EARTH PEOPLE TECHNOLOGY, Inc

UnoMax CPLD DEVELOPMENT SYSTEM User Manual

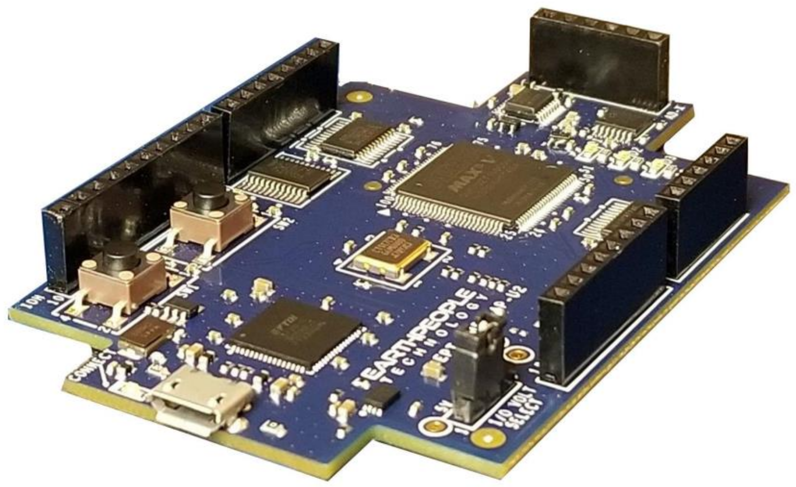
The UnoMax CPLD development system provides an innovative method of developing and debugging programmable logic code. It also provides a high speed data transfer mechanism between user code and a host PC. The UnoMax CPLD development system provides a convenient, user-friendly work flow by connecting seamlessly with Intel/Altera's Quartus Prime Lite software. The user will develop the code in the Quartus environment on a Windows Personal Computer. The programmable logic code is then synthesized and loaded into the CPLD using only the Quartus Programmer tool and a standard USB cable. The Active Host SDK provides a highly configurable communications interface between the UnoMax and host PC. The board connects transparently with the Active Transfer Library in the CPLD code. This Active Host/Active Transfer combination eliminates the complexity of designing a USB communication system. The UnoMax CPLD development system is a unique combination of hardware and software.

http://www.earthpeopletechnology.com/

Table of Contents

1 Introduction and General Description

The Earth People Technology UnoMax CPLD development system hardware consists of a High Speed USB chip, a CPLD and a four channel ADC. The USB interface provides both JTAG programming for the CPLD and a High Speed communications path. The CPLD is an Intel/Altera MAXV chip providing 570 Logic Cells. The ADC is a four channel 300KSample/sec data converter. The software consists of the Active Host SDK for the PC. The firmware includes the Active Transfer Library which is used in the CPLD to provide advanced functions for control and data transfer to/from a PC connection.

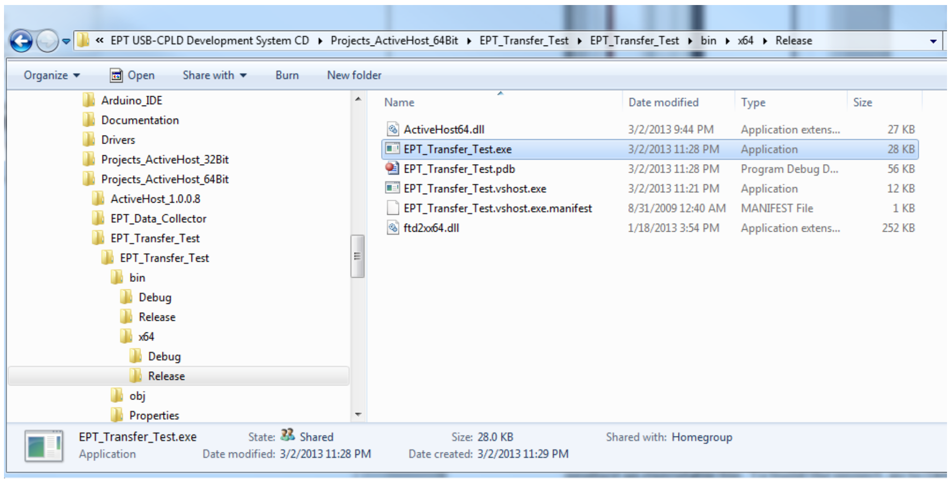


The UnoMax Development System allows users to write HDL code (either Verilog or VHDL) that will implement any digital logic circuit. The user's HDL code is compiled and synthesized and packaged into a programming file. The programming file is programmed into the CPLD using the JTAG channel of the USB to Serial chip, the

FT2232H.The Active Host SDK contains a dll which maintains device connection, polling, writes and includes a unique receive mechanism that automatically transfers data from UnoMax when data is ready. It also alerts the user code when the dll 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 dll in their projects. The Active Transfer Libraries must be included in the CPLD 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 UnoMax board comes pre-loaded with the EPT_Transfer_Test HDL project in the CPLD. This project allows the user to test out the functions of the Active Host API and the board hardware.

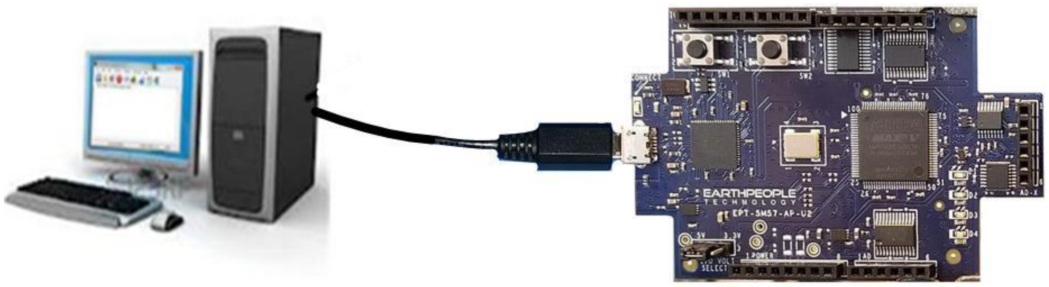

« EPT USB-CPLD Development System CD ▸ Projects_ActiveHost_64Bit ▸ EPT_Transfer_Test ▸ EPT_Transfer_Test ▸ bin ▸ x64 ▸ Release
Organize ▾
Open
Share with ▾
Burn
New folder

- Arduino_IDE
- Documentation
- Drivers
- Projects_ActiveHost_32Bit
- Projects_ActiveHost_64Bit
  - ActiveHost_1.0.0.8
  - EPT_Data_Collector
  - EPT_Transfer_Test
    - EPT_Transfer_Test
      - bin
        - Debug
        - Release
        - x64
          - Debug
          - Release
      - obj
      - Properties

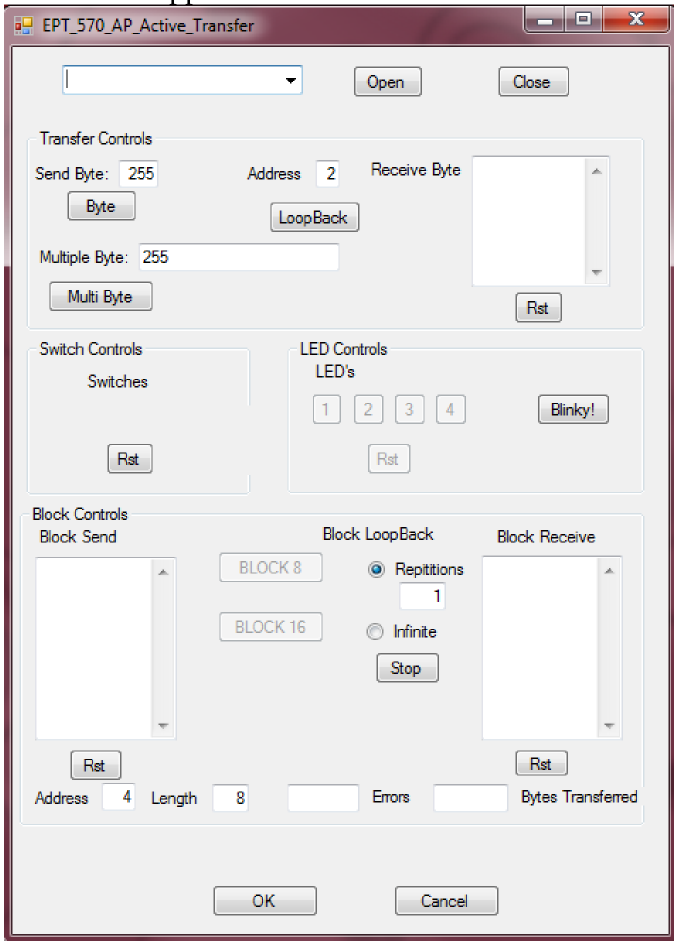
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	8/31/2009 12:40 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

EPT_Transfer_Test.exe State: Shared Size: 28.0 KB Shared with: Homegroup
Application Date modified: 3/2/2013 11:28 PM Date created: 3/2/2013 11:29 PM

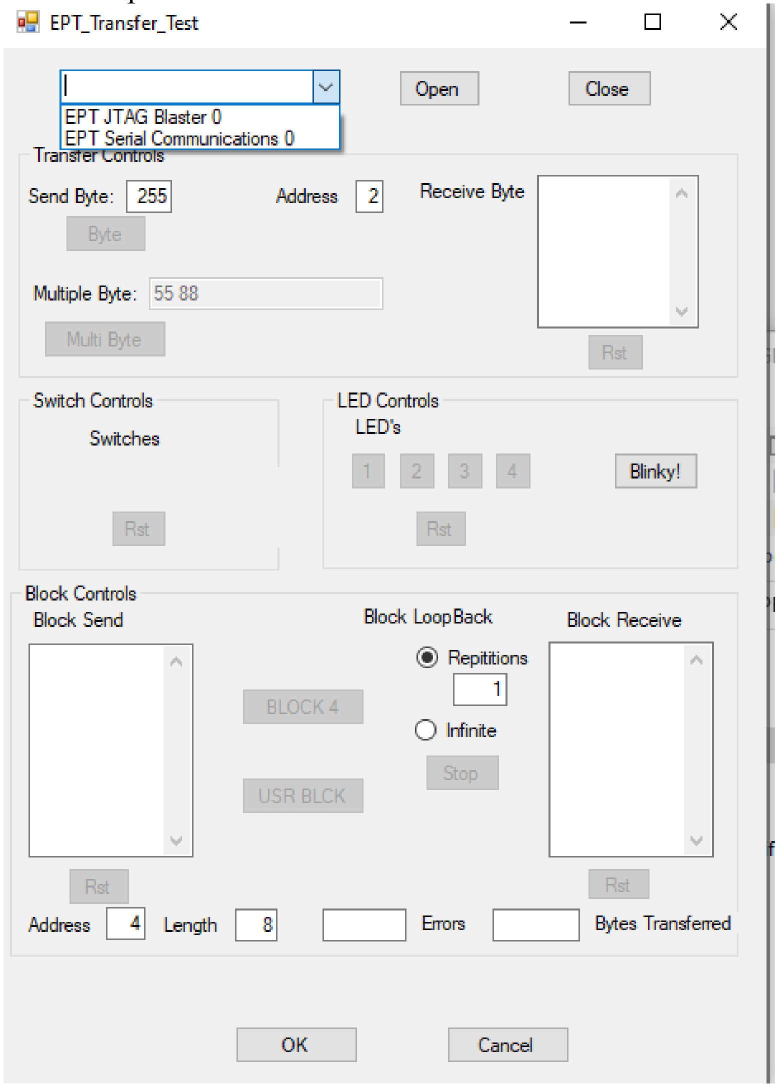
To test drive the application, connect the UnoMax to the Windows PC using Type A to Micro B USB cable. Load the driver for the board. See the section EPT Drivers for instructions on loading the EPT-5M57-AP-U3 driver.



Next, open a Windows Explorer browser. Browse to the Projects_ActiveHost_xxBit\EPT_Transfer_Test\EPT_Transfer_Test\bin\X64\Release\ folder on the UNO_USB_CPLD_PROJECT_CD. Double click on the EPT_Transfer_Text.exe. The application should load with a Windows form.



With the application loaded, select the USB-CPLD board from the dropdown combo box and click on the "Open" button.



Leave the Address set at 2 for the Transfer Controls Group. And, leave the Address set at 4 for the Block Controls Group.

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

To exercise the Single Byte Transfer EndTerm, click the "Byte" 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.

To exercise the Block Transfer EndTerm, click the “BLOCK4” or “USR BLOCK” button in the Block Controls group. A pre-selected group of numbers appear in the Block Receive textbox.

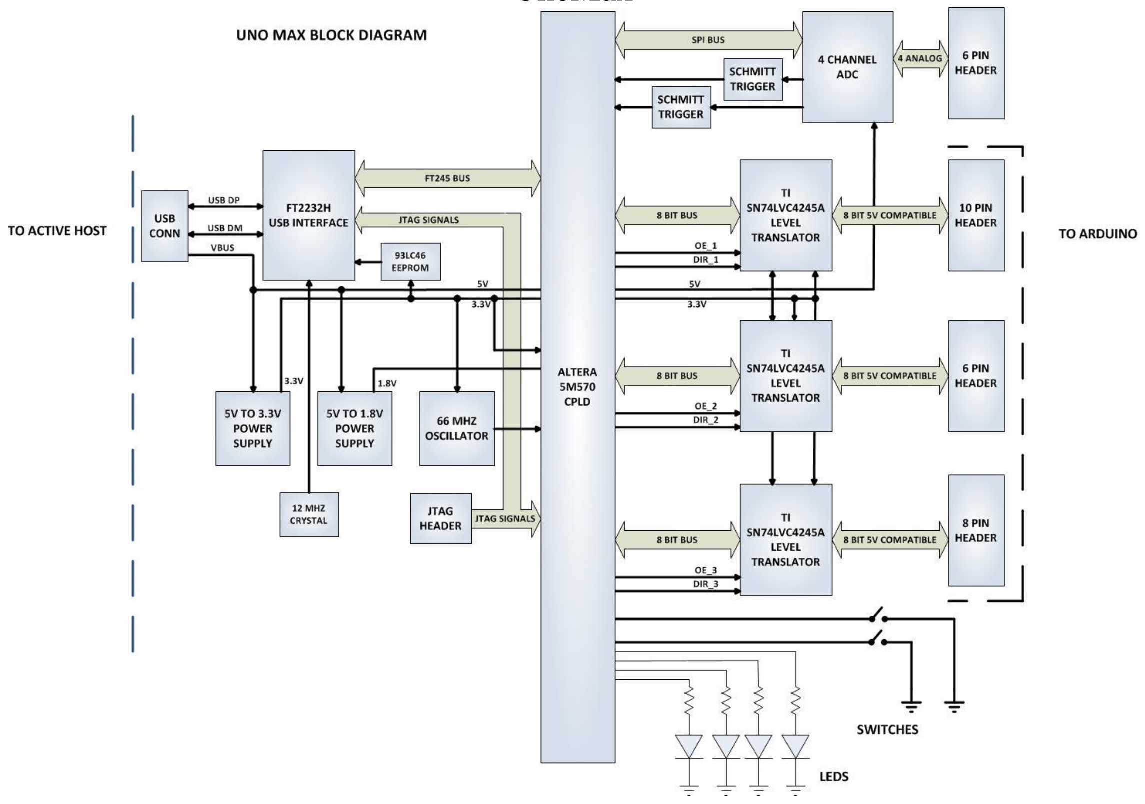
Press the PCB switches on the UnoMax to view the Switch Controls in action.

1.2 *EPT-5M57-AP-U3*

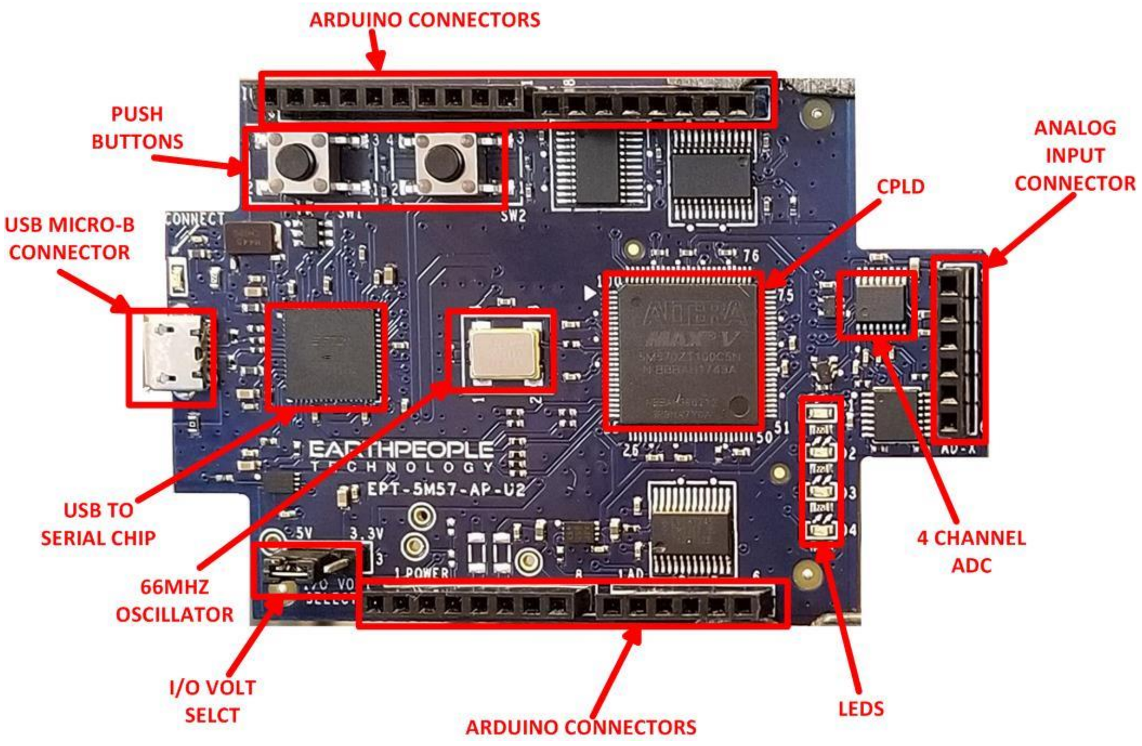
The UnoMax board (EPT-5M57-AP-U3) is equipped with an Intel/Altera 5M570 CPLD; which is programmed using the Quartus Prime software. The CPLD has 570 Logic Elements which is equivalent to 440 Macrocells. An on board 66 MHz oscillator is used by the EPT Active Transfer Library to provide data transfer rates of up to 0.1 Mega Bytes per second. A four channel ADC with 300KSamples/Second sample rate and dedicated 6 pin header. Twenty Four I/O’s from the CPLD are attached to three 8 bit transceivers to provide 3.3V/5 Volt compatible I/O’s. These 74LVC245 bidirectional voltage translator/bus transceivers are controlled by one enable and direction bit per transceiver. This means the direction of the individual bits of each transceiver cannot be selected; the direction is selected for all eight bits per transceiver. There are four green LED’s and two Push Buttons that are controllable by the user code. The hardware features are as follows.

- Intel/Altera 5M570 CPLD with 440 Macrocells
- 4 Channel ADC 300KSamples/Second
- 66 MHz oscillator for driving USB data transfers and users code
- Three bidirectional voltage translator/bus transceivers
- 24 user Input/Outputs available as three 8 bit ports
- Ports have jumper selectable 3.3V/5 Volt Input/Output
- Four Green LED’s accessible by the user
- Two PCB switches accessible by the user

UnoMax


UNO MAX BLOCK DIAGRAM
TO ACTIVE HOST
USB CONN
USB DP
USB DM
VBUS
FT2232H USB INTERFACE
FT245 BUS
JTAG SIGNALS
93LC46 EEPROM
5V
3.3V
5V TO 3.3V POWER SUPPLY
5V TO 1.8V POWER SUPPLY
1.8V
66 MHZ OSCILLATOR
12 MHZ CRYSTAL
JTAG HEADER
ALTERA 5M570 CPLD
SPI BUS
4 CHANNEL ADC
4 ANALOG
6 PIN HEADER
SCHMITT TRIGGER
SCHMITT TRIGGER
8 BIT BUS
OE_1
DIR_1
TI SN74LVC4245A LEVEL TRANSLATOR
8 BIT 5V COMPATIBLE
10 PIN HEADER
OE_2
DIR_2
6 PIN HEADER
OE_3
DIR_3
8 PIN HEADER
TO ARDUINO
SWITCHES
LEDS

UnoMax



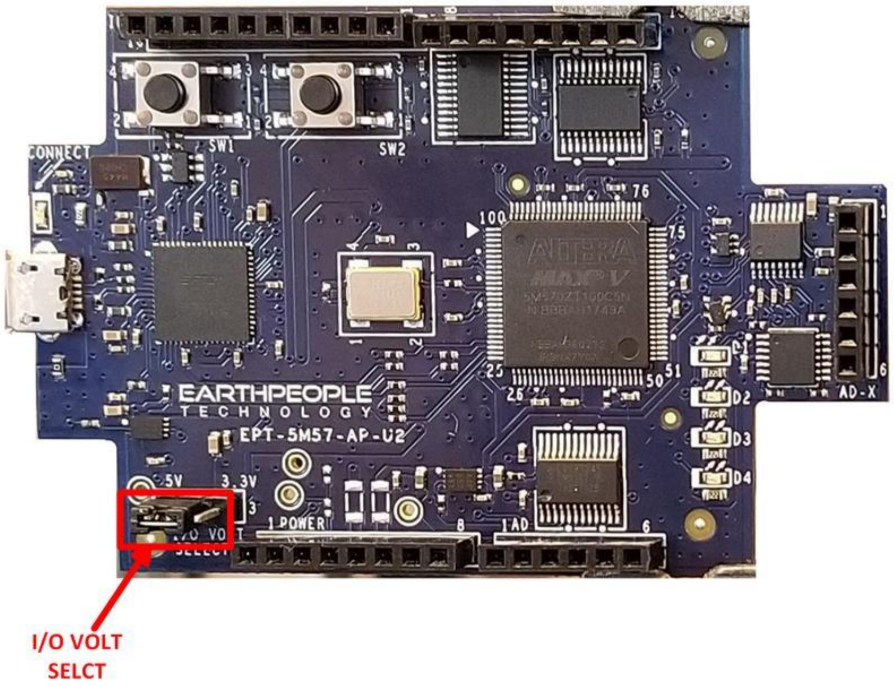
1.2.1 High Speed USB Communications

The UnoMax CPLD Development system connects an FT2232H Dual High Speed USB (480 Mbits/sec) chip to the CPLD. The CPLD uses a dedicated channel on the FT2232H for high speed transfers to the PC. Using the EPT Active Transfer Library, sustained speeds of 8 Mbytes/sec can be achieved. The transfers are bi-directional.

The FT2232H chip provides a means of data conversion from USB to serial/ parallel data and serial/parallel to USB for data being sent from the CPLD to the PC. Channel A is configured as a JTAG bus and Channel B is configured as an Serial bus. CPLD Programming commands are transmitted via the JTAG bus (channel A). Channel B has one dual port 4Kbyte FIFO for transmission from Host PC to the CPLD, it also has one dual port 4Kbyte FIFO for receiving data from the CPLD to the Host PC. The FT2232H chip provides its own 12 MHz clock and +3.3V and +1.8V power supplies. The +3.3V power supply output is used by the UnoMax for all of its +3.3V power budget.

1.2.2 Inputs and Outputs

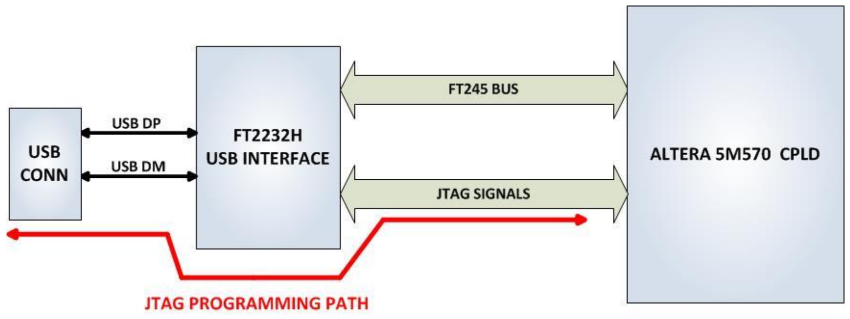
There are 24 Inputs/Outputs which are selectable between +3.3V and +5 Volt. JMP1 is used to select which voltage the 24 Inputs/Outputs are set to.



The I/O's are organized as three 8 bit directional ports. Each port must be defined as input or output. This means that all 8 bits of a port will point in the same direction, depending on the direction bit of the transceiver. The direction bit can be changed at any time, so that a port can change from input to output in minimum setup time of 6 nanoseconds. Each port also has an enable pin. This enable pin will enable or disable the bits of the port. If the port is disabled, the bits will "float".

1.2.3 JTAG

The UnoMax uses the second channel of the FT2232H chip as a dedicated CPLD programming port. The CPLD must be programmed via JTAG signals and the FT2232H has built in JTAG signals. The CPLD can be programmed directly from Quartus Prime Lite by using the "jtag_hw_mbftdi_blaster.dll". Just click on the Programmer button and select the EPT-Blaster.

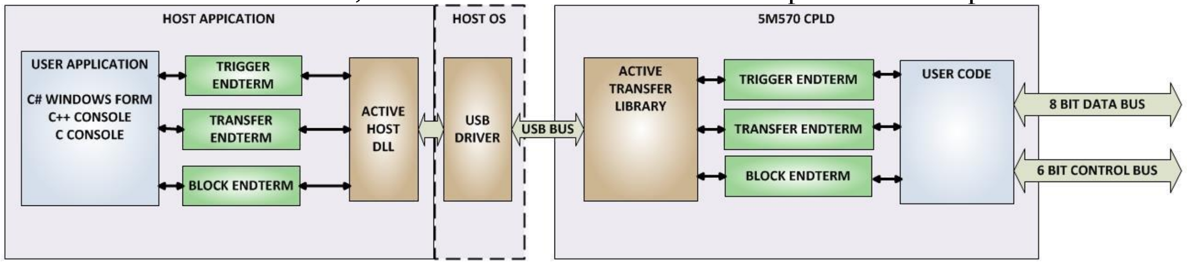


1.2.4 USB To Serial

The FT2232H chip has dual channels high speed (480 Mb/s) USB to FIFO (first in-first out) integrated circuit to interface between the Host PC and the CPLD. The FT2232H chip provides a means of data conversion from USB to serial/ parallel data and serial/parallel to USB for data being sent from the CPLD to the PC. Channel A is configured as a JTAG bus and Channel B is configured as an Serial bus. CPLD Programming commands are transmitted via the JTAG bus (channel A). Channel B has one dual port 4Kbyte FIFO for transmission from Host PC to the CPLD, it also has one dual port 4Kbyte FIFO for receiving data from the CPLD to the Host PC. The chip uses the +5Vbus from the Host USB for self power. The FT2232H has its own 12 MHz clock.

1.3 Active Host EndTerms

The Active Host SDK is provided as a dll which easily interfaces to application software written in C#, C++ or C. It runs on the PC and provides transparent connection



from PC application code through the USB driver to the user CPLD code. The user code connects to "Endterms" in the Active Host dll. 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 CPLD.

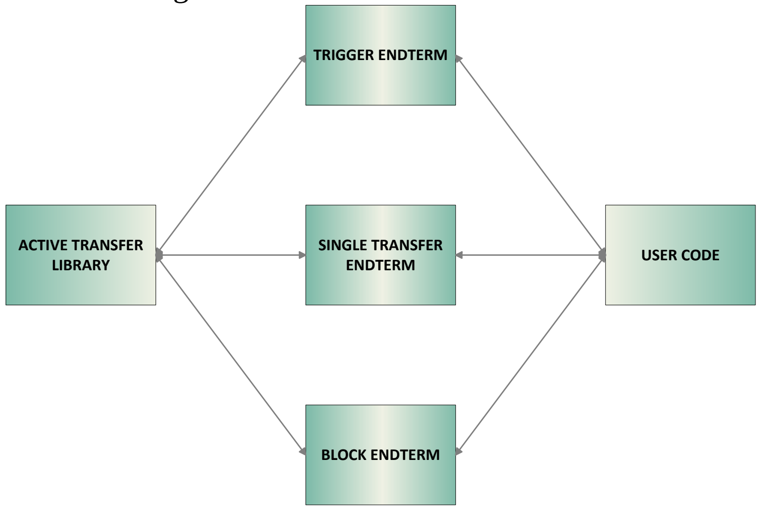
Receiving data from the CPLD is made simple by Active Host. Active Host transfers data from the CPLD as soon as it is available. It stores the transferred data into circular buffer. When the transfer is complete, Active Host invokes a callback function which is registered in the users application. This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

1.4 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.

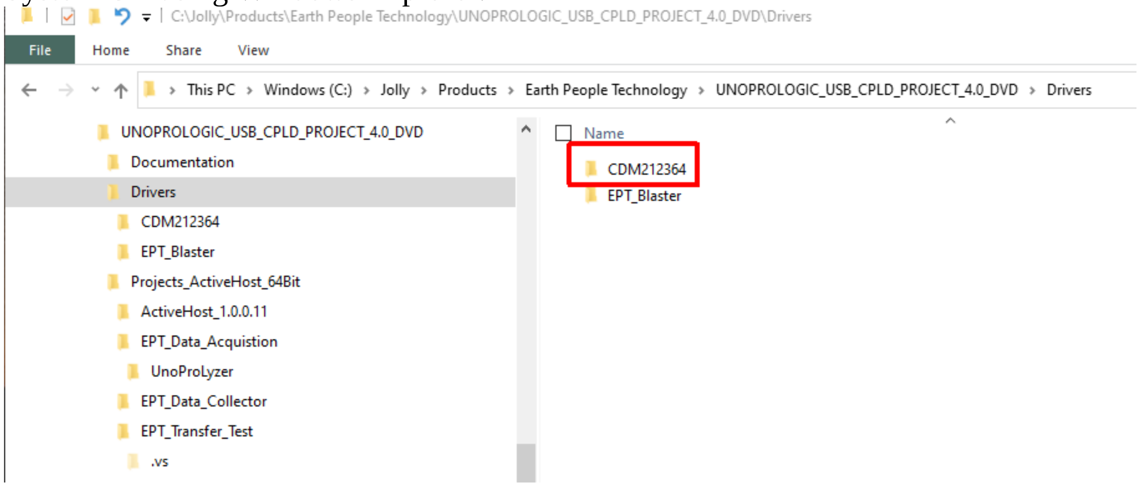
2 EPT Drivers

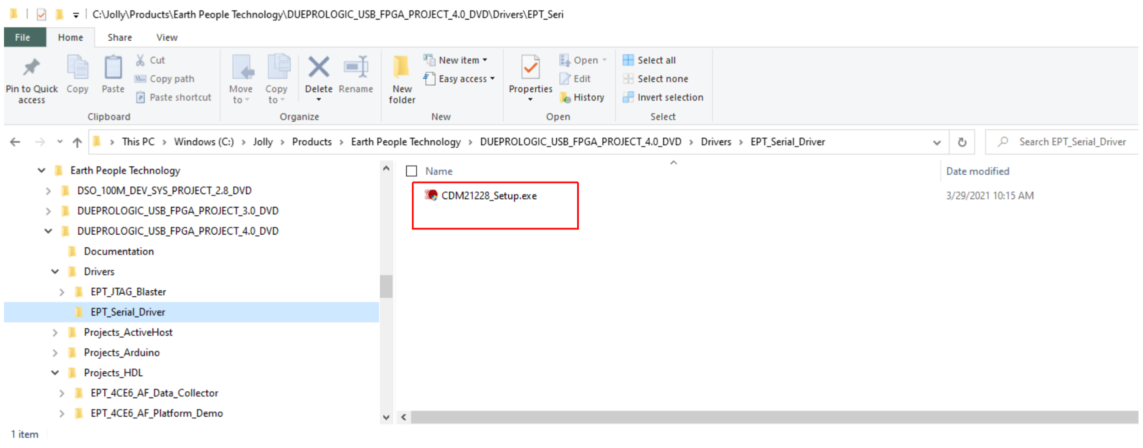
The UnoProLogic Development system requires drivers for any interaction between PC and the board. The communication between the two consists of programming the CPLD and data transfer. In both cases, the USB Driver is required. This will allow Windows to recognize the USB Chip and setup a pathway for Windows to communicate with the USB hardware.

2.1 USB Driver

The UnoProLogic uses an FTDI FT2232H USB to Serial chip. This chip provides the USB interface to the PC and the serial/FIFO interface to the CPLD. The FT2232H requires the use of the EPT USB driver. To install the driver onto your PC, use the CDM212xxx Folder. The installation of the FTDI 2.12.28 driver is easily accomplished by double clicking the CDM21228_Setup.exe.

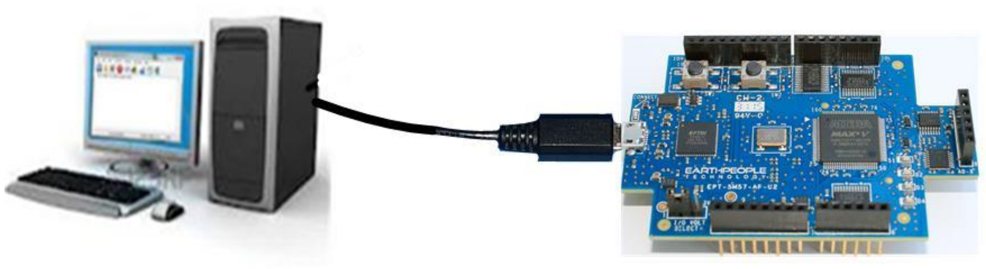
Locate the CDM212xxx folder in the Drivers folder of the UnoProLogic Development System CD using Windows Explorer.


C:\Jolly\Products\Earth People Technology\UNOPROLOGIC_USB_CPLD_PROJECT_4.0_DVD\Drivers
File
Home
Share
View
This PC › Windows (C:) › Jolly › Products › Earth People Technology › UNOPROLOGIC_USB_CPLD_PROJECT_4.0_DVD › Drivers
UNOPROLOGIC_USB_CPLD_PROJECT_4.0_DVD
Documentation
Drivers
CDM212364
EPT_Blaster
Projects_ActiveHost_64Bit
ActiveHost_1.0.0.11
EPT_Data_Acquistion
UnoProLyzer
EPT_Data_Collector
EPT_Transfer_Test
.vs
Name
CDM212364
EPT_Blaster

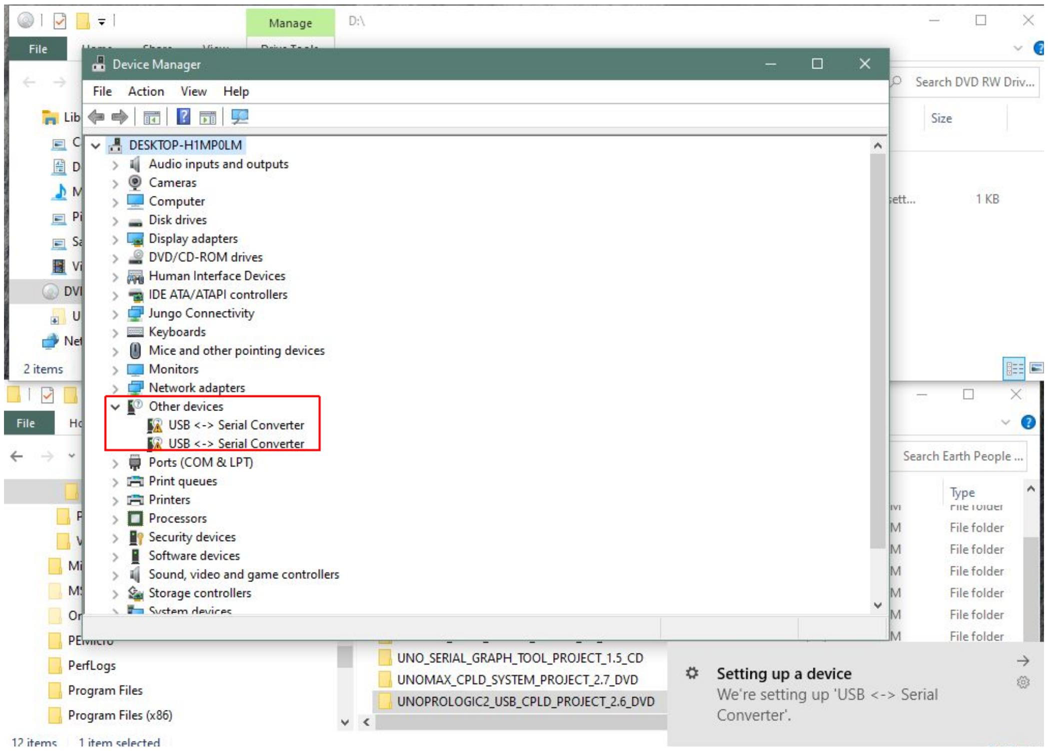


Double click on the *.exe file and select the default settings when the software tool queries the user.

Plug in the UnoProLogic device into an available USB port.

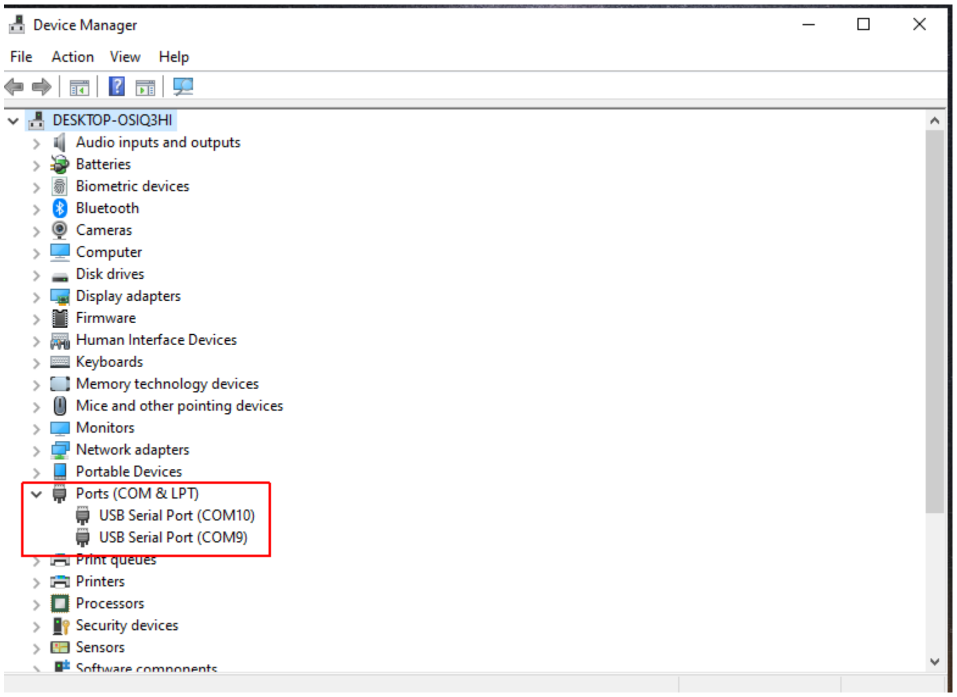


Windows will attempt to locate a driver for the USB device. When it does not find one, it will report a error, “Device driver software was not successfully installed”. Ignore this error.
If Windows cannot load a driver for the DPL, a notification window will inform the user that the driver load has failed for the device.



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

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.

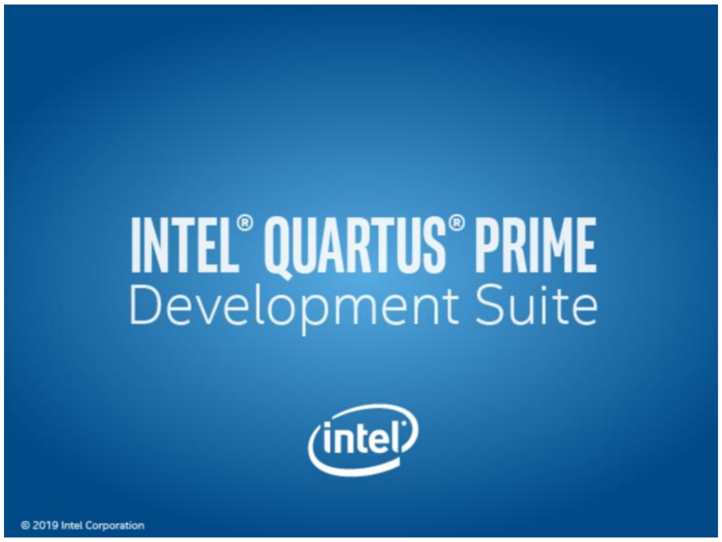
2.2 JTAG DLL Insert to Quartus Prime Lite

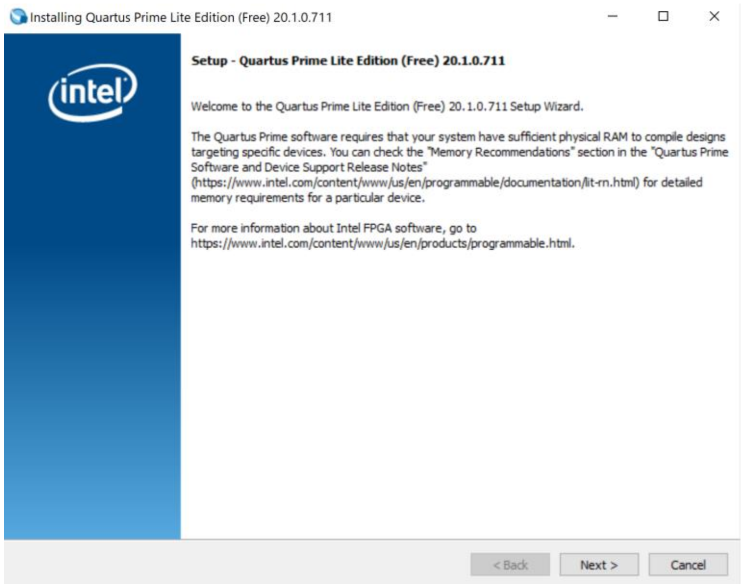
The JTAG DLL Insert to Quartus Prime Lite allows the Programmer Tool under Quartus to recognize the UnoProLogic. The UnoProLogic can then be selected and perform programming of the CPLD. The file, jtag_hw_mbftdi_blaster.dll must be placed into the folder that hosts the jtag_server for Quartus.

2.2.1 Installing Quartus

You can download the Quartus Prime Lite by following the directions in the Section Downloading Quartus.

If you don't need to download Quartus, double click on the QuartusLiteSetup-xxx.xxx.xxx-windows .exe (the xxx is the build number of the file, it is subject to change). The Quartus Prime Web Edition will start the installation process.





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.

When the install shield window pops up click “Yes” or if needed, enter the administrator password for the users PC. Click “Ok”

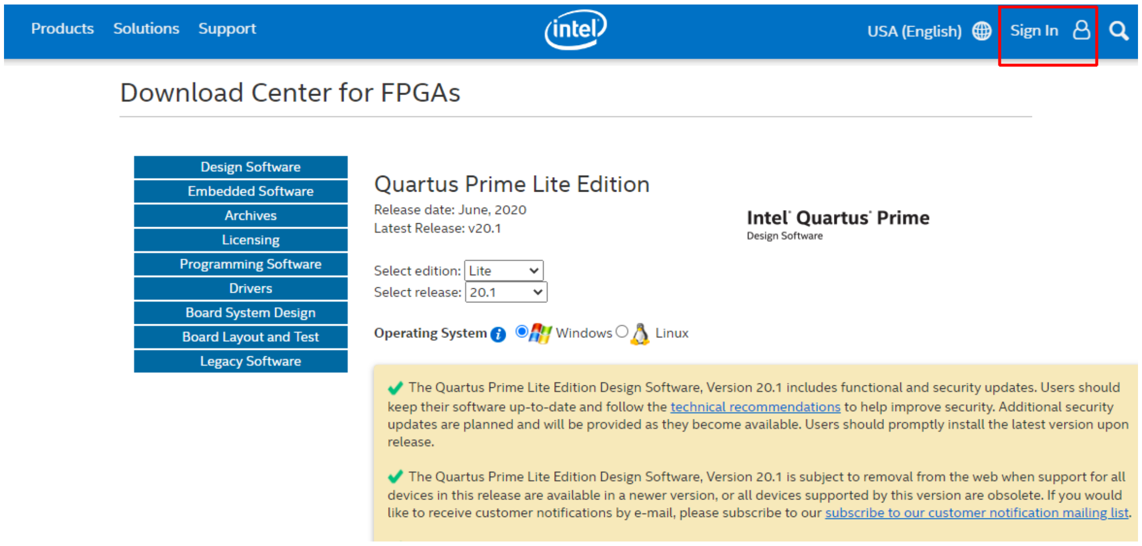
Next, skip the “Download Quartus” section. Go down to the “Quartus Installer” section to complete the Quartus installation.

