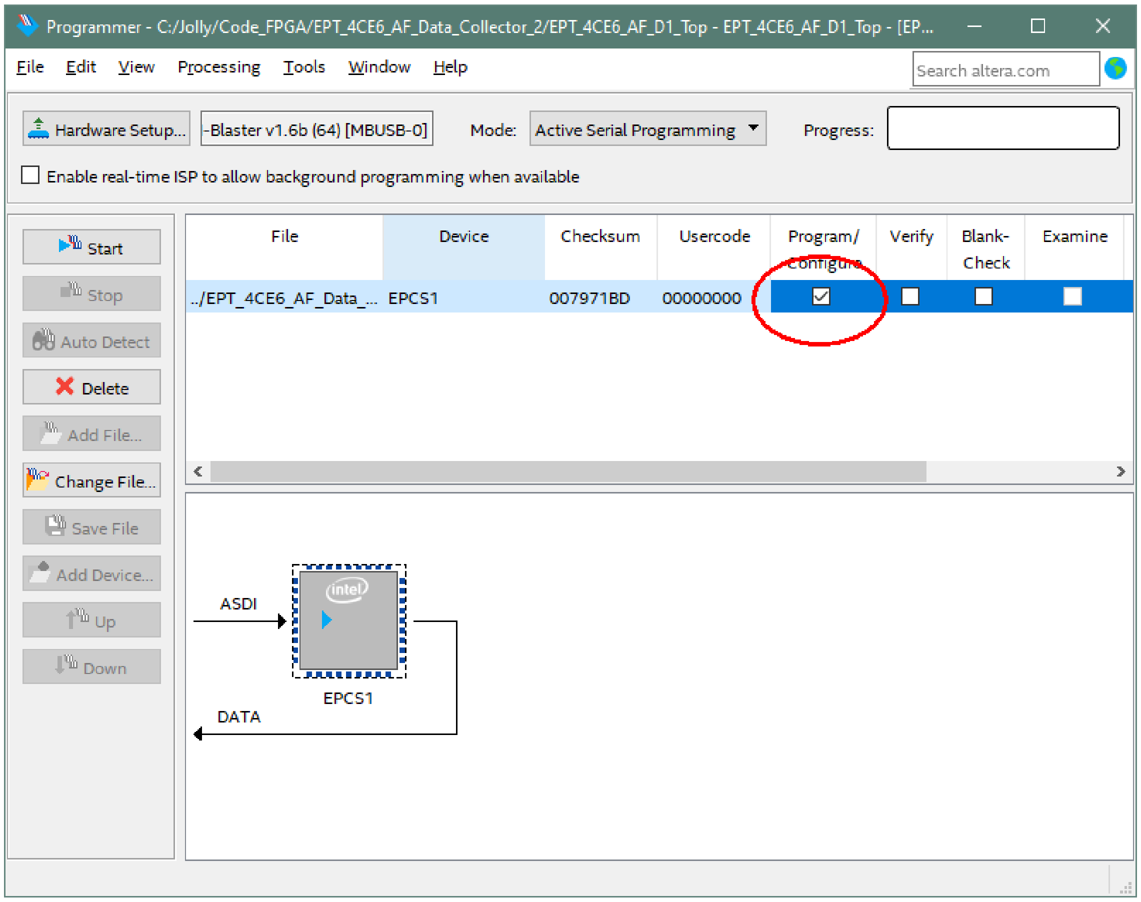
Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...

Hardware Setup... -Blaster v1.6b (64) [MBUSB-0]   Mode: Active Serial Programming   Progress:

☐ Enable real-time ISP to allow background programming when available

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| ../EPT_4CE6_AF_Data... | EPCS1 | 007971BD | 00000000 | ☐ | ☐ | ☐ | ☐ |

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

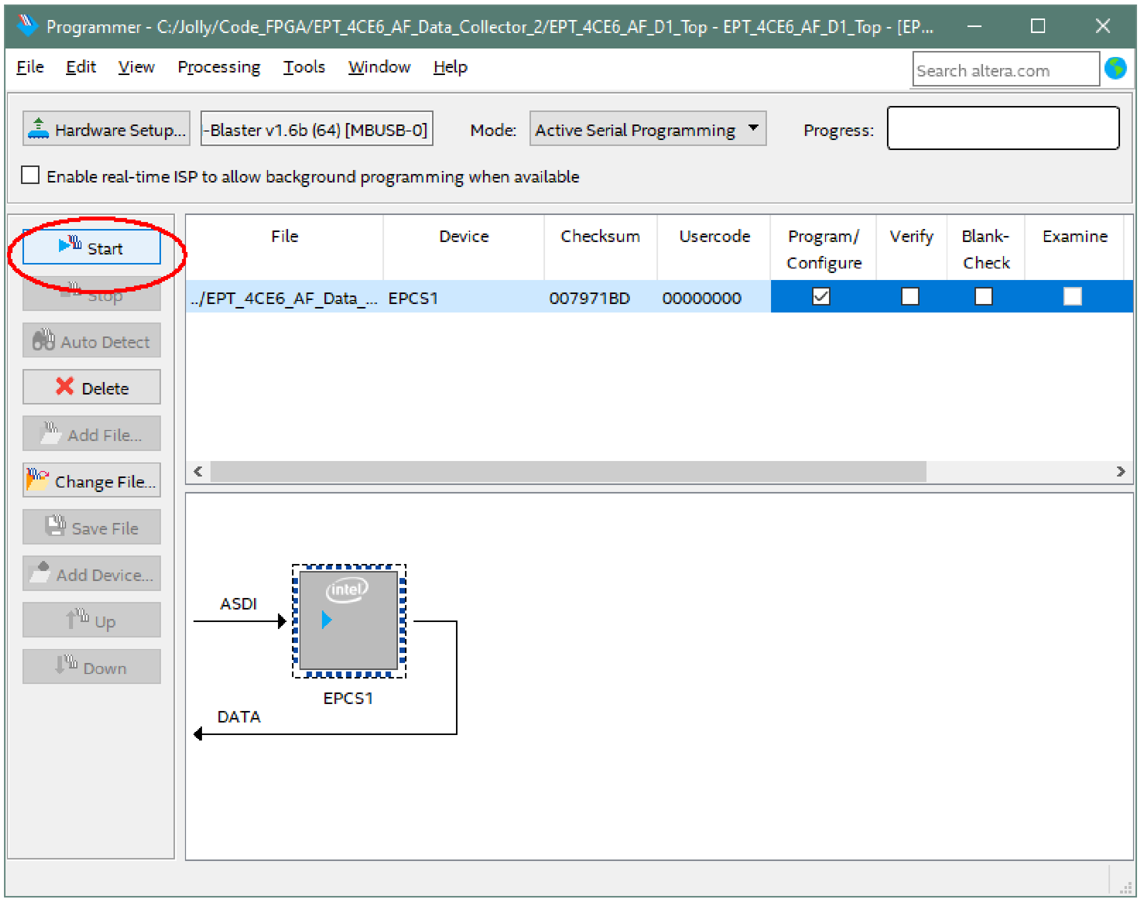

Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...

Hardware Setup... -Blaster v1.6b (64) [MBUSB-0] Mode: Active Serial Programming Progress:

☐ Enable real-time ISP to allow background programming when available

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| ../EPT_4CE6_AF_Data_... | EPCS1 | 007971BD | 00000000 | ☑ | ☐ | ☐ | ☐ |

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

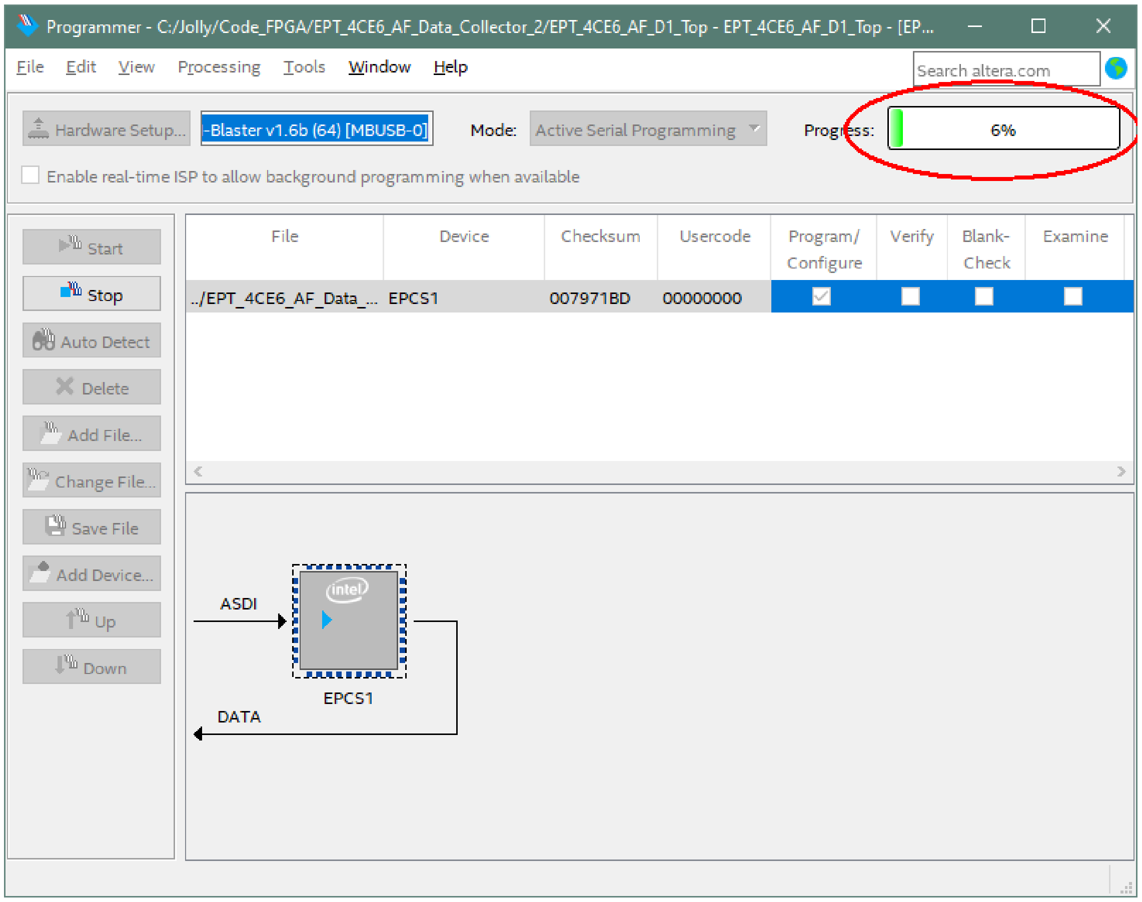

Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...

Hardware Setup... -Blaster v1.6b (64) [MBUSB-0] Mode: Active Serial Programming Progress:

☐ Enable real-time ISP to allow background programming when available

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| ../EPT_4CE6_AF_Data_... | EPCS1 | 007971BD | 00000000 | ✓ | ☐ | ☐ | ☐ |

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

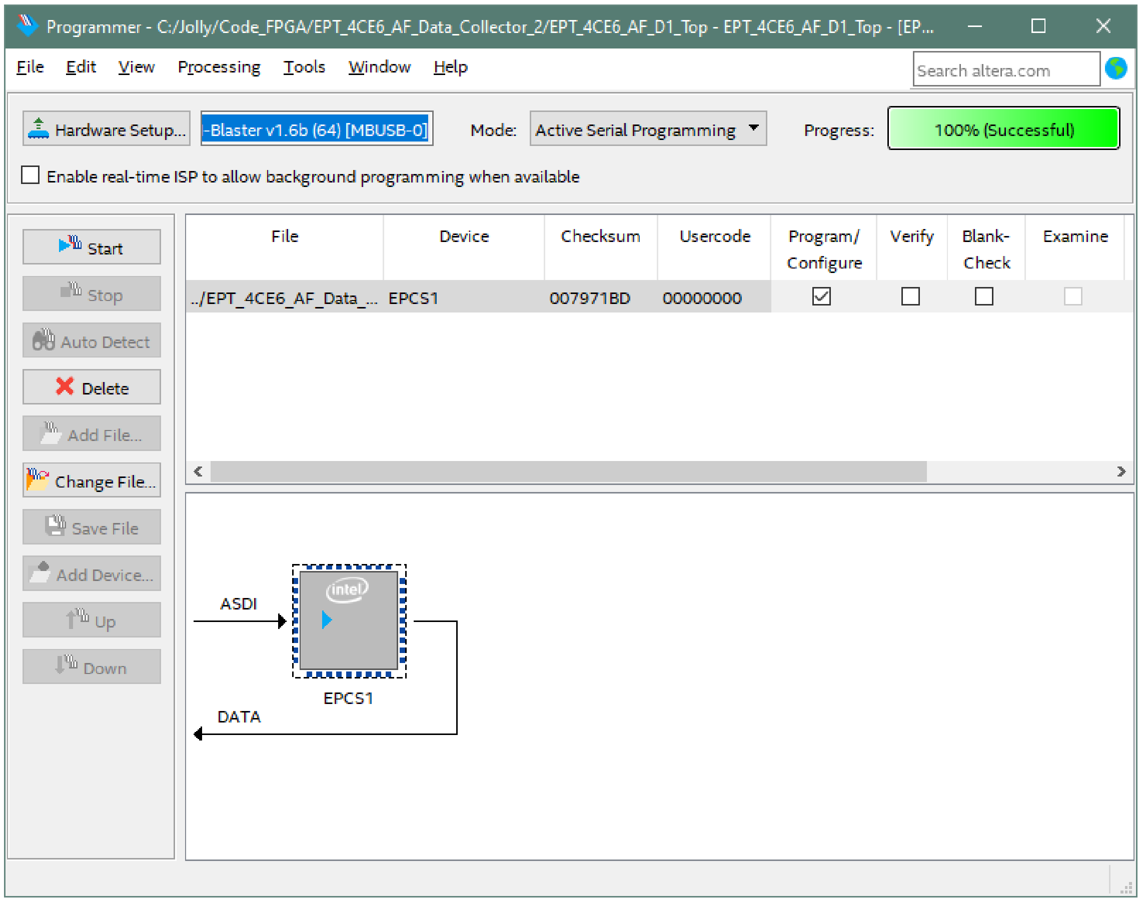

Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...

Hardware Setup... -Blaster v1.6b (64) [MBUSB-0]   Mode: Active Serial Programming   Progress: 6%

Enable real-time ISP to allow background programming when available

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| ../EPT_4CE6_AF_Data_... | EPCS1 | 007971BD | 00000000 | ✓ | ☐ | ☐ | ☐ |

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


Programmer - C:/Jolly/Code_FPGA/EPT_4CE6_AF_Data_Collector_2/EPT_4CE6_AF_D1_Top - EPT_4CE6_AF_D1_Top - [EP...

Hardware Setup... -Blaster v1.6b (64) [MBUSB-0] Mode: Active Serial Programming Progress: 100% (Successful)

☐ Enable real-time ISP to allow background programming when available

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| ../EPT_4CE6_AF_Data_... | EPCS1 | 007971BD | 00000000 | ☑ | ☐ | ☐ | ☐ |

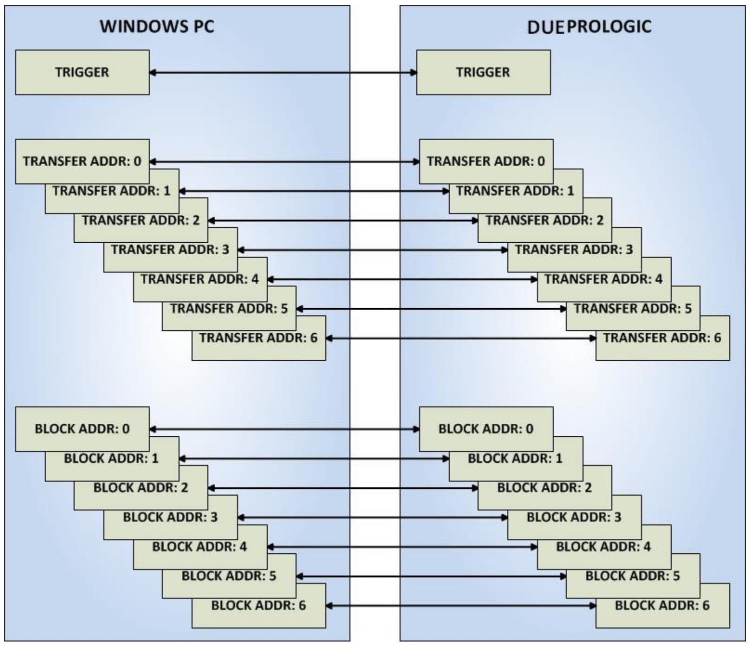
At this point, the DueProLogic is programmed and ready for use. To test that the FPGA is properly programmed, bring up the Platform Demo Tool. Click on one of the LED's and verify that the LED selected lights up. Press one of the switches on the board and ensure that the switch is captured on the Platform Demo Tool. Now you are ready to connect to the Arduino Due and write some code to transfer data between microcontroller and PC.

# 5 Active Host Application

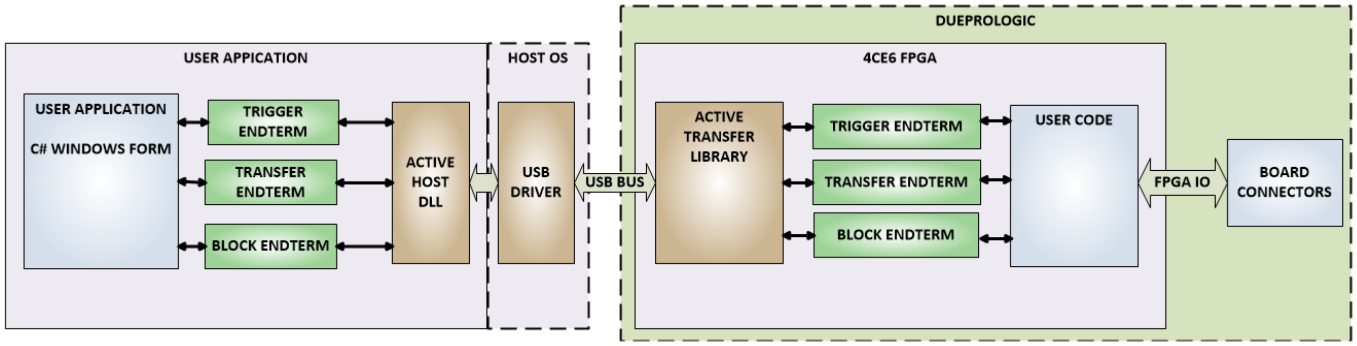
The Active Host SDK is provided which allows easy interface to application software written in C#. It runs on the PC and provides transparent connection from PC application code through the USB driver to the user FPGA code. The user code connects to "Endterms" in the Active Host SDK. These host "Endterms" have complementary HDL “Endterms” in the Active Transfer Library. Users have seamless bi-directional communications at their disposal in the form of:

- Trigger Endterm
- Transfer Endterm
- Block Endterm

User code writes to the Endterms as function calls. Just include the address of the individual module (there are eight individually addressable modules of each Endterm).