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

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.



Quartus Prime Lite Edition

Release date: June, 2020
Latest Release: v20.1

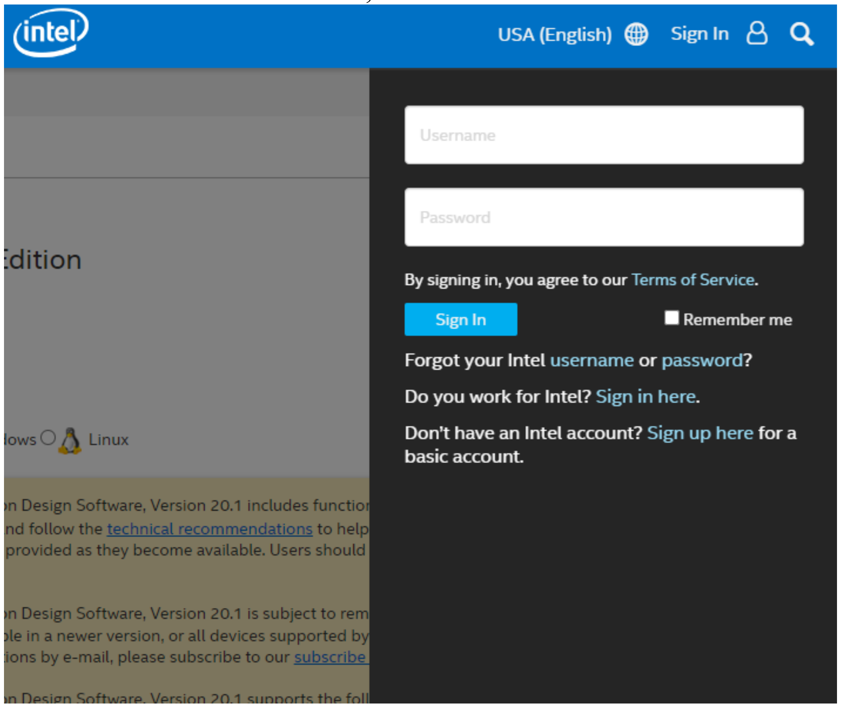
Select edition: Lite
Select release: 20.1

Operating System [✓ Windows] Linux

✓ The Quartus Prime Lite Edition Design Software, Version 20.1 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.

✓ The Quartus Prime Lite Edition Design Software, Version 20.1 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.

The next page will require you to sign into your "myAltera" account. If you do not have one, follow the directions under the box, "Don't have an account?"



By signing in, you agree to our Terms of Service.

Forgot your Intel username or password?

Do you work for Intel? Sign in here.

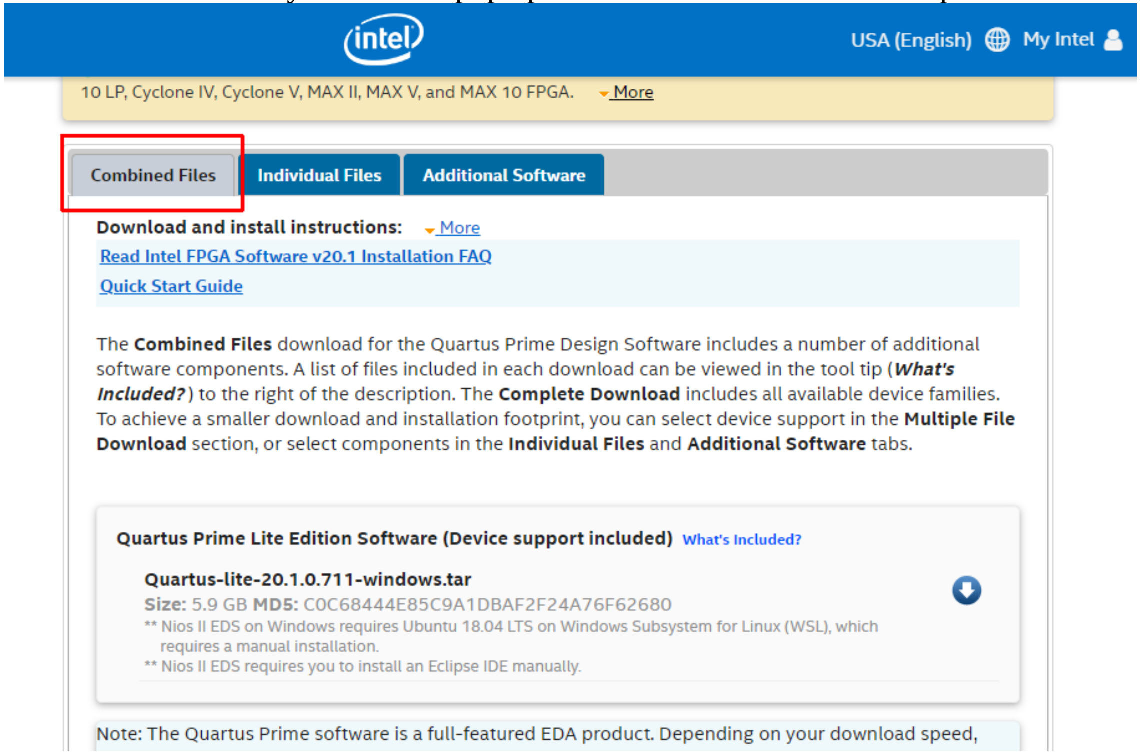
Don't have an Intel account? Sign up here for a basic account.

n Design Software, Version 20.1 includes functio
nd follow the technical recommendations to help
provided as they become available. Users should

n Design Software, Version 20.1 is subject to rem
le in a newer version, or all devices supported by
ions by e-mail, please subscribe to our subscribe

n Design Software, Version 20.1 supports the foll

Once you have created your myAltera account, enter the User Name and Password. The next window will ask you to allow pop ups so that the file download can proceed.


USA (English) My Intel

10 LP, Cyclone IV, Cyclone V, MAX II, MAX V, and MAX 10 FPGA. More

Combined Files | Individual Files | Additional Software

Download and install instructions: More

Read Intel FPGA Software v20.1 Installation FAQ

Quick Start Guide

The **Combined Files** download for the Quartus Prime Design Software includes a number of additional software components. A list of files included in each download can be viewed in the tool tip (***What's Included?***) to the right of the description. The **Complete Download** includes all available device families. To achieve a smaller download and installation footprint, you can select device support in the **Multiple File Download** section, or select components in the **Individual Files** and **Additional Software** tabs.

Quartus Prime Lite Edition Software (Device support included) What's Included?

Quartus-lite-20.1.0.711-windows.tar

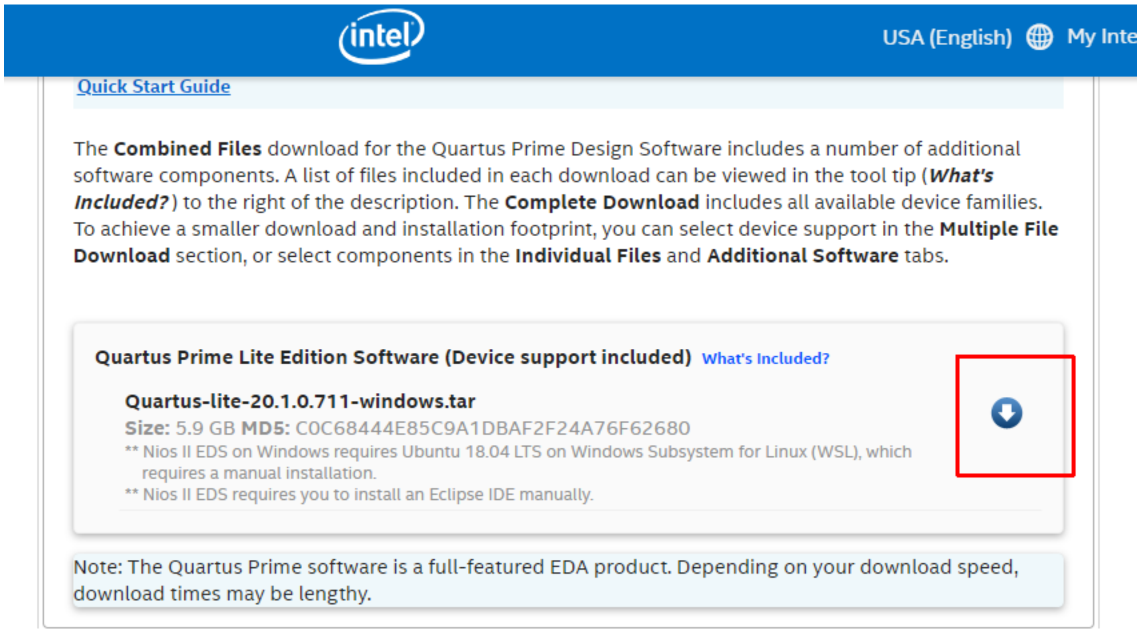
Size: 5.9 GB **MD5:** C0C68444E85C9A1DBAF2F24A76F62680

** 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.

Note: The Quartus Prime software is a full-featured EDA product. Depending on your download speed,

Click on the download icon.


USA (English) My Inte

Quick Start Guide

The **Combined Files** download for the Quartus Prime Design Software includes a number of additional software components. A list of files included in each download can be viewed in the tool tip (***What's Included?***) to the right of the description. The **Complete Download** includes all available device families. To achieve a smaller download and installation footprint, you can select device support in the **Multiple File Download** section, or select components in the **Individual Files** and **Additional Software** tabs.

Quartus Prime Lite Edition Software (Device support included) What's Included?

Quartus-lite-20.1.0.711-windows.tar

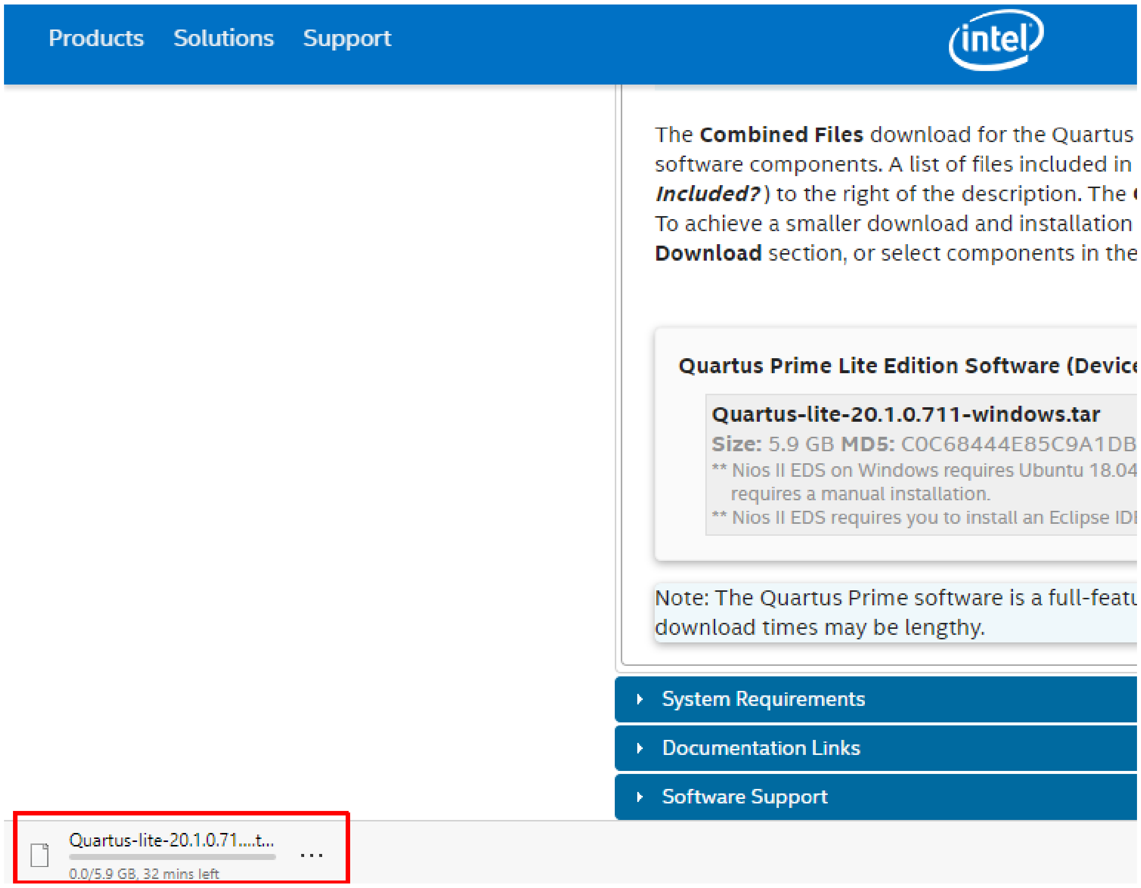
Size: 5.9 GB **MD5:** C0C68444E85C9A1DBAF2F24A76F62680

** 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.

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

This will start the download.



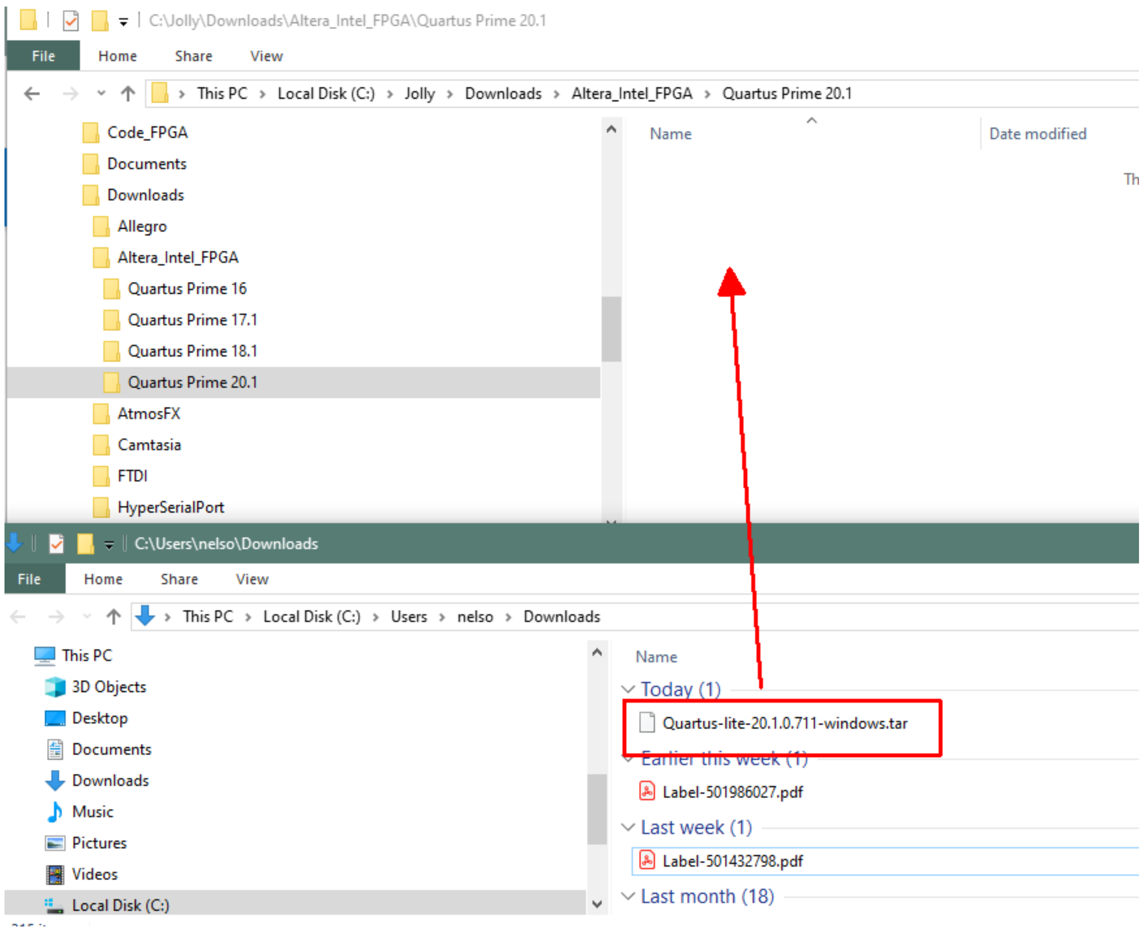
The **Combined Files** download for the Quartus
software components. A list of files included in
Included?) to the right of the description. The
To achieve a smaller download and installation
Download section, or select components in the

Quartus Prime Lite Edition Software (Devic

Quartus-lite-20.1.0.711-windows.tar
Size: 5.9 GB **MD5:** C0C68444E85C9A1DB
** Nios II EDS on Windows requires Ubuntu 18.04
requires a manual installation.
** Nios II EDS requires you to install an Eclipse ID

Note: The Quartus Prime software is a full-featu
download times may be lengthy.

The file is 5.9 GB, 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.


C:\Jolly\Downloads\Altera_Intel_FPGA\Quartus Prime 20.1

This PC > Local Disk (C:) > Jolly > Downloads > Altera_Intel_FPGA > Quartus Prime 20.1

C:\Users\nelso\Downloads

This PC > Local Disk (C:) > Users > nelso > Downloads

Today (1)

Quartus-lite-20.1.0.711-windows.tar

Earlier this week (1)

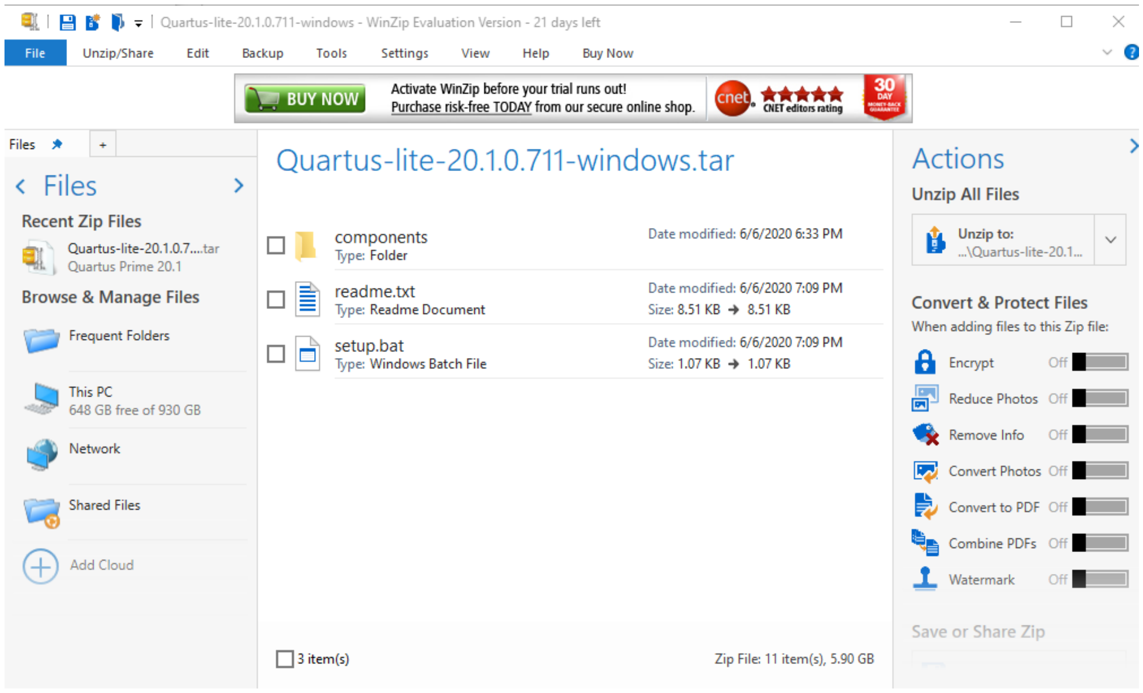
Label-501986027.pdf

Last week (1)

Label-501432798.pdf

Last month (18)

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

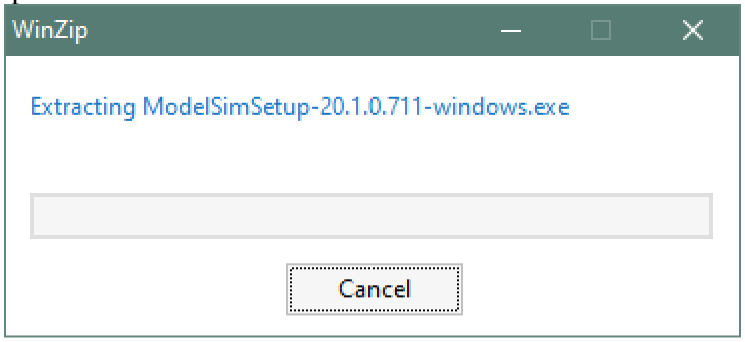

Quartus-lite-20.1.0.711-windows - WinZip Evaluation Version - 21 days left

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

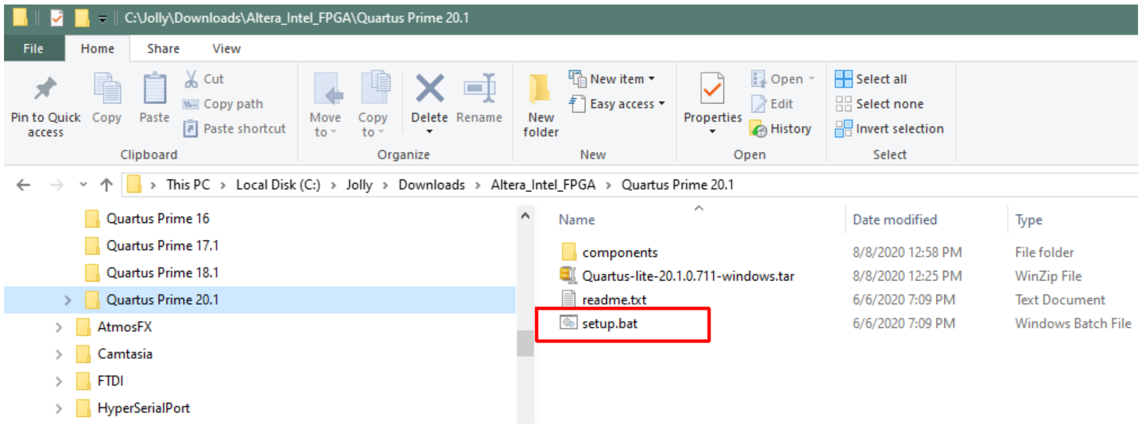
The tool will unpack all files.


WinZip

Extracting ModelSimSetup-20.1.0.711-windows.exe

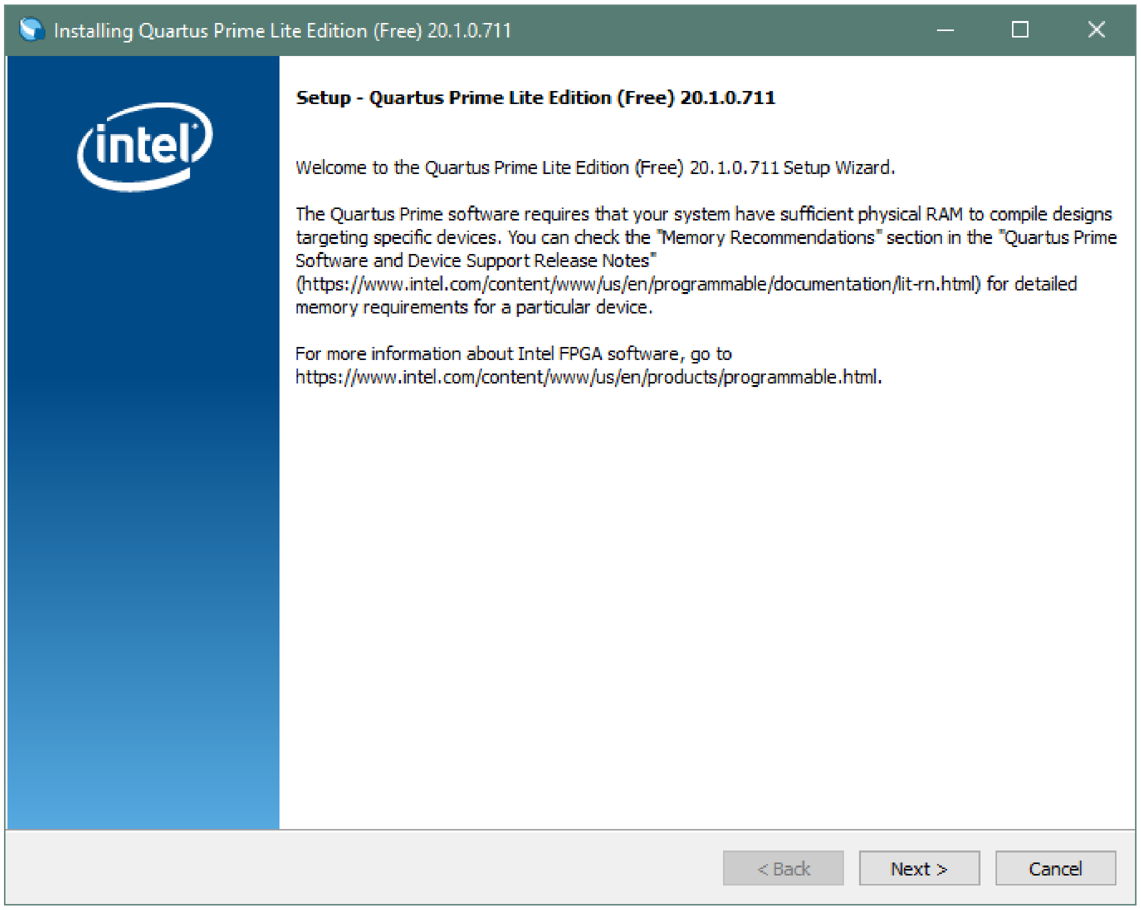
2.2.3 Quartus Installer

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.



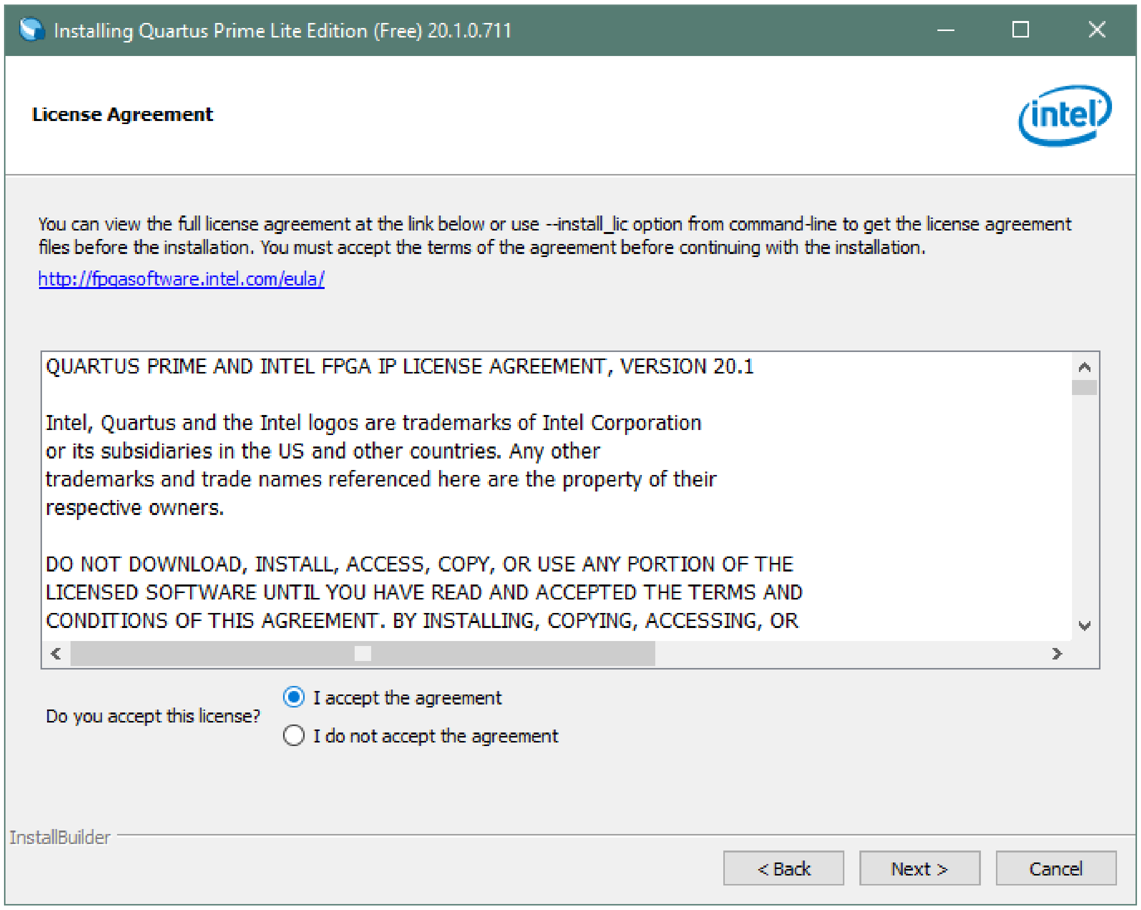
Setup - Quartus Prime Lite Edition (Free) 20.1.0.711

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”.



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

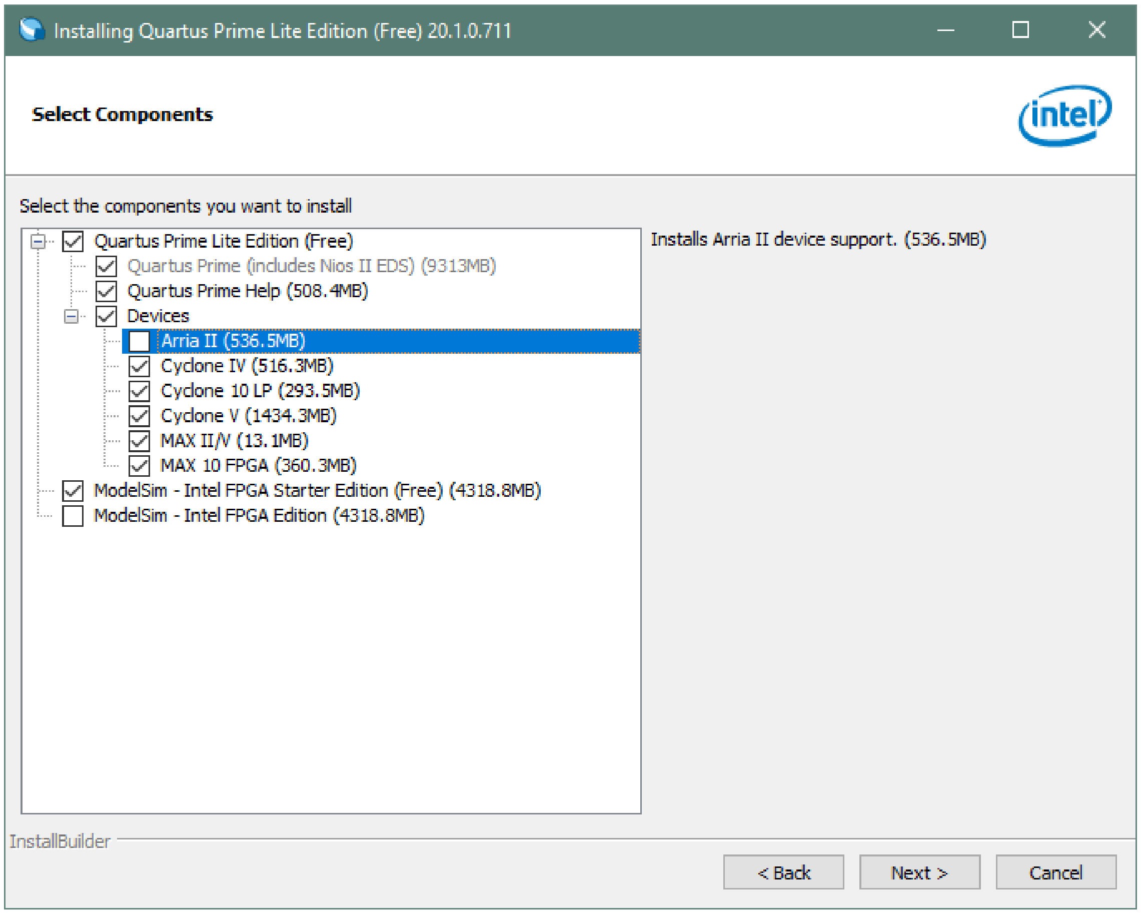
Intel, Quartus and the Intel logos are trademarks of Intel Corporation
or its subsidiaries in the US and other countries. Any other
trademarks and trade names referenced here are the property of their
respective owners.

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

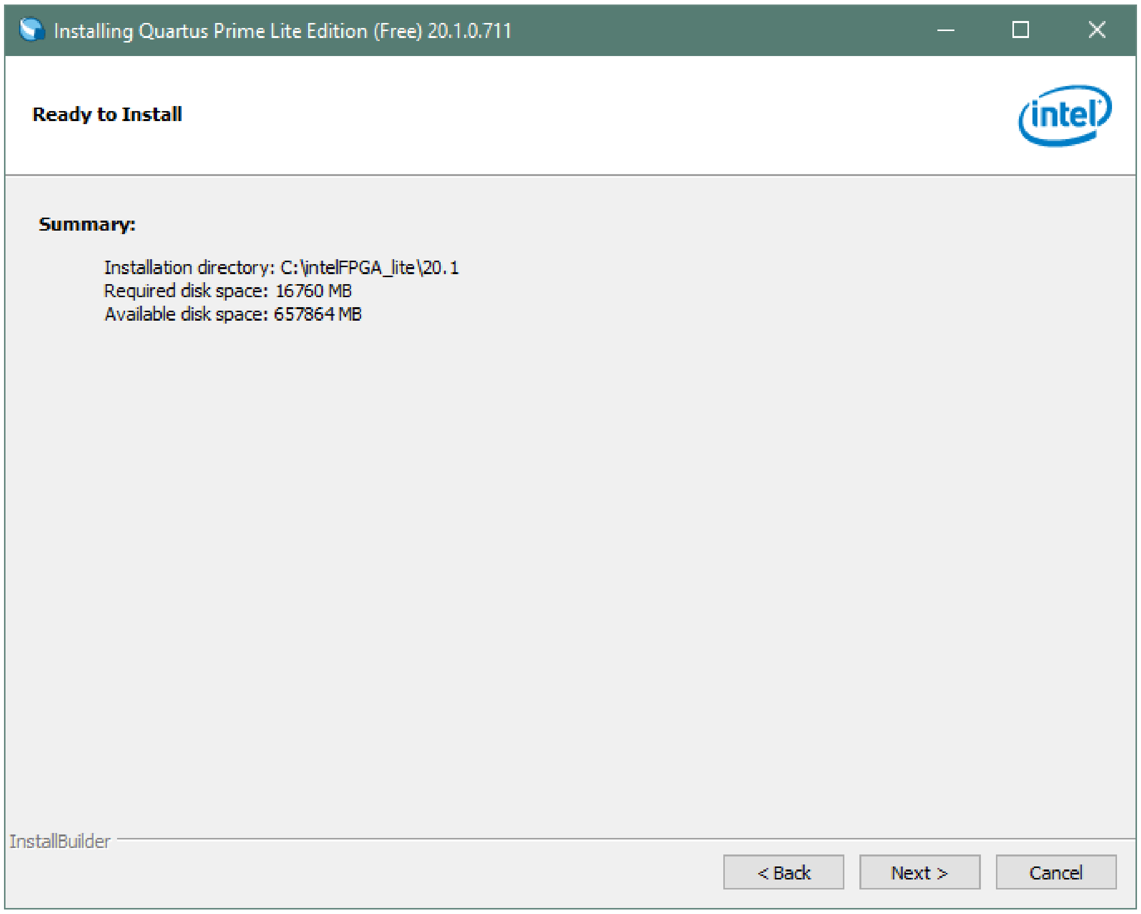
Select Components

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

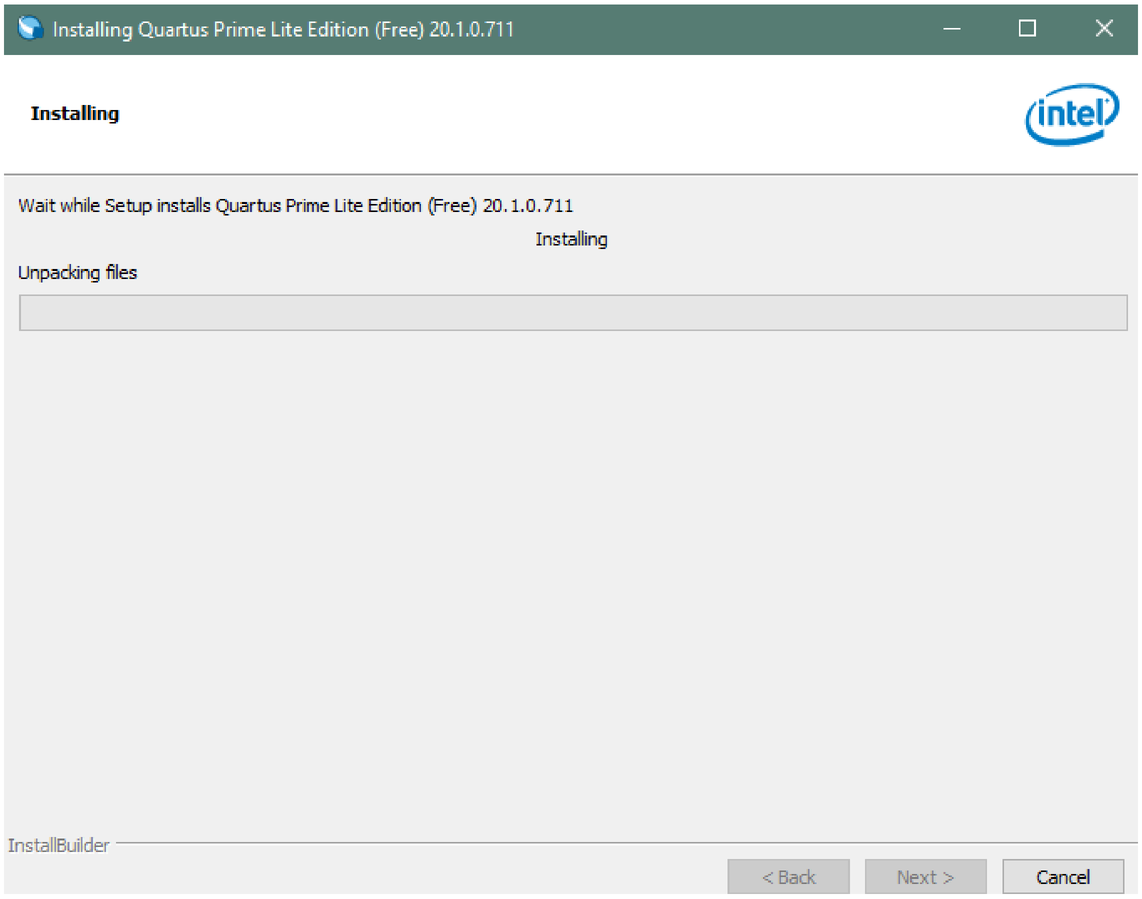

Installing Quartus Prime Lite Edition (Free) 20.1.0.711

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

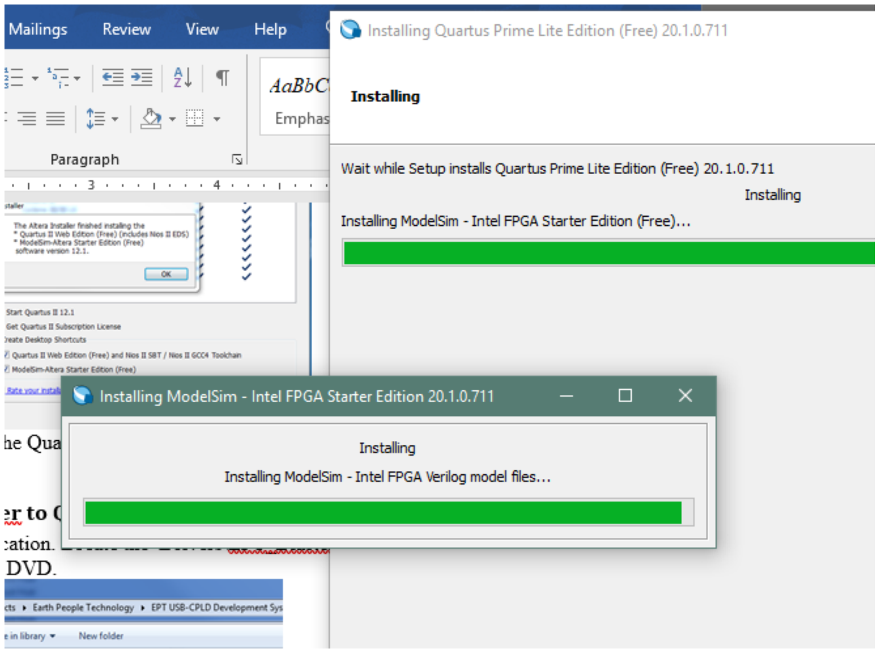
Installing

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

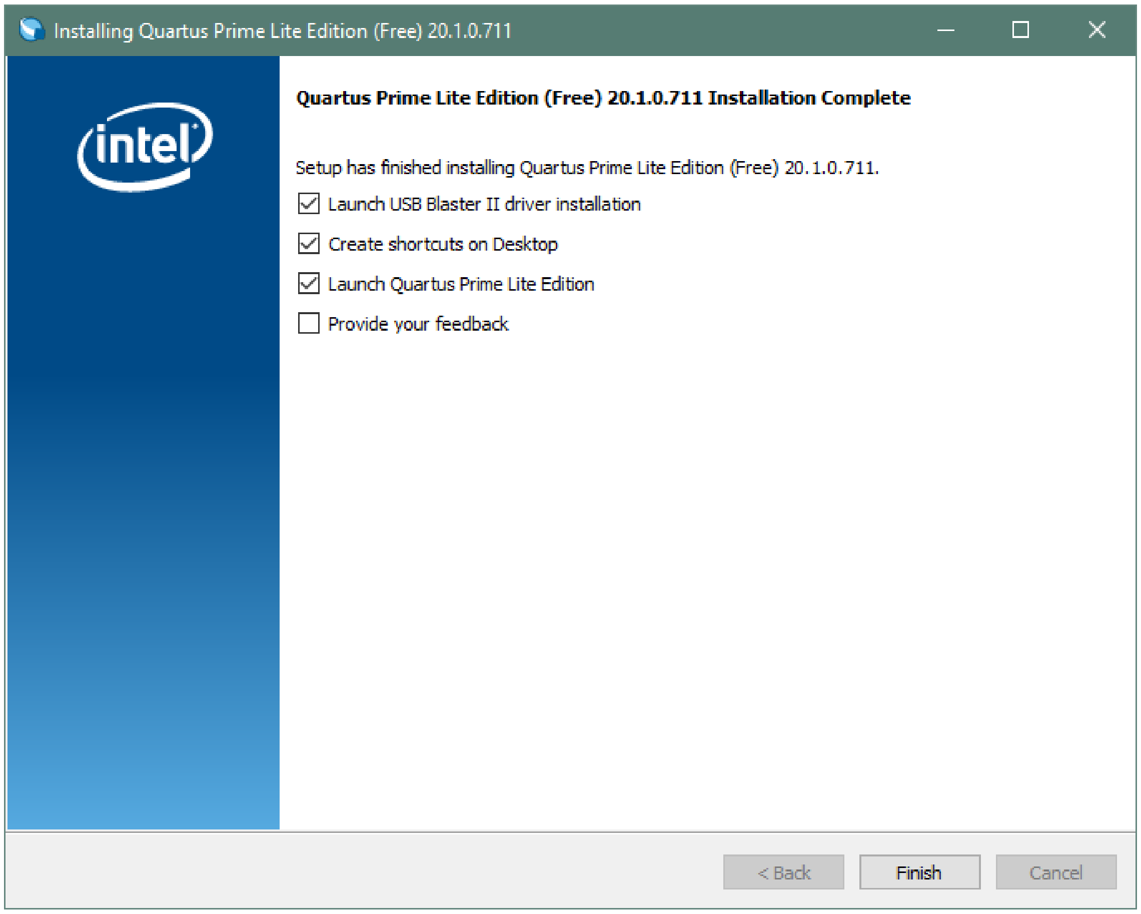
Installing

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...