Immediately after writing to the selected Endterm, the value is received at the HDL Endterm in the FPGA.  The Trigger Endterms are used as “switches”. The user code can set a Trigger bit in the FPGA and cause an event to occur. The Transfer Endterm sends one byte to the FPGA. The Block Endterm sends a block of bytes. By using one of the Active Host Endterms, the user can create a dynamic, bi-directional, and configurable data transfer design.



### 5.1 Trigger EndTerm

The Trigger EndTerm is a software component that provides a direct path from the users application to the commensurate Trigger EndTerm in the FPGA. The Trigger has eight bits and is intended to be used to provide a switch at the opposite EndTerm. They are fast acting and are not stored or buffered by memory. When the user code sets a Trigger, it is immediately passed through to the opposite EndTerm via the USB driver. When receiving Trigger, the user application is required to respond to a callback from the Active Host SDK.

### 5.2 Transfer(Byte) EndTerm

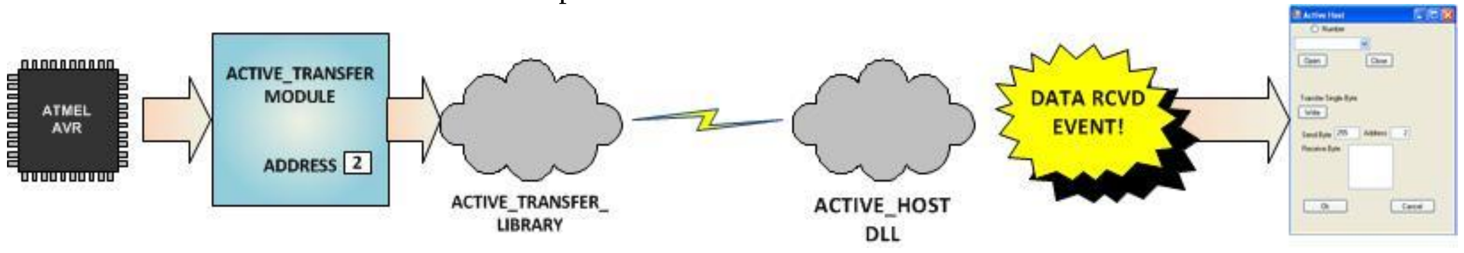
The Transfer EndTerm is a software component that provides a direct path from the users application to the commensurate Transfer EndTerm in the FPGA. It is used to transfer a byte to and from the FPGA. Eight separate Transfer EndTerm modules can be instantiated in the FPGA. Each module is addressed by the user application. Sending a byte is easy, just use the function call with the address and byte value. The byte is immediately sent to the corresponding EndTerm in the FPGA. Receiving a byte is just as easy, a callback function is registered at initialization. When the FPGA transmits a byte using its EndTerm, the callback function is called in the user application. The user code must store this byte in order to use it. The incoming Transfers are stored in a circular buffer in memory. This allows the user code to fetch the transfers with out losing bytes.

### 5.3 Block EndTerm

The Block EndTerm is a software component that provides a direct path from the users application to the commensurate Block EndTerm in the FPGA. The Block EndTerm is used to transfer a complete block to the FPGA. Block size is limited to 1 to 256 bytes. Eight separate Block EndTerm modules can be instantiated in the FPGA. Each module is addressed by the user application. Sending a block is easy, just use the function call with the address, block length, byte array. The block is buffered into a circular buffer in memory then transmitted via the USB bus to the Block EndTerm in the FPGA. Receiving a block is just as easy, a callback function is registered at initialization. When the FPGA transmits a block using its EndTerm, the callback function is called in the user application. The incoming Transfers are stored in a circular buffer in memory. This allows the user code to fetch the transfers with out losing bytes.

### *5.4 Active Host SDK*

The Active_Host SDK is designed to transfer data from the FPGA when it becomes available. The data will be stored into local memory of the PC, and an event will be triggered to inform the user code that data is available from the addressed module of the FPGA. This method of automatically moving data from the user code Endterm in the FPGA makes the data transfer transparent.



The data seamlessly appears in Host PC memory from the Arduino. The user code will direct the data to a control such as a textbox on a Windows Form. The transparent receive transfer path is made possible by a Callback mechanism in the Active Host SDK. The SDK calls a registered callback function in the user code. The user code callback can be designed to generate any number of events to handle the received data.

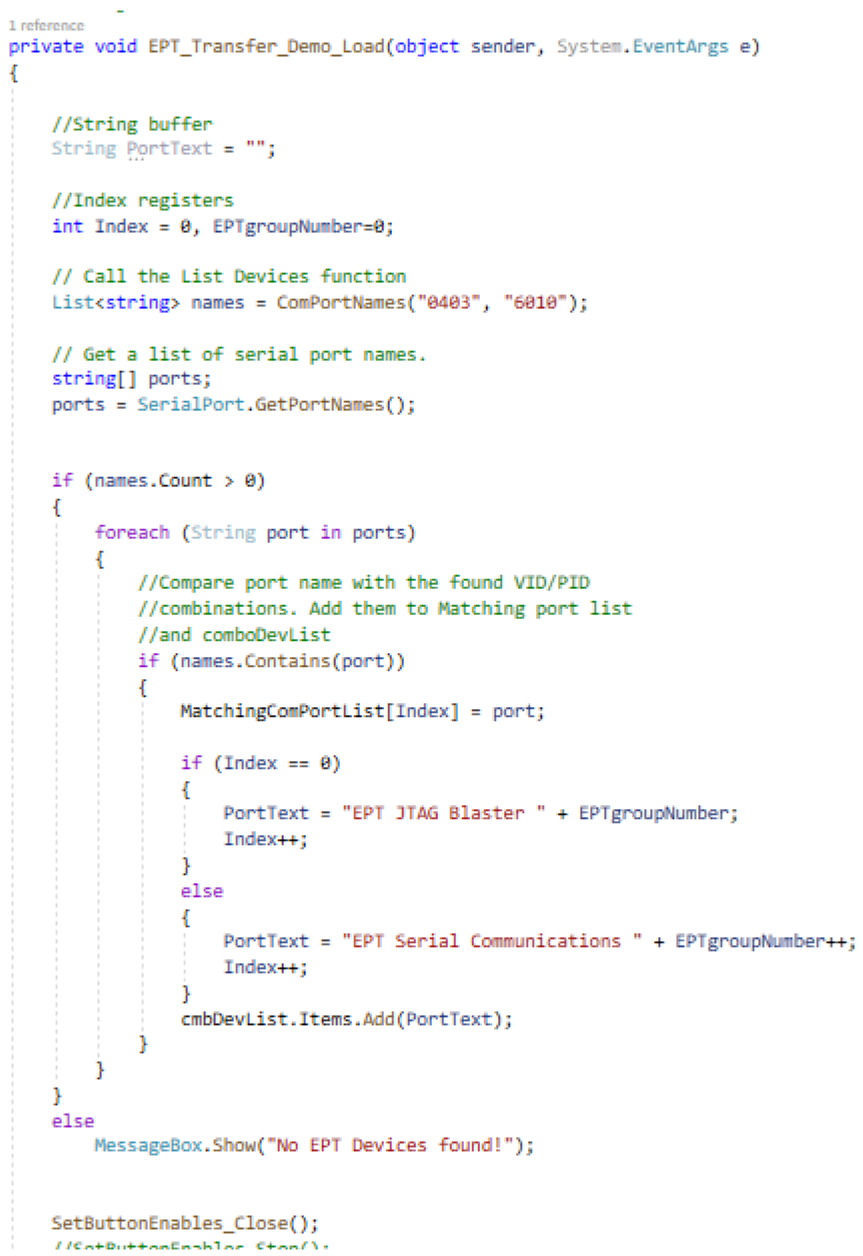
The user application will access the FPGA by use of functions contained in the Active Host SDK. The functions to access the FPGA are:

- EPT_AH_SendTrigger ()
- EPT_AH_SendByte ()
- EPT_AH_SendBlock ()
- EPT_AH_SendTransferControlByte()

#### 5.4.1 Active Host Open Device

To use the library functions for data transfer and triggering, an Earth People Technology device must be opened. EPT Devices use the COM Port library of the Visual Studio to provide communications. The first function called when the Windows

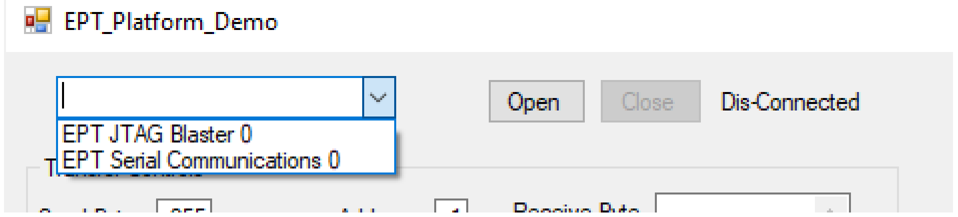
Form loads up is the <project_name>_Load(). This function is called automatically upon the completion of the Windows Form, so there is no need to do anything to call it. Once this function is called, it in turn calls the ComPortNames(). Use the function List Devices() to detect all EPT devices connected to the PC.



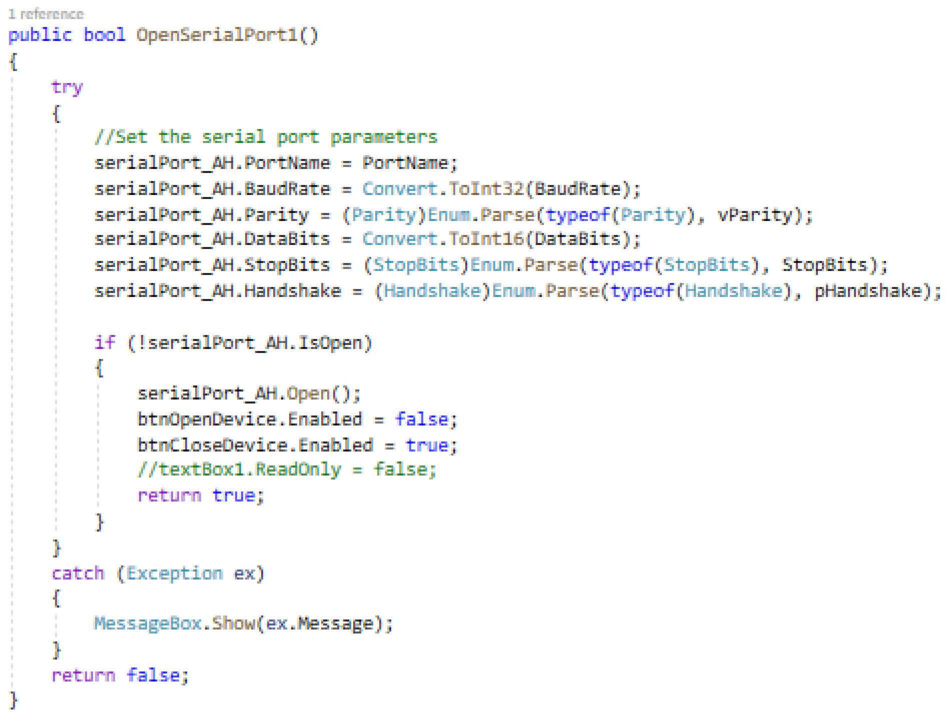
```
1 reference
private void EPT_Transfer_Demo_Load(object sender, System.EventArgs e)
{

    //String buffer
    String PortText = "";

    //Index registers
    int Index = 0, EPTgroupNumber=0;

    // Call the List Devices function
    List<string> names = ComPortNames("0403", "6010");

    // Get a list of serial port names.
    string[] ports;
    ports = SerialPort.GetPortNames();


    if (names.Count > 0)
    {
        foreach (String port in ports)
        {
            //Compare port name with the found VID/PID
            //combinations. Add them to Matching port list
            //and comboDevList
            if (names.Contains(port))
            {
                MatchingComPortList[Index] = port;

                if (Index == 0)
                {
                    PortText = "EPT JTAG Blaster " + EPTgroupNumber;
                    Index++;
                }
                else
                {
                    PortText = "EPT Serial Communications " + EPTgroupNumber++;
                    Index++;
                }
                cmbDevList.Items.Add(PortText);
            }
        }
    }
    else
        MessageBox.Show("No EPT Devices found!");


    SetButtonEnables_Close();
    //SetButtonEnables_Stop();
```

The ComPortNames() searches through all attached COM Ports and attempts to find the DueProLogic. Once it finds the DPL, a List is populated with each channel. The code in *_Load() function then adds the appropriate name to the COM Port channel and matches the COM Port number with the string in the list.

The user will select the device from the drop down combo box. This can be seen when the Windows Form is opened and the cmbDevList combo box is populated with all the devices. The selected device will be stored as an index number in the variable device_index.



In order to select the device, the user will click on the “Open” button which calls the OpenSerialPort1() function. The PortName is passed into the OpenSerialPort1 () function. If the function is successful, the device name is displayed in the label, labelDeviceCnt. Next, the device is made the active device the Open button is grayed out and the Close button is made active.

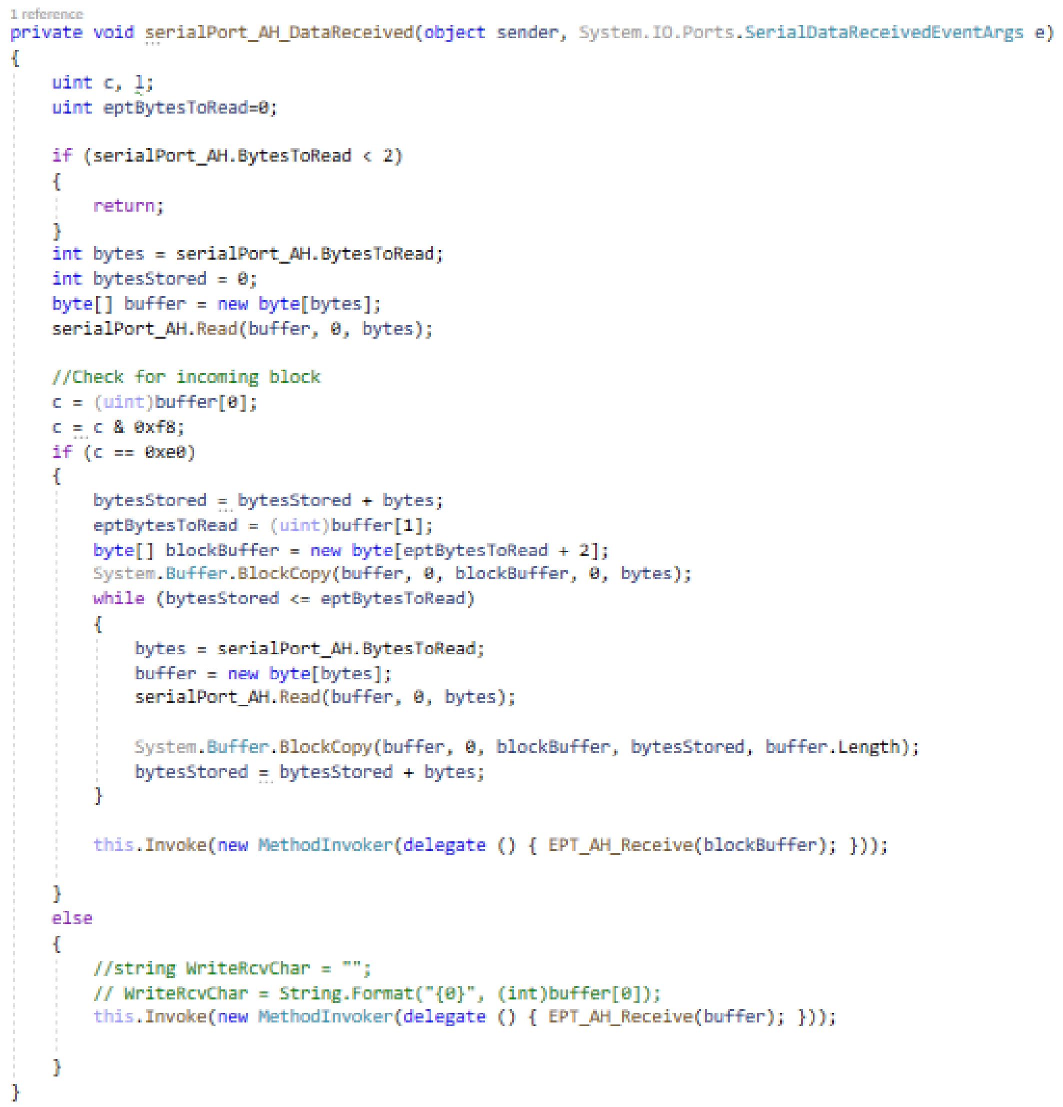


```
1 reference
public bool OpenSerialPort1()
{
    try
    {
        //Set the serial port parameters
        serialPort_AH.PortName = PortName;
        serialPort_AH.BaudRate = Convert.ToInt32(BaudRate);
        serialPort_AH.Parity = (Parity)Enum.Parse(typeof(Parity), vParity);
        serialPort_AH.DataBits = Convert.ToInt16(DataBits);
        serialPort_AH.StopBits = (StopBits)Enum.Parse(typeof(StopBits), StopBits);
        serialPort_AH.Handshake = (Handshake)Enum.Parse(typeof(Handshake), pHandshake);

        if (!serialPort_AH.IsOpen)
        {
            serialPort_AH.Open();
            btnOpenDevice.Enabled = false;
            btnCloseDevice.Enabled = true;
            //textBox1.ReadOnly = false;
            return true;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    return false;
}
```

### 5.4.2 Active Host Data Received Function

The serialPort_AH_DataReceived function is activated and assigned to the serial port received event. It resides in the active_transfer.cs file. This function will be called when the serial port receives bytes from the DueProLogic. The function must determine if the incoming transfer is:

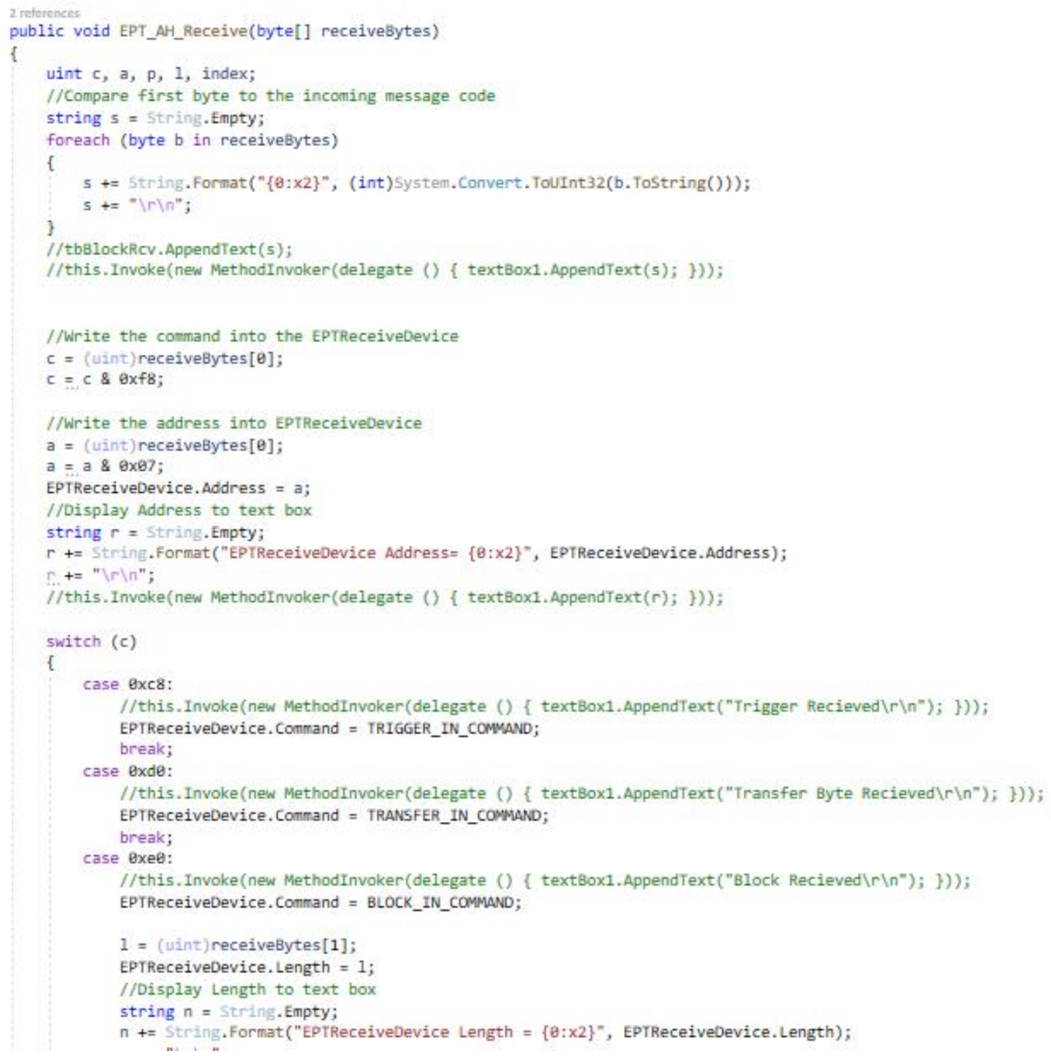
- Trigger Byte
- Transfer Byte
- Block Transfer



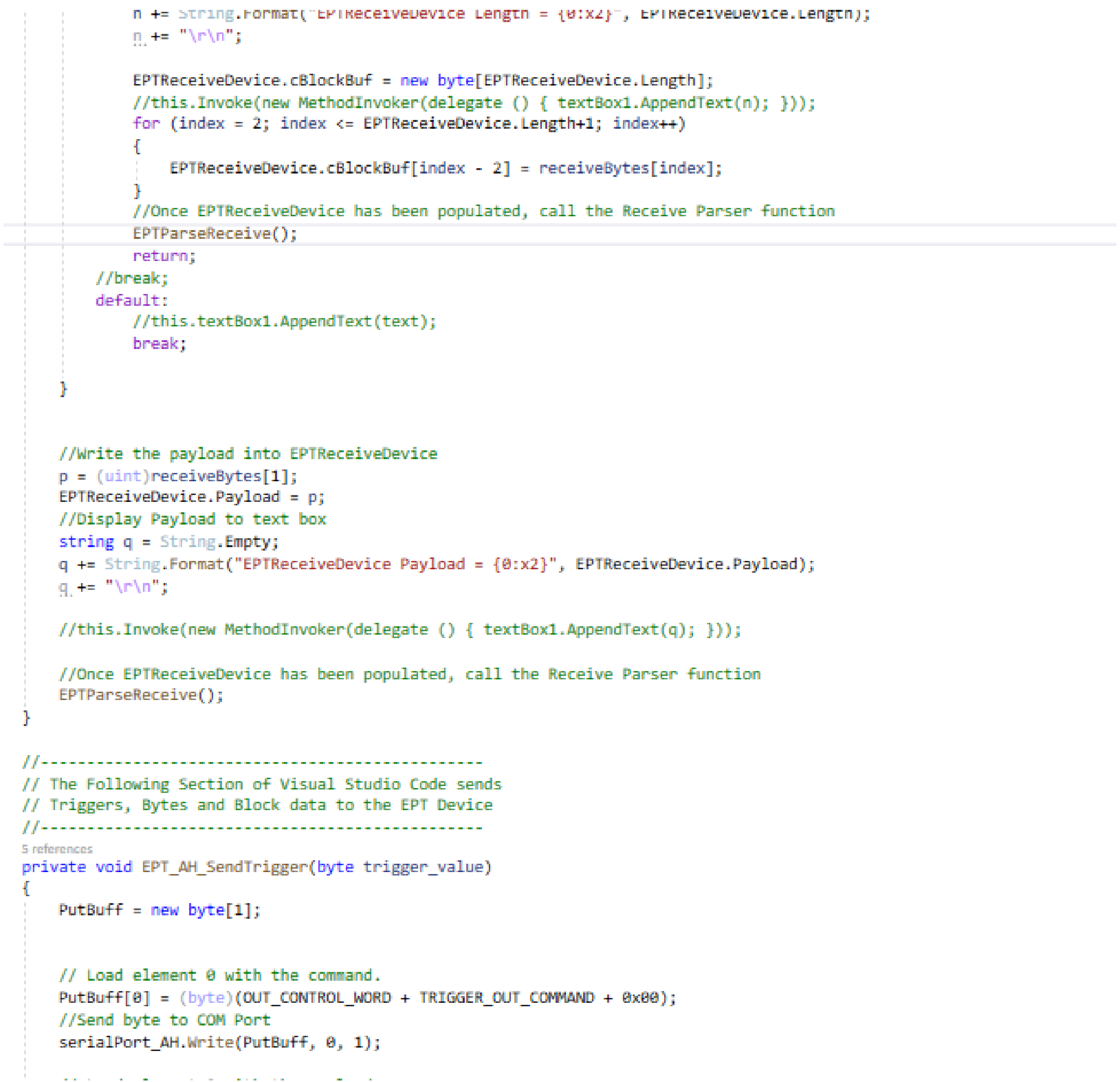
```
1 reference
private void serialPort_AH_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    uint c, l;
    uint eptBytesToRead=0;

    if (serialPort_AH.BytesToRead < 2)
    {
        return;
    }
    int bytes = serialPort_AH.BytesToRead;
    int bytesStored = 0;
    byte[] buffer = new byte[bytes];
    serialPort_AH.Read(buffer, 0, bytes);

    //Check for incoming block
    c = (uint)buffer[0];
    c = c & 0xf8;
    if (c == 0xe0)
    {
        bytesStored = bytesStored + bytes;
        eptBytesToRead = (uint)buffer[1];
        byte[] blockBuffer = new byte[eptBytesToRead + 2];
        System.Buffer.BlockCopy(buffer, 0, blockBuffer, 0, bytes);
        while (bytesStored <= eptBytesToRead)
        {
            bytes = serialPort_AH.BytesToRead;
            buffer = new byte[bytes];
            serialPort_AH.Read(buffer, 0, bytes);

            System.Buffer.BlockCopy(buffer, 0, blockBuffer, bytesStored, buffer.Length);
            bytesStored = bytesStored + bytes;
        }

        this.Invoke(new MethodInvoker(delegate () { EPT_AH_Receive(blockBuffer); }));

    }
    else
    {
        //string WriteRcvChar = "";
        // WriteRcvChar = String.Format("{0}", (int)buffer[0]);
        this.Invoke(new MethodInvoker(delegate () { EPT_AH_Receive(buffer); }));

    }
}
```

Once the serialPort_AH_DataReceived() has stored all incoming bytes due to the transfer, it calls the EPT_AH_Receive(). This function will process the incoming transfer and populate the class:

- EPTReceiveDevice



```
2 references
public void EPT_AH_Receive(byte[] receiveBytes)
{
    uint c, a, p, l, index;
    //Compare first byte to the incoming message code
    string s = String.Empty;
    foreach (byte b in receiveBytes)
    {
        s += String.Format("{0:x2}", (int)System.Convert.ToUInt32(b.ToString()));
        s += "\r\n";
    }
    //tbBlockRcv.AppendText(s);
    //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText(s); }));


    //Write the command into the EPTReceiveDevice
    c = (uint)receiveBytes[0];
    c = c & 0xf8;

    //Write the address into EPTReceiveDevice
    a = (uint)receiveBytes[0];
    a = a & 0x07;
    EPTReceiveDevice.Address = a;
    //Display Address to text box
    string r = String.Empty;
    r += String.Format("EPTReceiveDevice Address= {0:x2}", EPTReceiveDevice.Address);
    r += "\r\n";
    //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText(r); }));

    switch (c)
    {
        case 0xc8:
            //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText("Trigger Recieved\r\n"); }));
            EPTReceiveDevice.Command = TRIGGER_IN_COMMAND;
            break;
        case 0xd0:
            //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText("Transfer Byte Recieved\r\n"); }));
            EPTReceiveDevice.Command = TRANSFER_IN_COMMAND;
            break;
        case 0xe0:
            //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText("Block Recieved\r\n"); }));
            EPTReceiveDevice.Command = BLOCK_IN_COMMAND;

            l = (uint)receiveBytes[1];
            EPTReceiveDevice.Length = l;
            //Display Length to text box
            string n = String.Empty;
            n += String.Format("EPTReceiveDevice Length = {0:x2}", EPTReceiveDevice.Length);
```

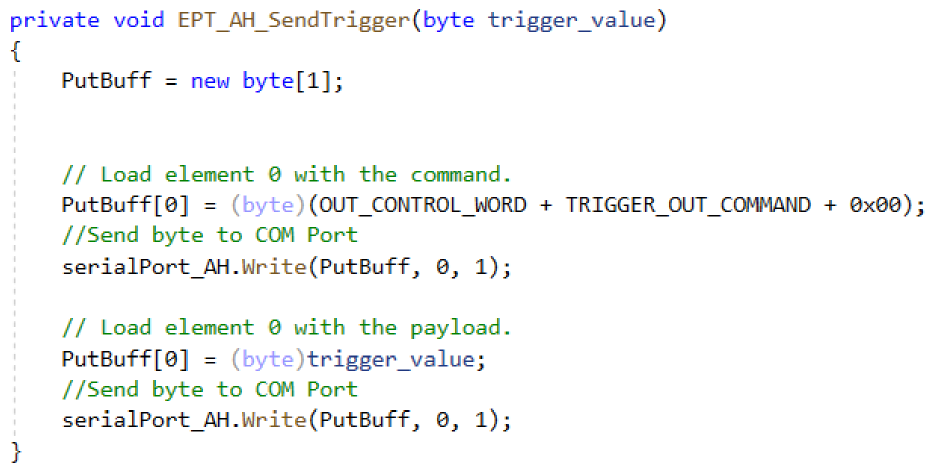


```
                n += String.Format("EPTReceiveDevice Length = {0:x2}", EPTReceiveDevice.Length);
                n += "\r\n";

                EPTReceiveDevice.cBlockBuf = new byte[EPTReceiveDevice.Length];
                //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText(n); }));
                for (index = 2; index <= EPTReceiveDevice.Length+1; index++)
                {
                    EPTReceiveDevice.cBlockBuf[index - 2] = receiveBytes[index];
                }
                //Once EPTReceiveDevice has been populated, call the Receive Parser function
                EPTParseReceive();
                return;
            //break;
            default:
                //this.textBox1.AppendText(text);
                break;

        }


        //Write the payload into EPTReceiveDevice
        p = (uint)receiveBytes[1];
        EPTReceiveDevice.Payload = p;
        //Display Payload to text box
        string q = String.Empty;
        q += String.Format("EPTReceiveDevice Payload = {0:x2}", EPTReceiveDevice.Payload);
        q += "\r\n";

        //this.Invoke(new MethodInvoker(delegate () { textBox1.AppendText(q); }));

        //Once EPTReceiveDevice has been populated, call the Receive Parser function
        EPTParseReceive();
    }

    //-----------------------------------------------
    // The Following Section of Visual Studio Code sends
    // Triggers, Bytes and Block data to the EPT Device
    //-----------------------------------------------
    5 references
    private void EPT_AH_SendTrigger(byte trigger_value)
    {
        PutBuff = new byte[1];


        // Load element 0 with the command.
        PutBuff[0] = (byte)(OUT_CONTROL_WORD + TRIGGER_OUT_COMMAND + 0x00);
        //Send byte to COM Port
        serialPort_AH.Write(PutBuff, 0, 1);
```

This object holds the command, address and payload for the transfer. Once the object is populated, the function calls the EPTParseReceive() function.

### 5.4.3 Active Host Triggers

The user application can send a trigger to the FPGA by using the EPT_AH_SendTrigger() function. First, open the EPT device to be used with openSerialPort1(). Call the function with the bit or bits to assert high on the trigger byte as the parameter. Then execute the function, the trigger bit or bits will momentarily assert high in the user code on the FPGA.



```
private void EPT_AH_SendTrigger(byte trigger_value)
{
    PutBuff = new byte[1];


    // Load element 0 with the command.
    PutBuff[0] = (byte)(OUT_CONTROL_WORD + TRIGGER_OUT_COMMAND + 0x00);
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);

    // Load element 0 with the payload.
    PutBuff[0] = (byte)trigger_value;
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);
}
```

To detect a trigger from the FPGA, the user application will set up functions in the EPTParseReceive() function.  A switch statement is used to decode which event should be called to handle the incoming received data.

- TRIGGER_IN
- TRANSFER_IN
- BLOCK_IN

```
private void EPTParseReceive()
{
    // Parse the command
    switch (EPTReceiveDevice.Command)
    {
        // This is a trigger command
        case TRIGGER_IN_COMMAND:
            TriggerOutReceive();
            break;
        // This is a single-byte transfer command
        case TRANSFER_IN_COMMAND:
            TransferOutReceive();
            break;
        // This is a data block transfer command
        case BLOCK_IN_COMMAND:
            BlockInReceive();
            break;
        default:
            break;
    }
}
```

The event handler function for the TRIGGER_IN's uses a switch statement to determine which trigger was asserted and what to do with it.

```
public void Receive_Trigger_In(object sender, EventArgs e)
{
    switch (ept_data.Payload)
    {
        case 0x01:
            lLableSwitch1.Text = "Switch 1\n Pressed";
            break;
        case 0x02:
            lLableSwitch2.Text = "Switch 2\n Pressed";
            break;
        case 0x04:
            lLableSwitch1.Text = "";
            lLableSwitch2.Text = "";
            break;
    }
}
```

### 5.4.4 Active Host Byte Transfers

The Active Host Byte Transfer EndTerm is designed to send/receive one byte to/from the EPT Device. To send a byte to the Device, the appropriate address must be selected for the Transfer module in the FPGA. Up to eight modules can be instantiated in the user code on the FPGA. Each module has its own address.

```
private void EPT_AH_SendByte(byte device_channel, byte data_byte)
{
    //byte tempVar = 0;
    PutBuff = new byte[1];

    // Load element 0 with the command.
    PutBuff[0] = (byte)(OUT_CONTROL_WORD + BYTE_TRANSFER_OUT_COMMAND + device_channel);
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);

    // Load element 0 with the payload.
    PutBuff[0] = (byte)data_byte;
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);
}
```

Use the function EPT_AH_SendByte() to send a byte the selected module. Then add the address of the transfer module as the first parameter of the EPT_AH_SendByte() function. Enter the byte to be transferred in the second parameter. Then execute the function, the byte will appear in the ports of the Active Transfer module in the user code on the FPGA.

To transfer data from the FPGA Device, a polling technique is used. This polling technique is because the Bulk Transfer USB is a Host initiated bus. The Device will not transfer any bytes until the Host commands it to. If the Device has data to send to the Host in an asynchronous manner (meaning the Host did not command the Device to send data), the Host must periodically check the Device for data in it's transmit FIFO. If data exists, the Host will command the Device to send it's data. The received data is then stored into local memory and register bits are set that will indicate data has been received from a particular address.