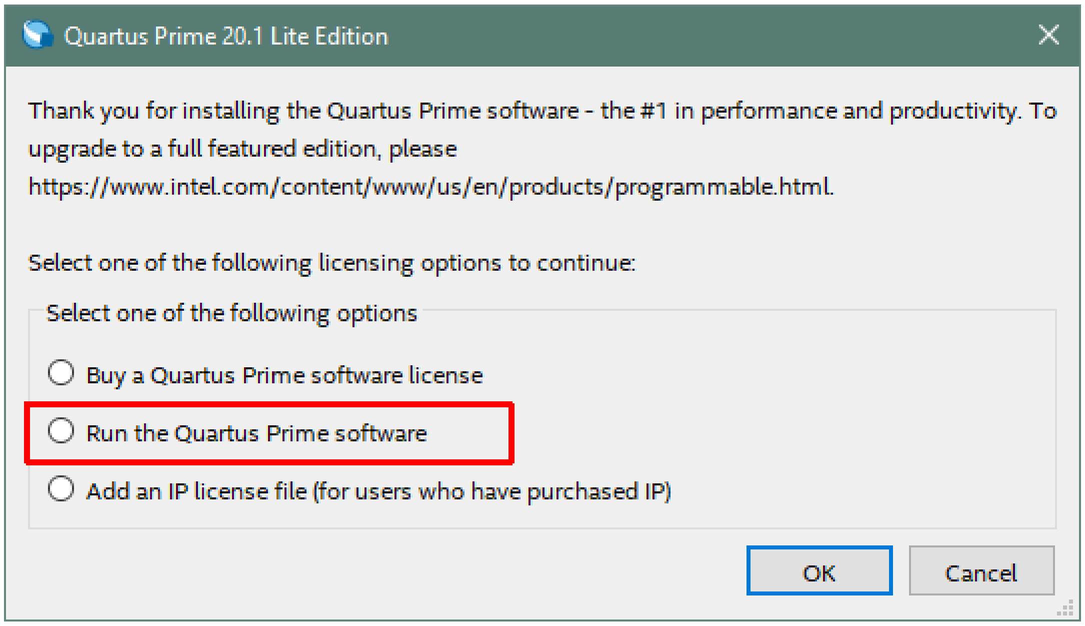

Installing Quartus Prime Lite Edition (Free) 20.1.0.711

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.

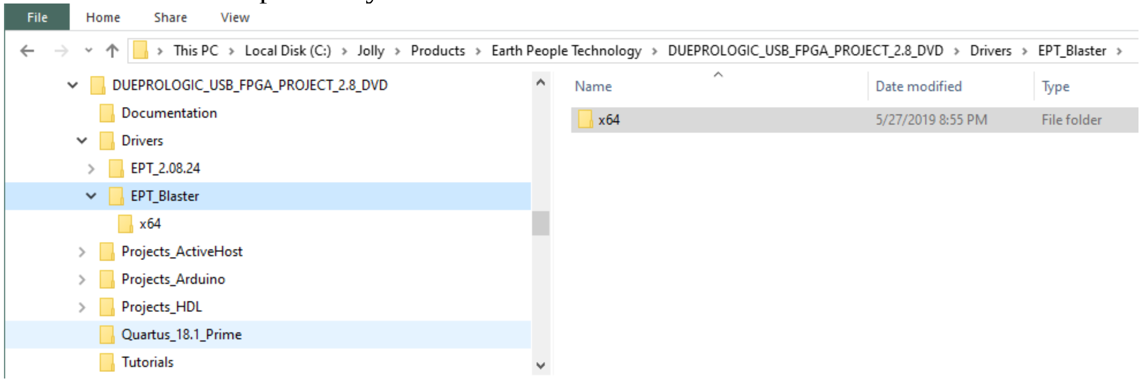
Select one of the following licensing options to continue:

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.

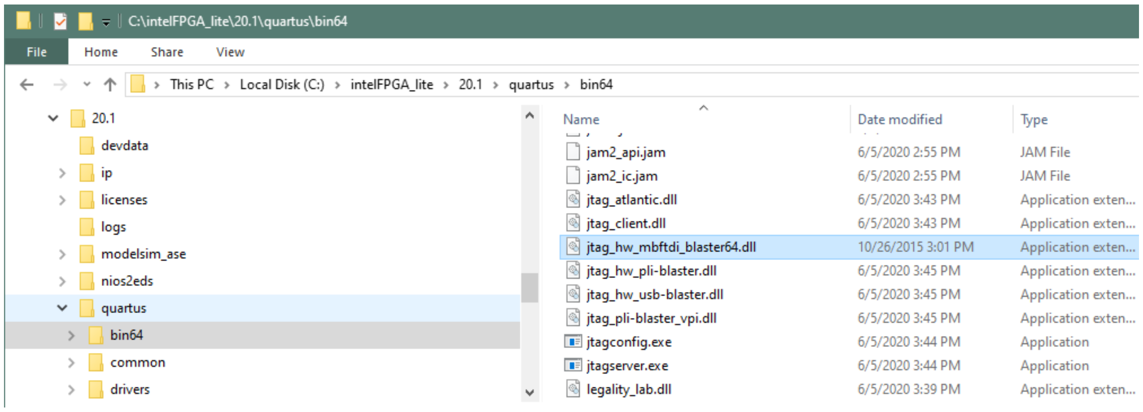


This PC › Local Disk (C:) › Jolly › Products › Earth People Technology › DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD › Drivers › EPT_Blaster ›

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...

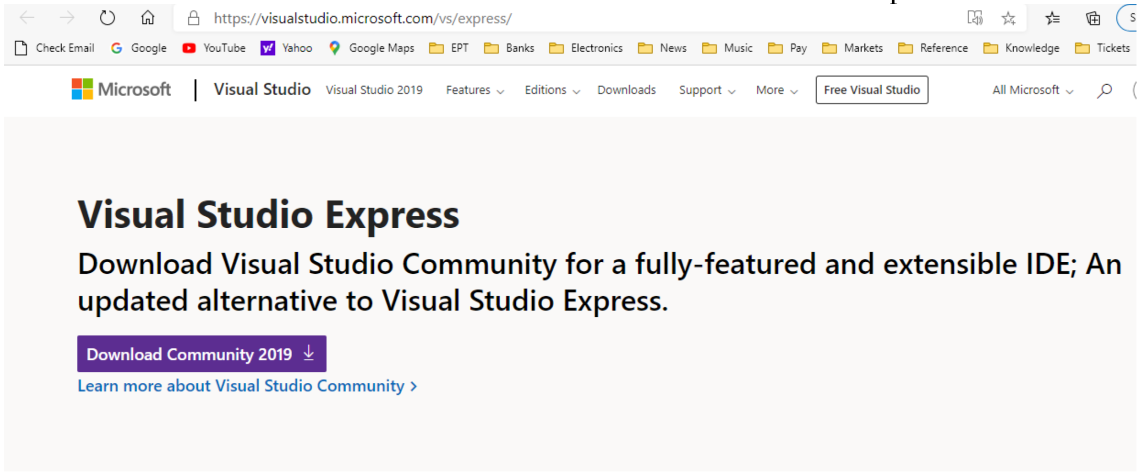
The DLL is installed and the JTAG server should recognize it. Go to the section "Programming the FPGA" of this manual for testing of the programming. If the driver is not found in the Programmer Tool->Hardware Setup box, see the JTAG DLL Insert to Quartus Prime Troubleshooting Guide.

2.3 Active Host Application DLL

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

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

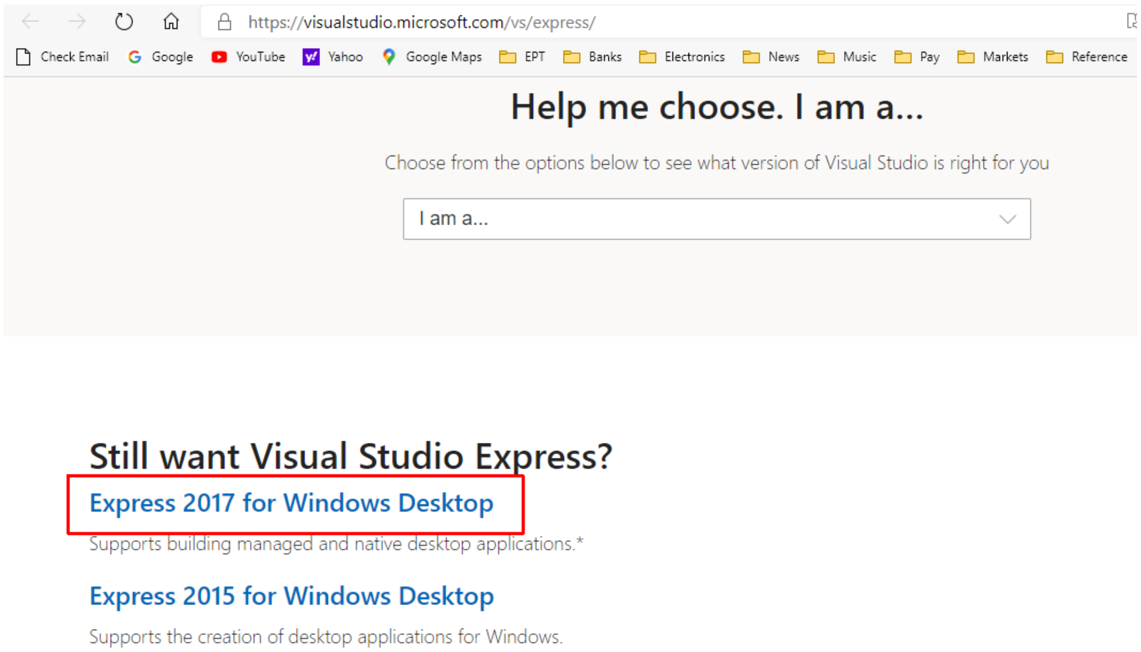
Go to the website and click on the "+" icon next to the Visual C# Express.


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



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

Check Email Google YouTube Yahoo Google Maps EPT Banks Electronics News Music Pay Markets Reference

Help me choose. I am a...

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

I am a...

Still want Visual Studio Express?

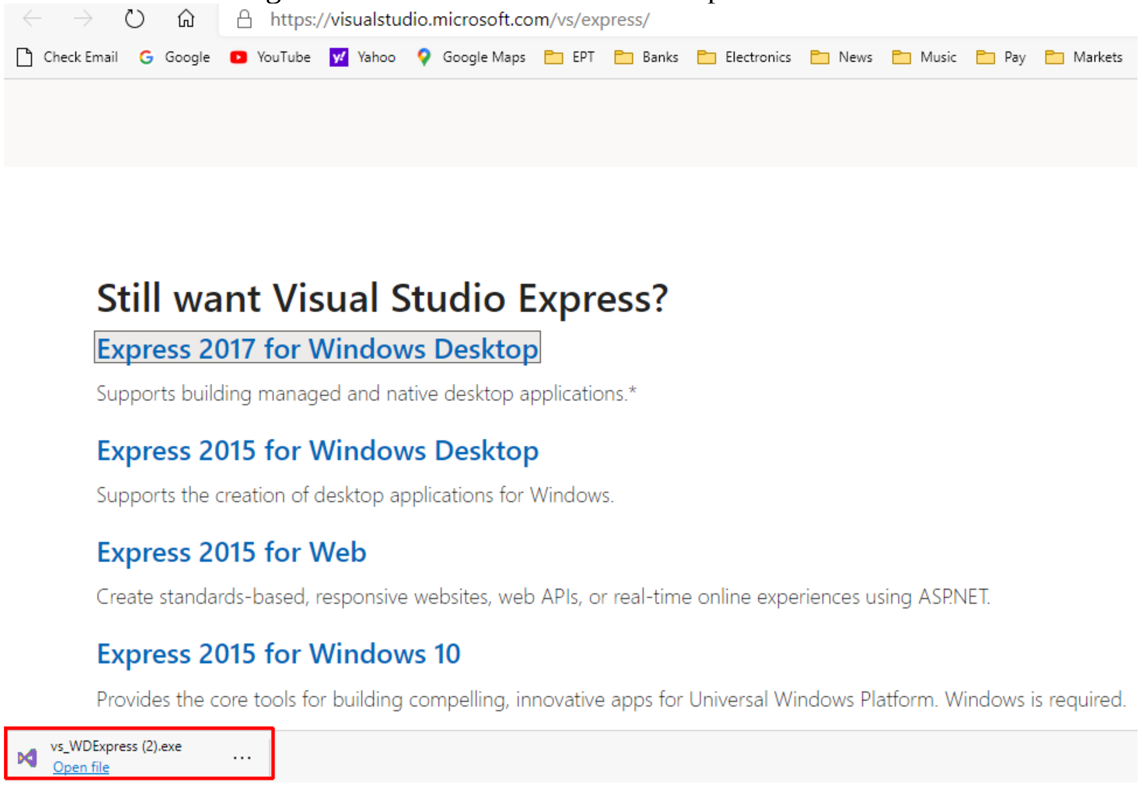
Express 2017 for Windows Desktop

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/

Check Email Google YouTube Yahoo Google Maps EPT Banks Electronics News Music Pay Markets

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.

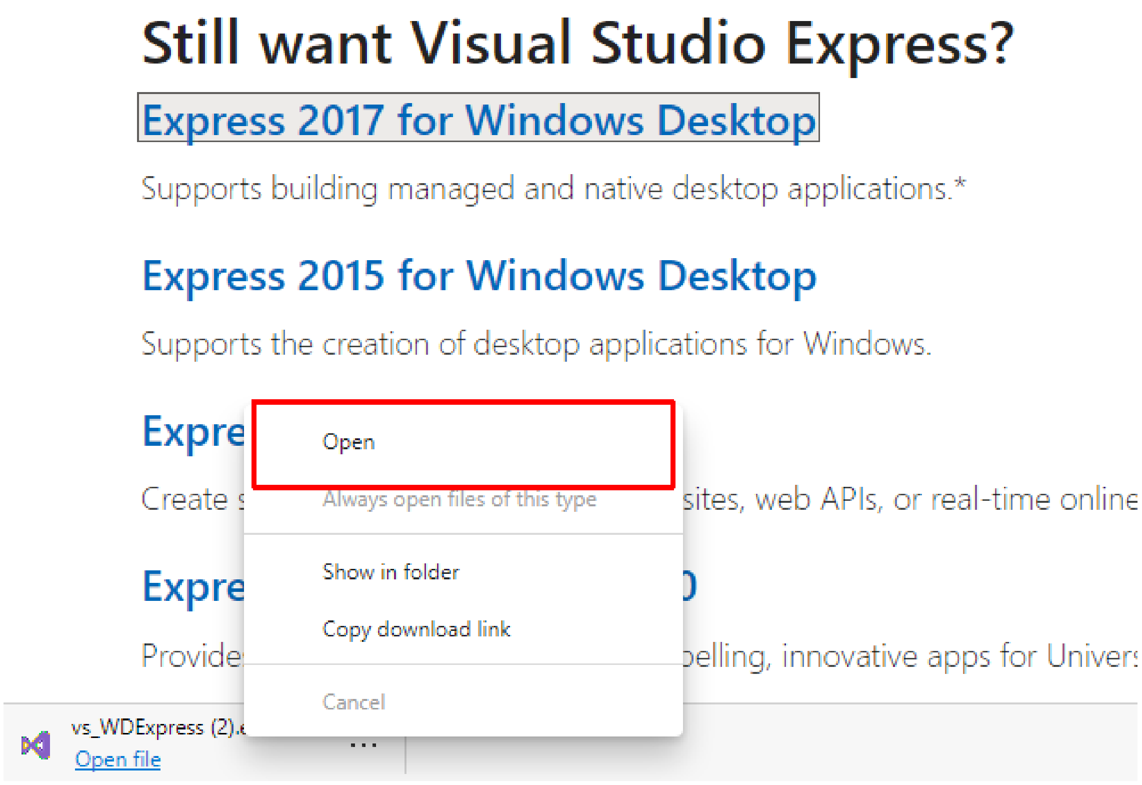
Express 2015 for Windows 10

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

vs_WDExpress (2).exe

Open file

Right click on the WDExpress.exe.


Still want Visual Studio Express?

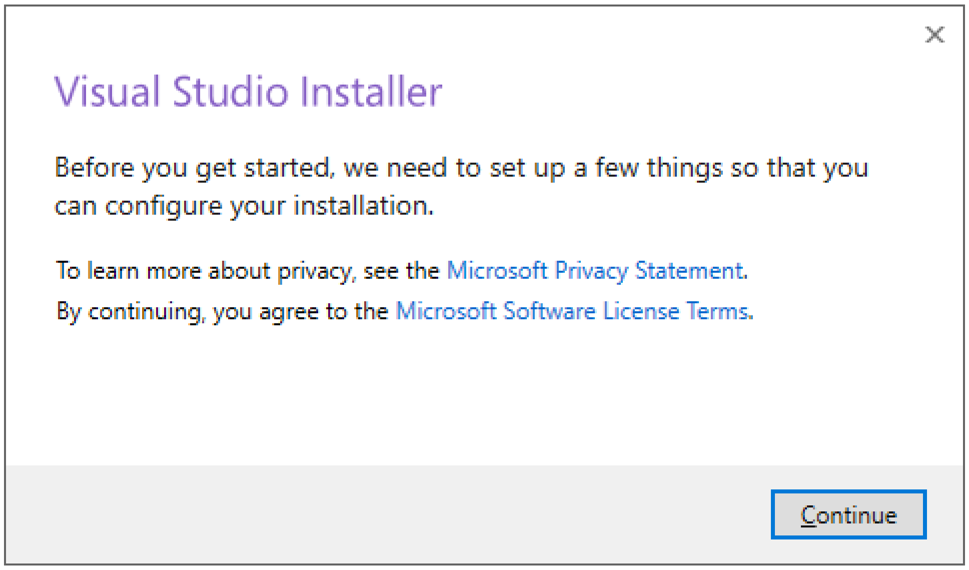
Express 2017 for Windows Desktop

Supports building managed and native desktop applications.*

Express 2015 for Windows Desktop

Supports the creation of desktop applications for Windows.

Click the “Continue” button.

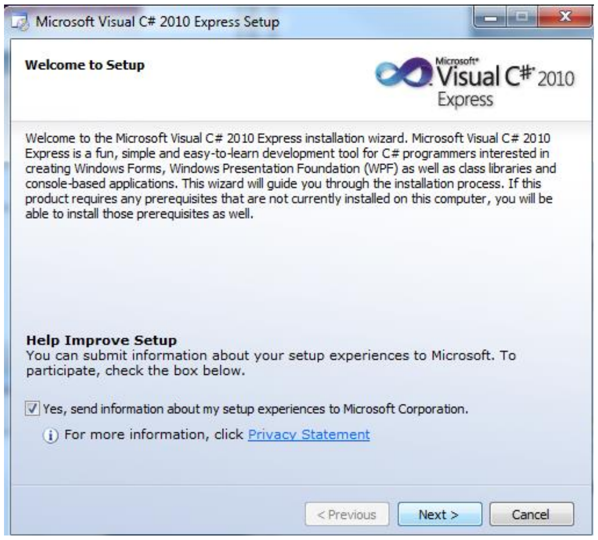

Visual Studio Installer

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

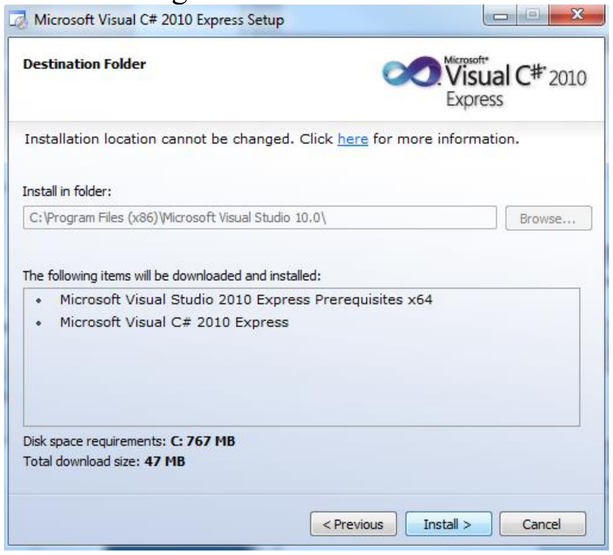
Welcome to the Microsoft Visual C# 2010 Express installation wizard. Microsoft Visual C# 2010 Express is a fun, simple and easy-to-learn development tool for C# programmers interested in creating Windows Forms, Windows Presentation Foundation (WPF) as well as class libraries and console-based applications. 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

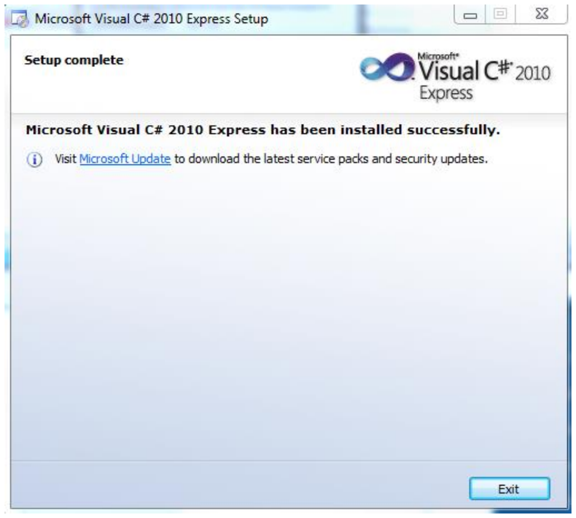
Installation location cannot be changed. Click here for more information.

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

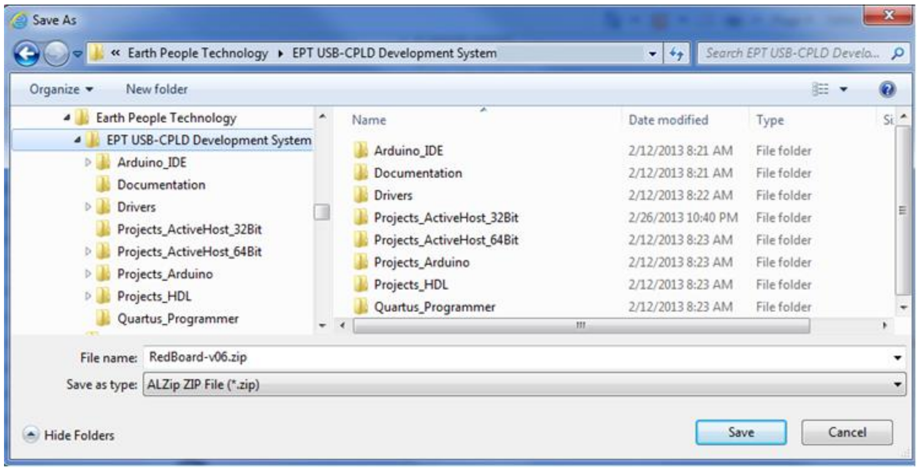
Setup complete

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.

To use the Active Host Application Software, the Active Host DLL and the ftd2xx DLL must be included in the Microsoft Visual project. 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 UnoProLogic. The methods and parameters of the Active Host DLL are explained in the Active Host Application section. Locate the \Projects_ActiveHost_64Bit and \Projects_ActiveHost_32Bit folders on the UnoProLogic Development System CD.



Save As

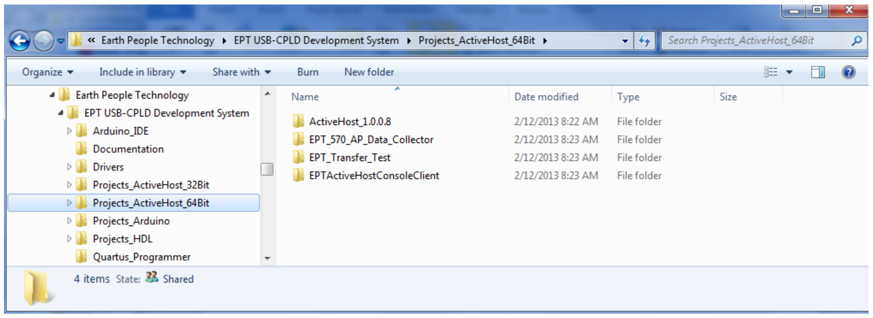
« Earth People Technology ▸ EPT USB-CPLD Development System

Name	Date modified	Type
Arduino_IDE	2/12/2013 8:21 AM	File folder
Documentation	2/12/2013 8:21 AM	File folder
Drivers	2/12/2013 8:22 AM	File folder
Projects_ActiveHost_32Bit	2/26/2013 10:40 PM	File folder
Projects_ActiveHost_64Bit	2/12/2013 8:23 AM	File folder
Projects_Arduino	2/12/2013 8:23 AM	File folder
Projects_HDL	2/12/2013 8:23 AM	File folder
Quartus_Programmer	2/12/2013 8:23 AM	File folder

File name: RedBoard-v06.zip

Save as type: ALZip ZIP File (*.zip)

Locate the Projects_ActiveHost_64Bit in the UnoProLogic Development System using Windows Explorer.



« Earth People Technology ▸ EPT USB-CPLD Development System ▸ Projects_ActiveHost_64Bit ▸

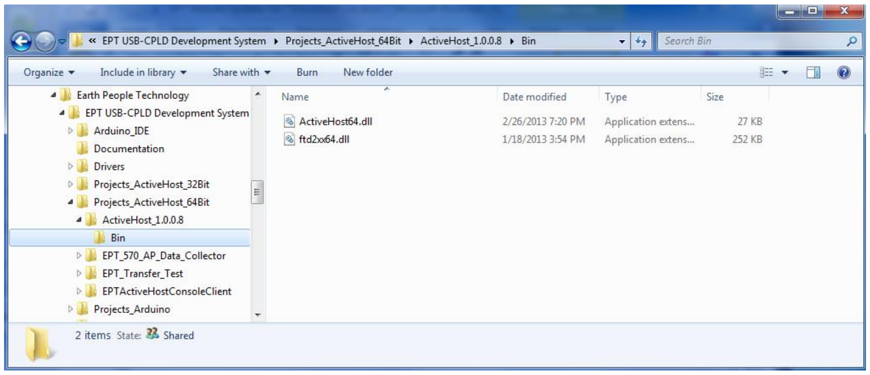
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

- Earth People Technology
  - EPT USB-CPLD Development System
    - Arduino_IDE
    - Documentation
    - Drivers
    - Projects_ActiveHost_32Bit
    - Projects_ActiveHost_64Bit
    - Projects_Arduino
    - Projects_HDL
    - Quartus_Programmer

Name	Date modified	Type	Size
ActiveHost_1.0.0.8	2/12/2013 8:22 AM	File folder	
EPT_570_AP_Data_Collector	2/12/2013 8:23 AM	File folder	
EPT_Transfer_Test	2/12/2013 8:23 AM	File folder	
EPTActiveHostConsoleClient	2/12/2013 8:23 AM	File folder	

4 items State: Shared

Locate the Projects_ActiveHost_64Bit \ActiveHost_1.0.0.8\Bin folder and copy the ActiveHost64.dll and the ftd2xx64.dll.



« EPT USB-CPLD Development System ▸ Projects_ActiveHost_64Bit ▸ ActiveHost_1.0.0.8 ▸ Bin

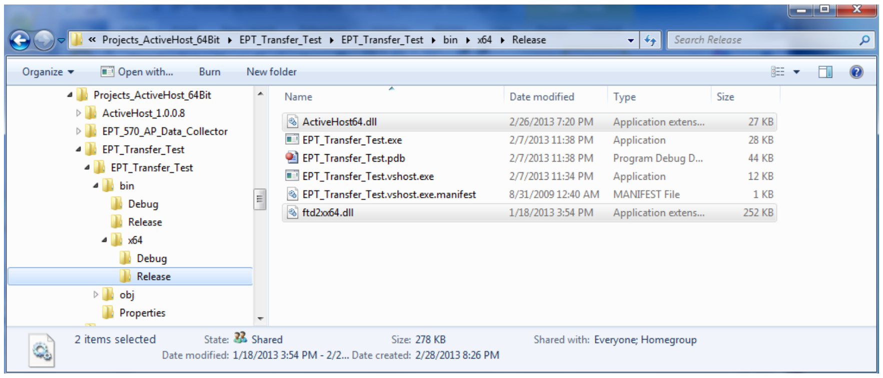
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

- Earth People Technology
  - EPT USB-CPLD Development System
    - Arduino_IDE
    - Documentation
    - Drivers
    - Projects_ActiveHost_32Bit
    - Projects_ActiveHost_64Bit
      - ActiveHost_1.0.0.8
        - Bin
      - EPT_570_AP_Data_Collector
      - EPT_Transfer_Test
      - EPTActiveHostConsoleClient
    - Projects_Arduino

Name	Date modified	Type	Size
ActiveHost64.dll	2/26/2013 7:20 PM	Application extens...	27 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

2 items State: Shared

Save the DLL's in the bin\x64\Release folder of the user project under the Microsoft C# Express project. See the Active Host Application section of the UnoProLogic Development System User Manuals for instructions on how to add the dll to the Microsoft C# Express project.



« Projects_ActiveHost_64Bit ▸ EPT_Transfer_Test ▸ EPT_Transfer_Test ▸ bin ▸ x64 ▸ Release

Organize ▾ Open with... Burn New folder

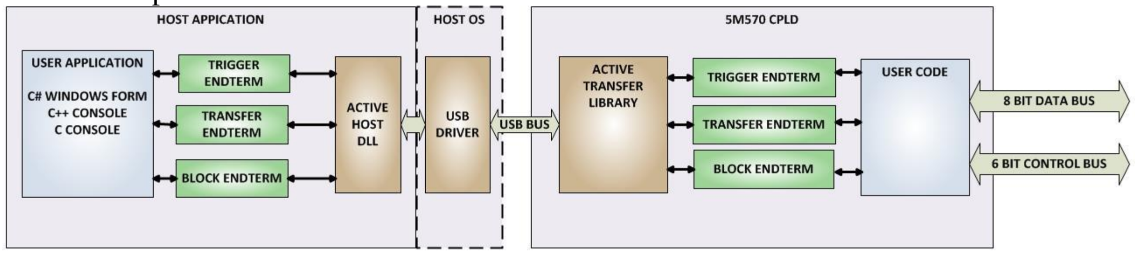
- Projects_ActiveHost_64Bit
  - ActiveHost_1.0.0.8
  - EPT_570_AP_Data_Collector
  - EPT_Transfer_Test
    - EPT_Transfer_Test
      - bin
        - Debug
        - Release
        - x64
          - Debug
          - Release
      - obj
      - Properties

Name	Date modified	Type	Size
ActiveHost64.dll	2/26/2013 7:20 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	2/7/2013 11:38 PM	Application	28 KB
EPT_Transfer_Test.pdb	2/7/2013 11:38 PM	Program Debug D...	44 KB
EPT_Transfer_Test.vshost.exe	2/7/2013 11:34 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	8/31/2009 12:40 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

2 items selected State: Shared Size: 278 KB Shared with: Everyone; Homegroup
Date modified: 1/18/2013 3:54 PM - 2/2... Date created: 2/28/2013 8:26 PM

3 Active Transfer Library

The Active Transfer Library is an HDL library designed to transfer data to and from the UnoMax via High Speed 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 Figure 6 for an overview of the EPT Active_Transfer system.

Figure 6 EPT Active Transfer Library Overview

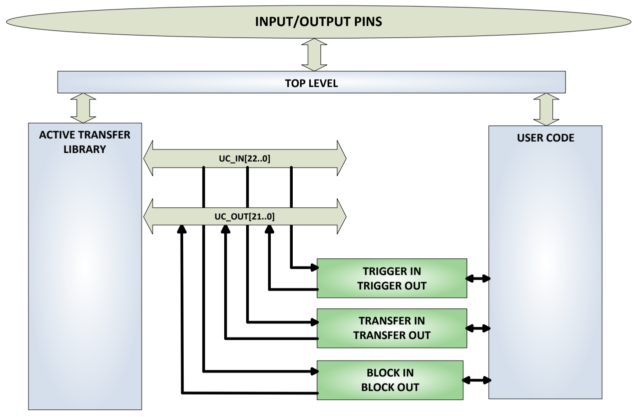
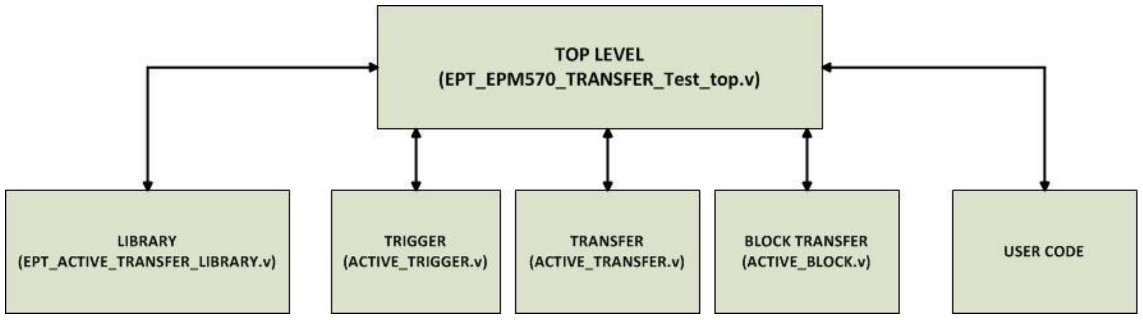
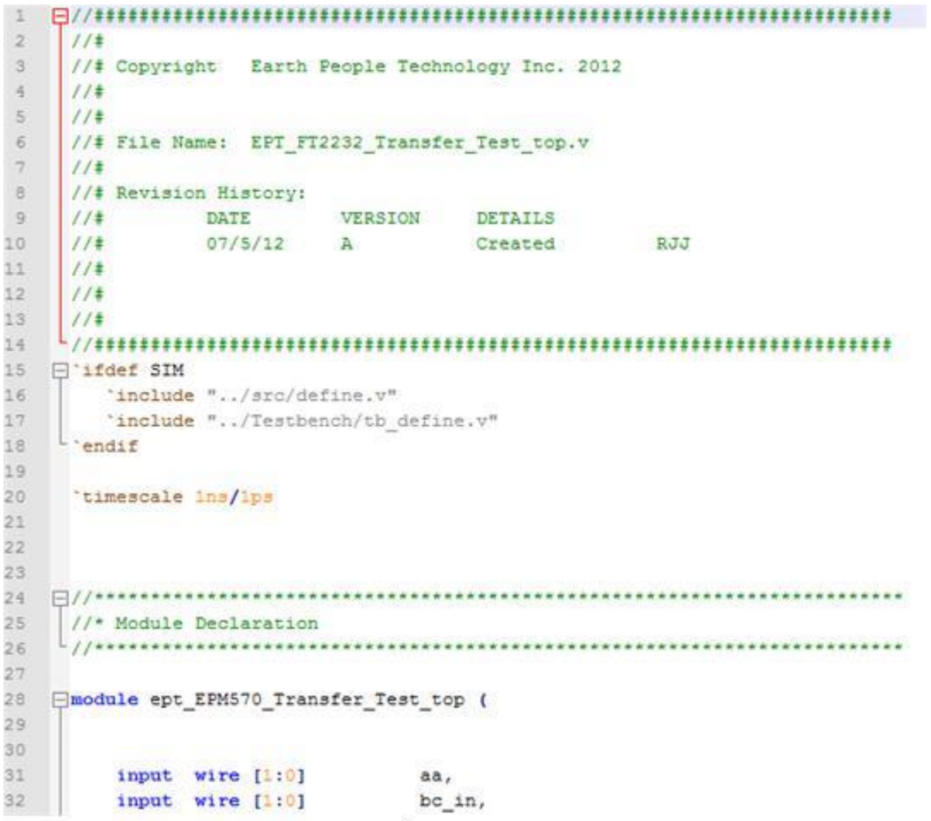


Figure 6 shows how the modules of the EPT Active Transfer Library attach to the overall user project. The EPT Active_Transfer_Library.vqm, Active_Trigger.v, Active_Transfer.v and Active_Block.v modules are instantiated in the top level of the user project. The User_Code.v module is also instantiated 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 CPLD. 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

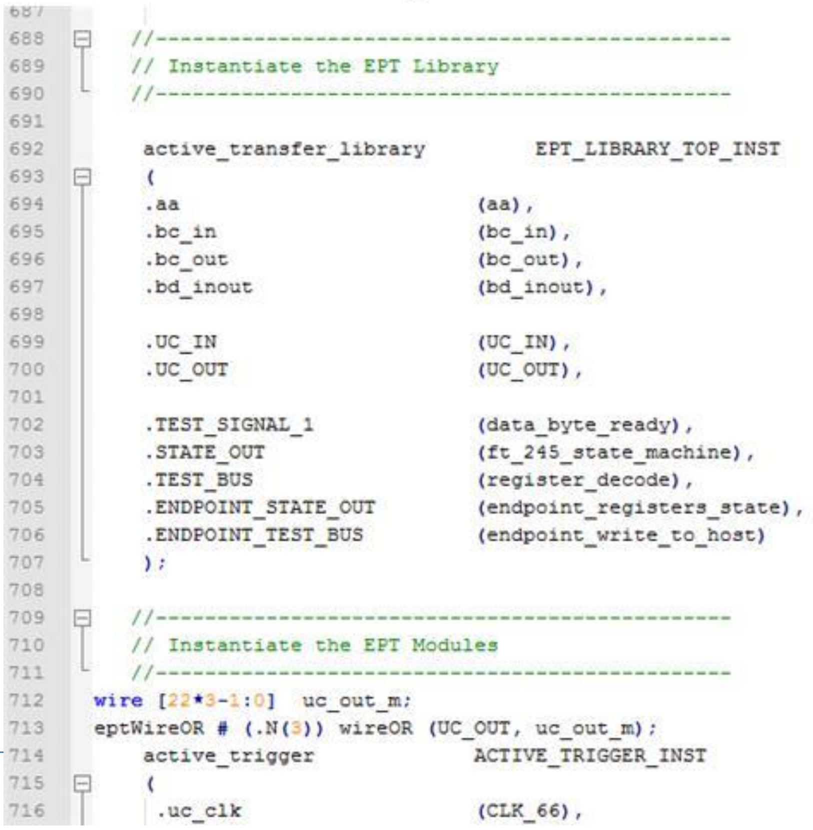
The Active Transfer Library contains the command, control, and data transfer mechanism that allows users to quickly build powerful communication schemes in the CPLD. Coupled with the Active Host application on the PC, this tools allows users to focus on creating programmable logic applications and not 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. 2012
//#
//#
//# File Name:  EPT_FT2232_Transfer_Test_top.v
//#
//# Revision History:
//#         DATE        VERSION     DETAILS
//#         07/5/12     A           Created         RJJ
//#
//#
//#
//########################################################################
`ifdef SIM
   `include "../src/define.v"
   `include "../Testbench/tb_define.v"
`endif

`timescale 1ns/1ps



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

module ept_EPM570_Transfer_Test_top (


    input  wire [1:0]           aa,
    input  wire [1:0]           bc_in,
```

●
●
●



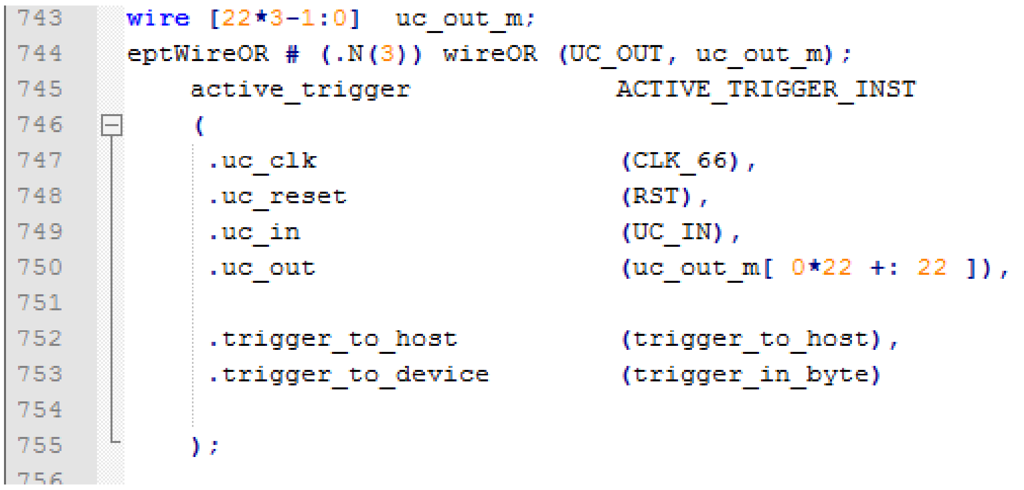
```
     //-----------------------------------------------
     // Instantiate the EPT Library
     //-----------------------------------------------

      active_transfer_library           EPT_LIBRARY_TOP_INST
      (
      .aa                           (aa),
      .bc_in                        (bc_in),
      .bc_out                       (bc_out),
      .bd_inout                     (bd_inout),

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

      .TEST_SIGNAL_1                (data_byte_ready),
      .STATE_OUT                    (ft_245_state_machine),
      .TEST_BUS                     (register_decode),
      .ENDPOINT_STATE_OUT           (endpoint_registers_state),
      .ENDPOINT_TEST_BUS            (endpoint_write_to_host)
      );

     //-----------------------------------------------
     // Instantiate the EPT Modules
     //-----------------------------------------------
   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),
```

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.

The control bus UART_IN and UART_OUT 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 CPLD.

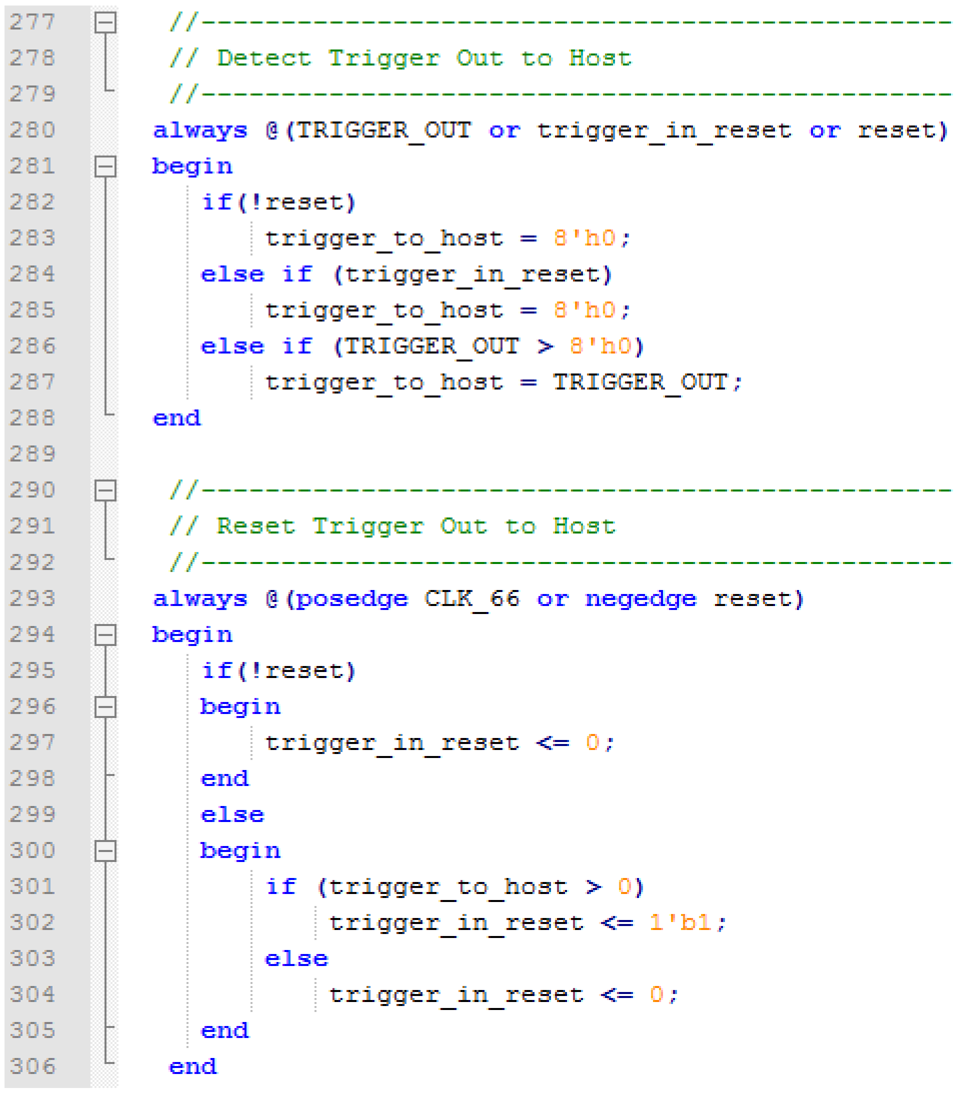
3.2.1 Active Trigger EndTerm

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)

    );
```

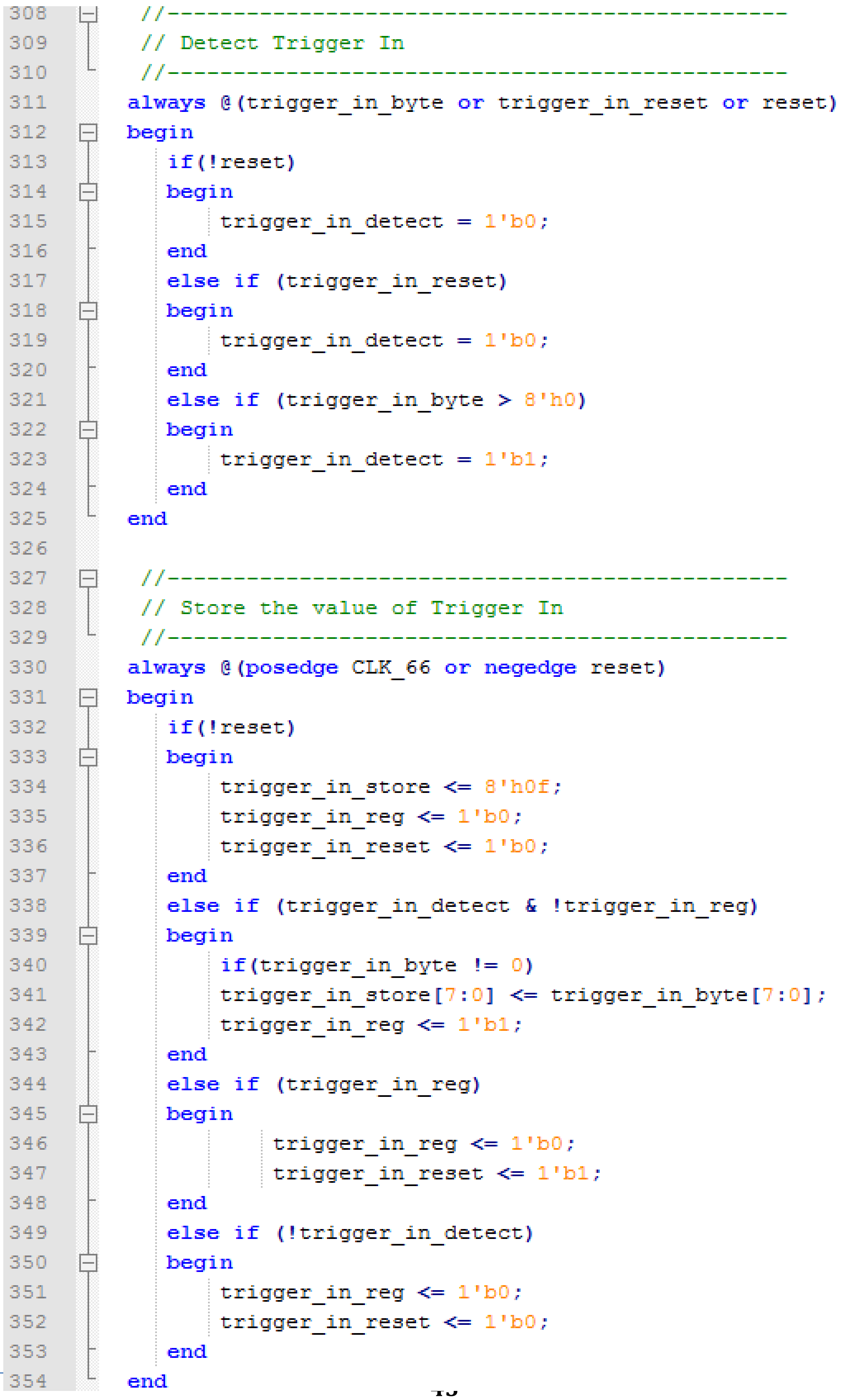
To send a trigger, decide which bit (or multiple bits) of the eight bits you want to send the trigger on. Then, set that bit (or bits) high. The Active Transfer Library will send a high on that trigger bit for one clock cycle (66 MHz), then reset itself to zero. The bit can stay high on the user code 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.