To receive a byte transfer from the Active host, user code must subscribe to the event created when the incoming byte transfer has arrived. A switch statement is used to decode which event should be called to handle the incoming received data. The event handler function will  check for any bytes read for that address.

```
private void EPTParseReceive()
{
    // Parse the command
    switch (EPTReceiveDevice.Command)
    {
        // This is a trigger command
        case TRIGGER_IN_COMMAND:
            TriggerOutReceive();
            break;
        // This is a single-byte transfer command
        case TRANSFER_IN_COMMAND:
            TransferOutReceive();
            break;
        // This is a data block transfer command
        case BLOCK_IN_COMMAND:
            BlockInReceive();
            break;
        default:
            break;
    }
}
```

The function EPTParseReceive() is called by the Read Callback function. The EPTParseReceive() function will examine the command of the incoming byte transfer and determine which receive function to call.

```
public void TransferOutReceive()
{
    string WriteRcvChar = "";
    WriteRcvChar = String.Format("{0}", (int)EPTReceiveData.Payload);
    tbDataBytes.AppendText(WriteRcvChar + ' ');
    tbAddress.Text = String.Format("{0:x2}", (uint)System.Convert.ToUInt32(EPTReceiveData.Address.ToString(
}
```

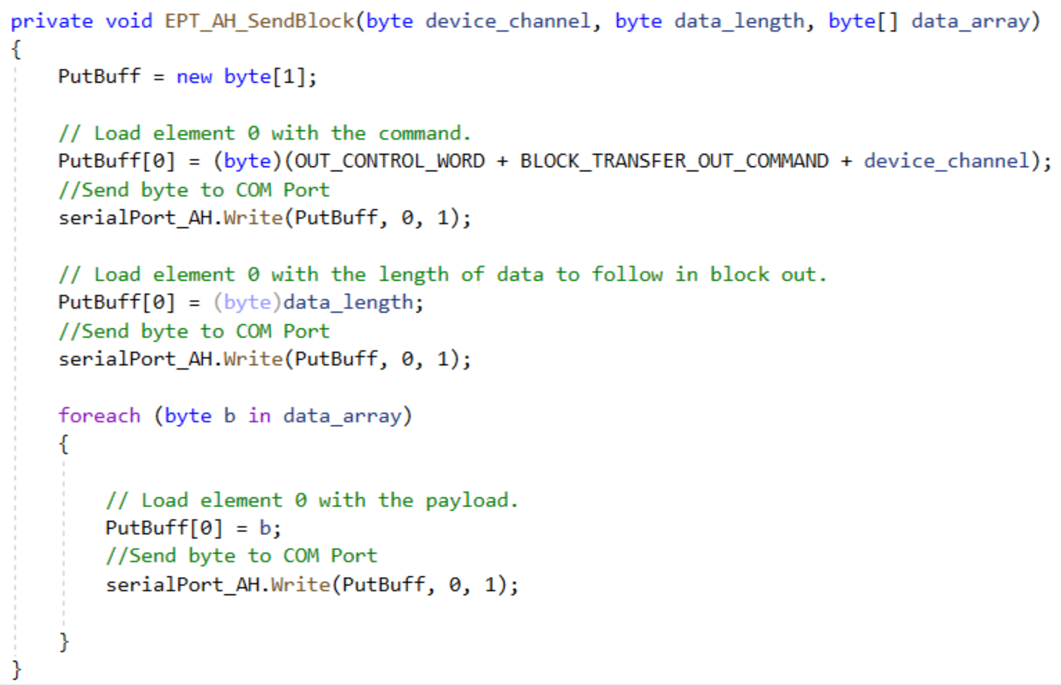
For our example project, the TransferOutReceive() function writes the Transfer byte received to a text block. The receive callback method is complex, however, Earth People Technology has created several projects which implement callbacks. Any part of these sample projects can copied and pasted into a user’s project.

### 5.4.5 Active Host Block Transfers

The Active Host Block Transfer is designed to transfer blocks of data between Host and FPGA and vice versa through the Block EndTerm. This allows buffers of data to be transferred with a minimal amount of code. The Active Host Block module (in the User

Code) is addressable, so up to eight individual modules can be instantiated and separately addressed. The length of the block to be transferred must also be specified. The Block EndTerm is limited to 1 to 256 bytes.

To send a block, first, open the EPT device to be used with openSerialPort1(), transfer module as the first parameter. Next, place the pointer to the buffer in the second parameter of EPT_AH_SendBlock(). Add the length of the buffer as the third parameter. Then execute the function, the entire buffer will be transferred to the USB chip. The data is available at the port of the Active Block module in the user code on the FPGA.



```
private void EPT_AH_SendBlock(byte device_channel, byte data_length, byte[] data_array)
{
    PutBuff = new byte[1];

    // Load element 0 with the command.
    PutBuff[0] = (byte)(OUT_CONTROL_WORD + BLOCK_TRANSFER_OUT_COMMAND + device_channel);
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);

    // Load element 0 with the length of data to follow in block out.
    PutBuff[0] = (byte)data_length;
    //Send byte to COM Port
    serialPort_AH.Write(PutBuff, 0, 1);

    foreach (byte b in data_array)
    {

        // Load element 0 with the payload.
        PutBuff[0] = b;
        //Send byte to COM Port
        serialPort_AH.Write(PutBuff, 0, 1);

    }
}
```

To receive a block transfer from the FPGA Device, a polling technique is used by the Active Host SDK. This is because the Bulk Transfer USB is a Host initiated bus. The Device will not transfer any bytes until the Host commands it to. If the Device has data to send to the Host in an asynchronous manner (meaning the Host did not command the Device to send data), the Host must periodically check the Device for data in its transmit FIFO. If data exists, the Host will command the Device to send its data. The received data is then stored into local memory and register bits are set that will indicate data has been received from a particular address. The receive callback function is then called from the Active Host SDK. This function start a thread to do something with the block data.

To receive a byte transfer from the callback function, user code must subscribe to the event created when the incoming byte transfer has arrived at the Read Callback function. The Read Callback must store the incoming transfer payload and module address in a local memory block. A switch statement is used to decode which event should be called to handle the incoming received data. The event handler function will check for any bytes read for that address.

```
private void EPTParseReceive()
{
    // Parse the command
    switch (EPTReceiveDevice.Command)
    {
        // This is a trigger command
        case TRIGGER_IN_COMMAND:
            TriggerOutReceive();
            break;
        // This is a single-byte transfer command
        case TRANSFER_IN_COMMAND:
            TransferOutReceive();
            break;
        // This is a data block transfer command
        case BLOCK_IN_COMMAND:
            BlockInReceive();
            break;
        default:
            break;
    }
}
```

The function EPTParseReceive() is called by the Read Callback function. The EPTParseReceive() function will examine the command of the incoming byte transfer and determine which receive function to call.

```
public void Receive_Block_In(object sender, EventArgs e)
{
    device[ept_data.Address].TransferPending = false;
    Thread.Sleep(5);
    if (device[ept_data.Address].ContinuosCountTest == false)
    {
        Thread t = new Thread(new ParameterizedThreadStart(BlockCompare));
        t.Start(ept_data.Address);
    }
    if (device[ept_data.Address].Repititions == 0)
    {
        Thread u = new Thread(new ParameterizedThreadStart(Display_Block_In));
        u.Start(BlockCount);
    }
    else if (BlockTransferInfinite | device[ept_data.Address].ContinuosCountTest)
    {
        if ((BlockCount % 100) == 0)
        {
            Thread u = new Thread(new ParameterizedThreadStart(Display_Block_In));
            u.Start(BlockCount);
        }
    }
}
```

For our example project, the Receive_Block_In() function writes the Transfer block received to a text block. The receive callback method is complex, however, Earth People Technology has created several projects which implement callbacks. Any part of these sample projects can copied and pasted into a user’s project.

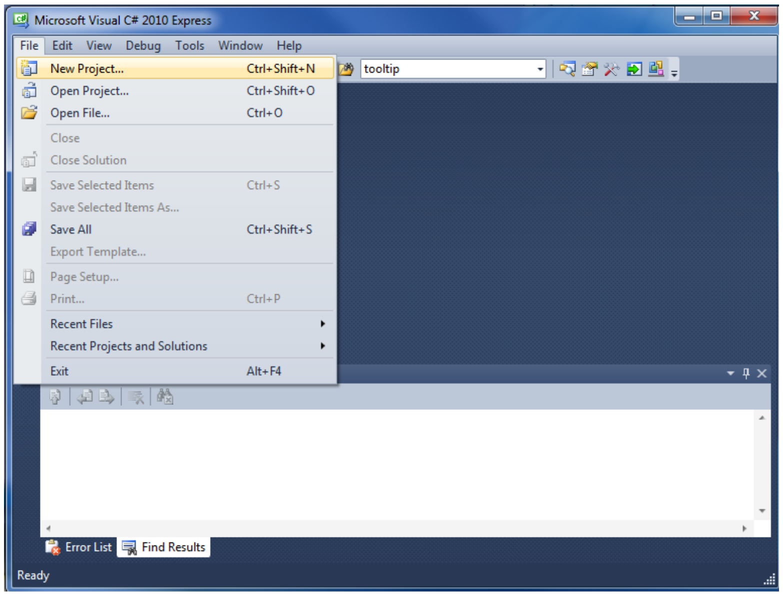
## 6 Assembling, Building, and Executing a .NET Project on the PC

The Active Host Application SDK is used to build a custom standalone executable on the PC that can perform Triggers and Transfer data to/from the DueProLogic. A standalone project can be range from a simple program to display and send data from the user to/from the Arduino Due. Or it can more complex to include receiving data, processing it, and start or end a process on the Arduino. This section will outline the procedures to take an example project and Assemble it, Build it, and Execute it.
This guide will focus on writing a Windows Forms application using the C# language for the Microsoft Visual Studio with .NET Framework. This is due to the idea that beginners can write effective Windows applications with the C# .NET Framework. They can focus on a subset of the language which is very similar to the C language. Anything that deviates from the subset of the C language, presented as in the Arduino

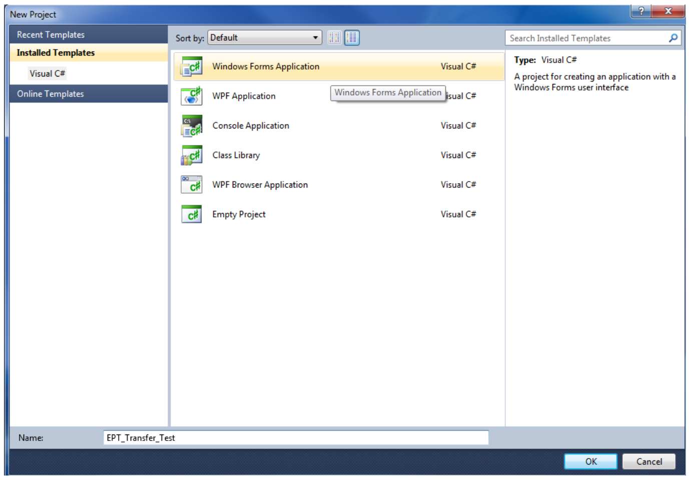
implication (such as events and controls), will be explained as the explanation progresses. Any language can be used with the Active Host Application SDK.

## 6.1 Creating a Project

Once the application is installed, open it up. Click on File->New Project.



At the New Project window, select the Windows Forms Application. Then, at the Name: box, type in EPT_Transfer_Demo

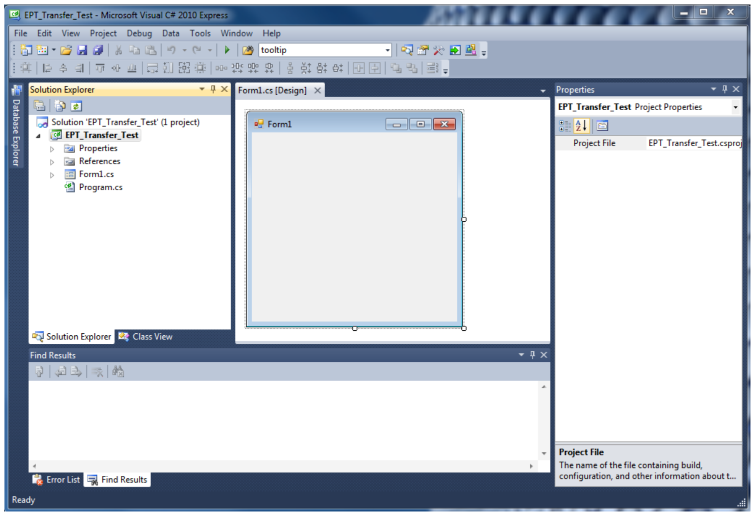

New Project
Recent Templates
Installed Templates
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Windows Forms Application
Console Application Visual C#
Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
OK
Cancel

Type: Visual C#

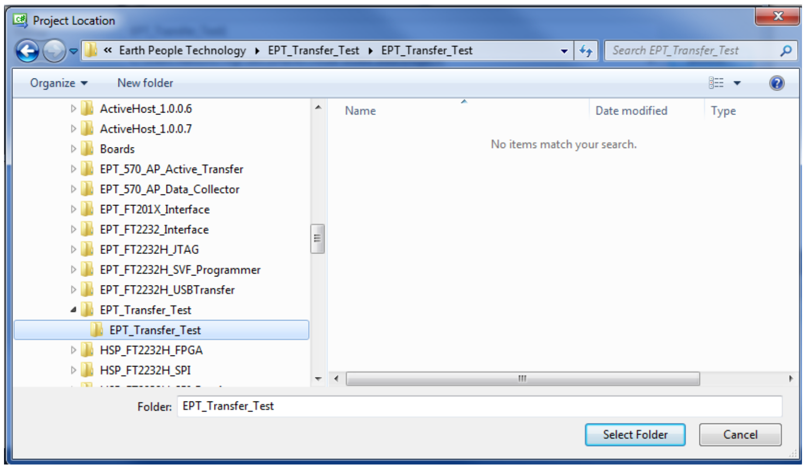
A project for creating an application with a Windows Forms user interface

Name: EPT_Transfer_Test

The project creation is complete.

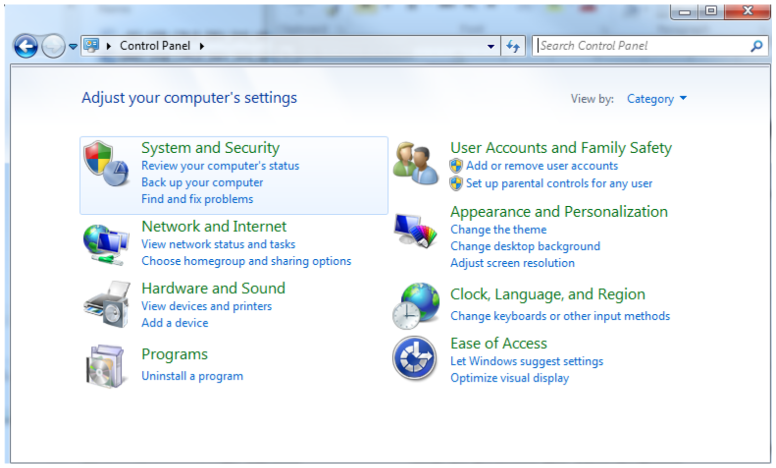

EPT_Transfer_Test - Microsoft Visual C# 2010 Express
File Edit View Project Debug Data Tools Window Help
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Form1.cs [Design]
Form1
Properties
EPT_Transfer_Test Project Properties
Project File
EPT_Transfer_Test.csproj
Find Results
Error List
Find Results
Project File
The name of the file containing build, configuration, and other information about t...
Ready

Save the project, go to File->Save as, browse to a folder to create EPT_Transfer_Demo folder. The default location is c:\Users\<Users Name>\documents\visual studio 2010\Projects.



### 6.1.1 Setting up the C# Express Environment x64 bit

The project environment must be set up correctly in order to produce an application that runs correctly on the target platform. If your system supports 64 bit operation, perform the following steps. Otherwise if your system is 32 bit skip to the Section, Assembling Files into the Project. Visual C# Express defaults to 32 bit operation. If you are unsure if your system supports, you can check it by going to Start->Control Panel->System and Security->System



Adjust your computer's settings

View by: Category

System and Security
Review your computer's status
Back up your computer
Find and fix problems

User Accounts and Family Safety
Add or remove user accounts
Set up parental controls for any user

Network and Internet
View network status and tasks
Choose homegroup and sharing options

Appearance and Personalization
Change the theme
Change desktop background
Adjust screen resolution

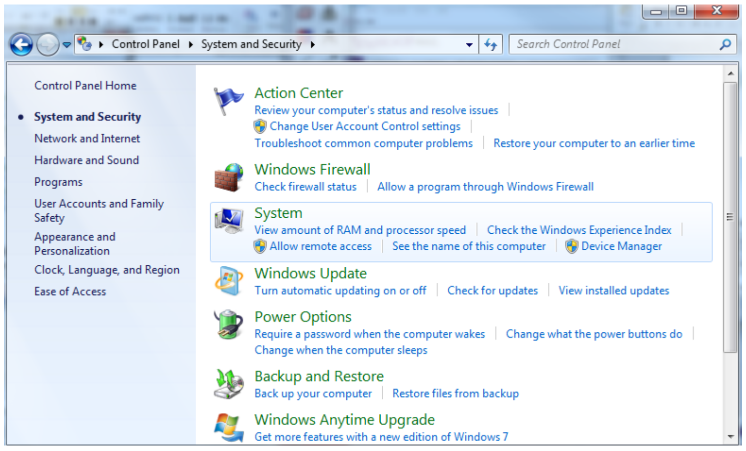
Hardware and Sound
View devices and printers
Add a device

Clock, Language, and Region
Change keyboards or other input methods

Programs
Uninstall a program

Ease of Access
Let Windows suggest settings
Optimize visual display

Click on System.


Control Panel ▸ System and Security
Search Control Panel
Control Panel Home
System and Security
Network and Internet
Hardware and Sound
Programs
User Accounts and Family Safety
Appearance and Personalization
Clock, Language, and Region
Ease of Access

Action Center
Review your computer's status and resolve issues | Change User Account Control settings | Troubleshoot common computer problems | Restore your computer to an earlier time

Windows Firewall
Check firewall status | Allow a program through Windows Firewall

System
View amount of RAM and processor speed | Check the Windows Experience Index | Allow remote access | See the name of this computer | Device Manager

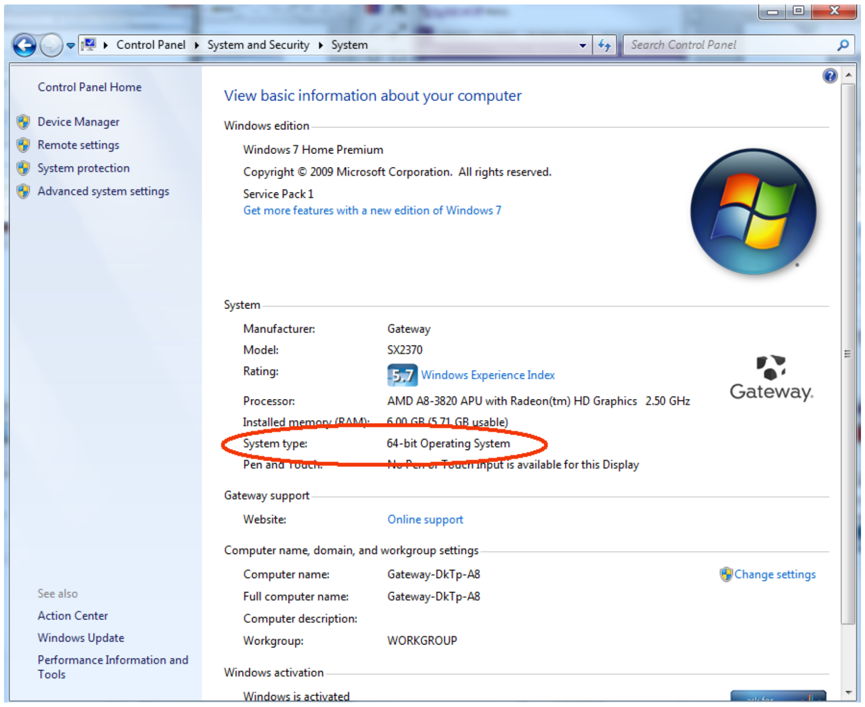
Windows Update
Turn automatic updating on or off | Check for updates | View installed updates

Power Options
Require a password when the computer wakes | Change what the power buttons do | Change when the computer sleeps

Backup and Restore
Back up your computer | Restore files from backup

Windows Anytime Upgrade
Get more features with a new edition of Windows 7

Check under System\System type:


Control Panel ▸ System and Security ▸ System

Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings

View basic information about your computer

Windows edition
Windows 7 Home Premium
Copyright © 2009 Microsoft Corporation. All rights reserved.
Service Pack 1
Get more features with a new edition of Windows 7

System
Manufacturer: Gateway
Model: SX2370
Rating: 5.7 Windows Experience Index
Processor: AMD A8-3820 APU with Radeon(tm) HD Graphics 2.50 GHz
Installed memory (RAM): 6.00 GB (5.71 GB usable)
System type: 64-bit Operating System
Pen and Touch: No Pen or Touch Input is available for this Display

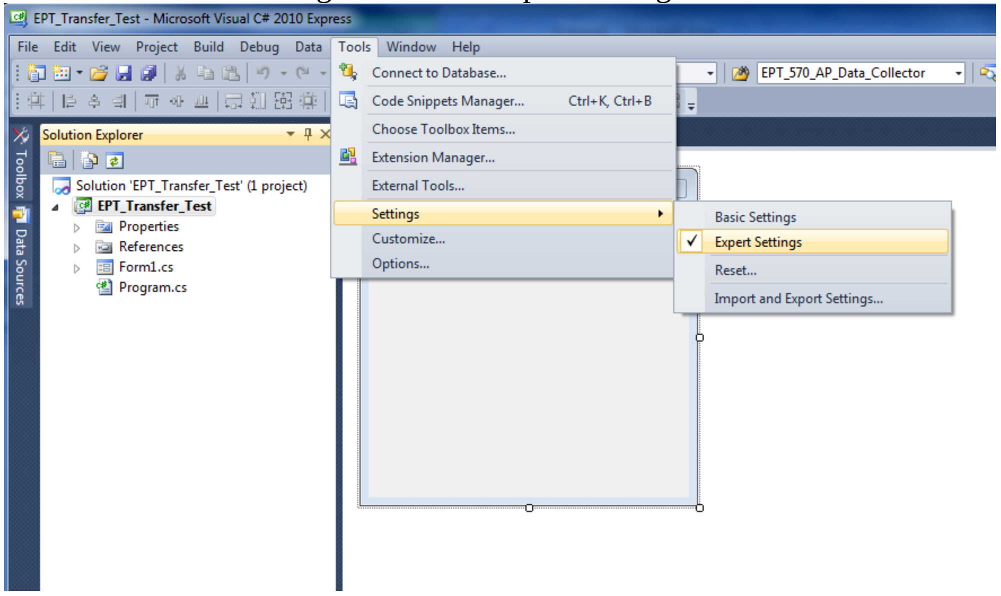
Gateway support
Website: Online support

Computer name, domain, and workgroup settings
Computer name: Gateway-DkTp-A8
Full computer name: Gateway-DkTp-A8
Computer description:
Workgroup: WORKGROUP
Change settings

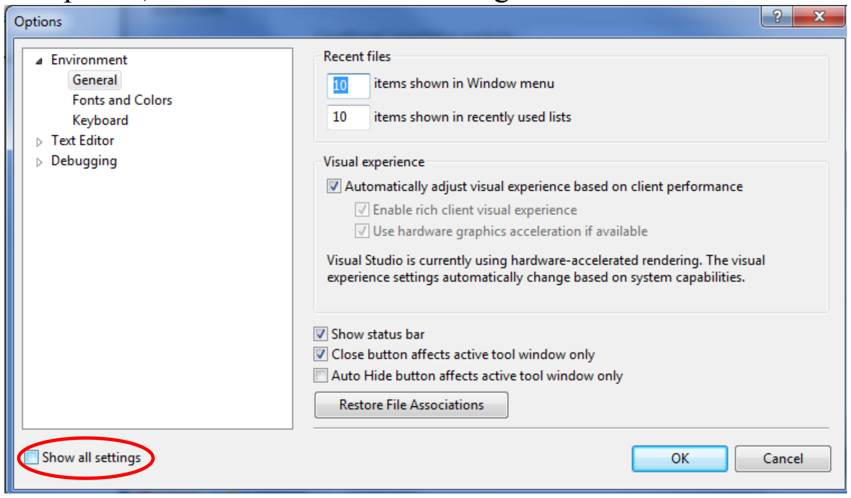
Windows activation
Windows is activated

See also
Action Center
Windows Update
Performance Information and Tools

First, we need tell C# Express to produce 64 bit code if we are running on a x64 platform. Go to Tools->Settings and select Expert Settings


EPT_Transfer_Test - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Tools Window Help
EPT_570_AP_Data_Collector
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Connect to Database...
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
Extension Manager...
External Tools...
Settings
Customize...
Options...
Basic Settings
✓ Expert Settings
Reset...
Import and Export Settings...
Toolbox
Data Sources

Go to Tools->Options, locate the “Show all settings” check box. Check the box.



Recent files
10 items shown in Window menu
10 items shown in recently used lists

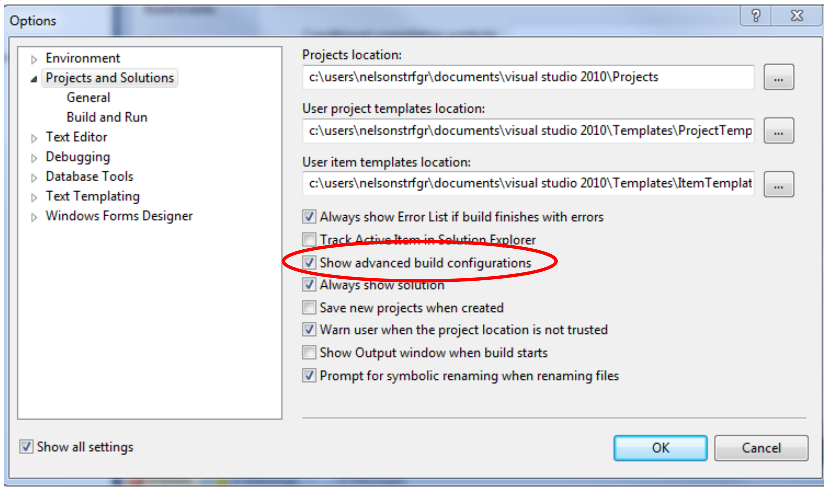
Visual experience
[✓] Automatically adjust visual experience based on client performance
[✓] Enable rich client visual experience
[✓] Use hardware graphics acceleration if available

Visual Studio is currently using hardware-accelerated rendering. The visual experience settings automatically change based on system capabilities.

[✓] Show status bar
[✓] Close button affects active tool window only
[ ] Auto Hide button affects active tool window only

[ ] Show all settings

In the window on the left, go to “Projects and Solutions”. Locate the “Show advanced build configurations” check box. Check the box.



Projects location:
c:\users\nelsonstrfgr\documents\visual studio 2010\Projects

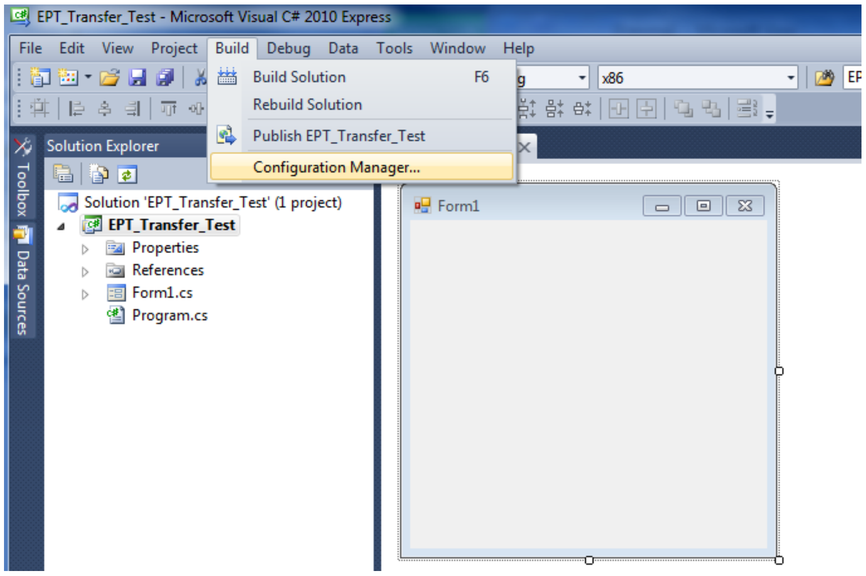
User project templates location:
c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ProjectTemp

User item templates location:
c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ItemTemplat

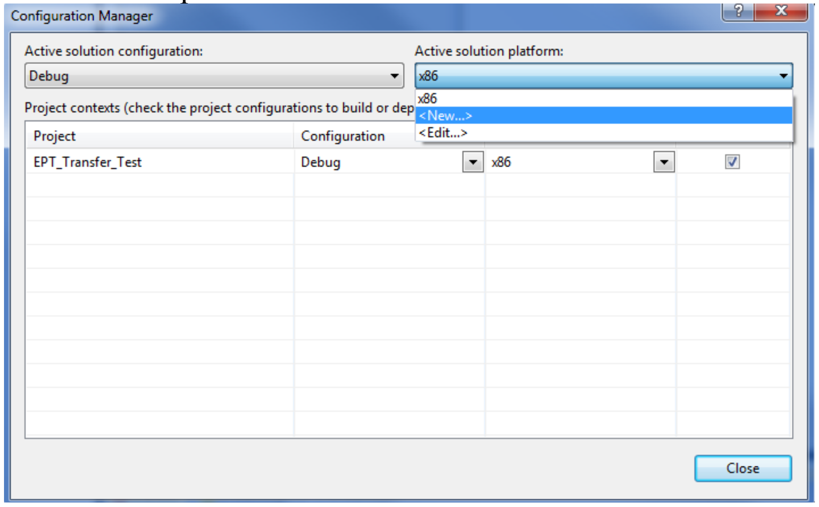
[✓] Always show Error List if build finishes with errors
[ ] Track Active Item in Solution Explorer
[✓] Show advanced build configurations
[✓] Always show solution
[ ] Save new projects when created
[✓] Warn user when the project location is not trusted
[ ] Show Output window when build starts
[✓] Prompt for symbolic renaming when renaming files

[✓] Show all settings

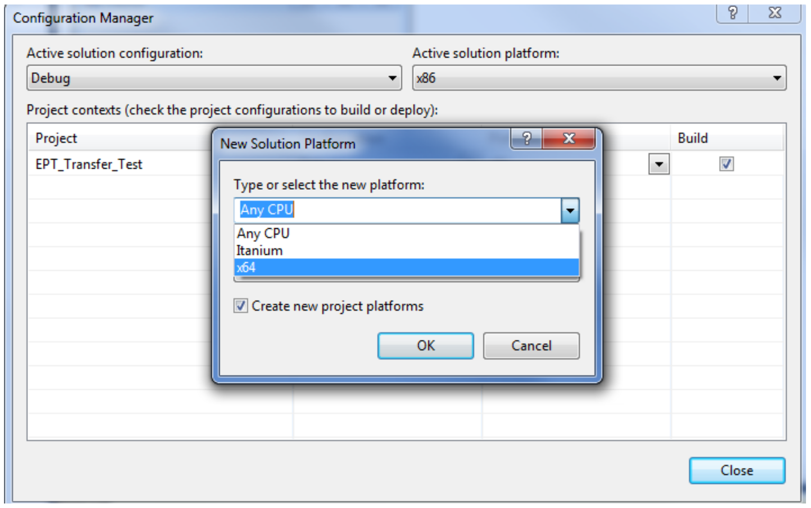
Go to Build->Configuration Manager.


EPT_Transfer_Test - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Tools Window Help
Build Solution F6
Rebuild Solution
Publish EPT_Transfer_Test
Configuration Manager...
x86
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Form1
Toolbox
Data Sources

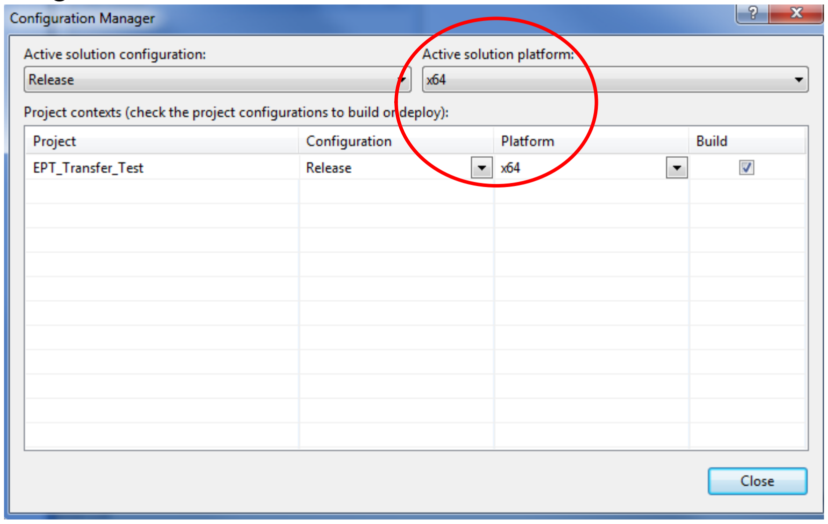
In the Configuration Manager window, locate the “Active solution platform:” label, select “New” from the drop down box.


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
x86
x86
<New...>
<Edit...>
Project contexts (check the project configurations to build or dep
Project
Configuration
EPT_Transfer_Test
Debug
x86
Close

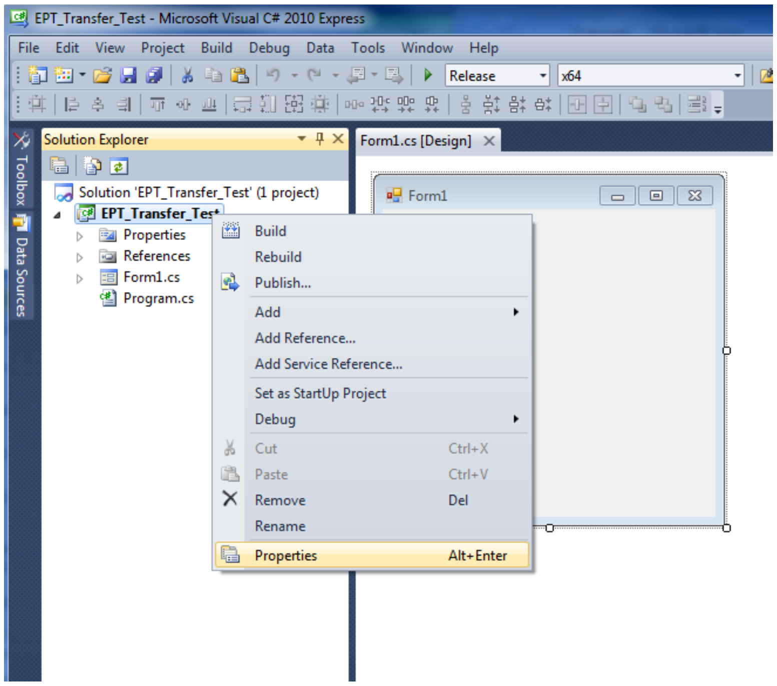
In the New Solution Platform window, click on the drop down box under “Type or select the new platform:”. Select “x64”.


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
x86
Project contexts (check the project configurations to build or deploy):
Project
EPT_Transfer_Test
Build
New Solution Platform
Type or select the new platform:
Any CPU
Any CPU
Itanium
x64
Create new project platforms
OK
Cancel
Close

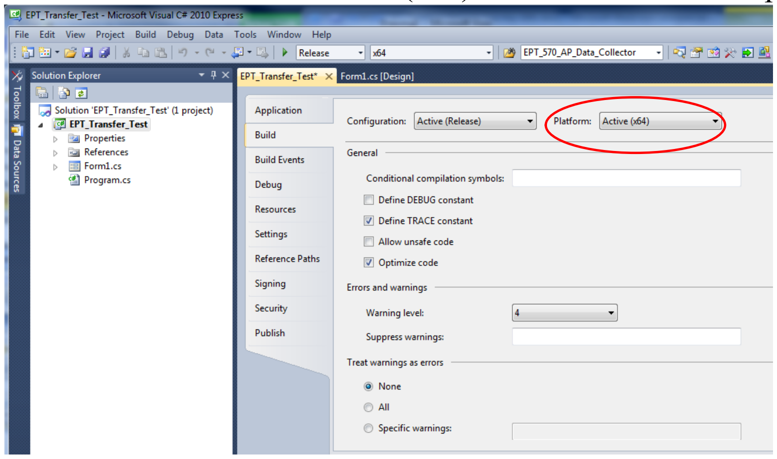
Click the Ok button. Verify that the "Active Solution Platform" and the "Platform" tab are both showing "x64".