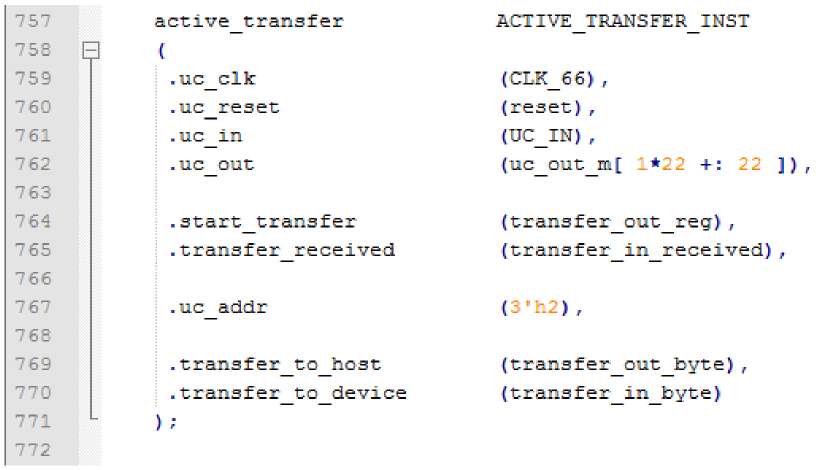
The user code must be setup to receive triggers from the Host. This can be done by using an asynchronous always block. Whenever a change occurs on a particular trigger bit (or bits), a conditional branch can detect if the trigger bit is for that block of code. Then, execute some code based on that trigger.



```verilog
 //------------------------------------------------
 // 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
end
```

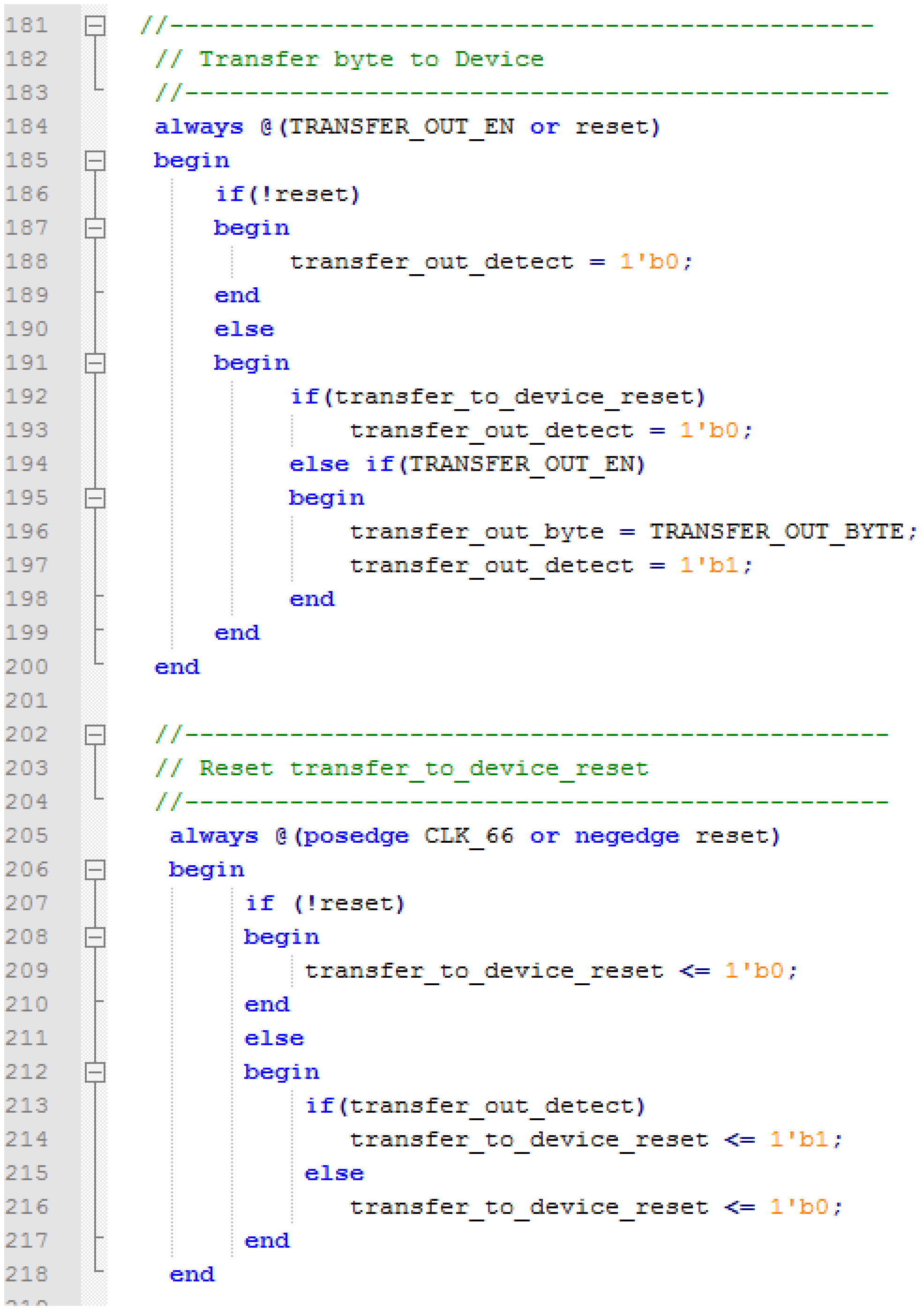
3.2.2 Active Transfer EndTerm

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.



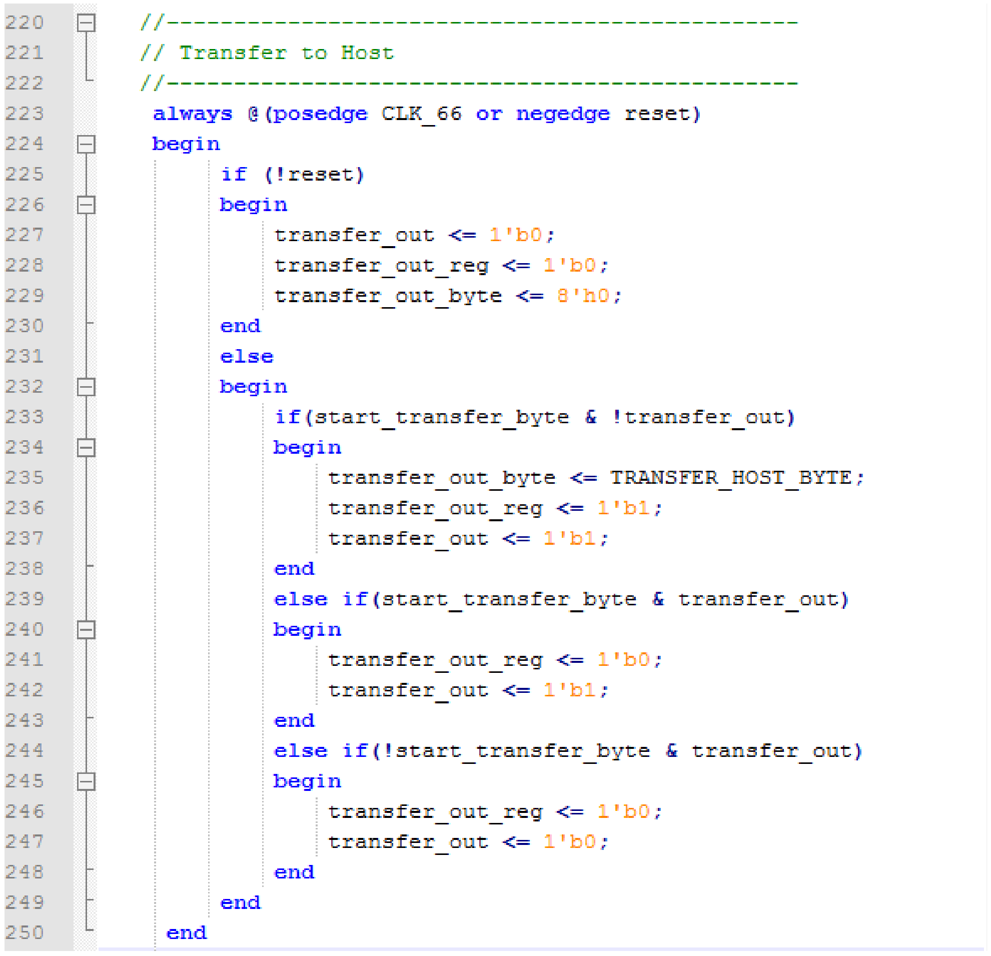
```
active_transfer                 ACTIVE_TRANSFER_INST
(
 .uc_clk                         (CLK_66),
 .uc_reset                       (reset),
 .uc_in                          (UC_IN),
 .uc_out                         (uc_out_m[ 1*22 +: 22 ]),

 .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
```

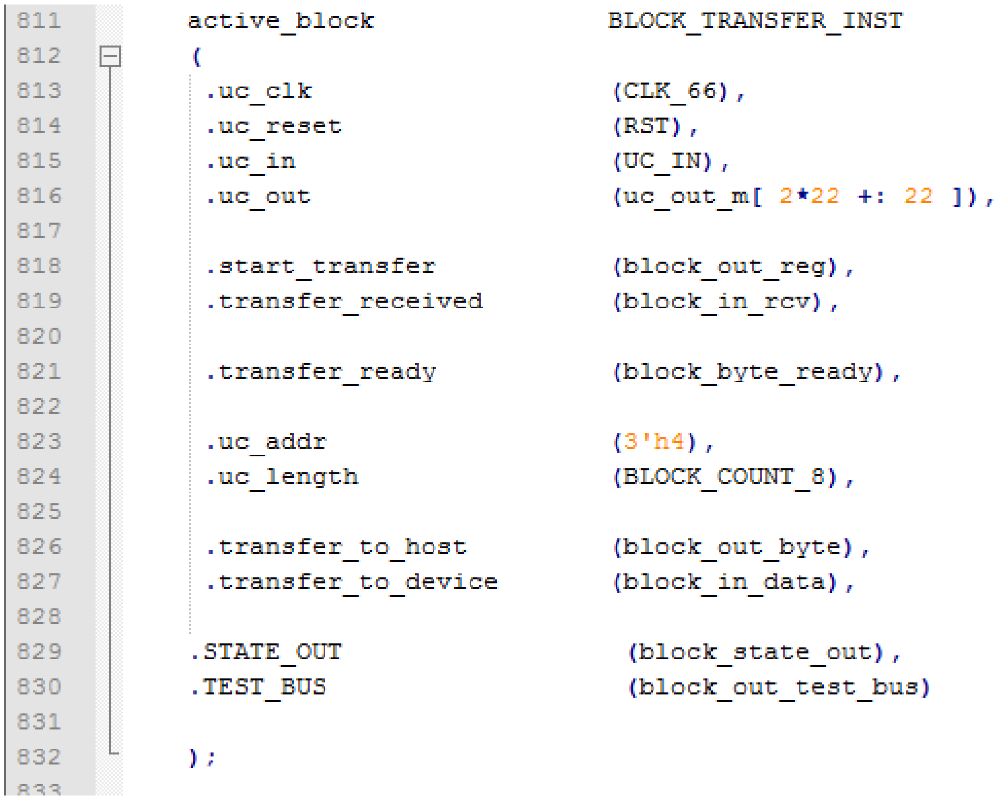
To receive a byte, the Active Host will send a byte using it's dll. The user code must monitor the transfer_received port. The transfer_received port will assert high for one clock cycle (66 MHz) when a byte is ready for reading on the transfer_to_device port. User code should use an asynchronous always 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

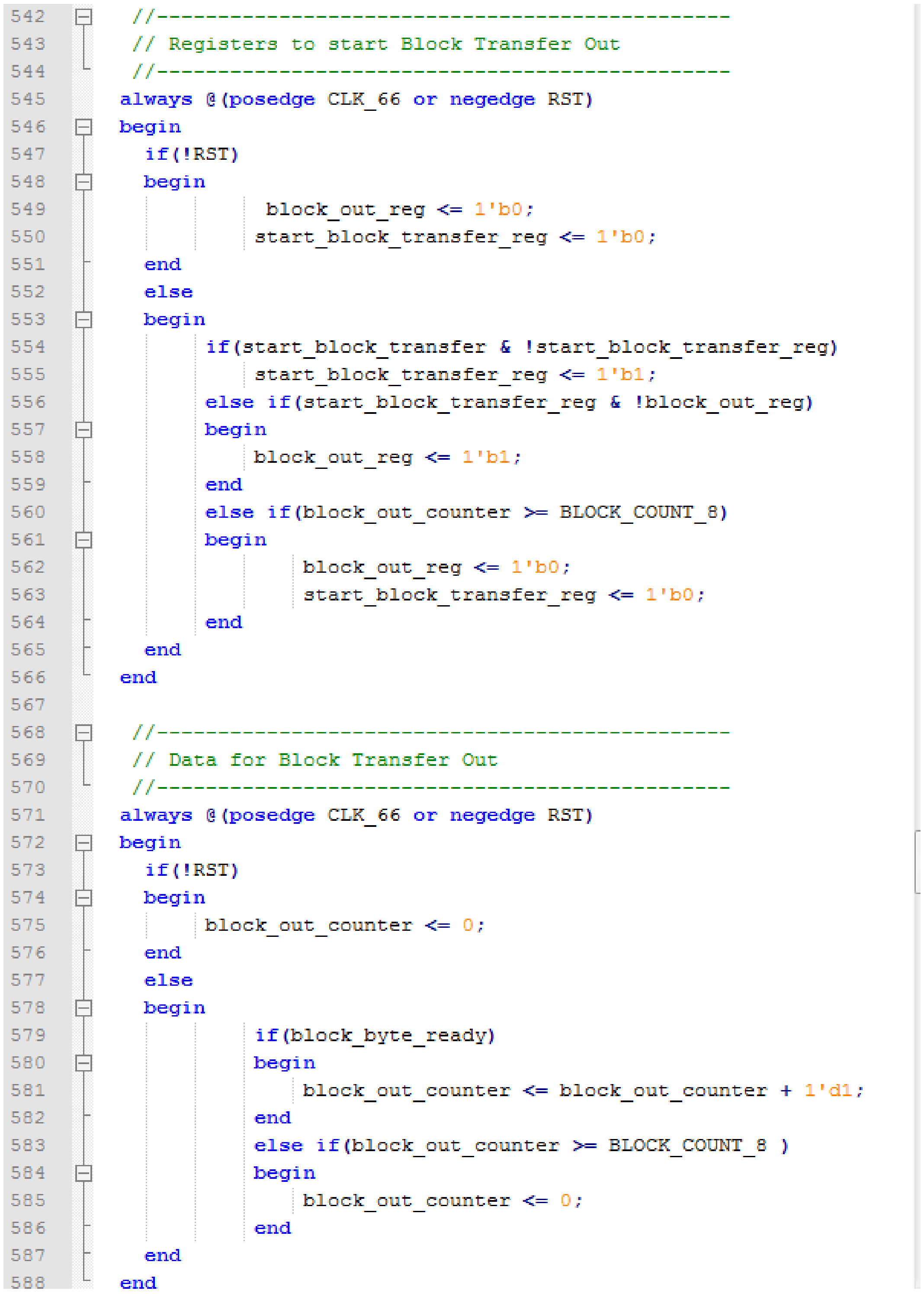
The Active Block module is designed to transfer blocks of data between Host and User Code and vice versa. This allows buffers of data to be transferred with a minimal amount of code. The Active Block module is addressable, so up to eight individual modules can be instantiated and separately addressed.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

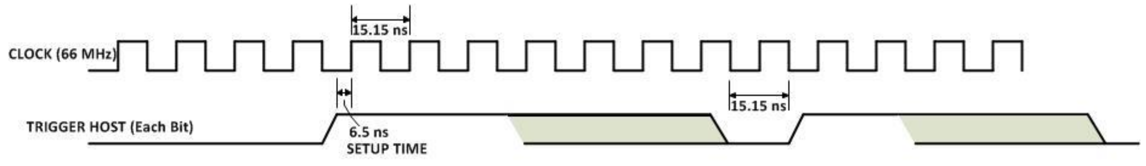


Figure xx Active Trigger to Host Timing

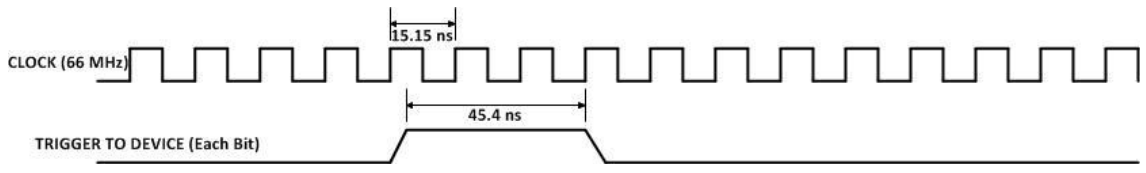


Figure xx Active Trigger to Device Timing

3.3.2 Active Transfer EndTerm Timing

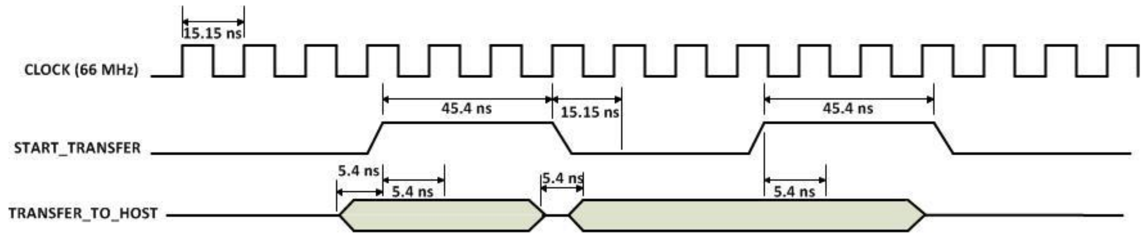


Figure xx Active Transfer To Host Timing

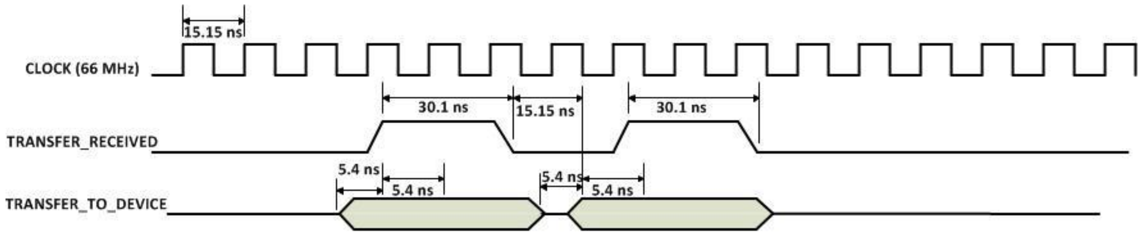


Figure xx Active Transfer To Device Timing

3.3.3 Active Block EndTerm Timing

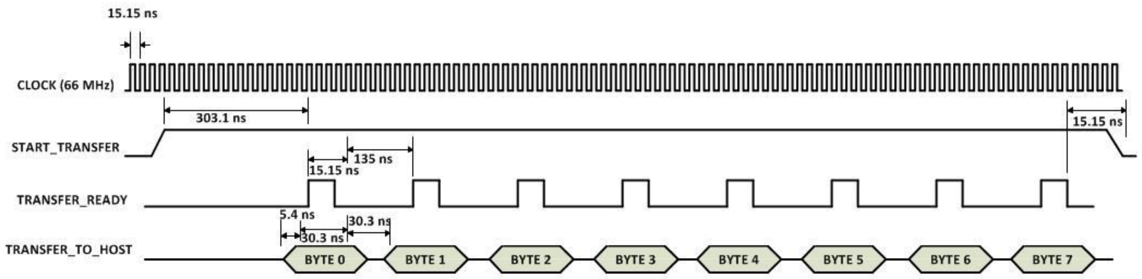


Figure xx Active Block To Host Timing

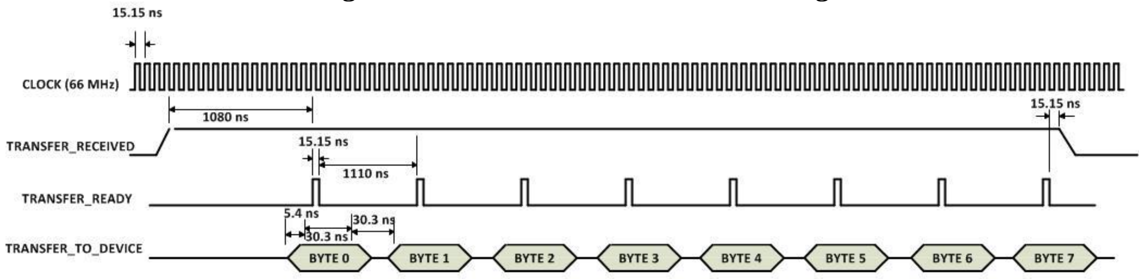
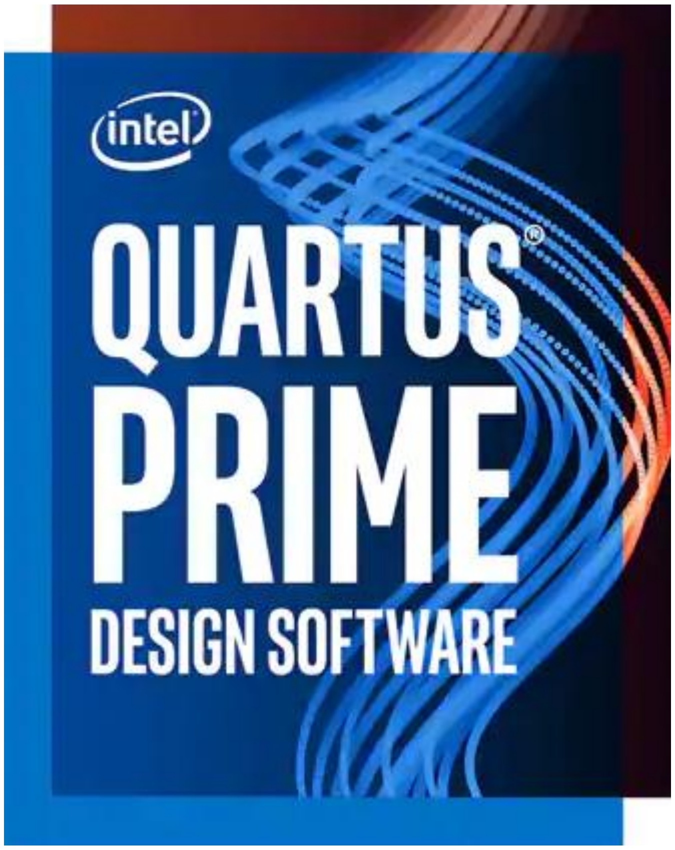


Figure xx Active Block To Device Timing

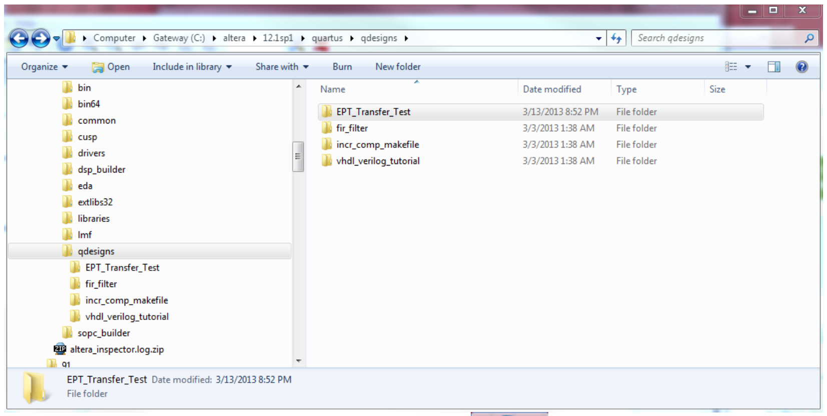
4 Compiling, Synthesizing, and Programming CPLD



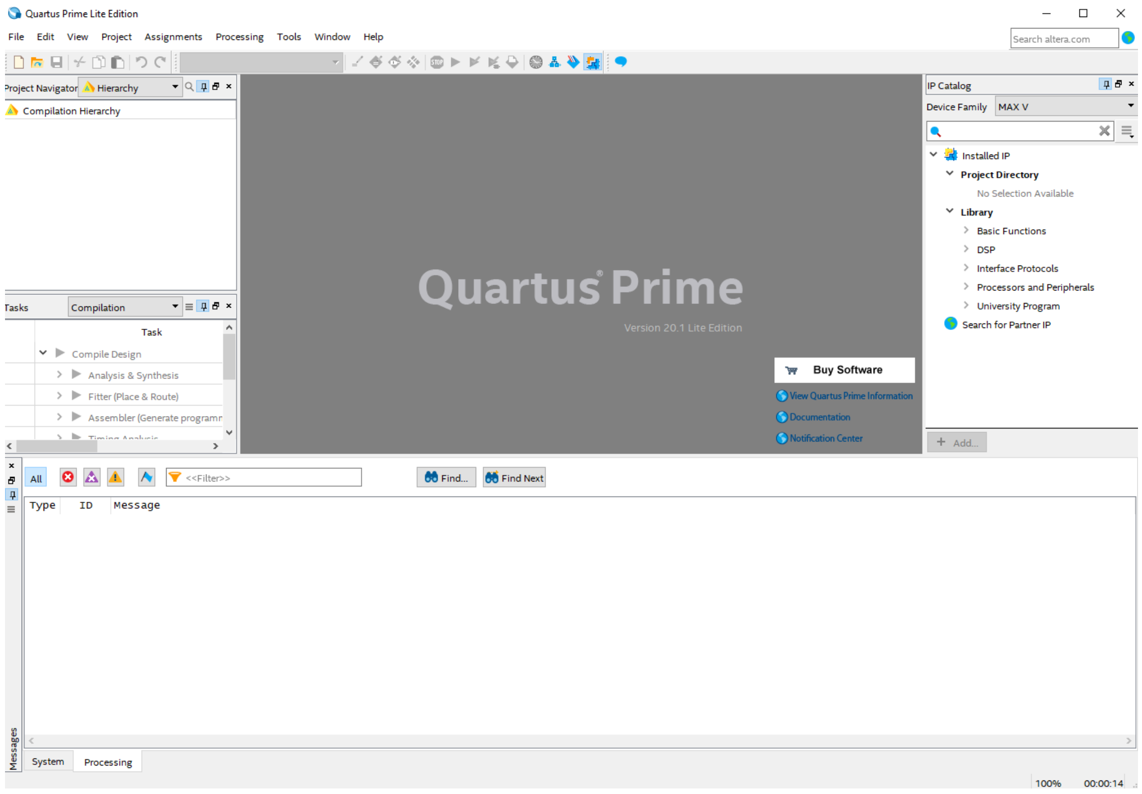
The CPLD on the UnoMax can be programmed with the Active Transfer Library and custom HDL code created by the user. Programming the CPLD requires the use of the Quartus Prime Lite software and a standard USB cable. There are no extra parts to buy, just plug in the USB cable. Once the user HDL code is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

4.1 Setting up the Project and Compiling

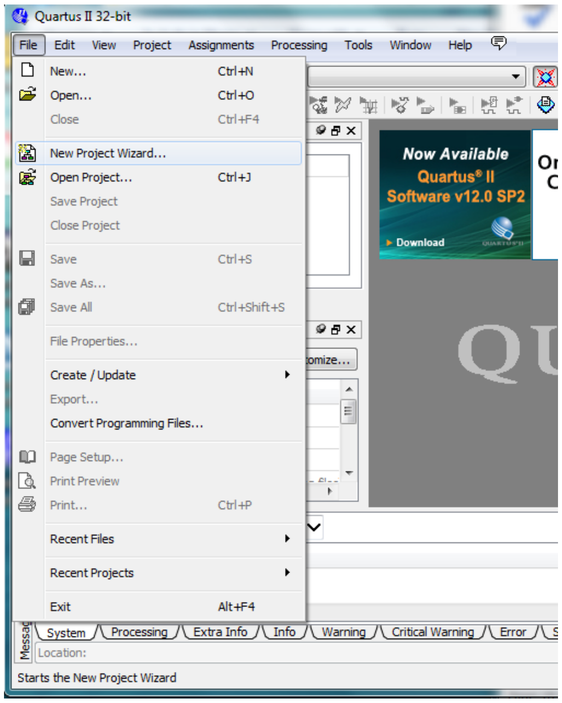
Once the HDL code (Verilog or VHDL) is written and verified using a simulator, a project can be created using Quartus Prime Lite. Writing the HDL code and simulating it will be covered in later sections. Bring up Quartus Prime Lite, then use Windows Explorer to browse to c:/altera/xxx/quartus/qdesigns create a new directory called: “EPT_Transfer_Test”.


Computer ▸ Gateway (C:) ▸ altera ▸ 12.1sp1 ▸ quartus ▸ qdesigns
Organize
Open
Include in library
Share with
Burn
New folder
bin
bin64
common
cusp
drivers
dsp_builder
eda
extlibs32
libraries
lmf
qdesigns
EPT_Transfer_Test
fir_filter
incr_comp_makefile
vhdl_verilog_tutorial
sopc_builder
altera_inspector.log.zip
Name
Date modified
Type
Size
EPT_Transfer_Test 3/13/2013 8:52 PM File folder
fir_filter 3/3/2013 1:38 AM File folder
incr_comp_makefile 3/3/2013 1:38 AM File folder
vhdl_verilog_tutorial 3/3/2013 1:38 AM File folder
EPT_Transfer_Test Date modified: 3/13/2013 8:52 PM
File folder

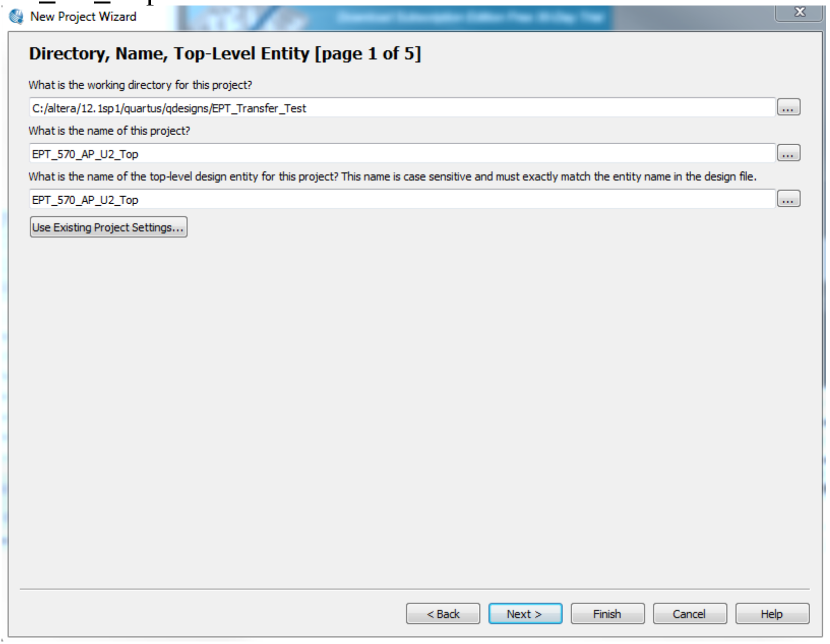
Open Quartus Prime Lite by clicking on the icon .



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



At the Top-Level Entity page, browse to the C:/intelFPGA_lite /xxx/quartus/qdesigns directory to store your project. Type in a name for your project "EPT_570_AP_U2_Top".


New Project Wizard

Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test

What is the name of this project?

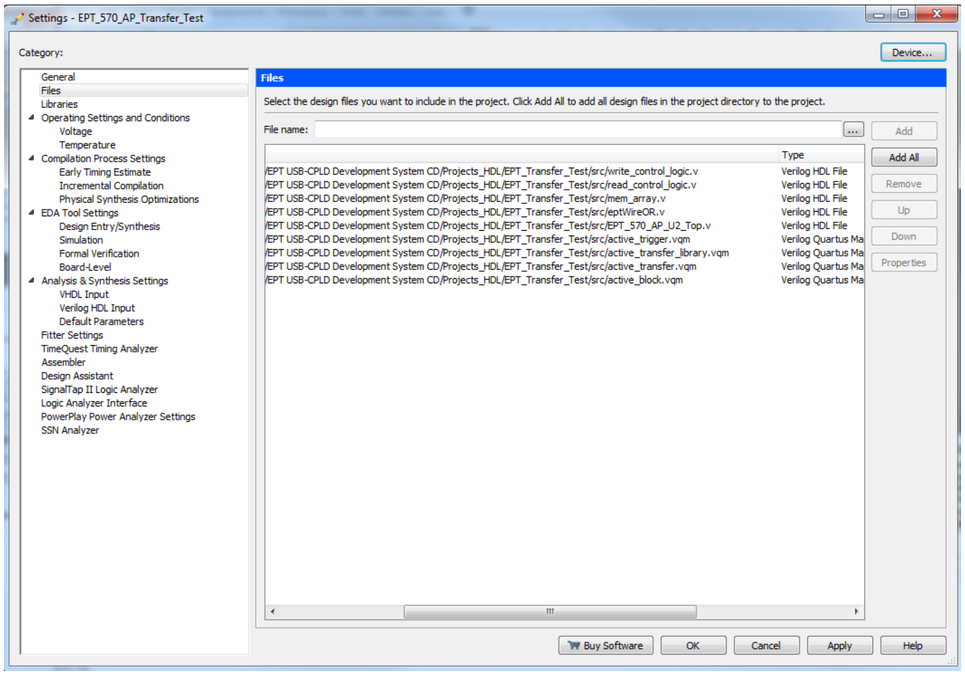
EPT_570_AP_U2_Top

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_570_AP_U2_Top

Select Next. At the Add Files window: Browse to the \Projects_HDL\EPT_Transfer_Test \src folder of the EPT USB-CPLD Development System CD. Copy the files from the \src directory.

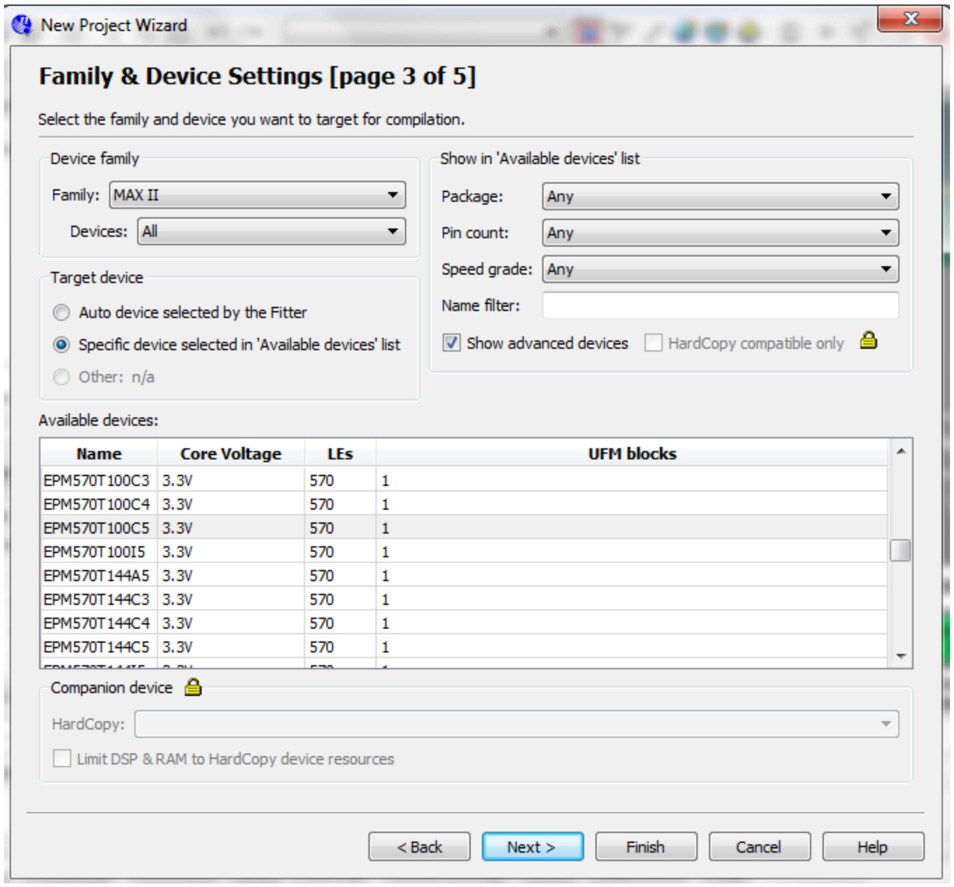
- Active_block.v
- Active_transfer.v
- Active_trigger.v
- Active_Serial_library.v
- eptWireOr.v
- mem_array.v
- read_control_logic.v
- write_control_logic.v
- EPT_570_AP_U2_Top.v



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.

	Type
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/write_control_logic.v	Verilog HDL File
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/read_control_logic.v	Verilog HDL File
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/mem_array.v	Verilog HDL File
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/eptWireOR.v	Verilog HDL File
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/EPT_570_AP_U2_Top.v	Verilog HDL File
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_trigger.vqm	Verilog Quartus Ma
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_transfer_library.vqm	Verilog Quartus Ma
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_transfer.vqm	Verilog Quartus Ma
/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_block.vqm	Verilog Quartus Ma

Select Next, at the Device Family group, select MAX V for Family. In the Available Devices group, browse down to 5M570ZT100C5 for Name.



New Project Wizard

Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family

Family: MAX II

Devices: 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

Speed grade: Any

Name filter:

[✓] Show advanced devices [ ] HardCopy compatible only

Available devices:

Name	Core Voltage	LEs	UFM blocks
EPM570T100C3	3.3V	570	1
EPM570T100C4	3.3V	570	1
EPM570T100C5	3.3V	570	1
EPM570T100I5	3.3V	570	1
EPM570T144A5	3.3V	570	1
EPM570T144C3	3.3V	570	1
EPM570T144C4	3.3V	570	1
EPM570T144C5	3.3V	570	1

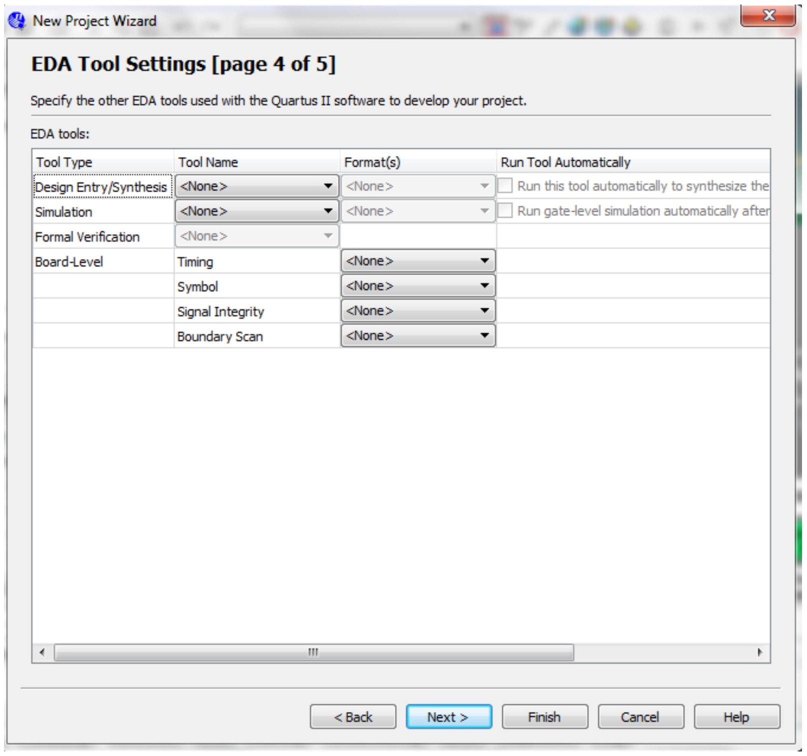
Companion device

HardCopy:

[ ] Limit DSP & RAM to HardCopy device resources

< Back | Next > | Finish | Cancel | Help

Select Next, leave defaults for the EDA Tool Settings.



New Project Wizard

EDA Tool Settings [page 4 of 5]

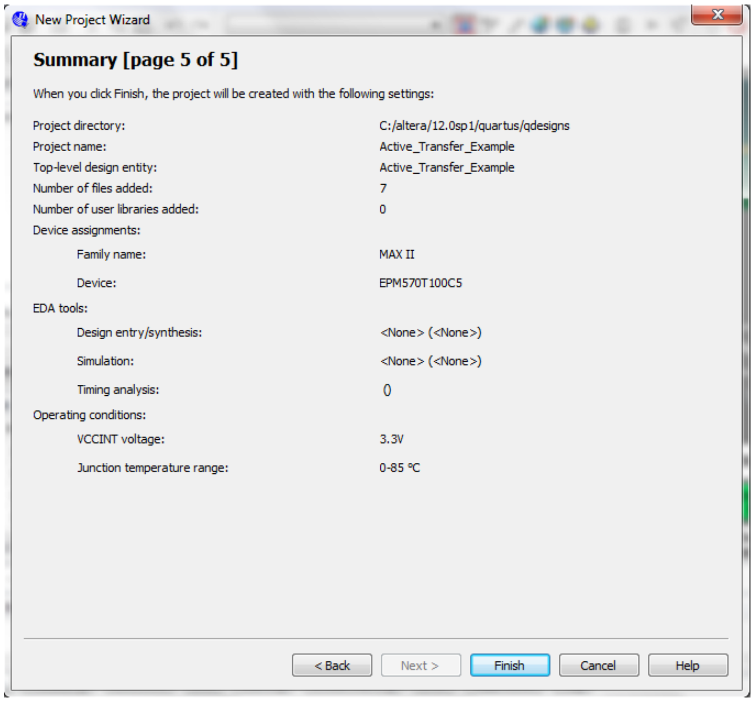
Specify the other EDA tools used with the Quartus II software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synthesis	<None>	<None>	Run this tool automatically to synthesize the
Simulation	<None>	<None>	Run gate-level simulation automatically after
Formal Verification	<None>		
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back | Next > | Finish | Cancel | Help

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



New Project Wizard

Summary [page 5 of 5]

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

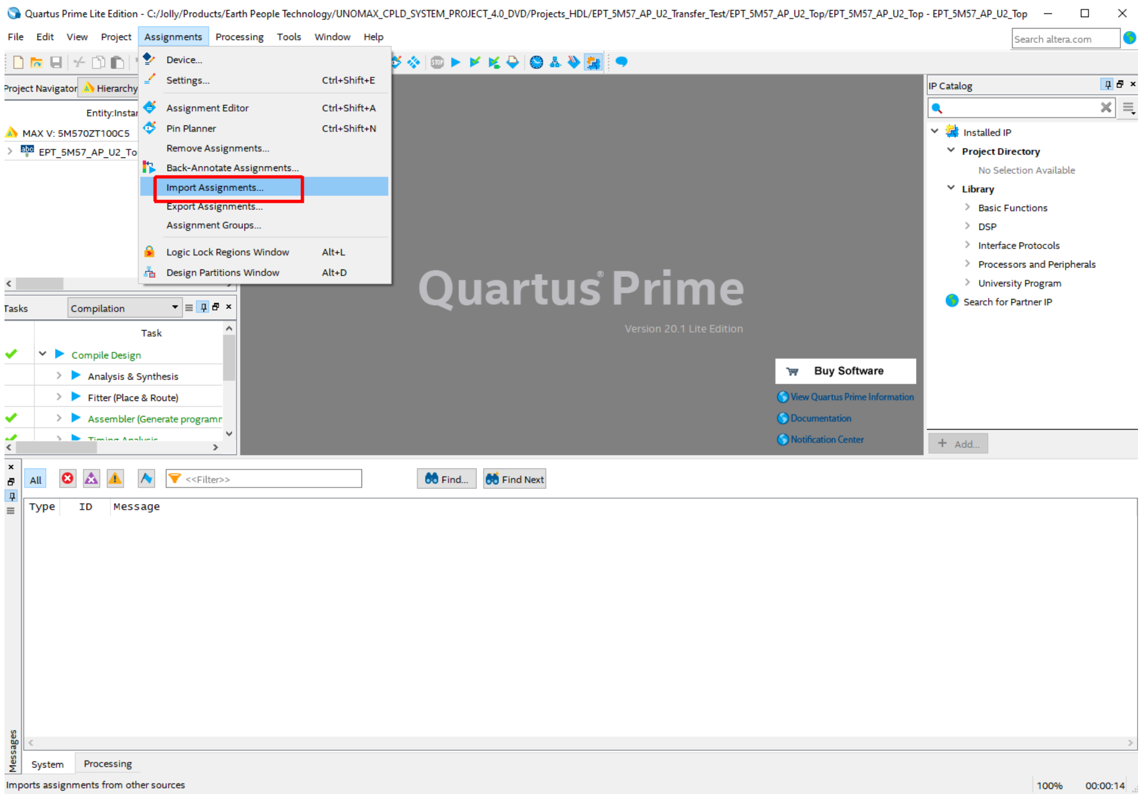
Project directory:	C:/altera/12.0sp1/quartus/qdesigns
Project name:	Active_Transfer_Example
Top-level design entity:	Active_Transfer_Example
Number of files added:	7
Number of user libraries added:	0
Device assignments:	
Family name:	MAX II
Device:	EPM570T100C5
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	0
Operating conditions:	
VCCINT voltage:	3.3V
Junction temperature range:	0-85 ℃

< Back | Next > | Finish | Cancel | Help

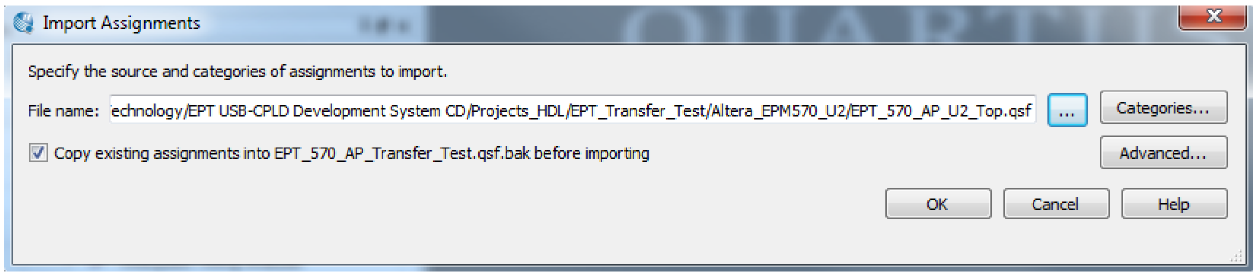
Next, we will select the pins and synthesize the project.

4.1.1 Selecting Pins and Synthesizing

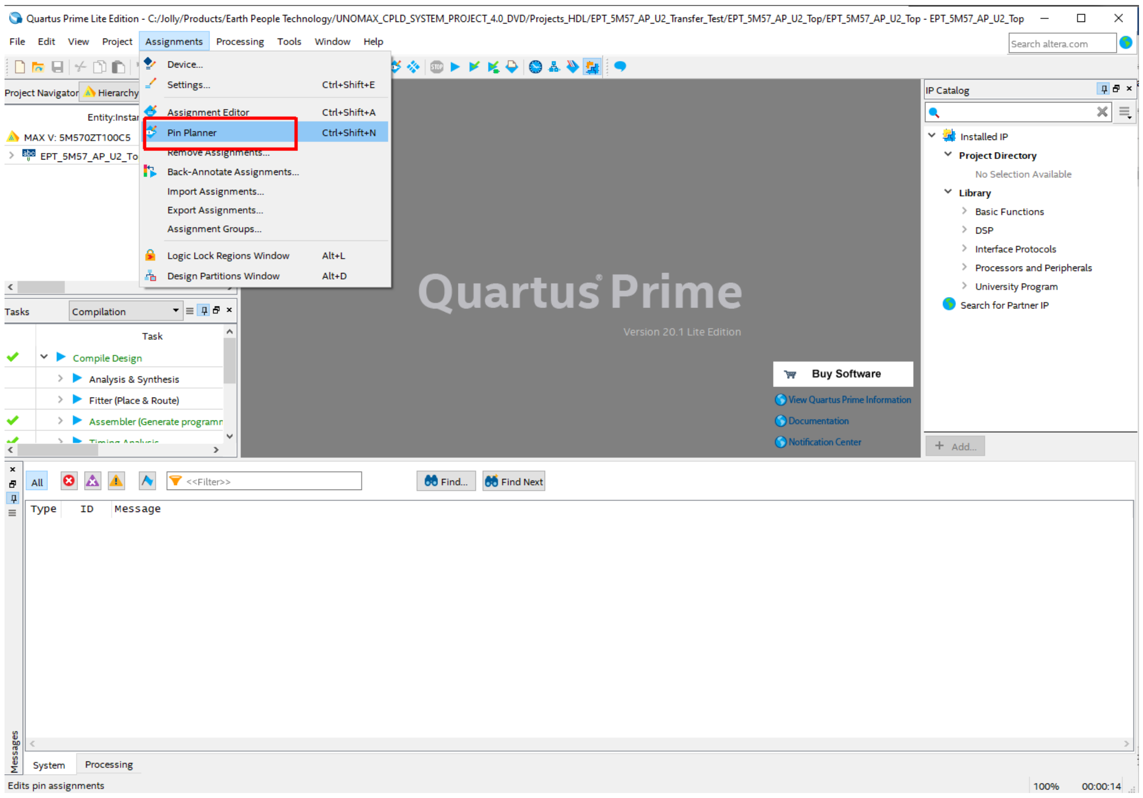
With the project created, we need to assign pins to the project. The signals defined in the top level file (in this case: EPT_570_AP_U2_Top.v) will connect directly to pins on the CPLD. The Pin Planner Tool from Quartus Prime Lite will add the pins and check to verify that our pin selections do not violate any restrictions of the device. In the case of this example we will import pin assignments that created at an earlier time. Under Assignments, Select Import Assignments.



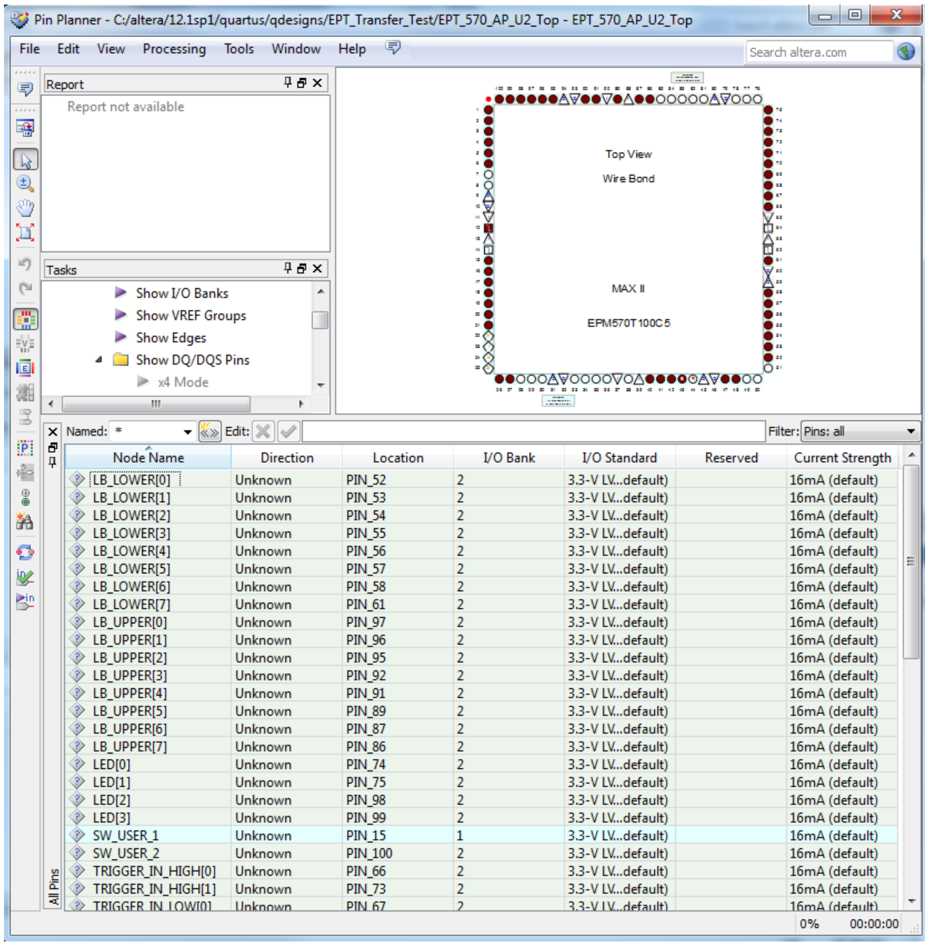
At the Import Assignment dialog box, Browse to the \Projects_HDL\EPT_Transfer_Test \ EPT -5M57-AP-U2_TOP folder of the EPT USB-CPLD Development System CD. Select the "EPT_570_AP_U2_Top.qsf" file.


Import Assignments
Specify the source and categories of assignments to import.
File name: echnology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/Altera_EPM570_U2/EPT_570_AP_U2_Top.qsf
...
Categories...
Copy existing assignments into EPT_570_AP_Transfer_Test.qsf.bak before importing
Advanced...
OK
Cancel
Help

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



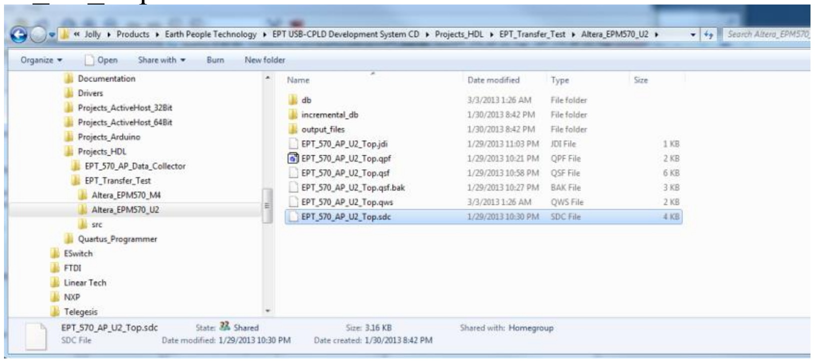
The pin locations should not need to be changed for EPT USB CPLD Development System. However, if you need to change any pin location, just click on the “location” column for the particular node you wish to change. Then, select the new pin location from the drop down box.



Node Name	Direction	Location	I/O Bank	I/O Standard	Reserved	Current Strength
LB_LOWER[0]	Unknown	PIN_52	2	3.3-V LV...default)		16mA (default)
LB_LOWER[1]	Unknown	PIN_53	2	3.3-V LV...default)		16mA (default)
LB_LOWER[2]	Unknown	PIN_54	2	3.3-V LV...default)		16mA (default)
LB_LOWER[3]	Unknown	PIN_55	2	3.3-V LV...default)		16mA (default)
LB_LOWER[4]	Unknown	PIN_56	2	3.3-V LV...default)		16mA (default)
LB_LOWER[5]	Unknown	PIN_57	2	3.3-V LV...default)		16mA (default)
LB_LOWER[6]	Unknown	PIN_58	2	3.3-V LV...default)		16mA (default)
LB_LOWER[7]	Unknown	PIN_61	2	3.3-V LV...default)		16mA (default)
LB_UPPER[0]	Unknown	PIN_97	2	3.3-V LV...default)		16mA (default)
LB_UPPER[1]	Unknown	PIN_96	2	3.3-V LV...default)		16mA (default)
LB_UPPER[2]	Unknown	PIN_95	2	3.3-V LV...default)		16mA (default)
LB_UPPER[3]	Unknown	PIN_92	2	3.3-V LV...default)		16mA (default)
LB_UPPER[4]	Unknown	PIN_91	2	3.3-V LV...default)		16mA (default)
LB_UPPER[5]	Unknown	PIN_89	2	3.3-V LV...default)		16mA (default)
LB_UPPER[6]	Unknown	PIN_87	2	3.3-V LV...default)		16mA (default)
LB_UPPER[7]	Unknown	PIN_86	2	3.3-V LV...default)		16mA (default)
LED[0]	Unknown	PIN_74	2	3.3-V LV...default)		16mA (default)
LED[1]	Unknown	PIN_75	2	3.3-V LV...default)		16mA (default)
LED[2]	Unknown	PIN_98	2	3.3-V LV...default)		16mA (default)
LED[3]	Unknown	PIN_99	2	3.3-V LV...default)		16mA (default)
SW_USER_1	Unknown	PIN_15	1	3.3-V LV...default)		16mA (default)
SW_USER_2	Unknown	PIN_100	2	3.3-V LV...default)		16mA (default)
TRIGGER_IN_HIGH[0]	Unknown	PIN_66	2	3.3-V LV...default)		16mA (default)
TRIGGER_IN_HIGH[1]	Unknown	PIN_73	2	3.3-V LV...default)		16mA (default)
TRIGGER_IN_LOW[0]	Unknown	PIN_67	2	3.3-V LV...default)		16mA (default)

Exit the Pin Planner. Next, we need to add the Synopsys Design Constraint file. This file contains timing constraints which forces the built in tool called TimeQuest Timing Analyzer to analyze the path of the synthesized HDL code with setup and hold times of the internal registers. It takes note of any path that may be too long to appropriately meet the timing qualifications. For more information on TimeQuest Timing Analyzer, see http://www.altera.com/literature/hb/qts/qts_qii53018.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=TimeQuest Timing Analyzer

Browse to the \Projects_HDL\EPT_Transfer_Test \ EPT_5M57_AP_U2_TOP folder of the EPT USB-CPLD Development System DVD. Select the "EPT_570_AP_U2_Top.sdc" file.

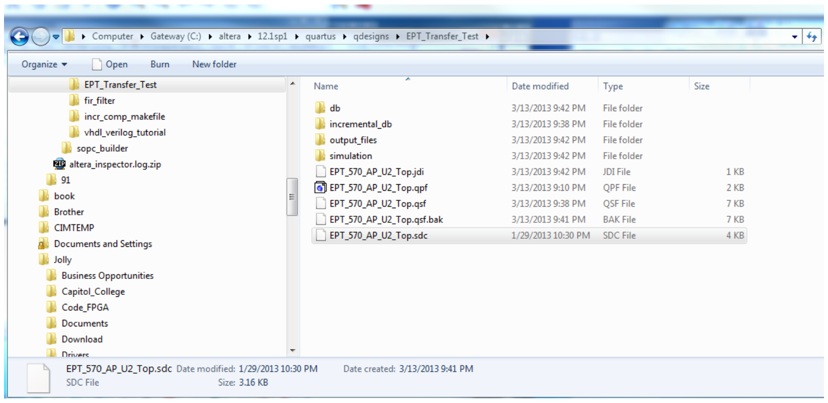

« Jolly ▸ Products ▸ Earth People Technology ▸ EPT USB-CPLD Development System CD ▸ Projects_HDL ▸ EPT_Transfer_Test ▸ Altera_EPM570_U2 ▸

Documentation
Drivers
Projects_ActiveHost_32Bit
Projects_ActiveHost_64Bit
Projects_Arduino
Projects_HDL
EPT_570_AP_Data_Collector
EPT_Transfer_Test
Altera_EPM570_M4
Altera_EPM570_U2
src
Quartus_Programmer
ESwitch
FTDI
Linear Tech
NXP
Telegesis

Name	Date modified	Type	Size
db	3/3/2013 1:26 AM	File folder	
incremental_db	1/30/2013 8:42 PM	File folder	
output_files	1/30/2013 8:42 PM	File folder	
EPT_570_AP_U2_Top.jdi	1/29/2013 11:03 PM	JDI File	1 KB
EPT_570_AP_U2_Top.qpf	1/29/2013 10:21 PM	QPF File	2 KB
EPT_570_AP_U2_Top.qsf	1/29/2013 10:58 PM	QSF File	6 KB
EPT_570_AP_U2_Top.qsf.bak	1/29/2013 10:27 PM	BAK File	3 KB
EPT_570_AP_U2_Top.qws	3/3/2013 1:26 AM	QWS File	2 KB
EPT_570_AP_U2_Top.sdc	1/29/2013 10:30 PM	SDC File	4 KB

EPT_570_AP_U2_Top.sdc State: Shared Size: 3.16 KB Shared with: Homegroup
SDC File Date modified: 1/29/2013 10:30 PM Date created: 1/30/2013 8:42 PM

Copy the file and browse to C:\intelFPGA_lite \xxx\quartus\qdesigns\EPT_Transfer_Test directory. Paste the file.

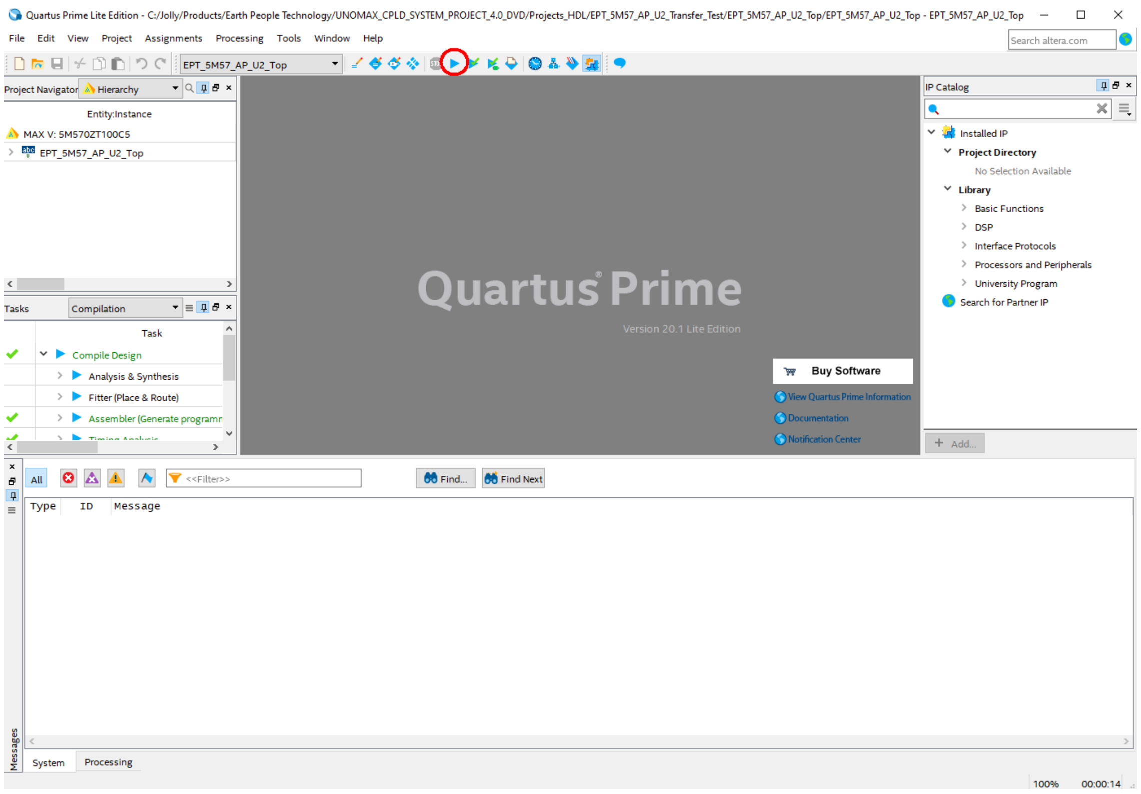

Computer ▸ Gateway (C:) ▸ altera ▸ 12.1sp1 ▸ quartus ▸ qdesigns ▸ EPT_Transfer_Test ▸

EPT_Transfer_Test
fir_filter
incr_comp_makefile
vhdl_verilog_tutorial
sopc_builder
altera_inspector.log.zip
91
book
Brother
CIMTEMP
Documents and Settings
Jolly
Business Opportunities
Capitol_College
Code_FPGA
Documents
Download
Drivers

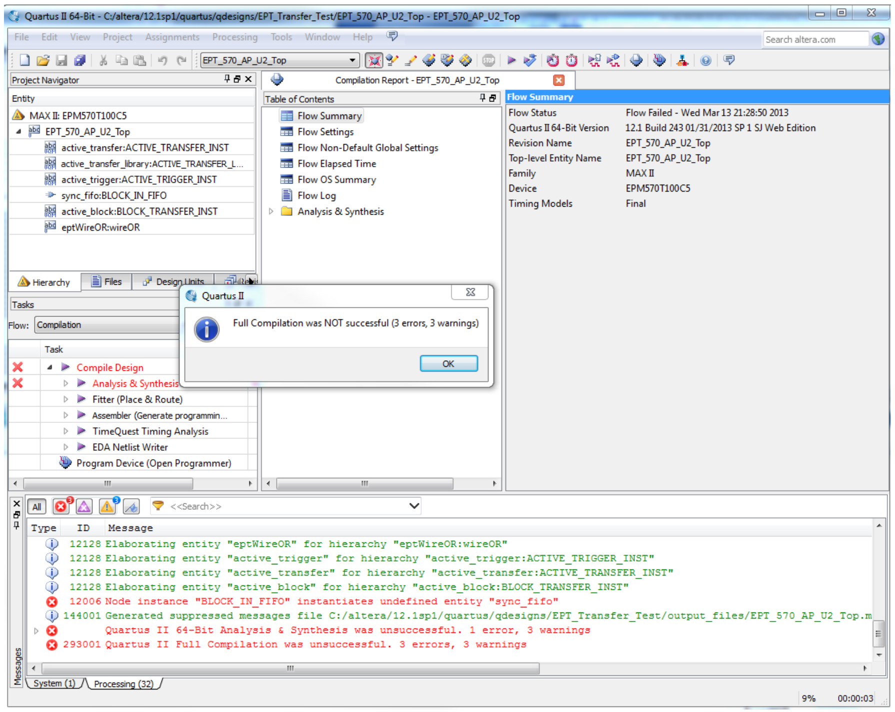
Name	Date modified	Type	Size
db	3/13/2013 9:42 PM	File folder	
incremental_db	3/13/2013 9:38 PM	File folder	
output_files	3/13/2013 9:42 PM	File folder	
simulation	3/13/2013 9:42 PM	File folder	
EPT_570_AP_U2_Top.jdi	3/13/2013 9:42 PM	JDI File	1 KB
EPT_570_AP_U2_Top.qpf	3/13/2013 9:10 PM	QPF File	2 KB
EPT_570_AP_U2_Top.qsf	3/13/2013 9:38 PM	QSF File	7 KB
EPT_570_AP_U2_Top.qsf.bak	3/13/2013 9:41 PM	BAK File	7 KB
EPT_570_AP_U2_Top.sdc	1/29/2013 10:30 PM	SDC File	4 KB

EPT_570_AP_U2_Top.sdc Date modified: 1/29/2013 10:30 PM Date created: 3/13/2013 9:41 PM
SDC File Size: 3.16 KB

Select the Start Compilation button.



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



Quartus II

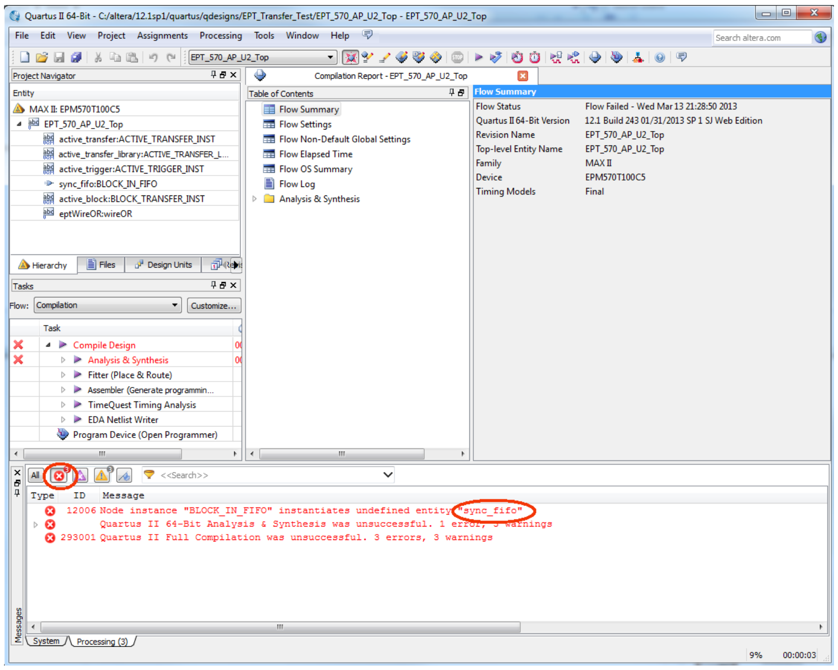
Full Compilation was NOT successful (3 errors, 3 warnings)

OK

Flow Status	Flow Failed - Wed Mar 13 21:28:50 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	EPT_570_AP_U2_Top
Top-level Entity Name	EPT_570_AP_U2_Top
Family	MAX II
Device	EPM570T100C5
Timing Models	Final

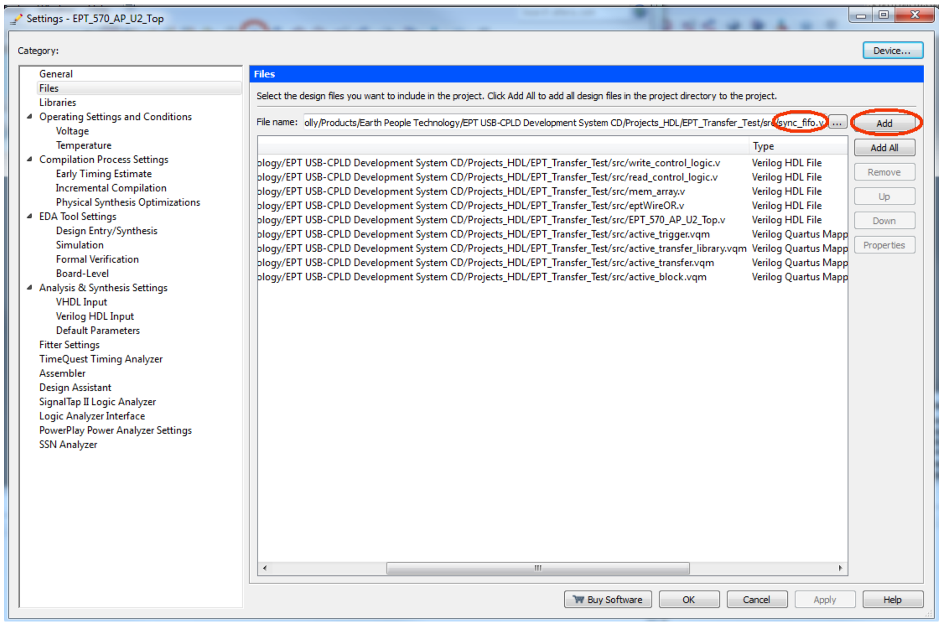
```
Type  ID     Message
      12128  Elaborating entity "eptWireOR" for hierarchy "eptWireOR:wireOR"
      12128  Elaborating entity "active_trigger" for hierarchy "active_trigger:ACTIVE_TRIGGER_INST"
      12128  Elaborating entity "active_transfer" for hierarchy "active_transfer:ACTIVE_TRANSFER_INST"
      12128  Elaborating entity "active_block" for hierarchy "active_block:BLOCK_TRANSFER_INST"
      12006  Node instance "BLOCK_IN_FIFO" instantiates undefined entity "sync_fifo"
      144001 Generated suppressed messages file C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/output_files/EPT_570_AP_U2_Top.m
             Quartus II 64-Bit Analysis & Synthesis was unsuccessful. 1 error, 3 warnings
      293001 Quartus II Full Compilation was unsuccessful. 3 errors, 3 warnings
```

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



```
Type  ID     Message
      12006  Node instance "BLOCK_IN_FIFO" instantiates undefined entity "sync_fifo"
             Quartus II 64-Bit Analysis & Synthesis was unsuccessful. 1 error, 3 warnings
      293001 Quartus II Full Compilation was unsuccessful. 3 errors, 3 warnings
```

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



Settings - EPT_570_AP_U2_Top

Category:

General
Files
Libraries
Operating Settings and Conditions
  Voltage
  Temperature
Compilation Process Settings
  Early Timing Estimate
  Incremental Compilation
  Physical Synthesis Optimizations
EDA Tool Settings
  Design Entry/Synthesis
  Simulation
  Formal Verification
  Board-Level
Analysis & Synthesis Settings
  VHDL Input
  Verilog HDL Input
  Default Parameters
Fitter Settings
TimeQuest Timing Analyzer
Assembler
Design Assistant
SignalTap II Logic Analyzer
Logic Analyzer Interface
PowerPlay Power Analyzer Settings
SSN Analyzer

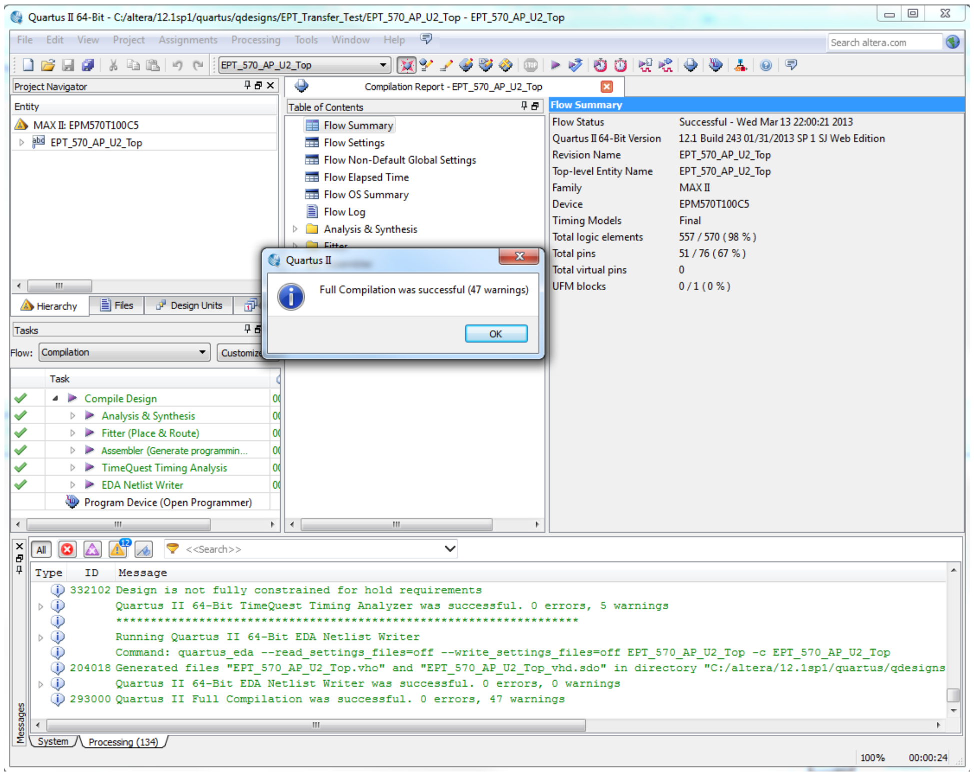
Files

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: olly/Products/Earth People Technology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/sync_fifo.v

	Type
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/write_control_logic.v	Verilog HDL File
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/read_control_logic.v	Verilog HDL File
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/mem_array.v	Verilog HDL File
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/eptWireOR.v	Verilog HDL File
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/EPT_570_AP_U2_Top.v	Verilog HDL File
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_trigger.vqm	Verilog Quartus Mapp
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_transfer_library.vqm	Verilog Quartus Mapp
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_transfer.vqm	Verilog Quartus Mapp
ology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_block.vqm	Verilog Quartus Mapp

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


Quartus II 64-Bit - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator
Compilation Report - EPT_570_AP_U2_Top
Table of Contents
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Customize
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programmin...
TimeQuest Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
All
<<Search>>
System
Processing (134)
100%
00:00:24

Entity: MAX II: EPM570T100C5 — EPT_570_AP_U2_Top

Quartus II: Full Compilation was successful (47 warnings) — OK

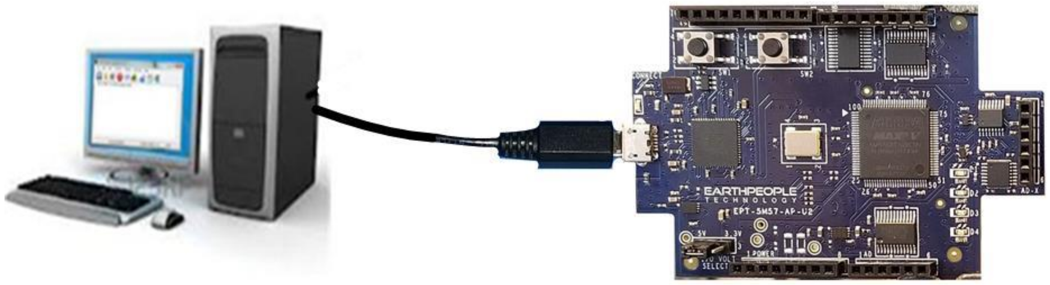
Flow Summary	
Flow Status	Successful - Wed Mar 13 22:00:21 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	EPT_570_AP_U2_Top
Top-level Entity Name	EPT_570_AP_U2_Top
Family	MAX II
Device	EPM570T100C5
Timing Models	Final
Total logic elements	557 / 570 (98 %)
Total pins	51 / 76 (67 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

```
Type  ID     Message
      332102 Design is not fully constrained for hold requirements
             Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 5 warnings
             ************************************************************************
             Running Quartus II 64-Bit EDA Netlist Writer
             Command: quartus_eda --read_settings_files=off --write_settings_files=off EPT_570_AP_U2_Top -c EPT_570_AP_U2_Top
      204018 Generated files "EPT_570_AP_U2_Top.vho" and "EPT_570_AP_U2_Top_vhd.sdo" in directory "C:/altera/12.1sp1/quartus/qdesigns
             Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
      293000 Quartus II Full Compilation was successful. 0 errors, 47 warnings
```

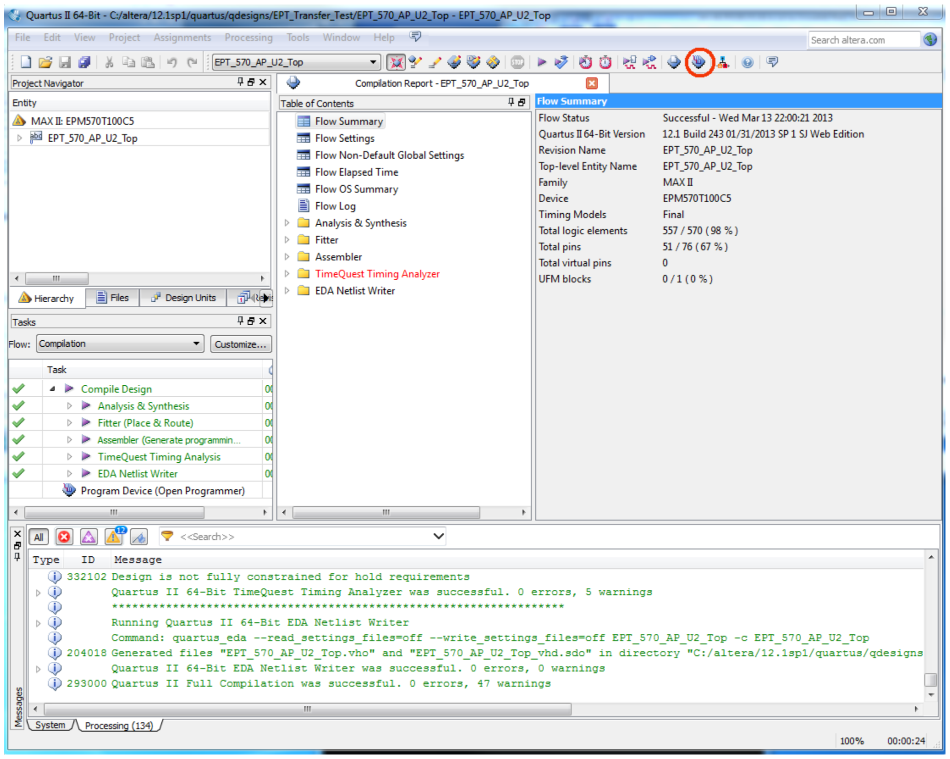
At this point the project has been successfully compiled, synthesized and a programming file has been produce. See the next section on how to program the CPLD.

4.1.2 Programming the CPLD

Programming the CPLD is quick and easy. All that is required is a standard USB cable with a Micro Type B connector on one end and the EPT_Blaster Driver DLL. Connect the UnoMax to the PC, open up Quartus Prime Lite , open the programmer tool, and click the Start button. To program the CPLD, follow the steps to install the USB Driver and the JTAG Driver Insert for Quartus Prime Lite.