Configuration Manager
Active solution configuration:
Release
Active solution platform:
x64
Project contexts (check the project configurations to build or deploy):
Project
Configuration
Platform
Build
EPT_Transfer_Test
Release
x64
Close

Also, select "Release" under "Active solution configuration". Click Close.
Then, using the Solution Explorer, you can right click on the project, select Properties and click on the Build tab on the right of the properties window.



Verify that the "Platform:" label has "Active (x64)" selected from the drop down box.



Configuration: Active (Release)  Platform: Active (x64)

General
Conditional compilation symbols:
- [ ] Define DEBUG constant
- [✓] Define TRACE constant
- [ ] Allow unsafe code
- [✓] Optimize code

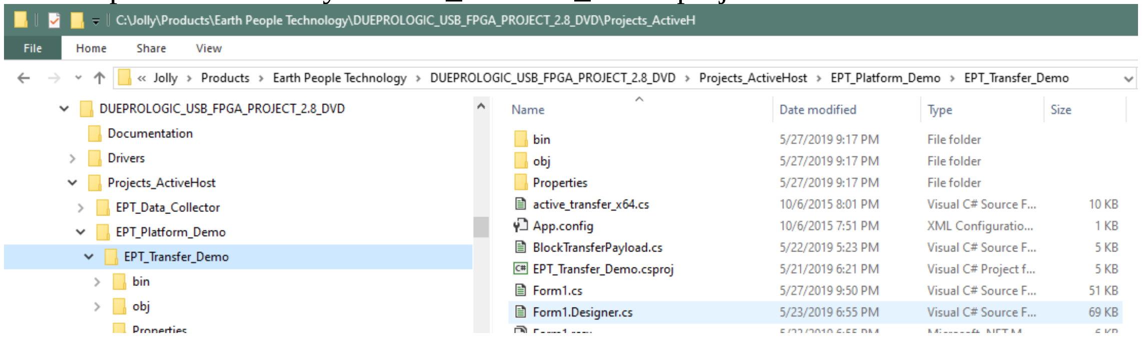
Errors and warnings
Warning level: 4
Suppress warnings:

Treat warnings as errors
- [✓ None]
- All
- Specific warnings:

Click on the Save All button on the tool bar. The project environment is now setup and ready for the project files. Close the Project.

## *6.2 Assembling Files into the Project*

Locate the EPT FPGA Development System DVD installed on your PC. Browse to the EPT_Platform_Demo folder where the Project files, copy the*.cs files, and install them in the top level folder of your EPT_Platform_Demo project.



C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD\Projects_ActiveH

« Jolly › Products › Earth People Technology › DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD › Projects_ActiveHost › EPT_Platform_Demo › EPT_Transfer_Demo

```
DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD
  Documentation
  Drivers
  Projects_ActiveHost
    EPT_Data_Collector
    EPT_Platform_Demo
      EPT_Transfer_Demo
        bin
        obj
        Properties
```

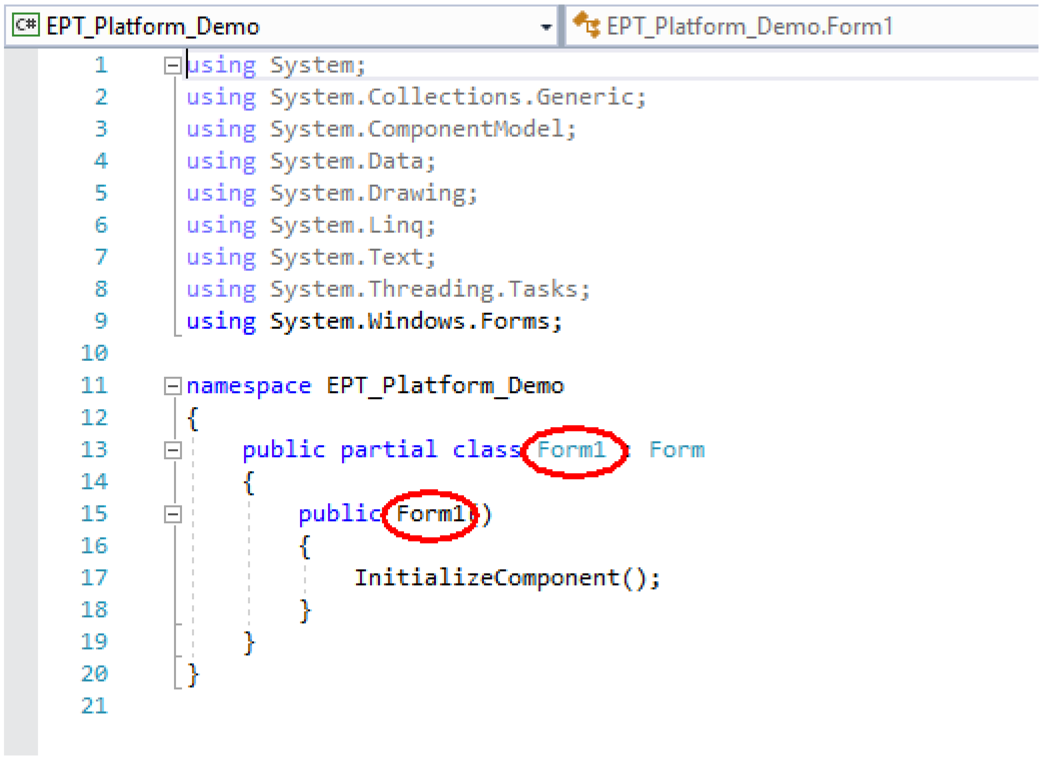
| Name | Date modified | Type | Size |
|---|---|---|---|
| bin | 5/27/2019 9:17 PM | File folder | |
| obj | 5/27/2019 9:17 PM | File folder | |
| Properties | 5/27/2019 9:17 PM | File folder | |
| active_transfer_x64.cs | 10/6/2015 8:01 PM | Visual C# Source F... | 10 KB |
| App.config | 10/6/2015 7:51 PM | XML Configuratio... | 1 KB |
| BlockTransferPayload.cs | 5/22/2019 5:23 PM | Visual C# Source F... | 5 KB |
| EPT_Transfer_Demo.csproj | 5/21/2019 6:21 PM | Visual C# Project f... | 5 KB |
| Form1.cs | 5/27/2019 9:50 PM | Visual C# Source F... | 51 KB |
| Form1.Designer.cs | 5/23/2019 6:55 PM | Visual C# Source F... | 69 KB |

### 6.2.1 Changing Project Name

***NOTE***

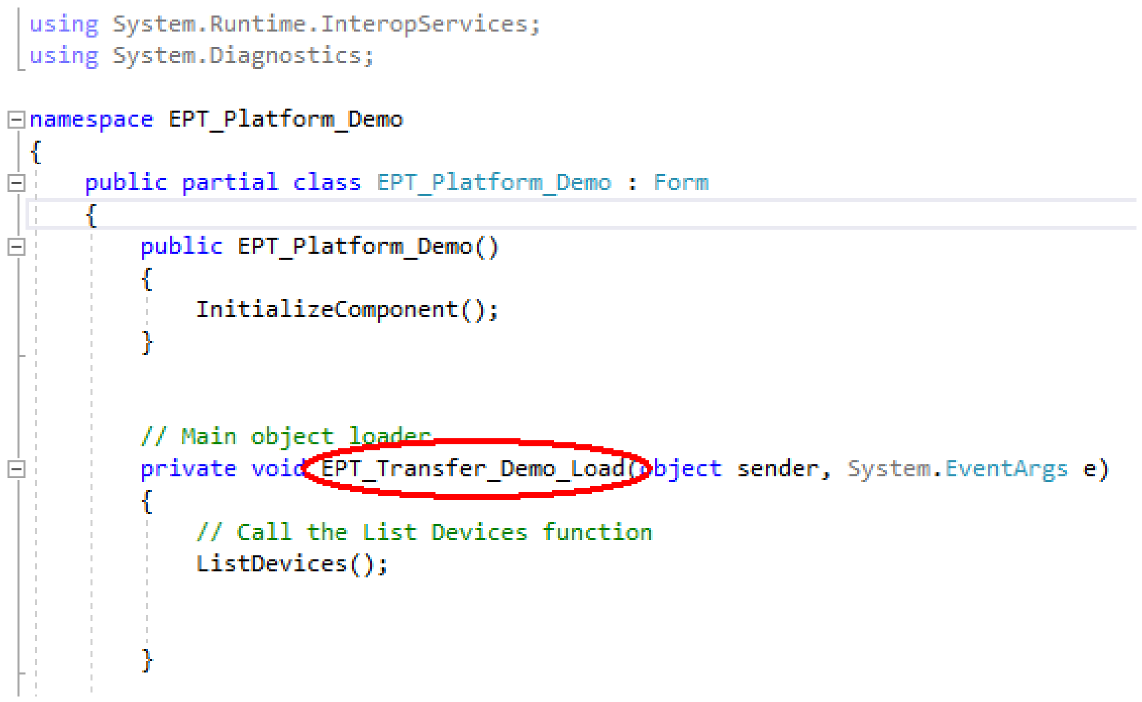
If you named your project something other than EPT_Platform_Demo, you will have to make changes to the *.cs files above. This is because Visual C# Express links the project files and program files together. These chages can be made by modifying the following:

1. Change namespace of Form1.cs to new project name.
2. Change class of Form1.cs to new project name.
3. Change constructor of Form1.cs to new project name.



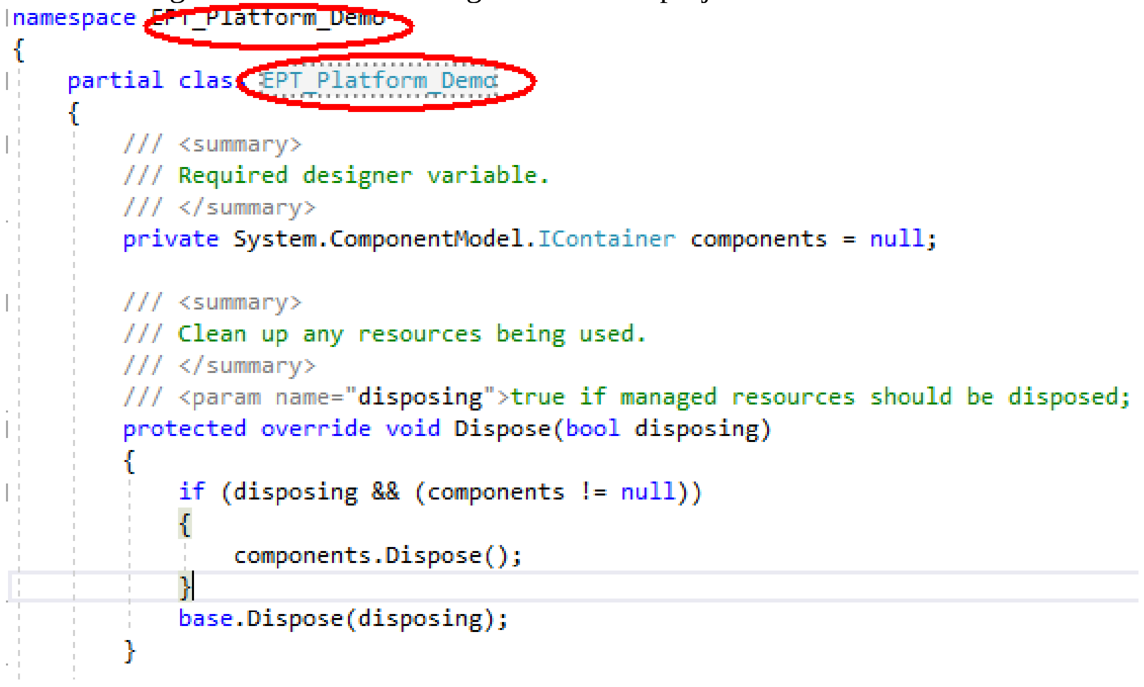
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EPT_Platform_Demo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

4. Change EPT_Transfer_Demo_Load of Form1.cs to new <project name>_Load



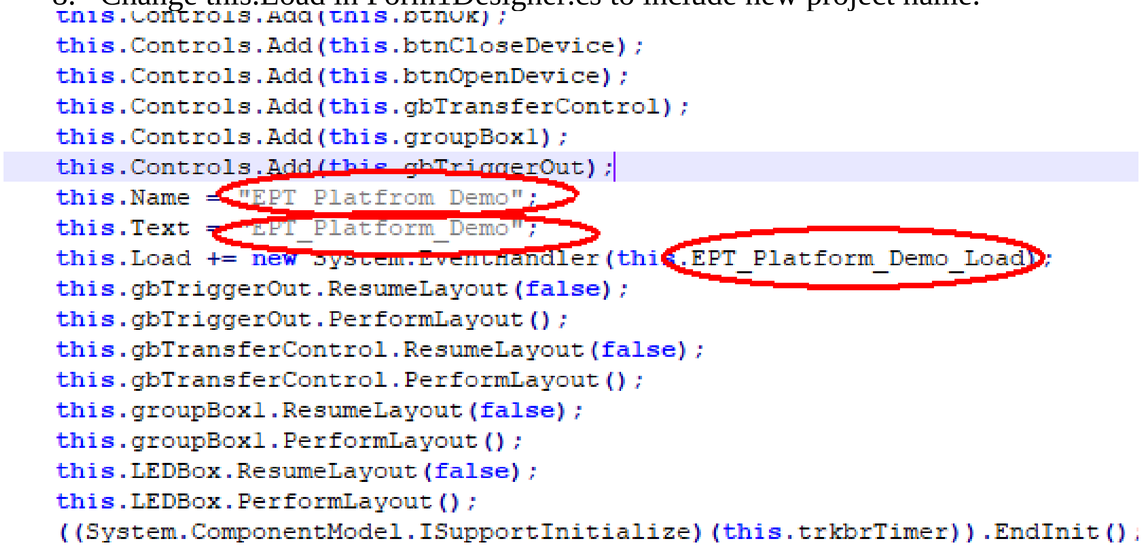
```
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace EPT_Platform_Demo
{
    public partial class EPT_Platform_Demo : Form
    {
        public EPT_Platform_Demo()
        {
            InitializeComponent();
        }


        // Main object loader
        private void EPT_Transfer_Demo_Load(object sender, System.EventArgs e)
        {
            // Call the List Devices function
            ListDevices();


        }
```

5. Change namespace of Form1.Designer.cs to new project name.
6. Change clase of Form1.Designer.cs to new project name.



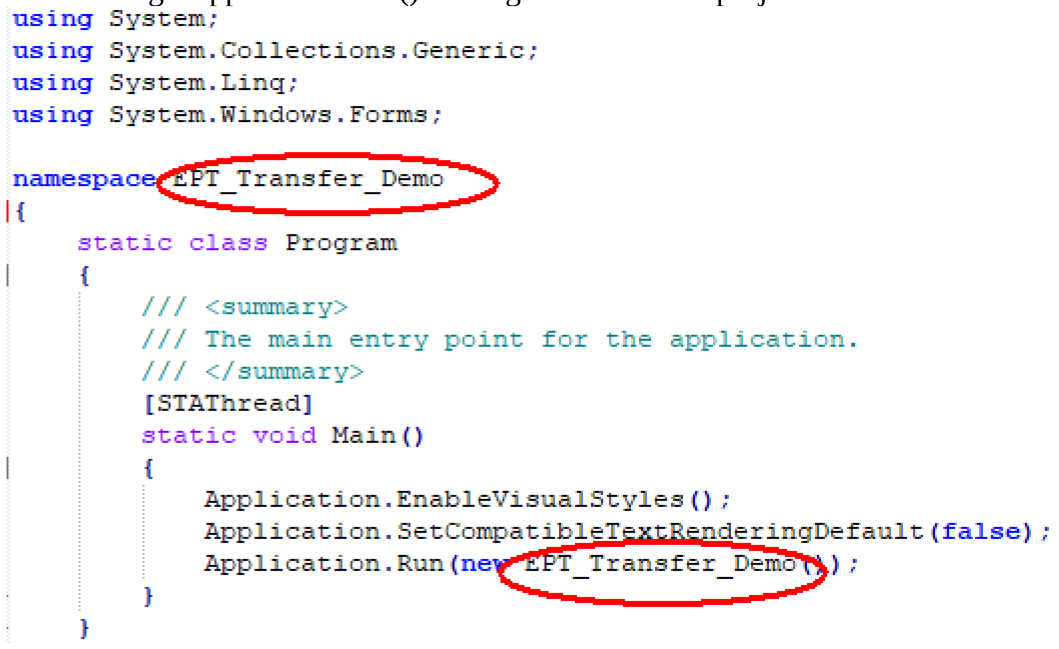
```
namespace EPT_Platform_Demo
{
    partial class EPT_Platform_Demo
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

7. Change the this.Name and this.Text in Form1Designer.cs to new project name.
8. Change this.Load in Form1Designer.cs to include new project name.