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



Quartus II 64-Bit - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top

Compilation Report - EPT_570_AP_U2_Top

Table of Contents: Flow Summary, Flow Settings, Flow Non-Default Global Settings, Flow Elapsed Time, Flow OS Summary, Flow Log, Analysis & Synthesis, Fitter, Assembler, TimeQuest Timing Analyzer, EDA Netlist Writer

Flow Summary

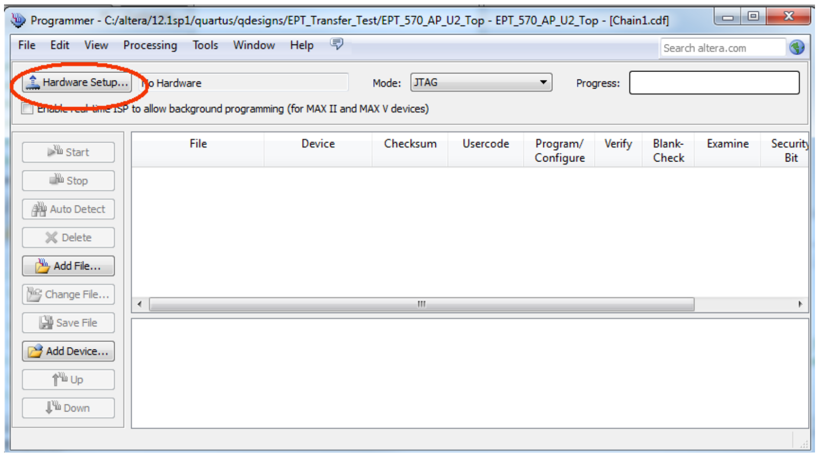
Flow Status	Successful - Wed Mar 13 22:00:21 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	EPT_570_AP_U2_Top
Top-level Entity Name	EPT_570_AP_U2_Top
Family	MAX II
Device	EPM570T100C5
Timing Models	Final
Total logic elements	557 / 570 (98 %)
Total pins	51 / 76 (67 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

Tasks — Flow: Compilation

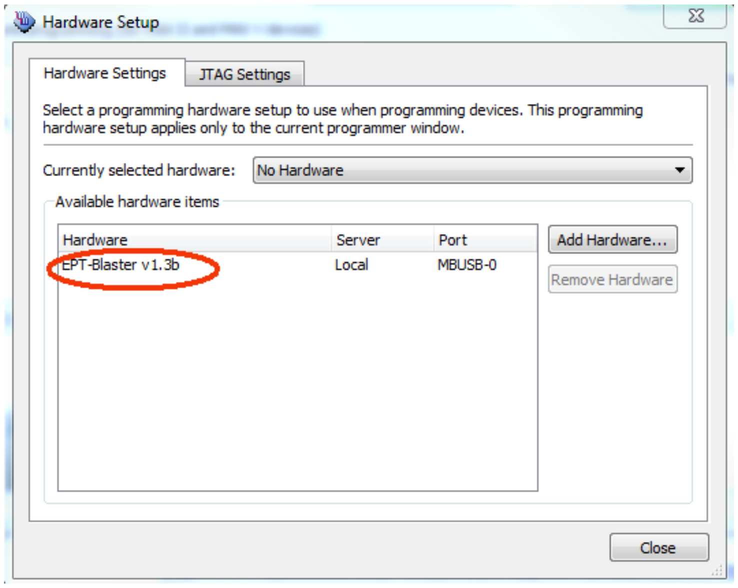
	Task
✓	Compile Design
✓	Analysis & Synthesis
✓	Fitter (Place & Route)
✓	Assembler (Generate programmin...
✓	TimeQuest Timing Analysis
✓	EDA Netlist Writer
	Program Device (Open Programmer)

```
Type  ID     Message
      332102 Design is not fully constrained for hold requirements
             Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 5 warnings
             ******************************************************************
             Running Quartus II 64-Bit EDA Netlist Writer
             Command: quartus_eda --read_settings_files=off --write_settings_files=off EPT_570_AP_U2_Top -c EPT_570_AP_U2_Top
      204018 Generated files "EPT_570_AP_U2_Top.vho" and "EPT_570_AP_U2_Top_vhd.sdo" in directory "C:/altera/12.1sp1/quartus/qdesigns
             Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
      293000 Quartus II Full Compilation was successful. 0 errors, 47 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:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain1.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 (for MAX II and MAX V devices)
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

The Hardware Setup Window will open. In the “Available hardware items”, double click on “EPT-Blaster v1.6b”.



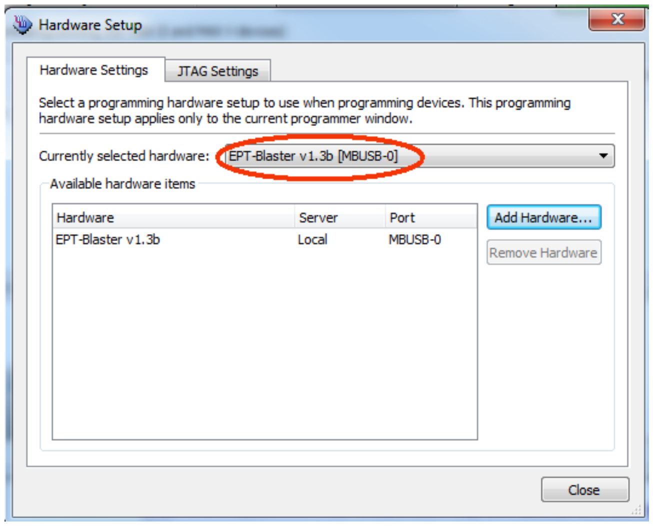
Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: No Hardware

Available hardware items

Hardware	Server	Port
EPT-Blaster v1.3b	Local	MBUSB-0

If you successfully double clicked, the “Currently selected hardware:” dropdown box will show the “EPT-Blaster v1.6b”.

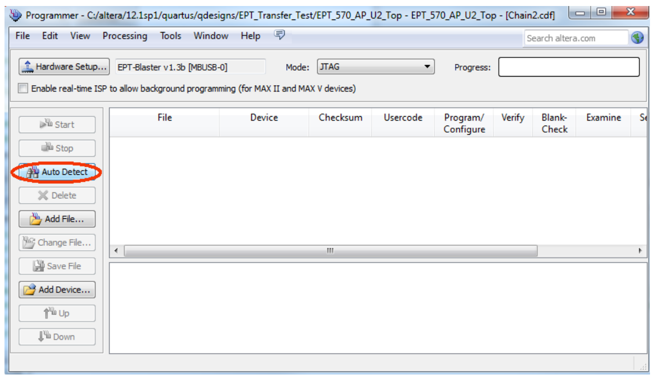


Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: EPT-Blaster v1.3b [MBUSB-0]

Hardware	Server	Port
EPT-Blaster v1.3b	Local	MBUSB-0

Click on the Auto-Detect button. This will verify that the EPT-Blaster driver can connect with the 5M570 device.

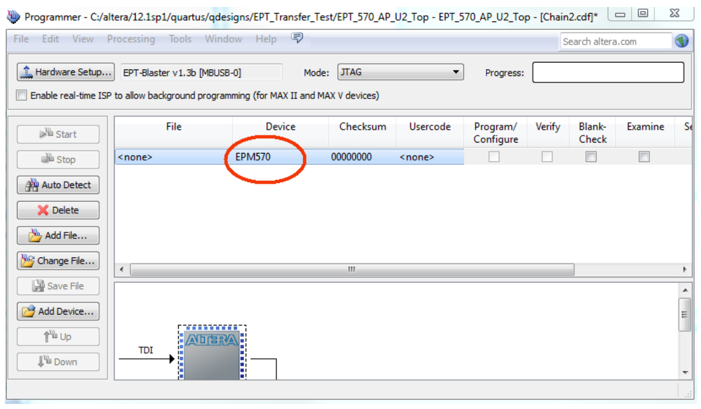


Programmer - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain2.cdf]

EPT-Blaster v1.3b [MBUSB-0] Mode: JTAG

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

Select the 5M570 under “Device”.



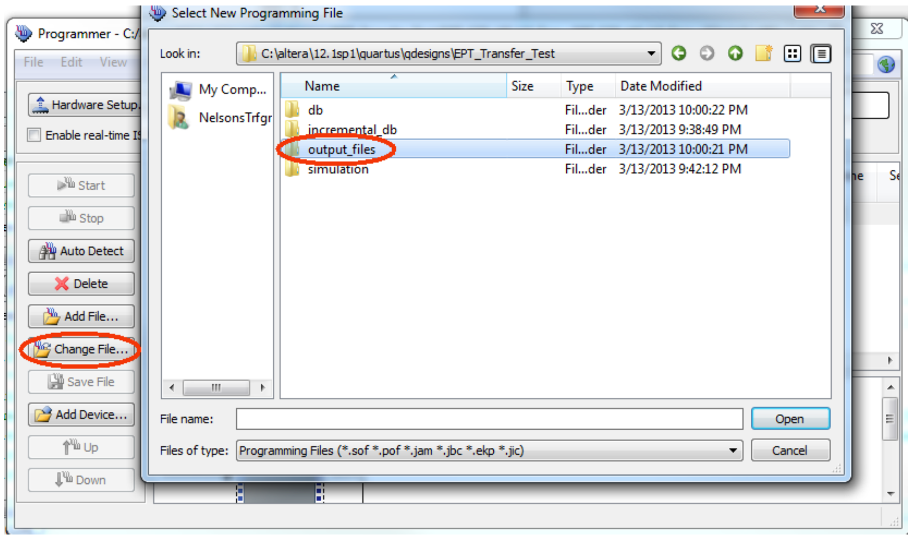
Programmer - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain2.cdf]*

EPT-Blaster v1.3b [MBUSB-0] Mode: JTAG

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

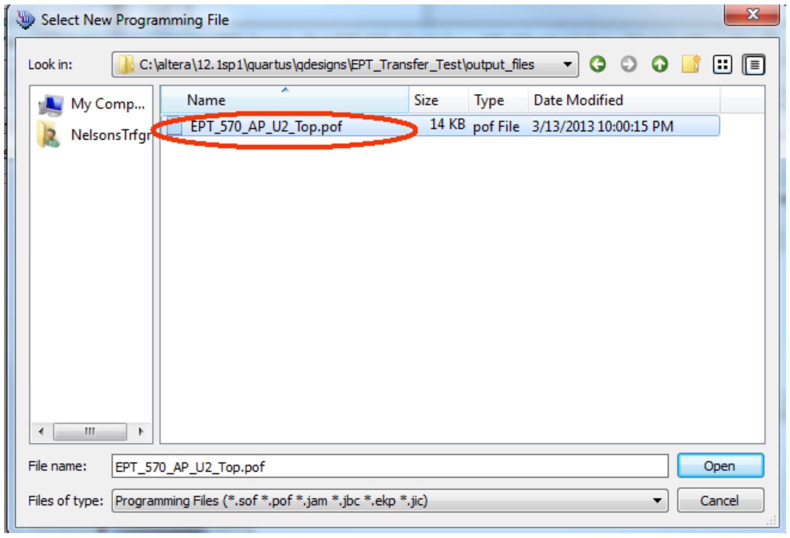
File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
<none>	EPM570	00000000	<none>	☐	☐	☐	☐

Click on the “Change File” button and browse to the output_files folder.



Name	Size	Type	Date Modified
db		Fil...der	3/13/2013 10:00:22 PM
incremental_db		Fil...der	3/13/2013 9:38:49 PM
output_files		Fil...der	3/13/2013 10:00:21 PM
simulation		Fil...der	3/13/2013 9:42:12 PM

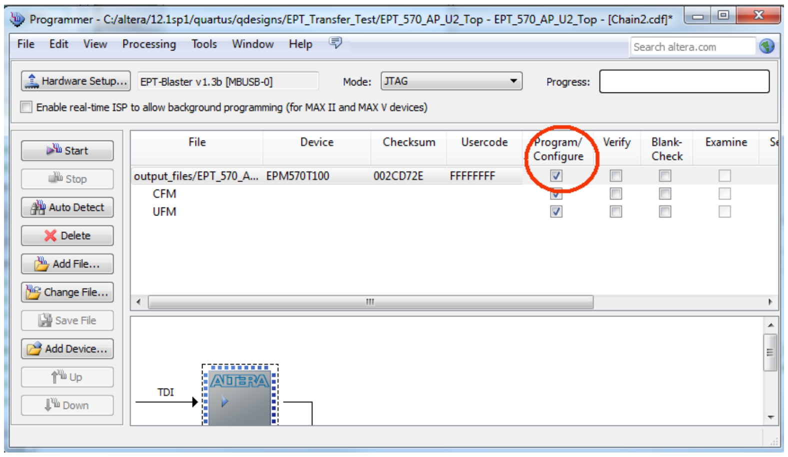
Click on the EPT_5M57_AP_U2_Top.pof file to select it.


Select New Programming File
Look in:
C:\altera\12.1sp1\quartus\qdesigns\EPT_Transfer_Test\output_files
My Comp...
NelsonsTrfgr
Name
Size
Type
Date Modified
File name:
EPT_570_AP_U2_Top.pof
Files of type:
Programming Files (*.sof *.pof *.jam *.jbc *.ekp *.jic)
Open
Cancel

Name	Size	Type	Date Modified
EPT_570_AP_U2_Top.pof	14 KB	pof File	3/13/2013 10:00:15 PM

Click the Open button in the lower right corner.

Next, selet the checkbox under the “Program/Configure” of the Programmer Tool. The checkboxes for the CFM and UFM will be selected automatically.



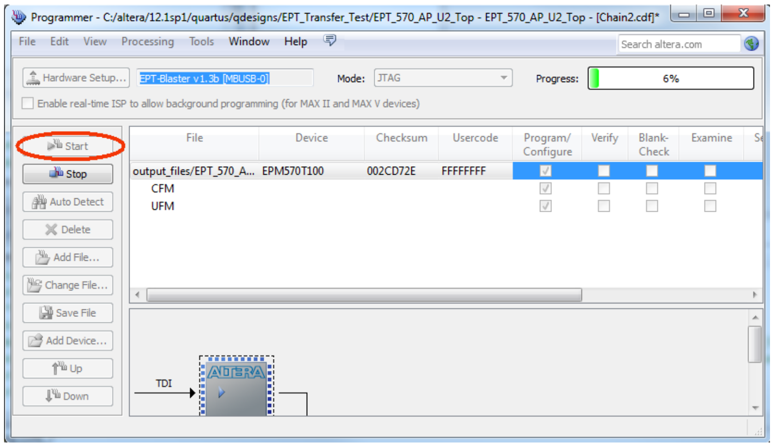
Programmer - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain2.cdf]*

EPT-Blaster v1.3b [MBUSB-0] Mode: JTAG Progress:

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/EPT_570_A...	EPM570T100	002CD72E	FFFFFFFF	✓			
CFM				✓			
UFM				✓			

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



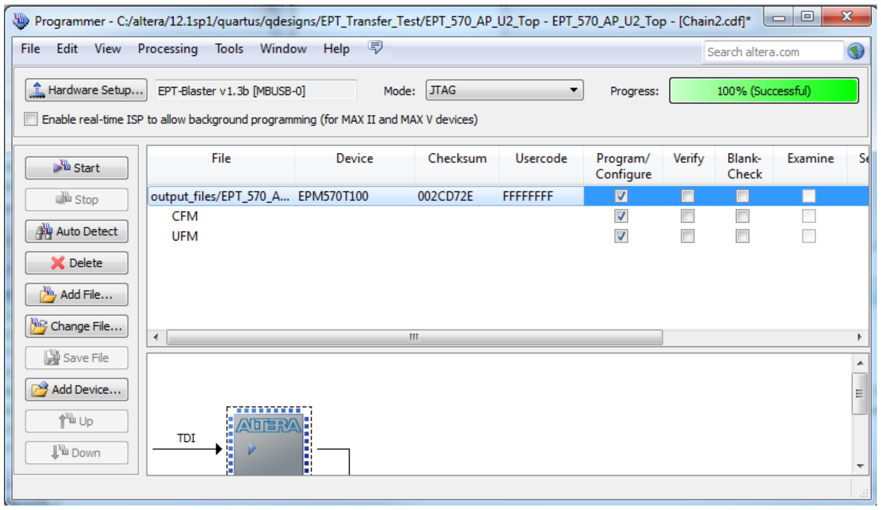
Programmer - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain2.cdf]*

EPT-Blaster v1.3b [MBUSB-0] Mode: JTAG Progress: 6%

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/EPT_570_A...	EPM570T100	002CD72E	FFFFFFFF	✓			
CFM				✓			
UFM				✓			

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



Programmer - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top - [Chain2.cdf]*

EPT-Blaster v1.3b [MBUSB-0]   Mode: JTAG   Progress: 100% (Successful)

Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/EPT_570_A...	EPM570T100	002CD72E	FFFFFFFF	✓	☐	☐	☐
CFM				✓	☐	☐	☐
UFM				✓	☐	☐	☐

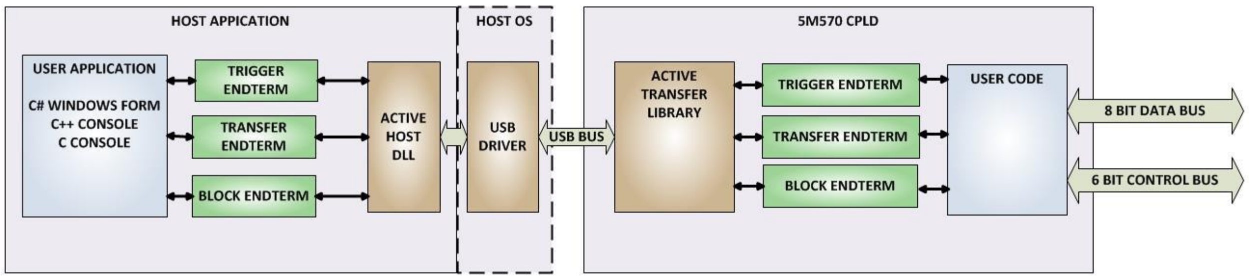
At this point, the UnoMax is programmed and ready for use. To test that the CPLD is properly programmed, bring up the Active Host Test Tool. Click on one of the LED's and verify that the LED selected lights up. Press one of the switches on the board and ensure that the switch is captured on the Active Host Test Tool. Now you are ready to write some code to transfer data between CPLD and PC.

5 Active Host Application

The Active Host SDK is provided as a dll which easily interfaces to application software written in C#, C++ or C. It runs on the PC and provides transparent connection from PC application code through the USB driver to the user CPLD code. The user code connects to "Endterms" in the Active Host dll. 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 CPLD. The Trigger Endterms are used as "switches". The user code can set a Trigger bit in the CPLD and cause an event to occur. The Transfer Endterm sends one byte to the CPLD. 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 CPLD. 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 dll.

5.2 Transfer(Byte) EndTerm

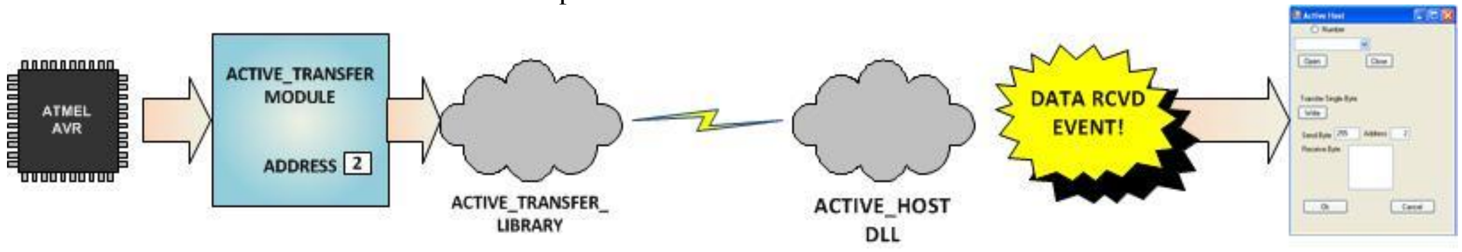
The Transfer EndTerm is a software component that provides a direct path from the users application to the commensurate Transfer EndTerm in the CPLD. It is used to transfer a byte to and from the CPLD. Eight separate Transfer EndTerm modules can be instantiated in the CPLD. Each module is addressed by 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 CPLD. Receiving a byte is just as easy, a callback function is registered at initialization. When the CPLD 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 CPLD. The Block EndTerm is used to transfer a complete block to the CPLD. Block size is limited to 1 to 256 bytes. Eight separate Block EndTerm modules can be instantiated in the CPLD. 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 CPLD. Receiving a block is just as easy, a callback function is registered at initialization. When the CPLD 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 DLL

The Active_Host DLL is designed to transfer data from the CPLD 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 CPLD. This method of automatically moving data from the user code Endterm in the CPLD makes the data transfer transparent.



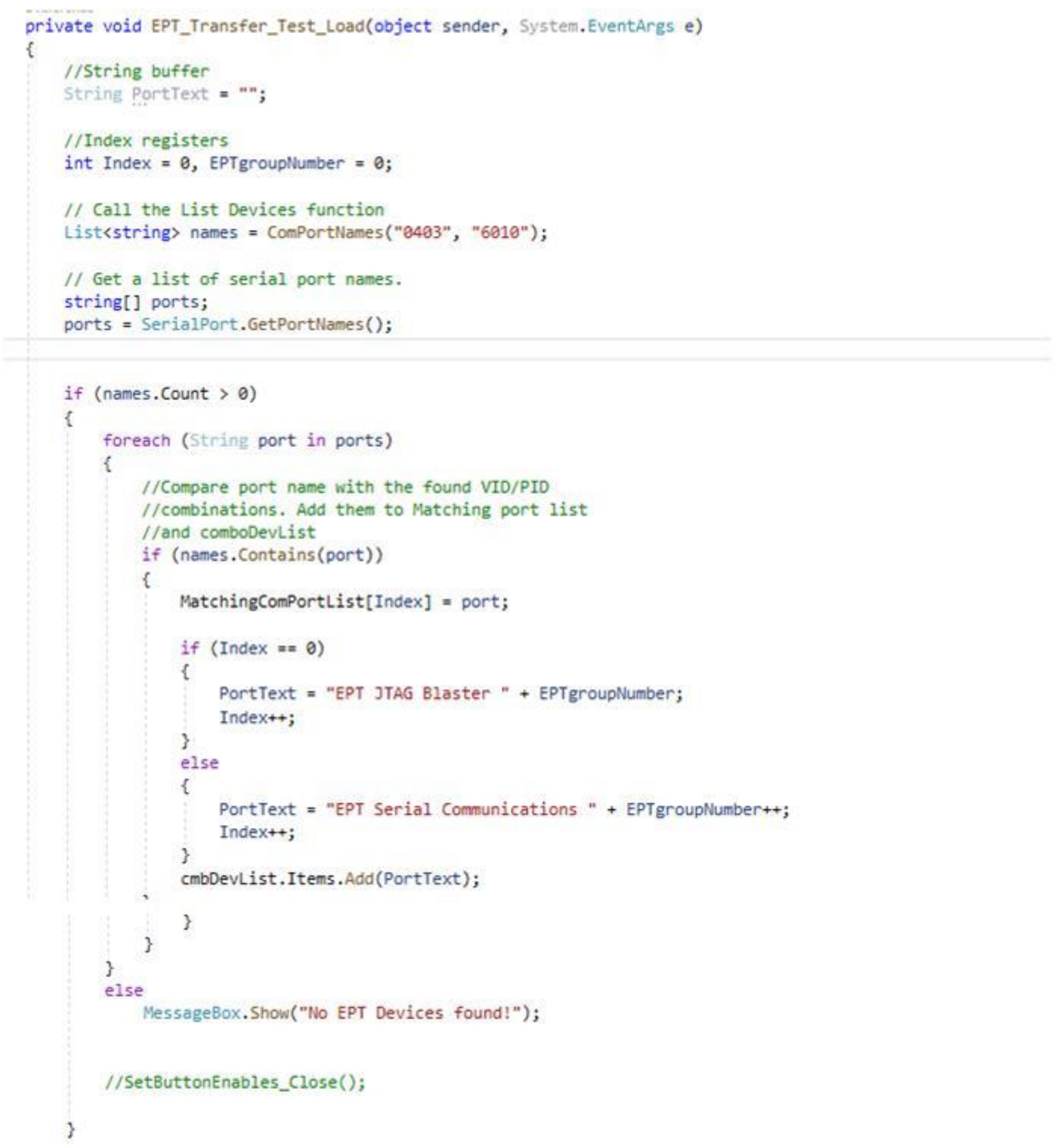
The data seamlessly appears in Host PC memory from the UnoMax. The user code will direct the data to a control such as a textbox on a Windows Form. The transparent receive transfer path is made possible by a Callback mechanism in the Active Host dll. The dll calls a registered callback function in the 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 CPLD by use of functions contained in the Active Host dll. The functions to access the CPLD are:

- EPT_AH_CloseDeviceByIndex()
- EPT_AH_CloseDeviceByName()
- 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. 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 ListDevices(). Use the function List Devices() to detect all EPT devices connected to the PC.



```
private void EPT_Transfer_Test_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();

    }
```

The ListDevices() function calls the

```
ports = SerialPort.GetPortNames();
```

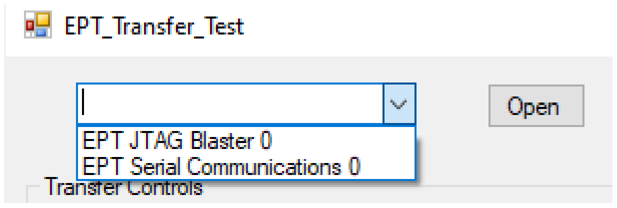
to determine the Serial devices attached to the PC. Next,

```
if (names.Contains(port))
```

is called inside a for loop to return the ASCII name of each Serial device attached to the PC. It will automatically populate the combo box, cmbDevList with all the EPT devices it finds.

```
cmbDevList.Items.Add(PortText);
```

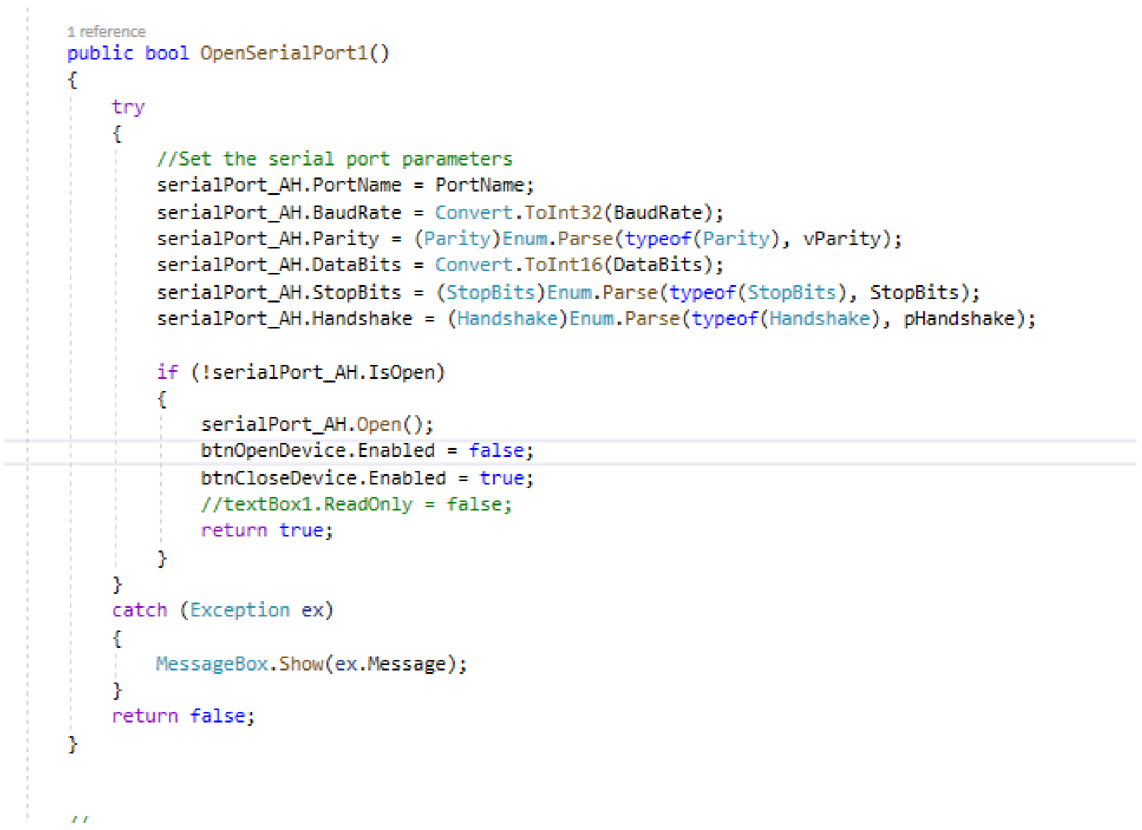
The user will select the device from the drop down combo box. This can be seen when the Windows Form is opened and the cmbDevList combo box is populated with all the devices. The selected device will be stored as an index number in the variable device_index.



In order to select the device, the user will click on the “Open” button which calls the

```
OpenSerialPort1()
```

function. The device_index is passed into the function. If the function is successful, the device name is displayed in the label, labelDeviceCnt. Next, 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;
}
```

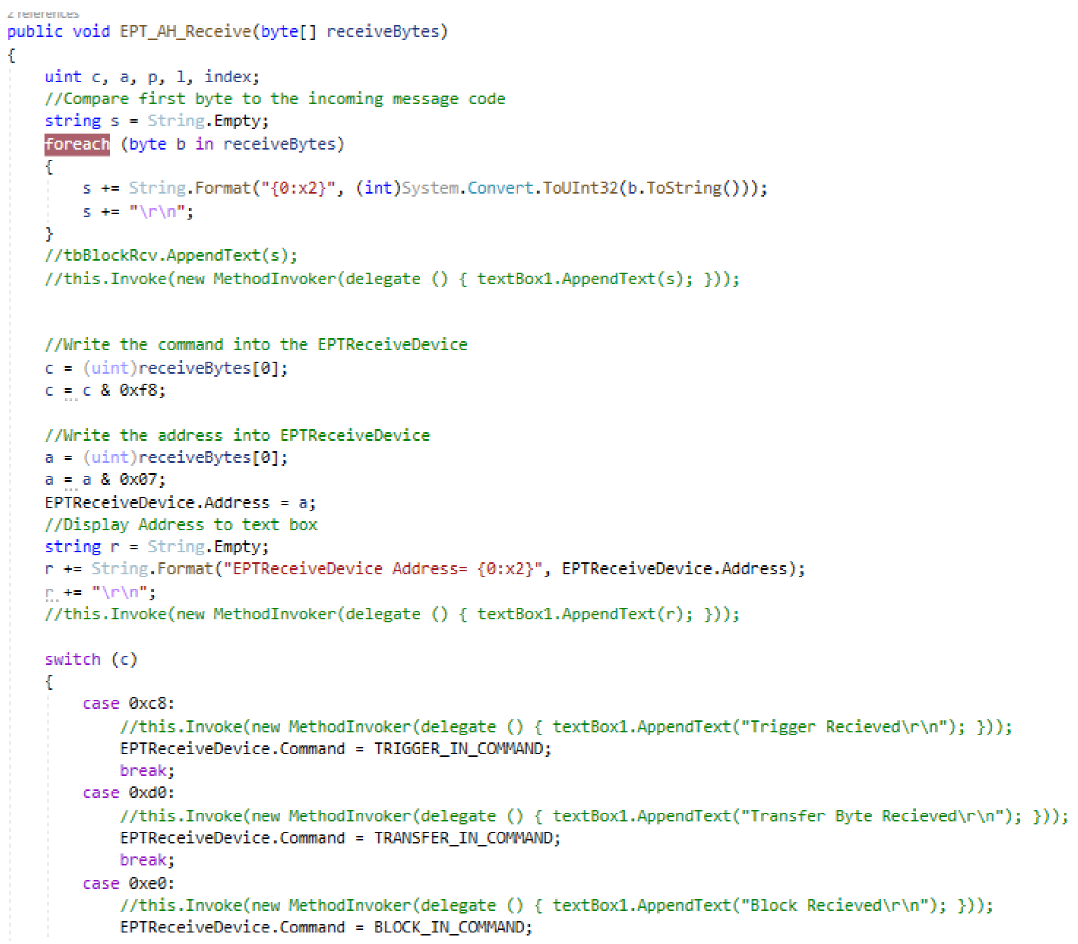
5.4.2 Active Host Triggers

The user application can send a trigger to the CPLD 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 CPLD.

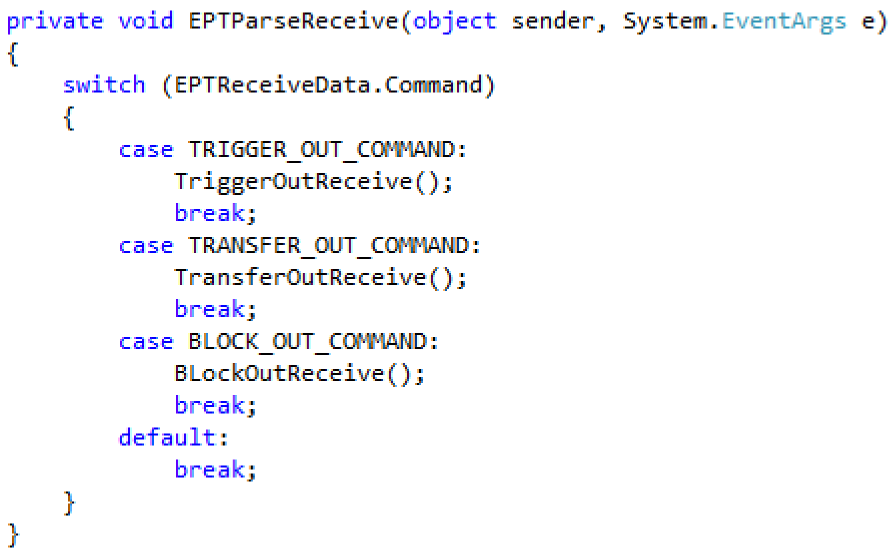
```
private void btnTrigger1_Click(object sender, EventArgs e)
{
    EPT_AH_SendTrigger((char) 1);
}
```

To detect a trigger from the CPLD, the user application must subscribe to the event created when the incoming trigger has arrived at the Read Callback function. The Read Callback must store the incoming trigger in a local variable. A switch statement is used to decode which event should be called to handle the incoming received data.

- TRIGGER_IN
- TRANSFER_IN
- BLOCK_IN



```
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;
```



```
private void EPTParseReceive(object sender, System.EventArgs e)
{
    switch (EPTReceiveData.Command)
    {
        case TRIGGER_OUT_COMMAND:
            TriggerOutReceive();
            break;
        case TRANSFER_OUT_COMMAND:
            TransferOutReceive();
            break;
        case BLOCK_OUT_COMMAND:
            BLockOutReceive();
            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;
    }
}
```

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.

5.4.3 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 CPLD. Up to eight modules can be instantiated in the user code on the CPLD. Each module has its own address.

```
private void btnWriteByte_Click(object sender, EventArgs e)
{
    int ibyte, address_to_device;
    ibyte = Convert.ToInt32(tbNumBytes.Text);
    address_to_device = Convert.ToInt32(tbAddress.Text);
    EPT_AH_SendByte(address_to_device, (char)ibyte);
}
```

Use the function EPT_AH_SendByte() to send a byte the selected module. First, open the EPT device to be used with `OpenSerialPort1()` . 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 CPLD.

To transfer data from the CPLD 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 dll, 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(object sender, System.EventArgs e)
{
    switch (EPTReceiveData.Command)
    {
        case TRIGGER_OUT_COMMAND:
            TriggerOutReceive();
            break;
        case TRANSFER_OUT_COMMAND:
            TransferOutReceive();
            break;
        case BLOCK_OUT_COMMAND:
            BLockOutReceive();
            break;
        default:
            break;
    }
}
```

The EventHandler function EPTParseReceive() is called by the Read 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.

5.4.4 Active Host Block Transfers

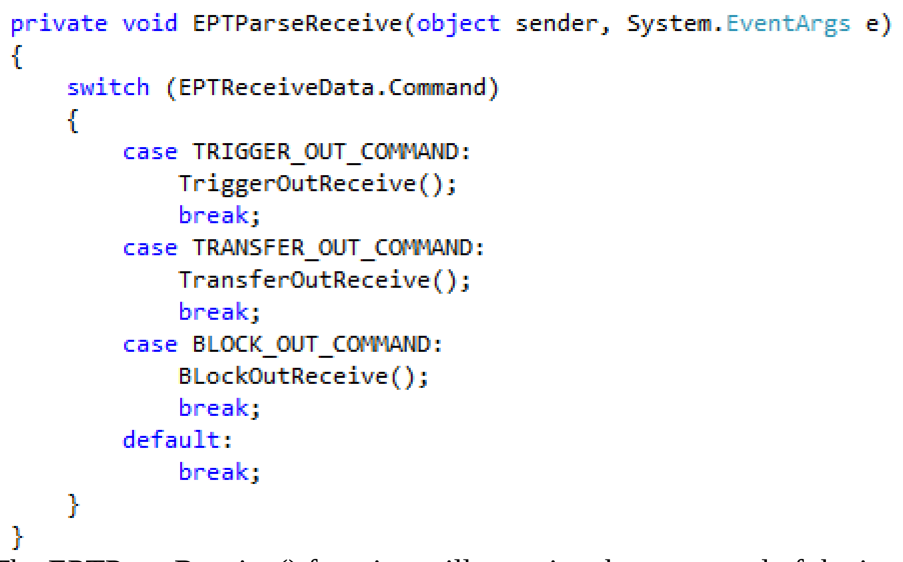
The Active Host Block Transfer is designed to transfer blocks of data between Host and CPLD and vice versa through the Block EndTerm. This allows buffers of data to be transferred with a minimal amount of code. The Active Host Block module (in the User Code) is addressable, so up to eight individual 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 EPT_AH_OpenDeviceByIndex(). Next, use the EPT_AH_SendBlock() function to send the block. Add the address of the 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 CPLD.

```
public unsafe void BlockCompare(object data)
{
    int BlockAddress = (int)data;
    byte[] cBuf = new Byte[device[BlockAddress].Length];

    if ((device[BlockAddress].Repititions > 0) &
        !device[BlockAddress].TransferPending & !BlockTransferStop)
    {
        device[BlockAddress].TransferPending = true;
        Buffer.BlockCopy(block_8_in_payload, 0, cBuf, 0,
            device[BlockAddress].Length);
        fixed (byte* pBuf = cBuf)
        {
          EPT_AH_SendBlock(device[BlockAddress].Address,
                           (void*)pBuf, (uint)device[BlockAddress].Length);
        }
        Thread.Sleep(1);
        EPT_AH_SendTransferControlByte((char)2, (char)2);
        Thread.Sleep(1);
        EPT_AH_SendTrigger((char)128);
        Thread.Sleep(1);
        EPT_AH_SendTransferControlByte((char)2, (char)0);

        if (BlockTransferInfinite)
            device[BlockAddress].Repititions = 1;
        else
            device[BlockAddress].Repititions--;
    }
}
```

To receive a block transfer from the CPLD Device, a polling technique is used by the Active Host dll. 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 dll. 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(object sender, System.EventArgs e)
{
    switch (EPTReceiveData.Command)
    {
        case TRIGGER_OUT_COMMAND:
            TriggerOutReceive();
            break;
        case TRANSFER_OUT_COMMAND:
            TransferOutReceive();
            break;
        case BLOCK_OUT_COMMAND:
            BLockOutReceive();
            break;
        default:
            break;
    }
}
```

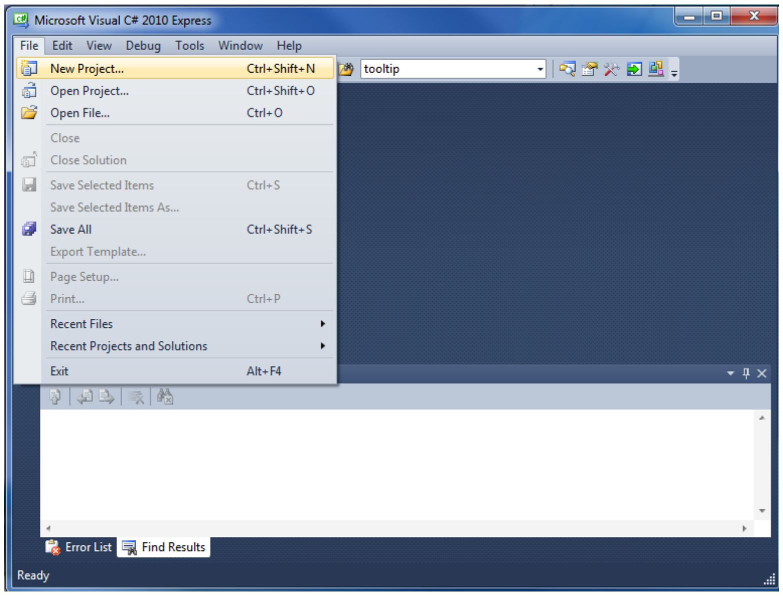
The EPTParseReceive() function will examine the command of the incoming byte transfer and determine which receive function to call.

```
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);
        }
    }
}
```

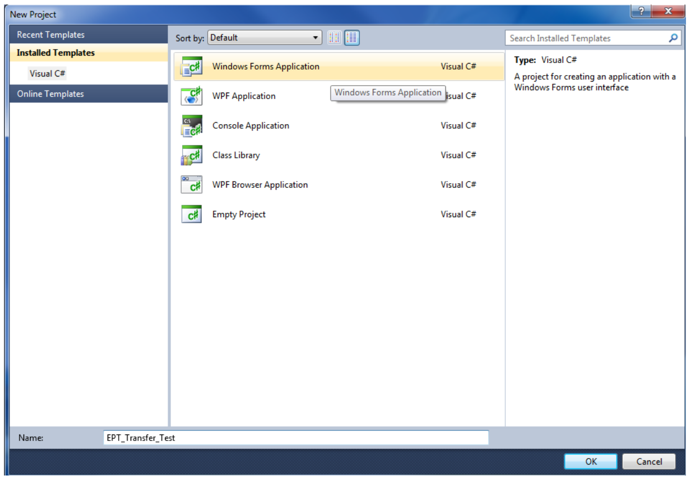
For our example project, the Receive_Block_In() function writes the Transfer block received to a text block. Assembling, Building, and Executing a .NET Project on the PC The Active Host Application DLL is used to build a custom standalone executable on the PC that can perform Triggers and Transfer data to/from the UnoMax. A standalone project can be range from a simple program to display and send data from the user to/from the CPLD. Or it can more complex to include receiving data, processing it, and start or end a process on the CPLD. 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, such as events and controls, will be explained as the explanation progresses. Any language can be used with the Active Host Application DLL.

5.5 Creating a Project

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

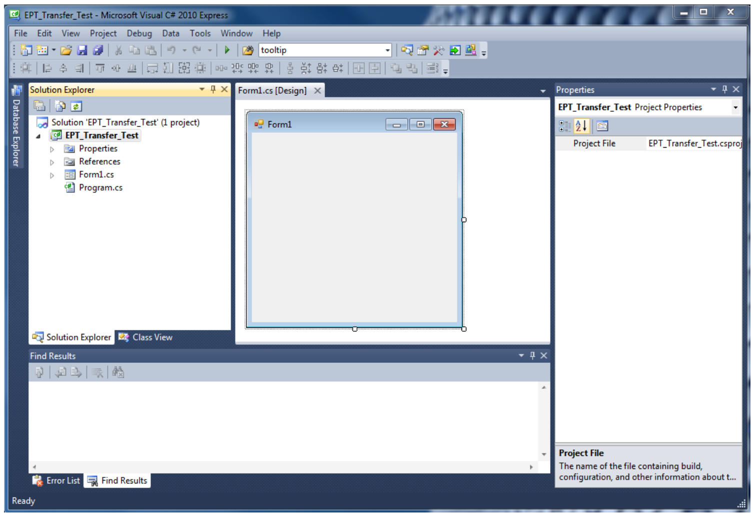

Microsoft Visual C# 2010 Express
File Edit View Debug Tools Window Help
New Project... Ctrl+Shift+N
Open Project... Ctrl+Shift+O
Open File... Ctrl+O
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects and Solutions
Exit Alt+F4
tooltip
Error List
Find Results
Ready

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

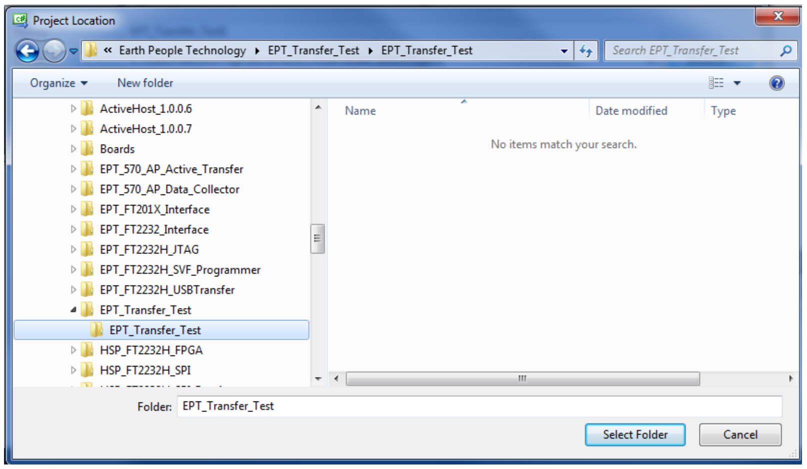

New Project
Recent Templates
Installed Templates
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Windows Forms Application
Console Application Visual C#
Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Type: Visual C#
A project for creating an application with a Windows Forms user interface
OK
Cancel

Name: EPT_Transfer_Test

The project creation is complete.

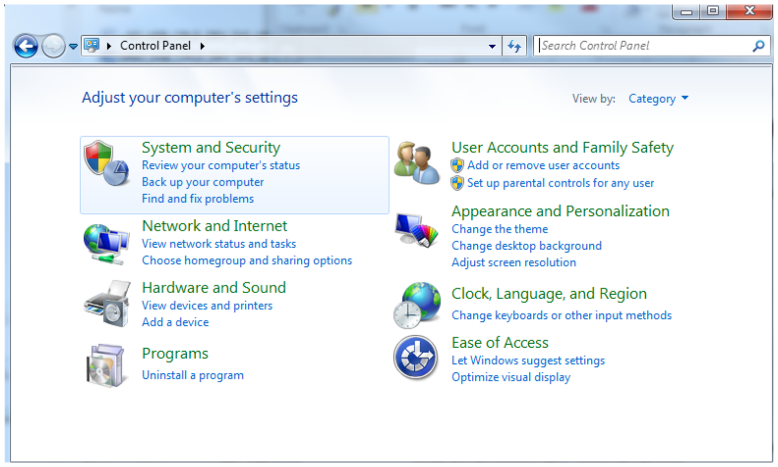

EPT_Transfer_Test - Microsoft Visual C# 2010 Express
File Edit View Project Debug Data Tools Window Help
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Form1.cs [Design]
Form1
Properties
EPT_Transfer_Test Project Properties
Project 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_Test folder. The default location is c:\Users\<Users Name>\documents\visual studio 2010\Projects.



5.5.1 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



Control Panel ▸

Search Control Panel

Adjust your computer's settings

View by: Category ▾

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

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

Hardware and Sound
View devices and printers
Add a device

Programs
Uninstall a program

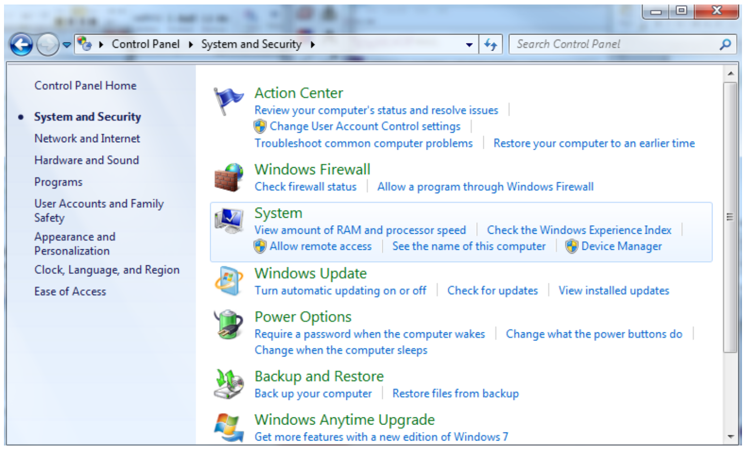
User Accounts and Family Safety
Add or remove user accounts
Set up parental controls for any user

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

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

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

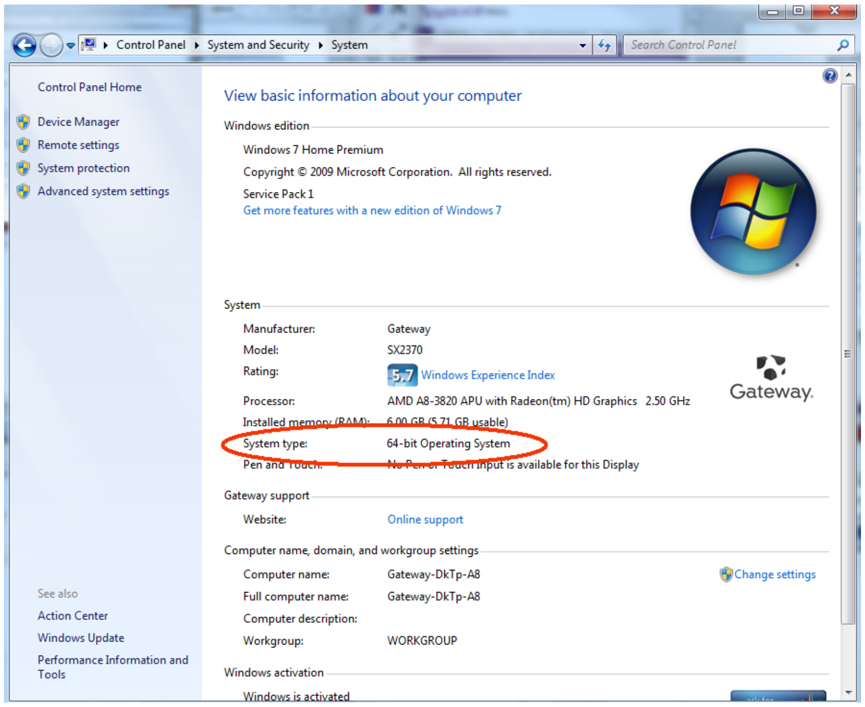
Windows Update
Turn automatic updating on or off | Check for updates | View installed updates

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

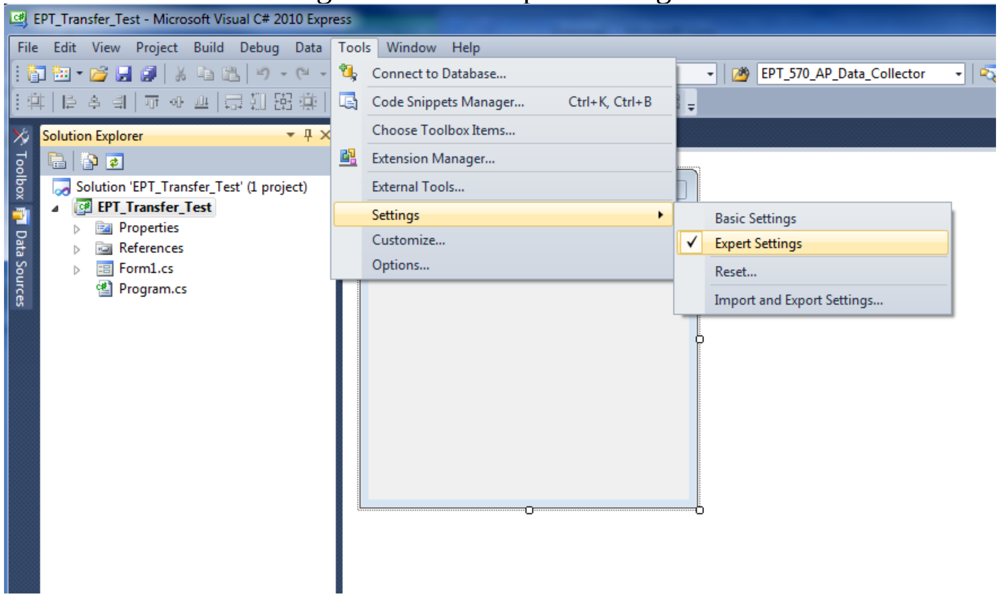
Change settings

See also
Action Center
Windows Update
Performance Information and Tools

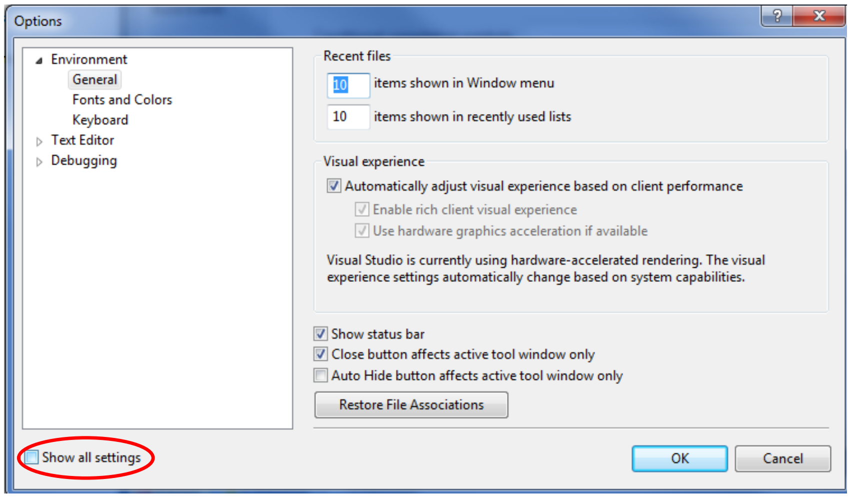
Windows activation

Windows is activated

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



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.

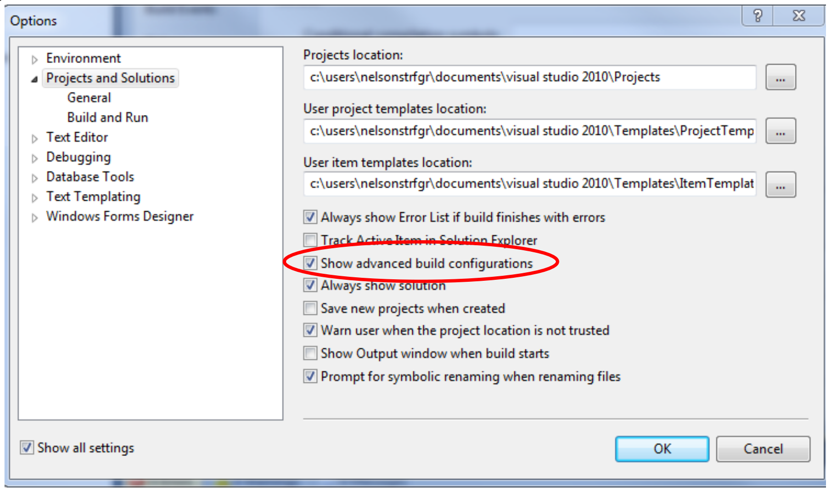
[✓] Show status bar

[✓] 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

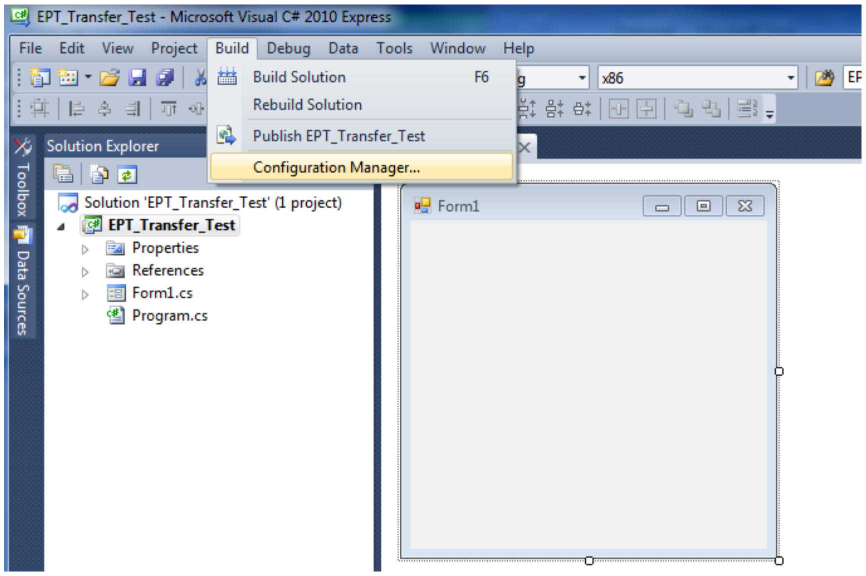
[✓] Warn user when the project location is not trusted

[ ] Show Output window when build starts

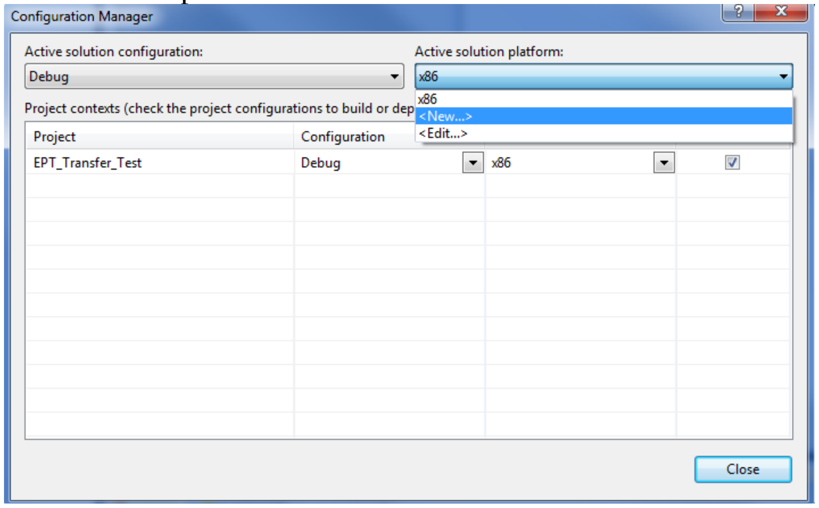
[✓] Prompt for symbolic renaming when renaming files

[✓] Show all settings

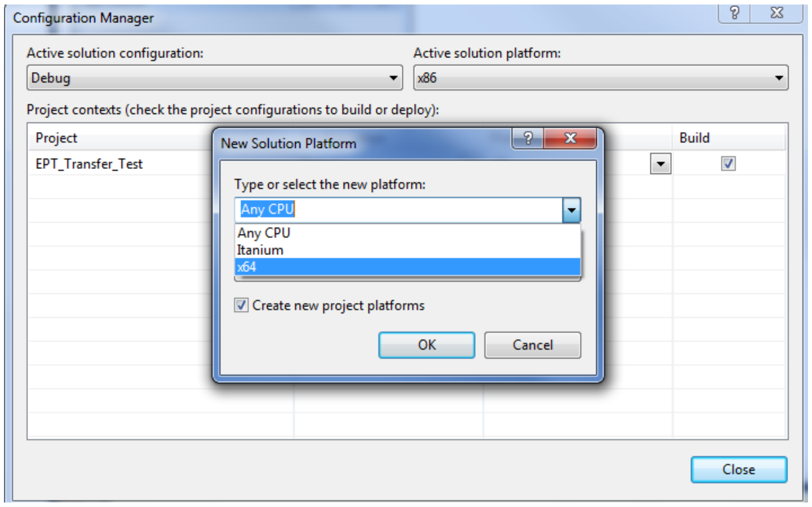
Go to Build->Configuration Manager.


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...
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Form1
Toolbox
Data Sources
x86

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	Build
EPT_Transfer_Test	✓

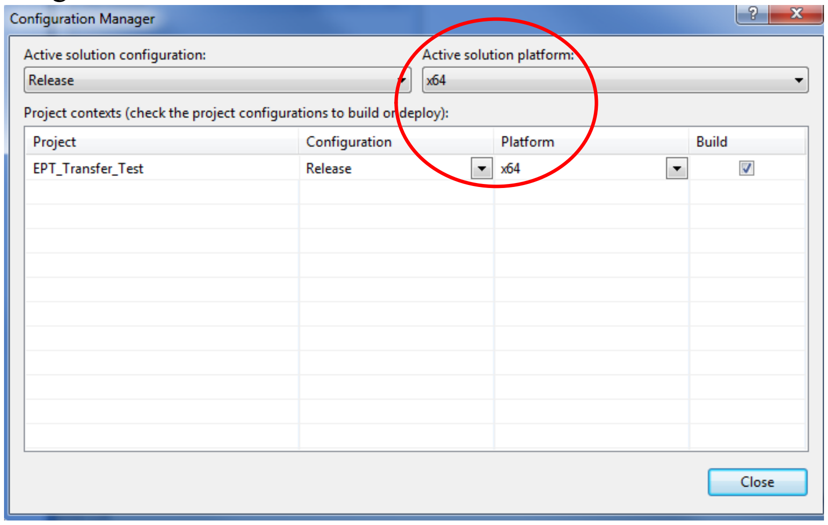
New Solution Platform

Type or select the new platform: Any CPU

Any CPU
Itanium
x64

✓ Create new project platforms

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


Configuration Manager

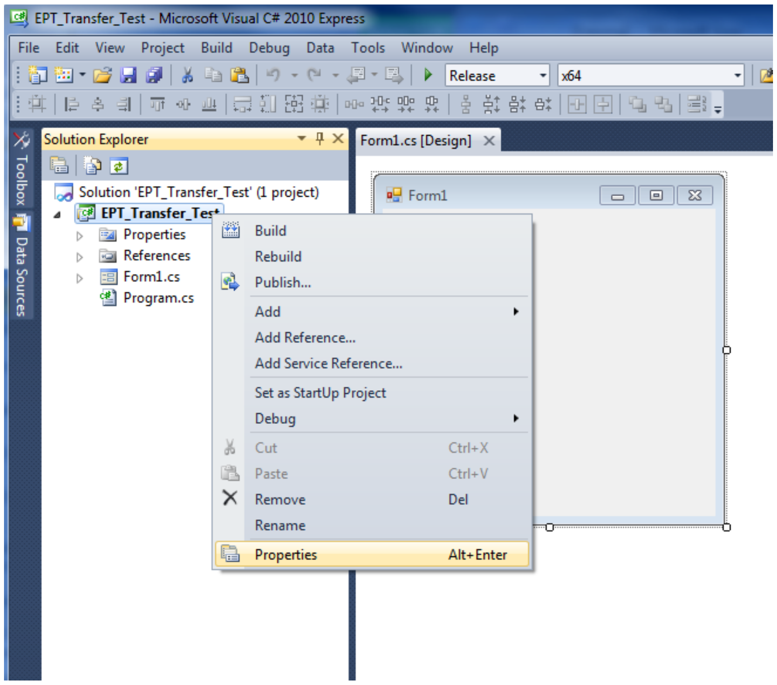
Active solution configuration: Release

Active solution platform: x64

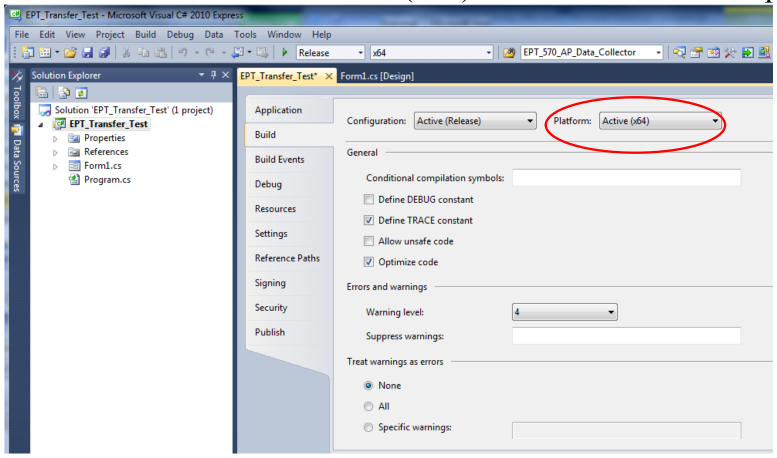
Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build
EPT_Transfer_Test	Release	x64	✓

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.



Release | x64 | EPT_570_AP_Data_Collector

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

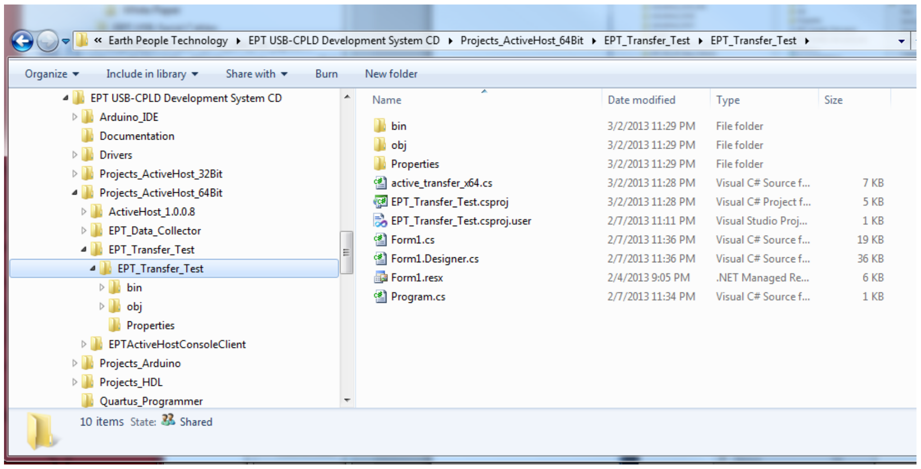
○ All

○ 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.

5.6 Assembling Files into the Project

Locate the EPT USB-CPLD Development System CD installed on your PC. Browse to the EPT_Transfer_Test folder where the Project files reside (choose either the 32 bit or 64 bit version, depending on whether your OS is 32 or 64 bit), copy the*.cs files, and install them in the top level folder of your EPT_Transfer_Test project.



« Earth People Technology ▸ EPT USB-CPLD Development System CD ▸ Projects_ActiveHost_64Bit ▸ EPT_Transfer_Test ▸ EPT_Transfer_Test ▸

```
EPT USB-CPLD Development System CD
  Arduino_IDE
  Documentation
  Drivers
  Projects_ActiveHost_32Bit
  Projects_ActiveHost_64Bit
    ActiveHost_1.0.0.8
    EPT_Data_Collector
    EPT_Transfer_Test
      EPT_Transfer_Test
        bin
        obj
        Properties
    EPTActiveHostConsoleClient
  Projects_Arduino
  Projects_HDL
  Quartus_Programmer
```

Name	Date modified	Type	Size
bin	3/2/2013 11:29 PM	File folder	
obj	3/2/2013 11:29 PM	File folder	
Properties	3/2/2013 11:29 PM	File folder	
active_transfer_x64.cs	3/2/2013 11:28 PM	Visual C# Source f...	7 KB
EPT_Transfer_Test.csproj	3/2/2013 11:28 PM	Visual C# Project f...	5 KB
EPT_Transfer_Test.csproj.user	2/7/2013 11:11 PM	Visual Studio Proj...	1 KB
Form1.cs	2/7/2013 11:36 PM	Visual C# Source f...	19 KB
Form1.Designer.cs	2/7/2013 11:36 PM	Visual C# Source f...	36 KB
Form1.resx	2/4/2013 9:05 PM	.NET Managed Re...	6 KB
Program.cs	2/7/2013 11:34 PM	Visual C# Source f...	1 KB

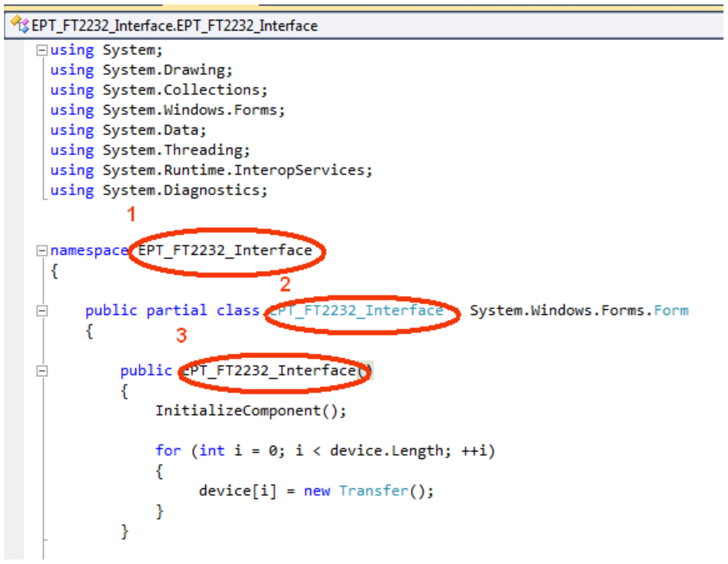
10 items State: Shared

5.6.1 Changing Project Name

NOTE

If you named your project something other than EPT_Transfer_Test, 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:

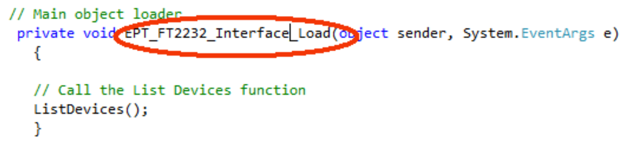
1. Change namespace of Form1.cs to new project name.
2. Change class of Form1.cs to new project name.
3. Change constructor of Form1.cs to new project name.



EPT_FT2232_Interface.EPT_FT2232_Interface

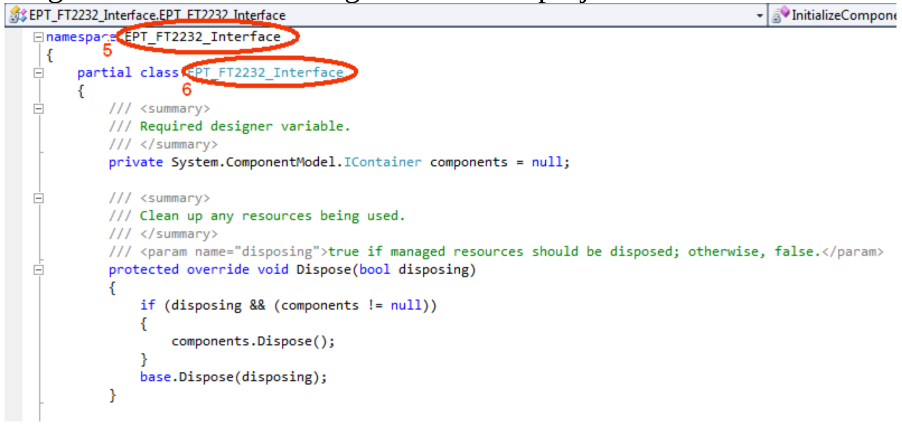
```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace EPT_FT2232_Interface
{
    public partial class EPT_FT2232_Interface : System.Windows.Forms.Form
    {
        public EPT_FT2232_Interface()
        {
            InitializeComponent();

            for (int i = 0; i < device.Length; ++i)
            {
                device[i] = new Transfer();
            }
        }
```

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



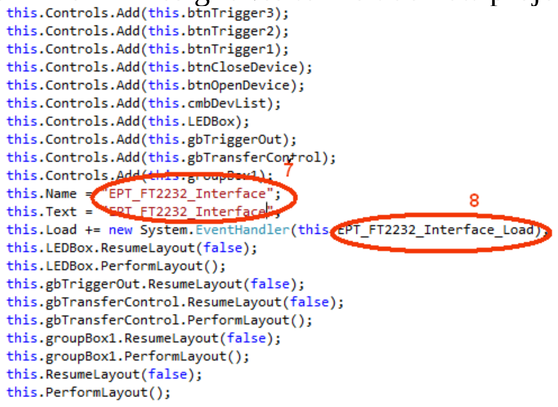
```
// Main object loader
 private void EPT_FT2232_Interface_Load(object sender, System.EventArgs e)
   {

   // 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.



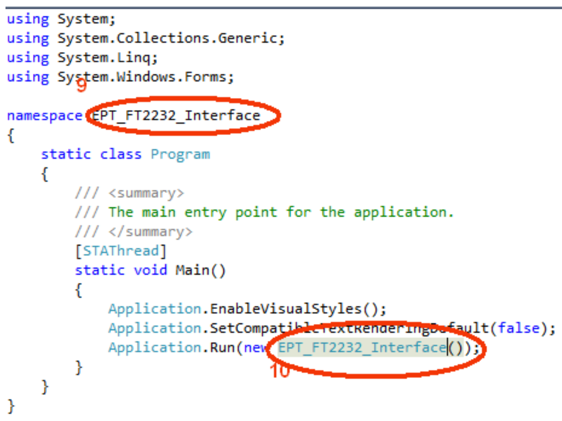
```
namespace EPT_FT2232_Interface
{
    partial class EPT_FT2232_Interface
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        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.btnTrigger3);
this.Controls.Add(this.btnTrigger2);
this.Controls.Add(this.btnTrigger1);
this.Controls.Add(this.btnCloseDevice);
this.Controls.Add(this.btnOpenDevice);
this.Controls.Add(this.cmbDevList);
this.Controls.Add(this.LEDBox);
this.Controls.Add(this.gbTriggerOut);
this.Controls.Add(this.gbTransferControl);
this.Controls.Add(this.groupBox1);
this.Name = "EPT_FT2232_Interface";
this.Text = "EPT_FT2232_Interface";
this.Load += new System.EventHandler(this.EPT_FT2232_Interface_Load);
this.LEDBox.ResumeLayout(false);
this.LEDBox.PerformLayout();
this.gbTriggerOut.ResumeLayout(false);
this.gbTransferControl.ResumeLayout(false);
this.gbTransferControl.PerformLayout();
this.groupBox1.ResumeLayout(false);
this.groupBox1.PerformLayout();
this.ResumeLayout(false);
this.PerformLayout();
```

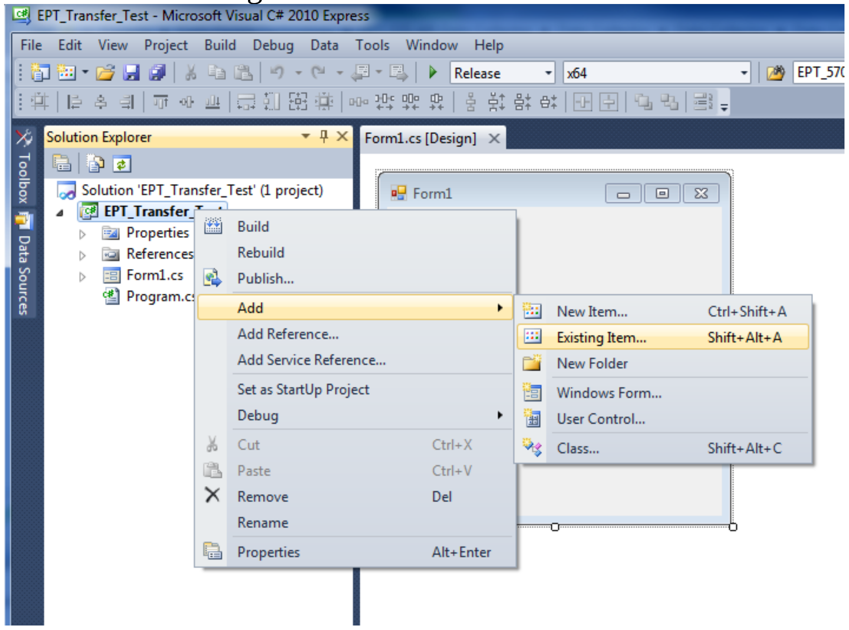
9. Change namespace in Program.cs to new project name
10. Change Application.Run() in Program .cs to new projectname.



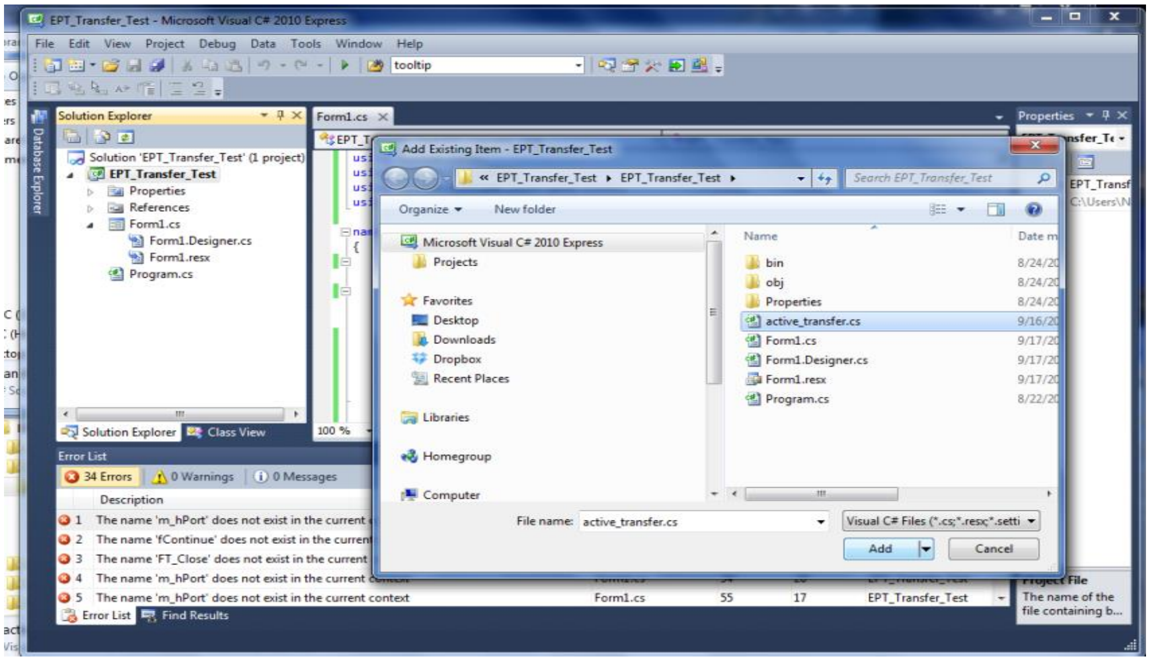
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace EPT_FT2232_Interface
{
    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_FT2232_Interface());
        }
    }
}
```

5.6.2 Add Files to Project

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



Browse to the EPT_Transfer_Test project folder and select the active_transfer_xx.cs file (choose either the 32 bit or 64 bit version, depending on whether your OS is 32 or 64 bit). Click Add.

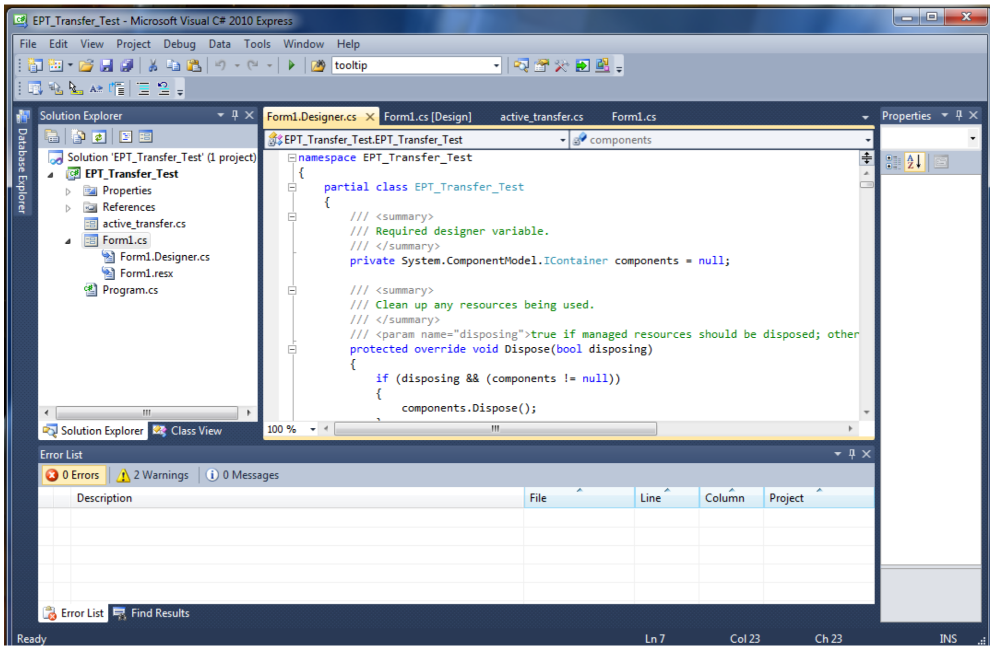


File name: active_transfer.cs

Visual C# Files (*.cs;*.resx;*.setti

	Description	File	Line	Column	Project
1	The name 'm_hPort' does not exist in the current				
2	The name 'fContinue' does not exist in the curren				
3	The name 'FT_Close' does not exist in the current				
4	The name 'm_hPort' does not exist in the current				
5	The name 'm_hPort' does not exist in the current context	Form1.cs	55	17	EPT_Transfer_Test

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.



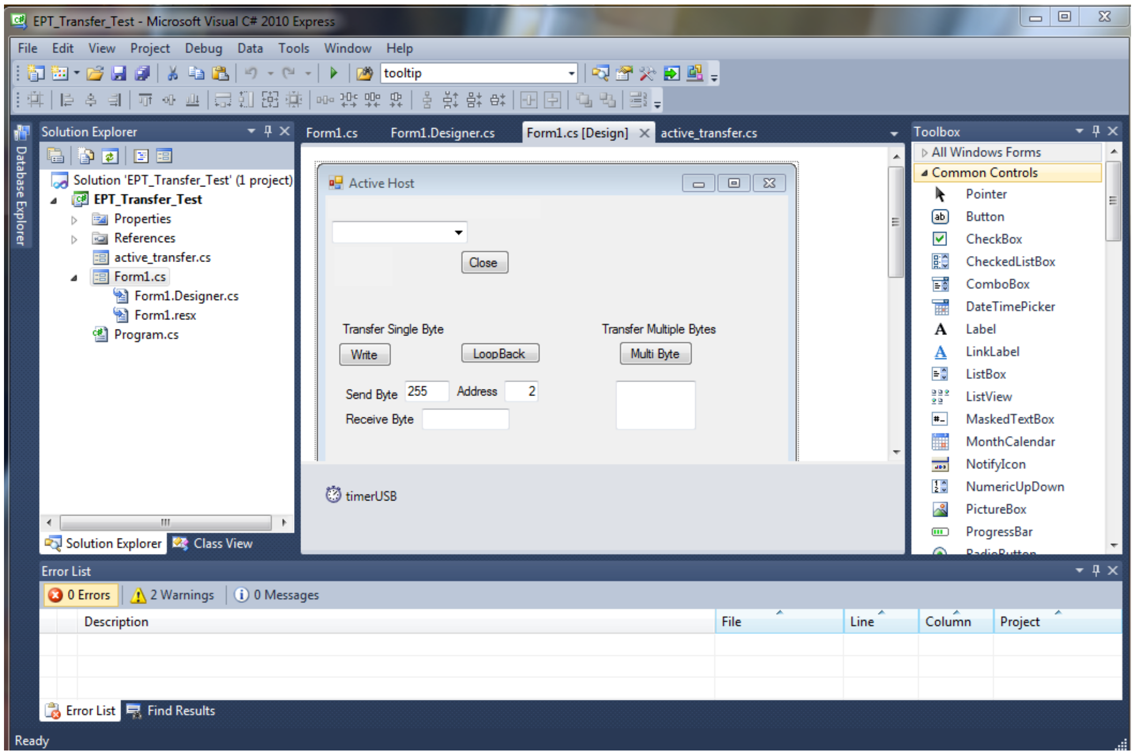
```
namespace EPT_Transfer_Test
{
    partial class EPT_Transfer_Test
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

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

5.6.3 Adding Controls to the Project

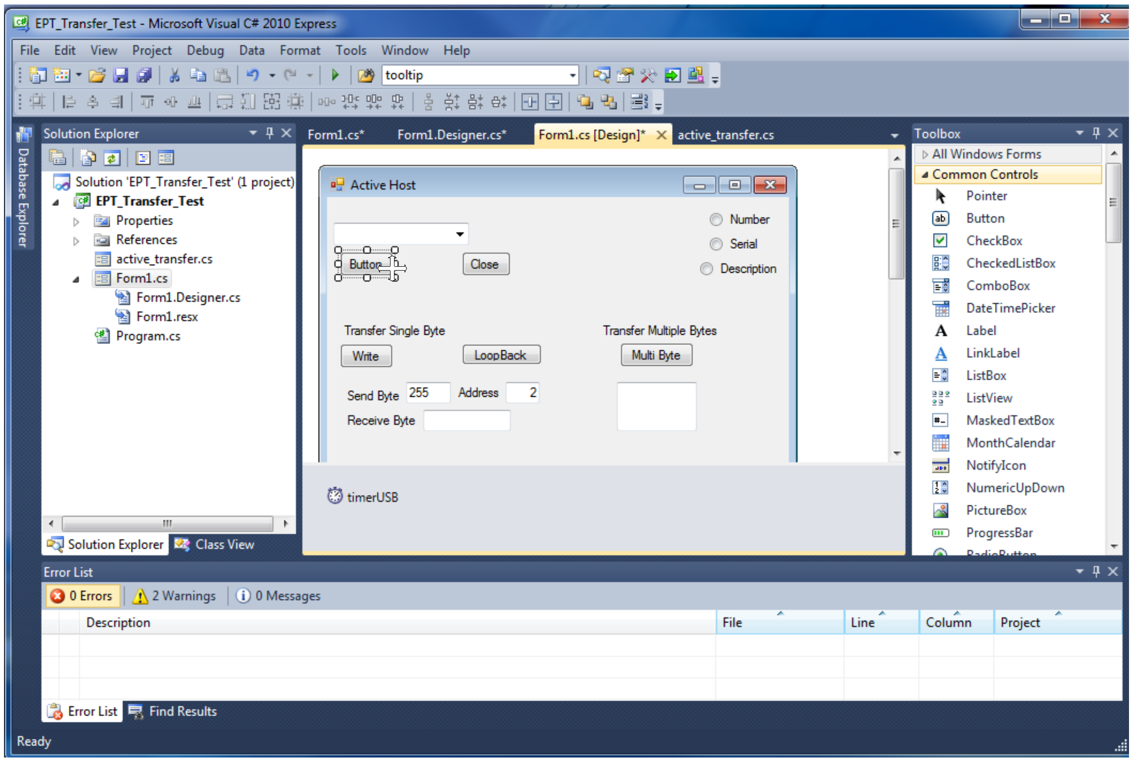
Although, the C# language is very similar to C Code, there are a few major differences. The first is C# .NET environment is event based. A second is C# utilizes classes. This guide will keep the details of these items hidden to keep things simple. However, a brief introduction to events and classes will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

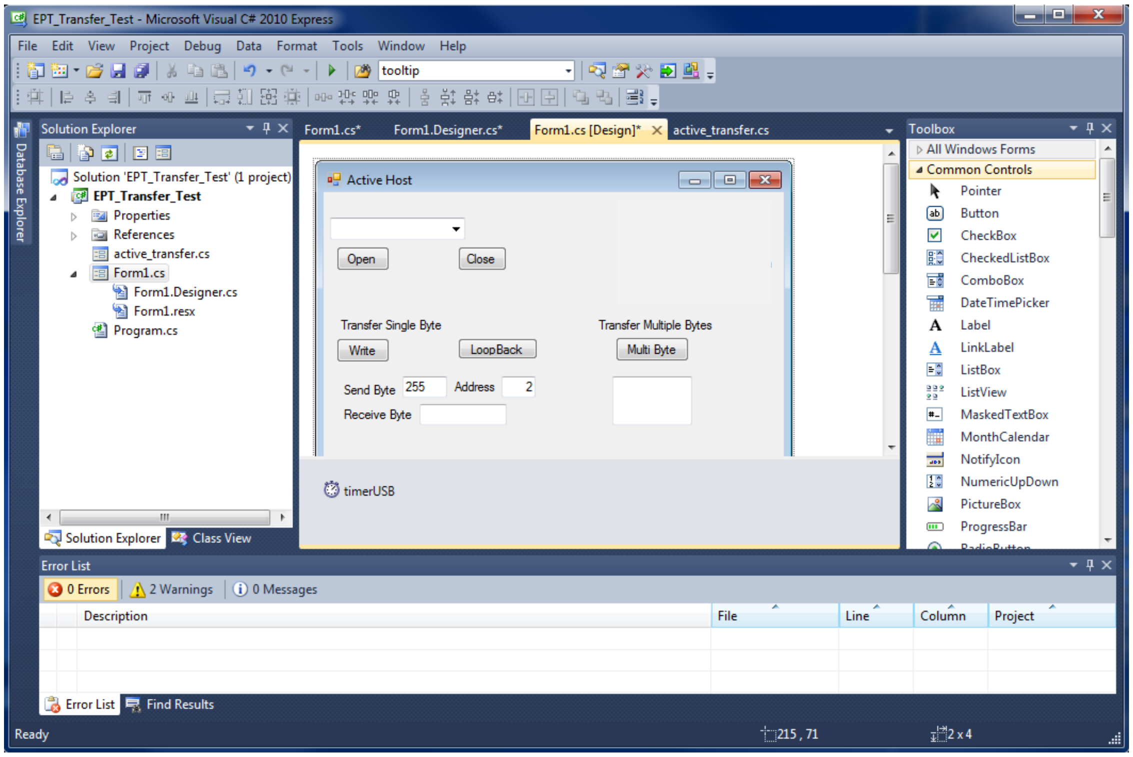
The easiest way to add a button to a form is to double click the Form1.cs in the Solution Explorer. Click on the button to launch the Toolbox.



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;
}
```

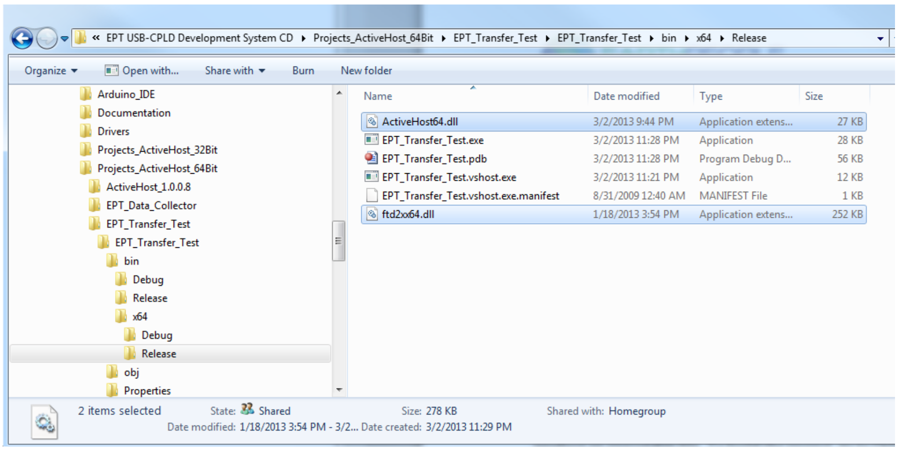
When btnOpenDevice_Click is called, it calls the function "OpenDevice()". This function is defined in the dll and will connect to the device selected in the combo box. This is a quick view of how to create, add files, and add controls to a C# project. The user is encouraged to spend some time reviewing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

5.6.4 Adding the DLL's to the Project

Locate the EPT USB-CPLD Development System CD installed on your PC. Browse to the Projects_ActiveHost folder (choose either the 32 bit or 64 bit version, depending on whether your OS is 32 or 64 bit). Open the Bin folder, copy the following files:

- ActiveHostXX.dll
- ftd2xxXX.dll

and install them in the bin\x64\x64 folder of your EPT_Transfer_Test project.