```
this.Controls.Add(this.btnOK);
this.Controls.Add(this.btnCloseDevice);
this.Controls.Add(this.btnOpenDevice);
this.Controls.Add(this.gbTransferControl);
this.Controls.Add(this.groupBox1);
this.Controls.Add(this.gbTriggerOut);
this.Name = "EPT Platfrom Demo";
this.Text = "EPT_Platform_Demo";
this.Load += new System.EventHandler(this.EPT_Platform_Demo_Load);
this.gbTriggerOut.ResumeLayout(false);
this.gbTriggerOut.PerformLayout();
this.gbTransferControl.ResumeLayout(false);
this.gbTransferControl.PerformLayout();
this.groupBox1.ResumeLayout(false);
this.groupBox1.PerformLayout();
this.LEDBox.ResumeLayout(false);
this.LEDBox.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.trkbrTimer)).EndInit();
```

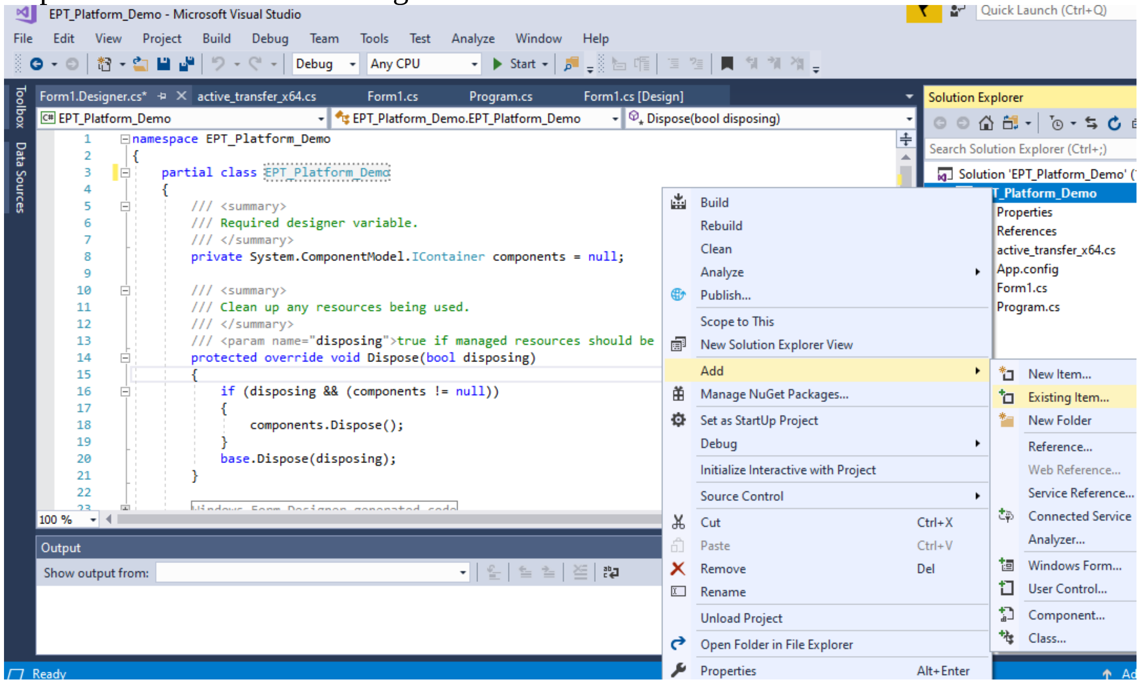
9. Change namespace in Program.cs to new project name
10. Change Application.Run() in Program .cs to new projectname.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace EPT_Transfer_Demo
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new EPT_Transfer_Demo());
        }
    }
```

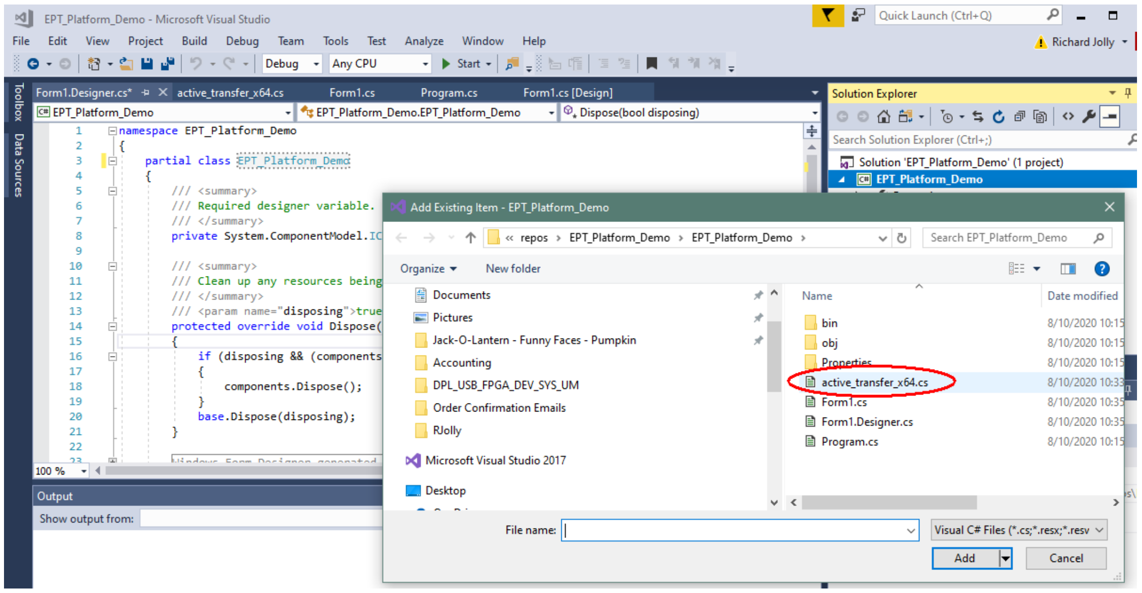
### 6.2.2 Add Files to Project

Open the EPT_Platform_Demo project. Right click on the project in the Solutions Explorer. Select Add->Existing Item.



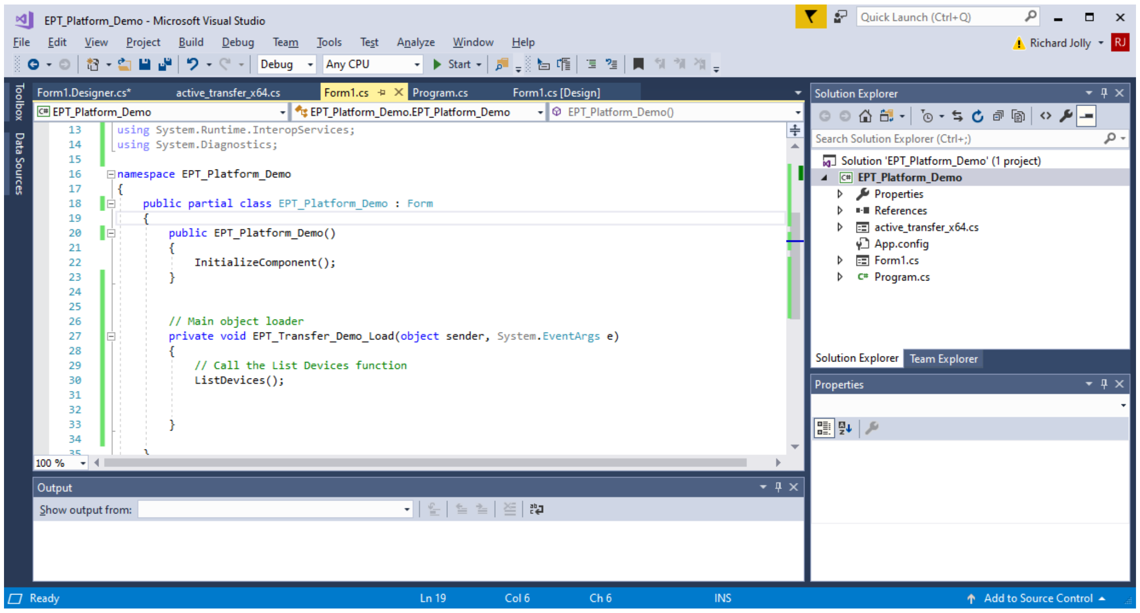
```
namespace EPT_Platform_Demo
{
    partial class EPT_Platform_Demo
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

Browse to the EPT_Platform_Demo project folder and select the active_transfer_64.cs file. Click Add.


EPT_Platform_Demo - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
Richard Jolly
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
Any CPU
Start
Form1.Designer.cs*
active_transfer_x64.cs
Form1.cs
Program.cs
Form1.cs [Design]
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'EPT_Platform_Demo' (1 project)
EPT_Platform_Demo
Add Existing Item - EPT_Platform_Demo
« repos › EPT_Platform_Demo › EPT_Platform_Demo ›
Search EPT_Platform_Demo
Organize
New folder
Documents
Pictures
Jack-O-Lantern - Funny Faces - Pumpkin
Accounting
DPL_USB_FPGA_DEV_SYS_UM
Order Confirmation Emails
RJolly
Microsoft Visual Studio 2017
Desktop
Name
Date modified
bin 8/10/2020 10:1
obj 8/10/2020 10:1
Properties 8/10/2020 10:1
active_transfer_x64.cs 8/10/2020 10:3
Form1.cs 8/10/2020 10:3
Form1.Designer.cs 8/10/2020 10:3
Program.cs 8/10/2020 10:1
File name:
Visual C# Files (*.cs;*.resx;*.resv
Add
Cancel
100 %
Output
Show output from:

```
namespace EPT_Platform_Demo
{
    partial class EPT_Platform_Demo
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IC
        
        /// <summary>
        /// Clean up any resources being
        /// </summary>
        /// <param name="disposing">true
        protected override void Dispose(
        {
            if (disposing && (components
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

In the C# Express Solution Explorer, you should be able to browse the files by clicking on them. There should be no errors noted in the Error List box.



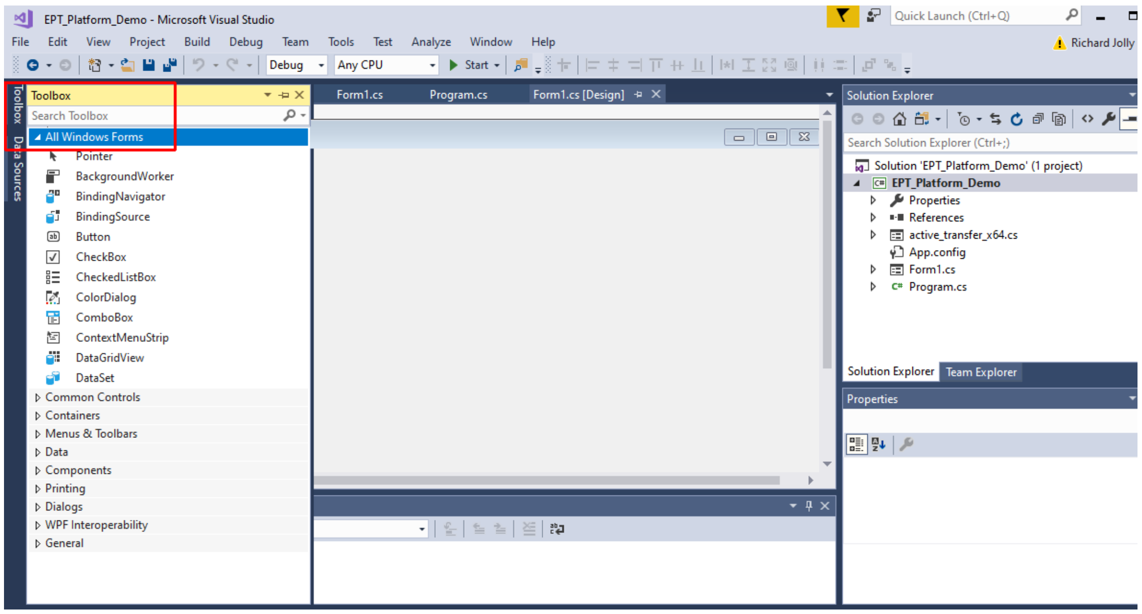
```
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace EPT_Platform_Demo
{
    public partial class EPT_Platform_Demo : Form
    {
        public EPT_Platform_Demo()
        {
            InitializeComponent();
        }


        // Main object loader
        private void EPT_Transfer_Demo_Load(object sender, System.EventArgs e)
        {
            // Call the List Devices function
            ListDevices();


        }
```

## 6.2.3 Adding Controls to the Project

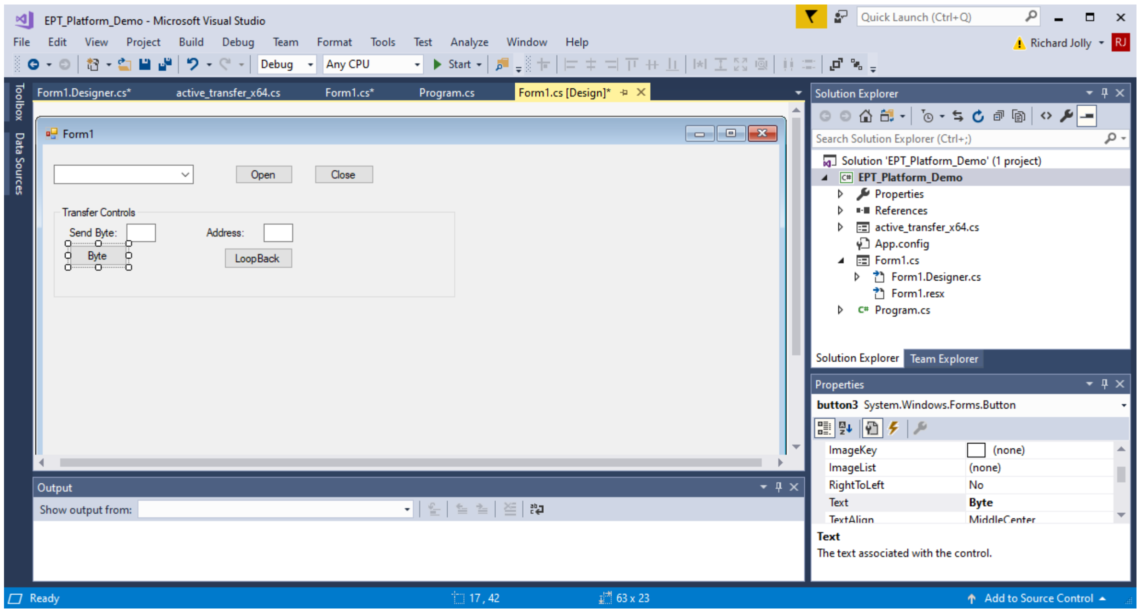
Although, the C# language is very similar to C Code, there are a few major differences. The first is C# .NET environment is event based. A second is C# utilizes classes. This guide will keep the details of these items hidden to keep things simple. However, a brief introduction to events and classes will allow the beginner to create effective programs.

Event based programming means the software responds to events created by the user, a timer event, external events such as serial communication into PC, internal events such as the OS, or other events. The events we are concerned with for our example program are user events and the timer event. The user events occur when the user clicks on a button on the Windows Form or selects a radio button. We will add a button to our example program to show how the button adds an event to the Windows Form and a function that gets executed when the event occurs.

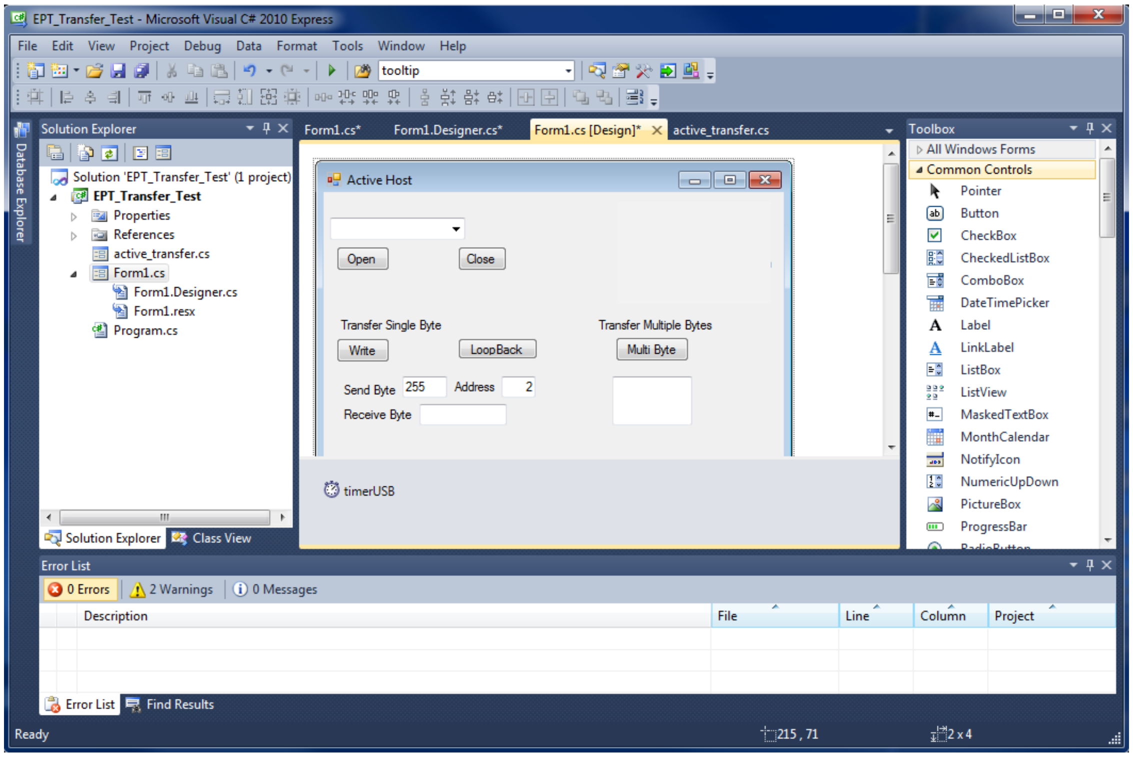
The easiest way to add a button to a form is to double click the Form1.cs in the Solution Explorer. Click on the button to launch the Toolbox.



Locate the button on the Toolbox, grab and drag the button onto the Form1.cs [Design] and drop it near the top.



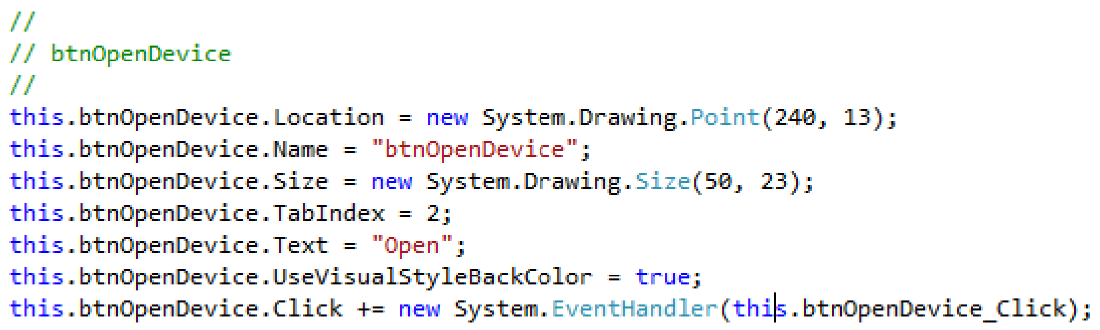
Go to the Properties box and locate the (Name) cell. Change the name to “btnOpenDevice”. Locate the Text cell, and change the name to Open.



Double click on the Open button. The C# Explorer will automatically switch to the Form1.cs code view. The callback function will be inserted with the name of the button along with "_click" appended to it. The parameter list includes (object sender, System.EventArgs e). These two additions are required for the callback function to initiate when the "click" event occurs.

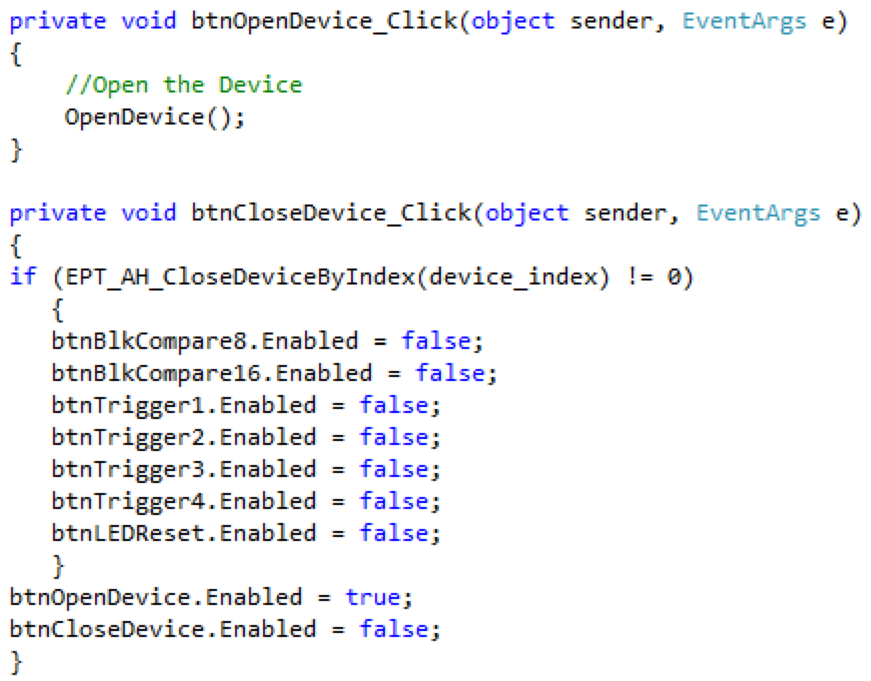
Private void btnOpenDevice_click(object sender, System.EventArgs e)

There is one more addition to the project files. Double click on the Form1.Designer.cs file in the Solution Explorer. Locate the following section of code.



```
//
// btnOpenDevice
//
this.btnOpenDevice.Location = new System.Drawing.Point(240, 13);
this.btnOpenDevice.Name = "btnOpenDevice";
this.btnOpenDevice.Size = new System.Drawing.Size(50, 23);
this.btnOpenDevice.TabIndex = 2;
this.btnOpenDevice.Text = "Open";
this.btnOpenDevice.UseVisualStyleBackColor = true;
this.btnOpenDevice.Click += new System.EventHandler(this.btnOpenDevice_Click);
```

This code sets up the button, size, placement, and text. It also declares the "System.EventHandler()". This statement sets the click method (which is a member of the button class) of the btnOpenDevice button to call the EventHandler – btnOpenDevice_Click. This is where the magic of the button click event happens.



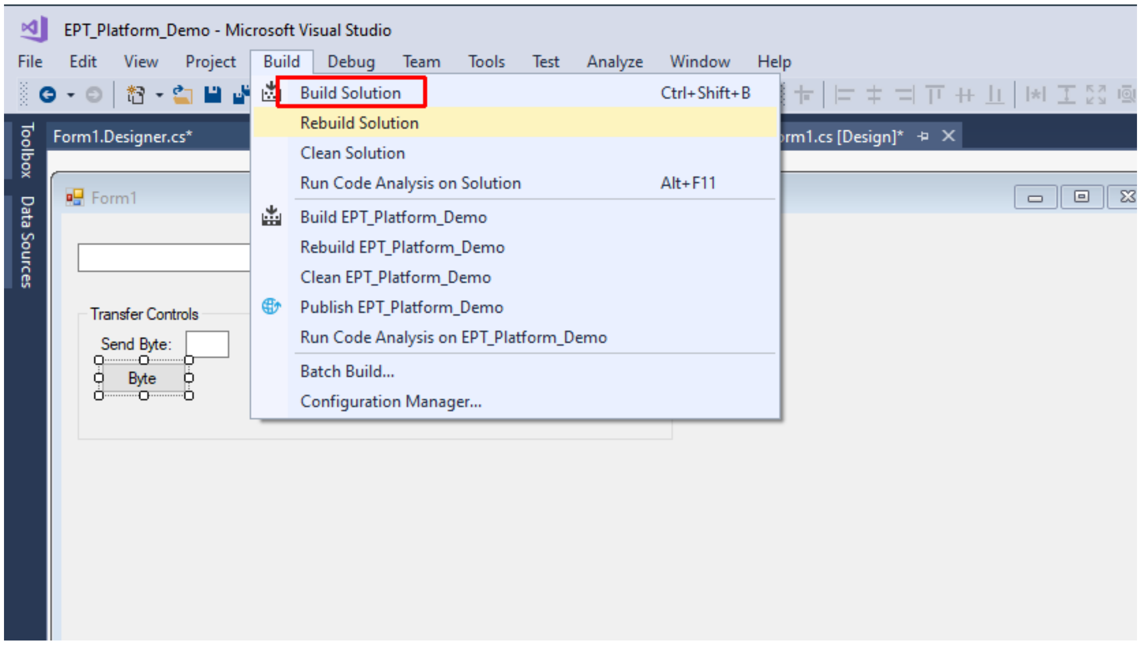
```
private void btnOpenDevice_Click(object sender, EventArgs e)
{
    //Open the Device
    OpenDevice();
}

private void btnCloseDevice_Click(object sender, EventArgs e)
{
if (EPT_AH_CloseDeviceByIndex(device_index) != 0)
   {
   btnBlkCompare8.Enabled = false;
   btnBlkCompare16.Enabled = false;
   btnTrigger1.Enabled = false;
   btnTrigger2.Enabled = false;
   btnTrigger3.Enabled = false;
   btnTrigger4.Enabled = false;
   btnLEDReset.Enabled = false;
   }
btnOpenDevice.Enabled = true;
btnCloseDevice.Enabled = false;
}
```

When btnOpenDevice_Click is called, it calls the function "OpenDevice()". This function will connect to the device selected in the combo box. This is a quick view of how to create, add files, and add controls to a C# project. The user is encouraged to spend some time reviewing the online tutorial at

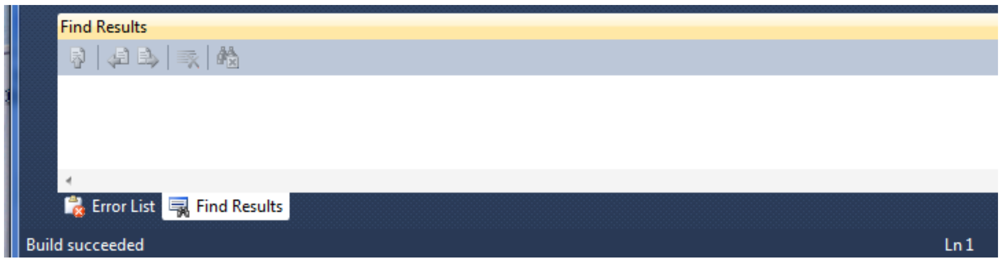
http://www.homeandlearn.co.uk/csharp/csharp.html to become intimately familiar with Visual C# .NET programming. In the meantime, follow the examples from the Earth People Technology to perform some simple reads and writes to the EPT USB-FPGA Development System.

### 6.2.4 Building the Project

Building the EPT_Platform_Demo project will compile the code in the project and produce an executable file. To build the project, go to Debug->Build Solution.



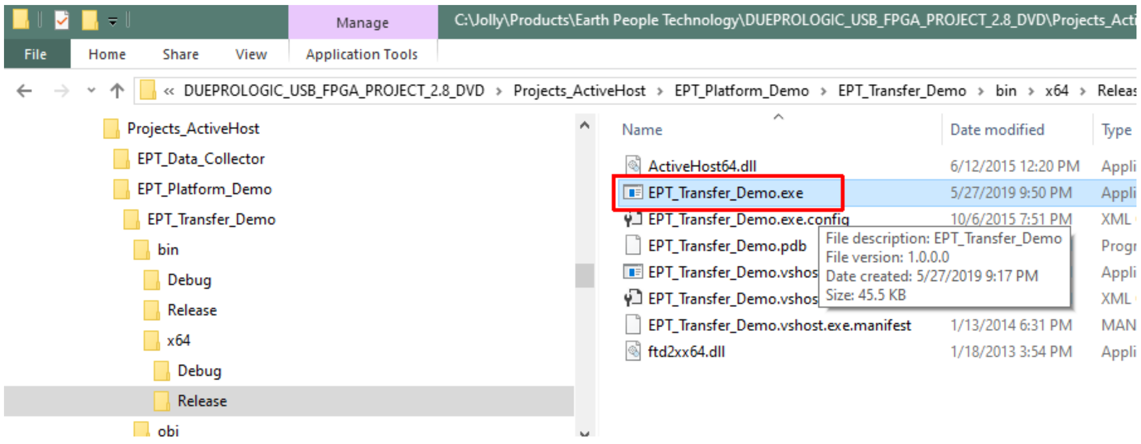
The C# Express compiler will start the building process. If there are no errors with code syntax, function usage, or linking, then the environment responds with “Build Succeeded”.



Build succeeded

### 6.2.5 Testing the Project

Once the project has been successfully built, it produces an *.exe file. The file will be saved in the Release or Debug folders.



C:\Jolly\Products\Earth People Technology\DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD\Projects_Acti

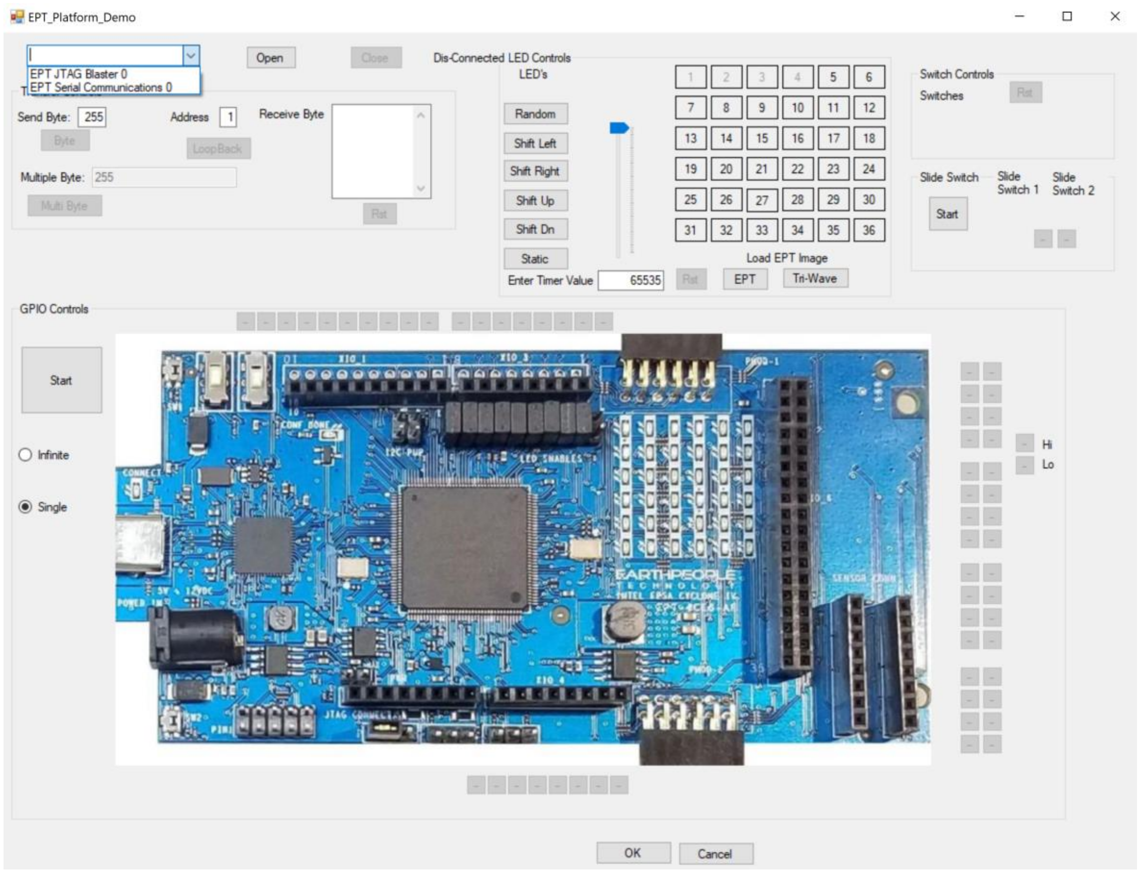
« DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD › Projects_ActiveHost › EPT_Platform_Demo › EPT_Transfer_Demo › bin › x64 › Releas

Projects_ActiveHost
EPT_Data_Collector
EPT_Platform_Demo
EPT_Transfer_Demo
bin
Debug
Release
x64
Debug
Release
obj

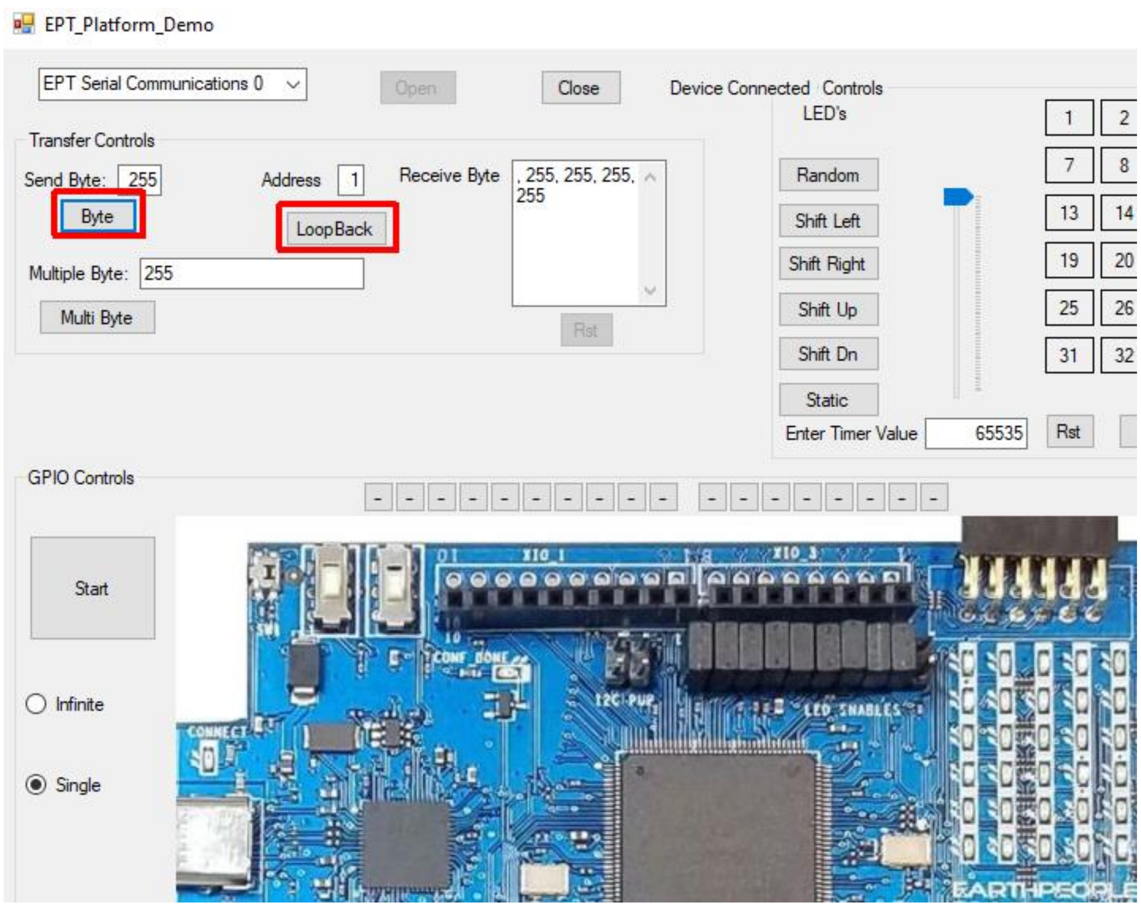
| Name | Date modified | Type |
|---|---|---|
| ActiveHost64.dll | 6/12/2015 12:20 PM | Appli |
| EPT_Transfer_Demo.exe | 5/27/2019 9:50 PM | Appli |
| EPT_Transfer_Demo.exe.config | 10/6/2015 7:51 PM | XML |
| EPT_Transfer_Demo.pdb | | Progr |
| EPT_Transfer_Demo.vshos | | Appli |
| EPT_Transfer_Demo.vshos | | XML |
| EPT_Transfer_Demo.vshost.exe.manifest | 1/13/2014 6:31 PM | MAN |
| ftd2xx64.dll | 1/18/2013 3:54 PM | Appli |

File description: EPT_Transfer_Demo
File version: 1.0.0.0
Date created: 5/27/2019 9:17 PM
Size: 45.5 KB

The EPT_Platform_Demo.exe file can now be tested using the DueProLogic board. To test the file, connect the DueProLogic to the Windows PC using Type A to USB-C cable. Make sure the driver for the board loads. If the USB driver fails to load, the Windows OS will indicate that no driver was loaded for the device. Go to the folder where the EPT_Platform_Demo.exe file resides, and double click on the file. The application should load with a Windows form.



With the application loaded, select the USB-FPGA board from the dropdown combo box and click on the “Open” button.



Click on one of the LED buttons in the middle of the window. The corresponding LED on the DueProLogic board should light up.

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

If there are any questions about the hardware, software, drivers, user manual, please contact Earth People Technology. There are three methods to contact EPT for support:

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