« EPT USB-CPLD Development System CD ▸ Projects_ActiveHost_64Bit ▸ EPT_Transfer_Test ▸ EPT_Transfer_Test ▸ bin ▸ x64 ▸ Release
Organize
Open with...
Share with
Burn
New folder

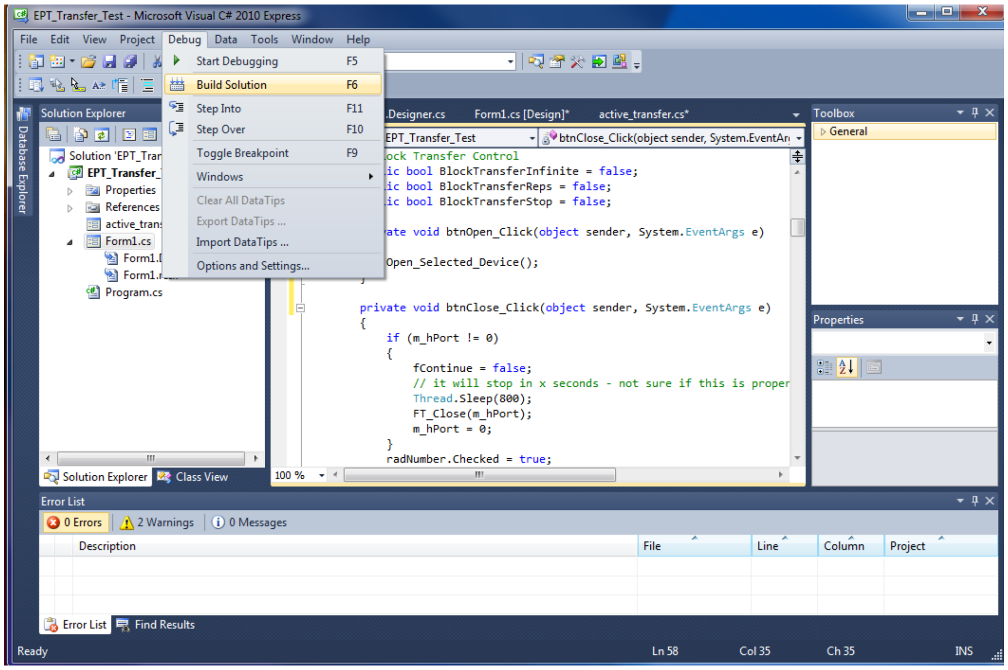
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	8/31/2009 12:40 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

2 items selected State: Shared Size: 278 KB Shared with: Homegroup
Date modified: 1/18/2013 3:54 PM - 3/2... Date created: 3/2/2013 11:29 PM

Save the project.

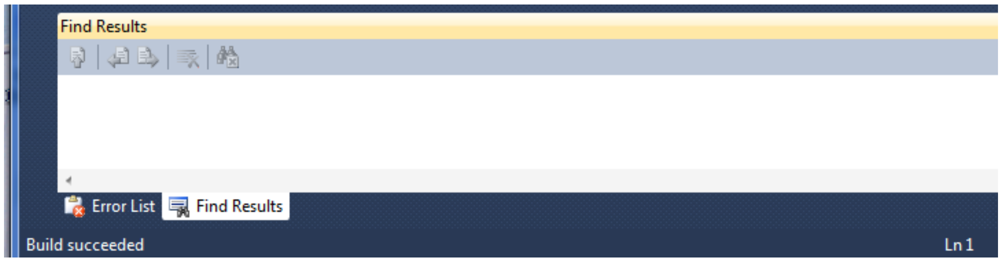
5.6.5 Building the Project

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



```
ock Transfer Control
ic bool BlockTransferInfinite = false;
ic bool BlockTransferReps = false;
ic bool BlockTransferStop = false;

ate void btnOpen_Click(object sender, System.EventArgs e)

Open_Selected_Device();

private void btnClose_Click(object sender, System.EventArgs e)
{
    if (m_hPort != 0)
    {
        fContinue = false;
        // it will stop in x seconds - not sure if this is proper
        Thread.Sleep(800);
        FT_Close(m_hPort);
        m_hPort = 0;
    }
    radNumber.Checked = true;
```

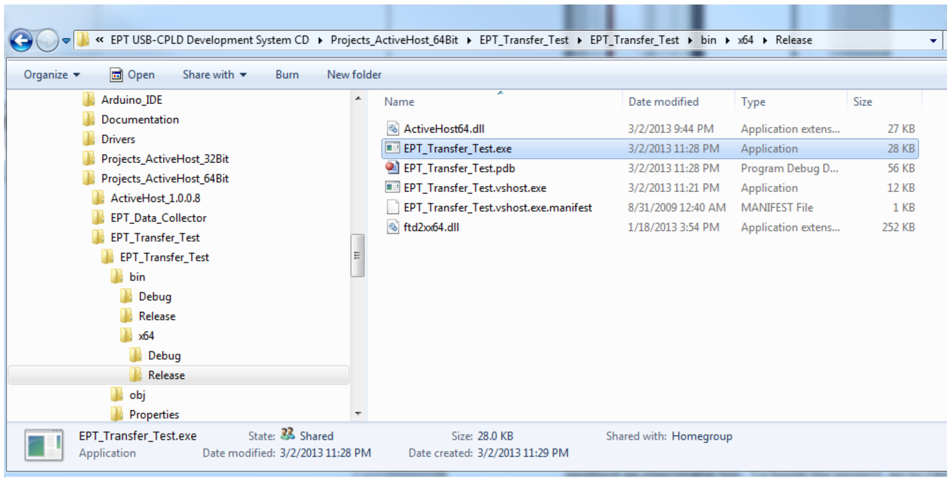
The C# Express compiler will start the building process. If there are no errors with code syntax, function usage, or linking, then the environment responds with “Build Succeeded”.



Build succeeded

5.6.6 Testing the Project

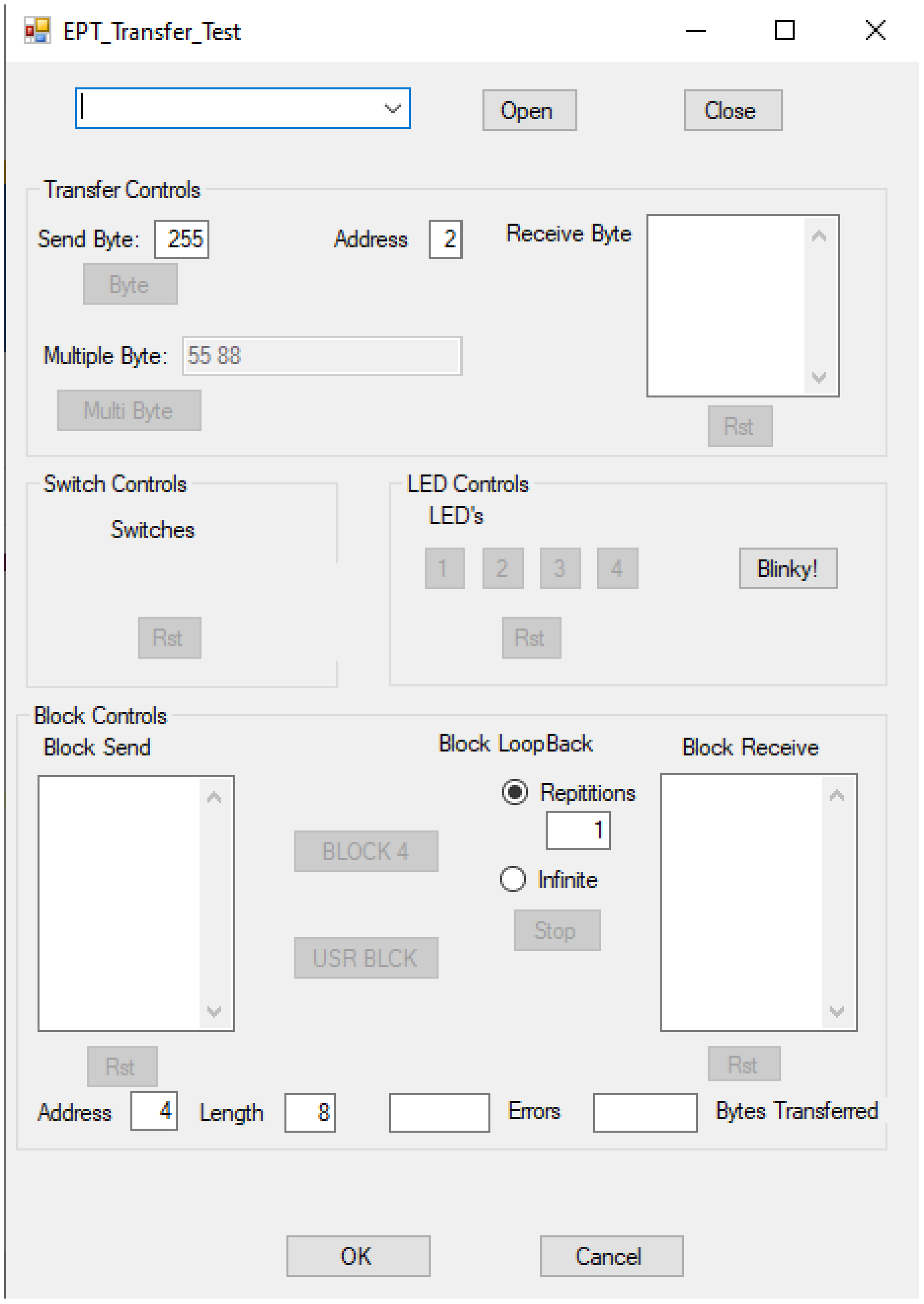
Once the project has been successfully built, it produces an *.exe file. The file will be saved in the Release or Debug folders.



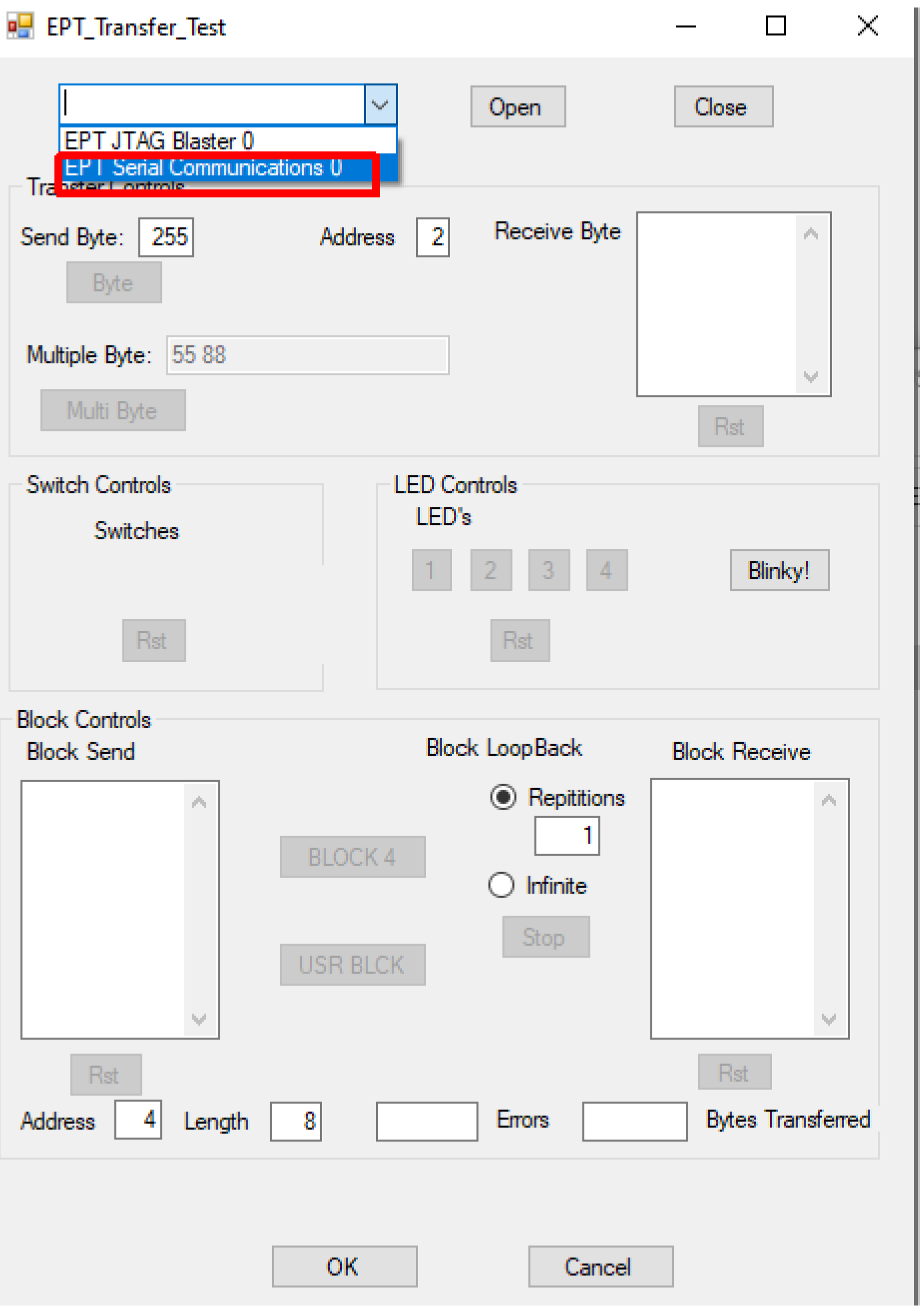
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	8/31/2009 12:40 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

EPT_Transfer_Test.exe State: Shared Size: 28.0 KB Shared with: Homegroup
Application Date modified: 3/2/2013 11:28 PM Date created: 3/2/2013 11:29 PM

The EPT_Transfer_Text.exe file can now be tested using the UnoMax board. To test the file, connect the UnoMax to the Windows PC using Type A to Type Micro B USB 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_Transfer_Text.exe file resides, and double click on the file. The application should load with a Windows form.

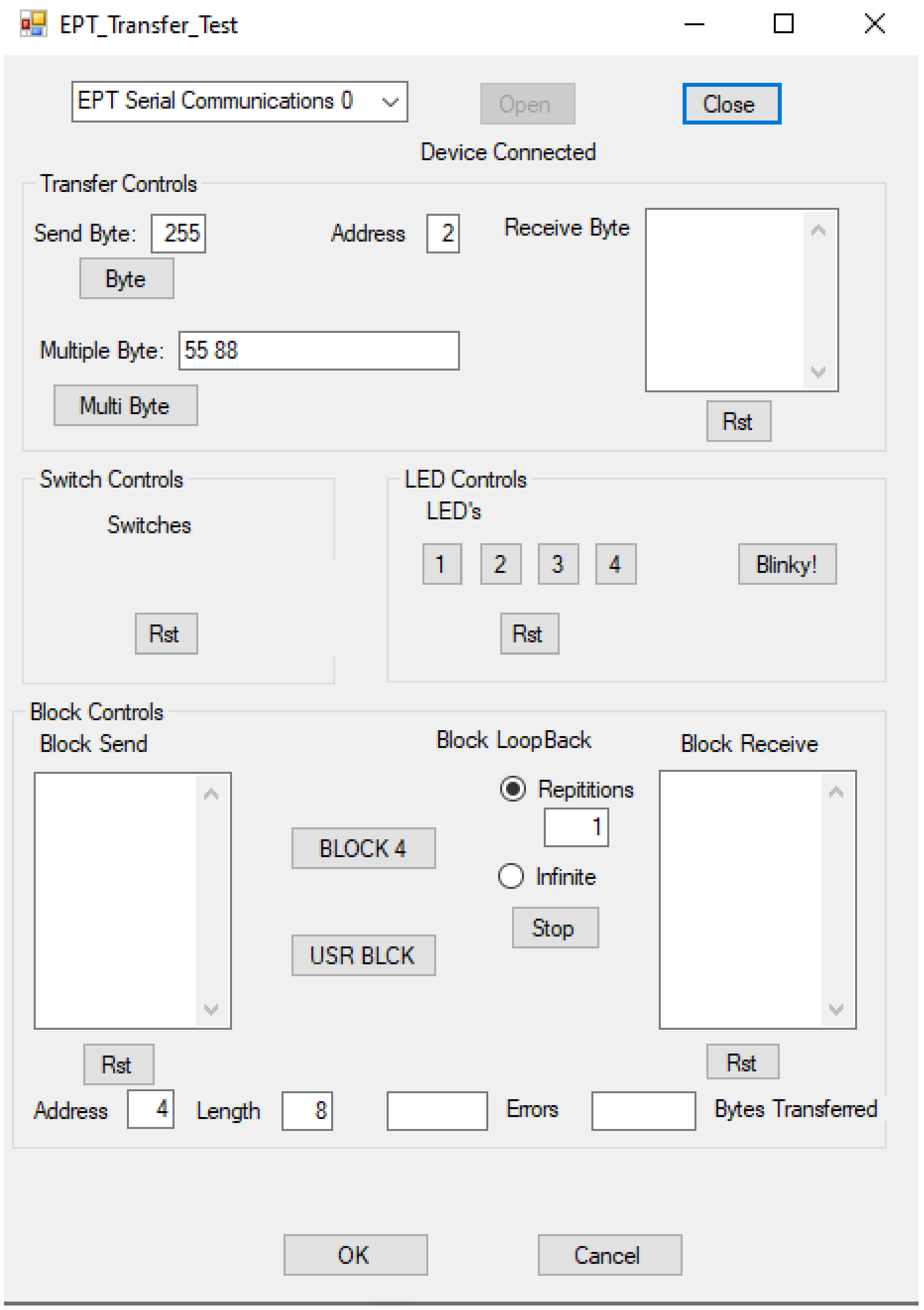


With the application loaded, select the USB-CPLD board from the dropdown combo box and click on the “Open” button.

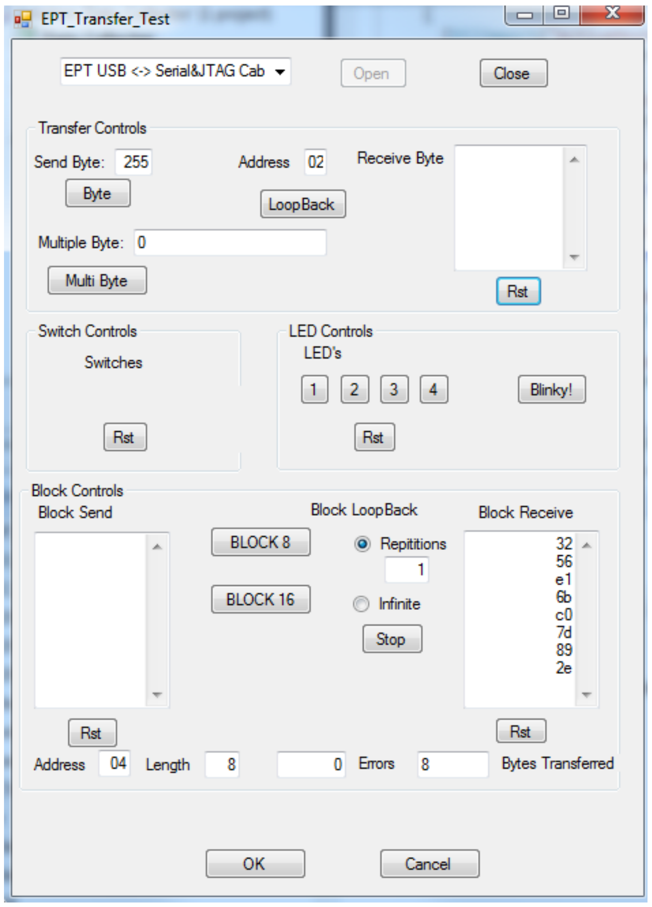


Click on one of the LED buttons in the middle of the window. The corresponding LED on the UnoMax board should light up.

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.



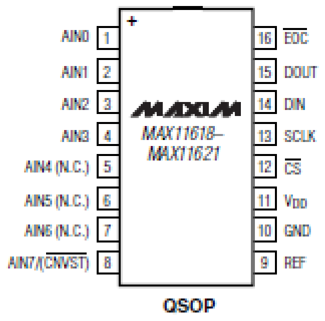
To exercise the Block Transfer EndTerm, click the “Block4” or “USR Block” button in the Block Controls group. A pre-selected group of numbers appear in the Block Receive textbox.



Press the PCB switches on the UnoMax to view the Switch Controls in action.

6 Using the Analog to Digital Converter

The EPT 5M57-AP-U2 has an onboard Four Channel, 10 Bit, 300 KSamples/second Analog to Digital Converter. It has a serial SPI communications that allow the host to send setup commands and retrieve the sampled data.



PIN	NAME	FUNCTION
1–4	AIN0–AIN3	Analog Inputs
5, 6, 7	N.C.	No Connection
8	CONVST	Active – low Conversion Start Input
9	REF	Reference Input
10	GND	Ground
11	VDD	Power Input
12	CS	Active Low Chip Select Input. When CS is Low the interface is enabled. When CS is high MOSI is high impedance
13	SCLK	Serial Clock input. Clocks data in and out of the serial interface.
14	MISO	Serial Data input. MISO data is latched into the interface on the rising edge of SCLK
15	MOSI	Serial Data Output. Data is clocked out on the falling edge of SCLK. High impedance when CS is connected to VDD.
16	EOC	End of Conversion Output. Data is valide after EOC pulls low.

6.1 Register Descriptions

The MAX11618 communicate between the internal registers and the external circuitry through the SPI-/QSPI-compatible serial interface. Table 1 details the registers and the bit names. Tables 2–5 show the various functions within the conversion register, setup register, averaging register, and reset register.

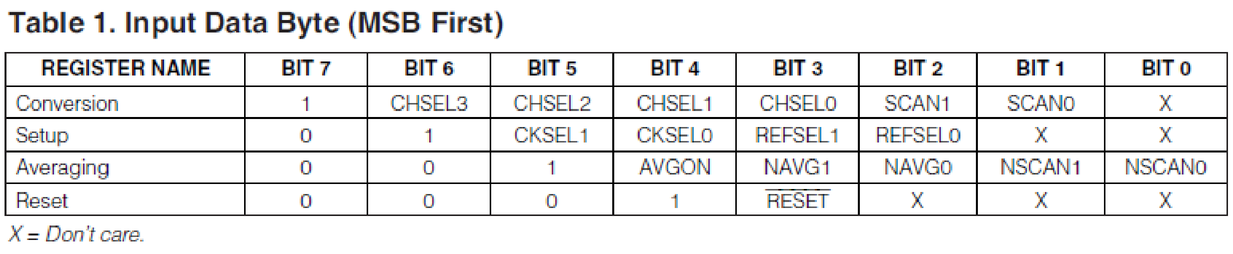


Table 1. Input Data Byte (MSB First)

REGISTER NAME	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Conversion	1	CHSEL3	CHSEL2	CHSEL1	CHSEL0	SCAN1	SCAN0	X
Setup	0	1	CKSEL1	CKSEL0	REFSEL1	REFSEL0	X	X
Averaging	0	0	1	AVGON	NAVG1	NAVG0	NSCAN1	NSCAN0
Reset	0	0	0	1	$\overline{\text{RESET}}$	X	X	X

X = Don't care.

6.2 Conversion Register

Select active analog input channels per scan and scan modes by writing to the conversion register. Table 2 details channel selection and the four scan modes. Request a scan by writing to the conversion register when in clock mode 10 or 11, or by applying a low pulse to the CNVST pin when in clock mode 00 or 01.

A conversion is not performed if it is requested on a channel that has been configured as CNVST. Select scan mode 00 or 01 to return one result per single- ended channel within the requested range. Select scan mode 10 to scan a single input channel numerous times, depending on NSCAN1 and NSCAN0 in the averaging register (Table 4). Select scan mode 11 to return only one result from a single channel.

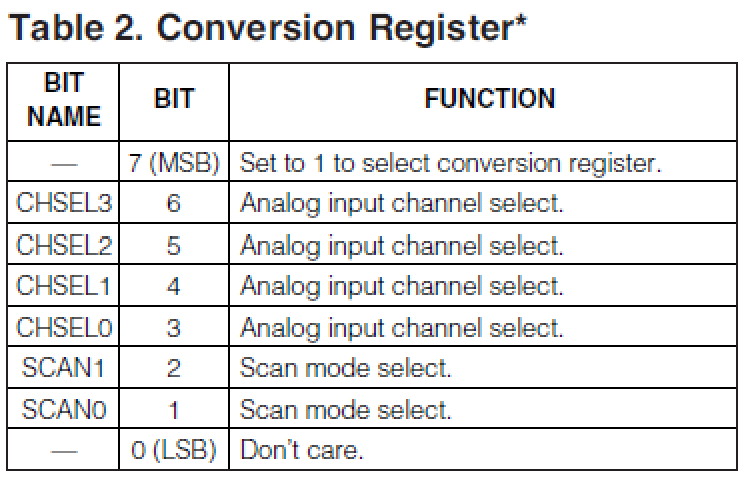
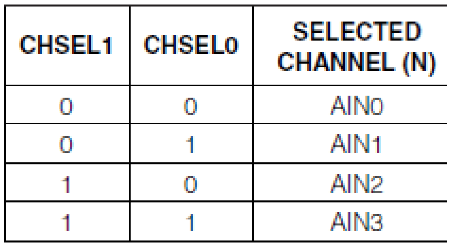
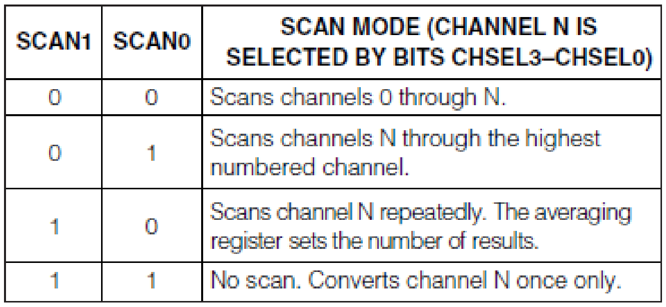


Table 2. Conversion Register*

BIT NAME	BIT	FUNCTION
—	7 (MSB)	Set to 1 to select conversion register.
CHSEL3	6	Analog input channel select.
CHSEL2	5	Analog input channel select.
CHSEL1	4	Analog input channel select.
CHSEL0	3	Analog input channel select.
SCAN1	2	Scan mode select.
SCAN0	1	Scan mode select.
—	0 (LSB)	Don't care.



CHSEL1	CHSEL0	SELECTED CHANNEL (N)
0	0	AIN0
0	1	AIN1
1	0	AIN2
1	1	AIN3



SCAN1	SCAN0	SCAN MODE (CHANNEL N IS SELECTED BY BITS CHSEL3–CHSEL0)
0	0	Scans channels 0 through N.
0	1	Scans channels N through the highest numbered channel.
1	0	Scans channel N repeatedly. The averaging register sets the number of results.
1	1	No scan. Converts channel N once only.

6.3 Setup Register

Write a byte to the setup register to configure the clock,reference, and power-down modes. Table 3 details the bits in the setup register. Bits 5 and 4 (CKSEL1 and CKSEL0) control the clock mode, acquisition and sampling, and the conversion start. Bits 3 and 2 (REFSEL1 and REFSEL0) control internal or external reference use.

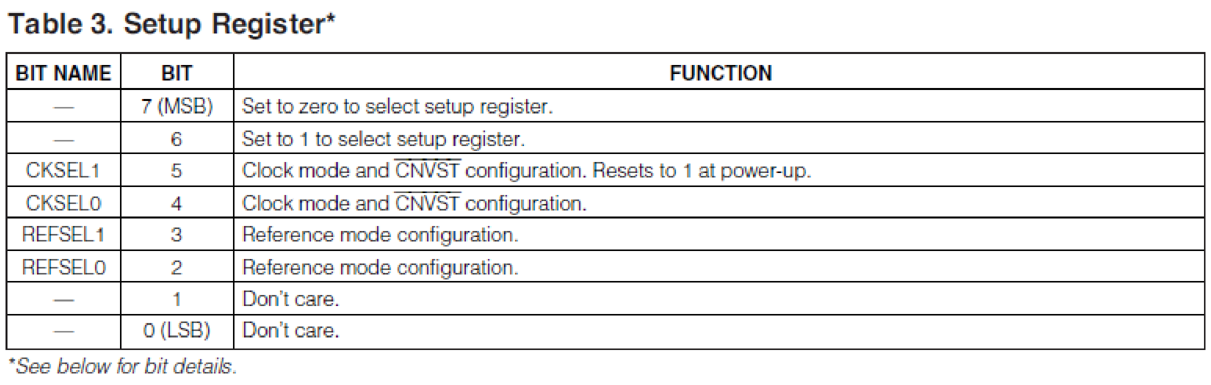
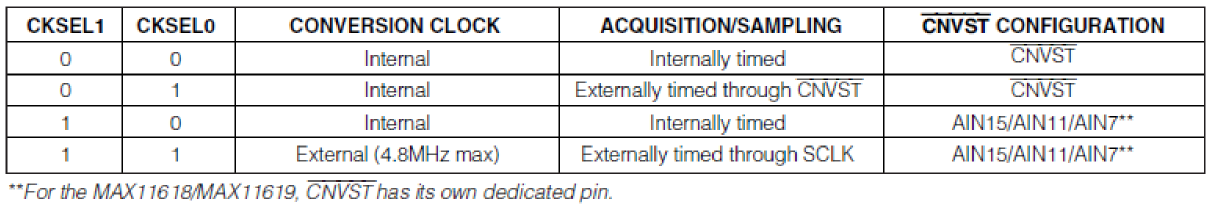


Table 3. Setup Register*

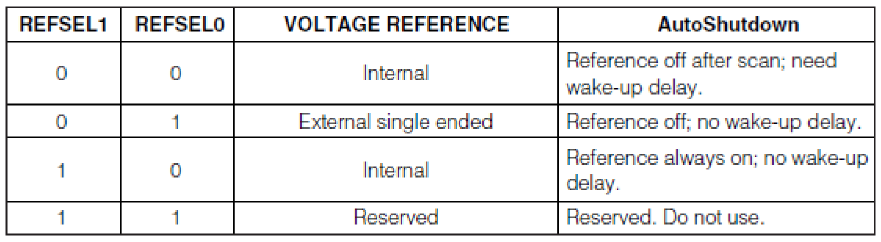
BIT NAME	BIT	FUNCTION
—	7 (MSB)	Set to zero to select setup register.
—	6	Set to 1 to select setup register.
CKSEL1	5	Clock mode and $\overline{\text{CNVST}}$ configuration. Resets to 1 at power-up.
CKSEL0	4	Clock mode and $\overline{\text{CNVST}}$ configuration.
REFSEL1	3	Reference mode configuration.
REFSEL0	2	Reference mode configuration.
—	1	Don't care.
—	0 (LSB)	Don't care.

*See below for bit details.



CKSEL1	CKSEL0	CONVERSION CLOCK	ACQUISITION/SAMPLING	$\overline{\text{CNVST}}$ CONFIGURATION
0	0	Internal	Internally timed	$\overline{\text{CNVST}}$
0	1	Internal	Externally timed through $\overline{\text{CNVST}}$	$\overline{\text{CNVST}}$
1	0	Internal	Internally timed	AIN15/AIN11/AIN7**
1	1	External (4.8MHz max)	Externally timed through SCLK	AIN15/AIN11/AIN7**

**For the MAX11618/MAX11619, $\overline{\text{CNVST}}$ has its own dedicated pin.



REFSEL1	REFSEL0	VOLTAGE REFERENCE	AutoShutdown
0	0	Internal	Reference off after scan; need wake-up delay.
0	1	External single ended	Reference off; no wake-up delay.
1	0	Internal	Reference always on; no wake-up delay.
1	1	Reserved	Reserved. Do not use.

6.4 Averaging Register

Write to the averaging register to configure the ADC to average up to 32 samples for each requested result, and to independently control the number of results requested for single-channel scans. Table 2 details the four scan modes available in the conversion register. All four scan modes allow averaging as long as the AVGON bit, bit 4 in the averaging register, is set to 1. Select scan mode 10 to scan the same channel multiple times. Clock mode 11 disables averaging.

Table 4. Averaging Register*

BIT NAME	BIT	FUNCTION
—	7 (MSB)	Set to 0 to select averaging register.
—	6	Set to 0 to select averaging register.
—	5	Set to 1 to select averaging register.
AVGON	4	Set to 1 to turn averaging on. Set to zero to turn averaging off.
NAVG1	3	Configures the number of conversions for single-channel scans.
NAVG0	2	Configures the number of conversions for single-channel scans.
NSCAN1	1	Single-channel scan count. (Scan mode 10 only.)
NSCAN0	0 (LSB)	Single-channel scan count. (Scan mode 10 only.)

**See below for bit details.*

AVGON	NAVG1	NAVG0	FUNCTION
0	X	X	Performs 1 conversion for each requested result.
1	0	0	Performs 4 conversions and returns the average for each requested result.
1	0	1	Performs 8 conversions and returns the average for each requested result.
1	1	0	Performs 16 conversions and returns the average for each requested result.
1	1	1	Performs 32 conversions and returns the average for each requested result.

X = Don't care.

NSCAN1	NSCAN0	FUNCTION (APPLIES ONLY IF SCAN MODE 10 IS SELECTED)
0	0	Scans channel N and returns 4 results.
0	1	Scans channel N and returns 8 results.
1	0	Scans channel N and returns 12 results.
1	1	Scans channel N and returns 16 results.

6.5 Reset Register

Write to the reset register (as shown in Table 5) to clear the FIFO or to reset all registers to their default states. Set the RESET bit to 1 to reset the FIFO. Set the reset bit to zero to return the MAX11618 to the default power-up state.

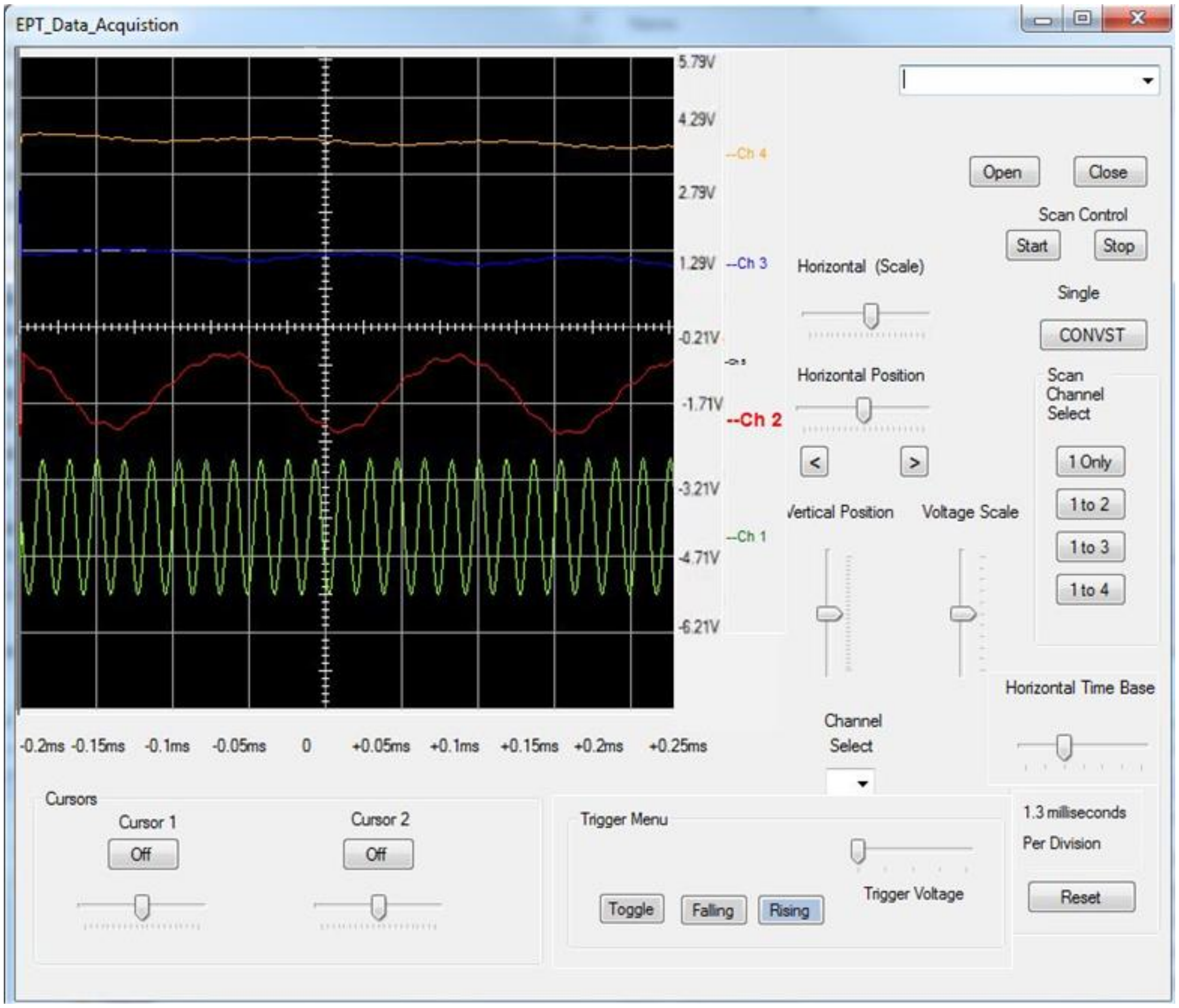
Table 5. Reset Register

BIT NAME	BIT	FUNCTION
—	7 (MSB)	Set to 0 to select reset register.
—	6	Set to 0 to select reset register.
—	5	Set to 0 to select reset register.
—	4	Set to 1 to select reset register.
$\overline{\text{RESET}}$	3	Set to zero to reset all registers. Set to 1 to clear the FIFO only.
X	2	Don't care.
X	1	Don't care.
X	0 (LSB)	Don't care.

7 The UnoProLyzer Application

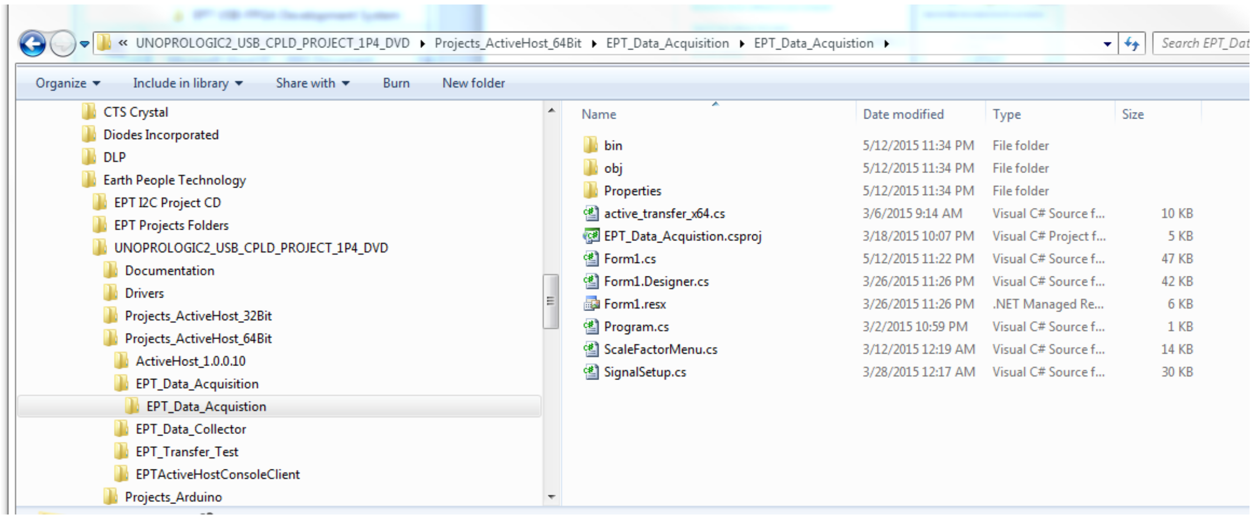
The source DVD for the UnoMax2 comes with the UnoProLyzer application project. This project allows the user to display 1 to 4 channels of analog input in a graphing application. The project utilizes the PC to perform all data storage and graphing. The PC sends commands to and receives the data from the UnoMax2 and stores each channel data in its own separate buffer in memory. The UnoProLyzer collects all samples from each channel by streaming across up to four dedicated communication “pipes”.

The UnoMax2 commands the ADC to start a conversion on the channels selected by the user. It then waits for the ADC to complete the conversion on all channels. It transfers the data for each channel across its own dedicated communication pipe. Then starts the process over again. The UnoProLyzer application will accept each data word and decode the pipe number it came across. It stores each word into a separate buffer for each channel. The UnoProLyzer then performs post processing on each data word. It performs trigger detection, smoothing, sorting, scaling and searching. It then displays the data set in 500 data point segments.



7.1 Accessing the UnoProLyzer Application

Locate the EPT_Data_Acquistion folder in the Drivers folder of the EPT USB-CPLD Development System DVD using Windows Explorer.



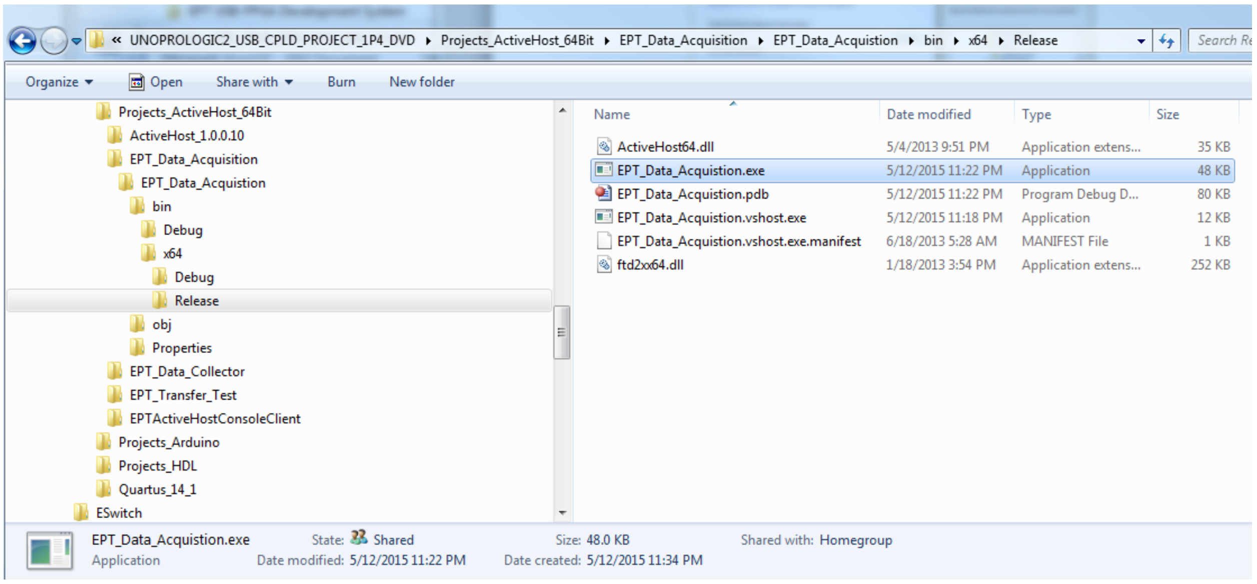
« UNOPROLOGIC2_USB_CPLD_PROJECT_1P4_DVD ▸ Projects_ActiveHost_64Bit ▸ EPT_Data_Acquisition ▸ EPT_Data_Acquistion ▸

Organize ▾ Include in library ▾ Share with ▾ Burn New folder

CTS Crystal
Diodes Incorporated
DLP
Earth People Technology
EPT I2C Project CD
EPT Projects Folders
UNOPROLOGIC2_USB_CPLD_PROJECT_1P4_DVD
Documentation
Drivers
Projects_ActiveHost_32Bit
Projects_ActiveHost_64Bit
ActiveHost_1.0.0.10
EPT_Data_Acquisition
EPT_Data_Acquistion
EPT_Data_Collector
EPT_Transfer_Test
EPTActiveHostConsoleClient
Projects_Arduino

Name	Date modified	Type	Size
bin	5/12/2015 11:34 PM	File folder	
obj	5/12/2015 11:34 PM	File folder	
Properties	5/12/2015 11:34 PM	File folder	
active_transfer_x64.cs	3/6/2015 9:14 AM	Visual C# Source f...	10 KB
EPT_Data_Acquistion.csproj	3/18/2015 10:07 PM	Visual C# Project f...	5 KB
Form1.cs	5/12/2015 11:22 PM	Visual C# Source f...	47 KB
Form1.Designer.cs	3/26/2015 11:26 PM	Visual C# Source f...	42 KB
Form1.resx	3/26/2015 11:26 PM	.NET Managed Re...	6 KB
Program.cs	3/2/2015 10:59 PM	Visual C# Source f...	1 KB
ScaleFactorMenu.cs	3/12/2015 12:19 AM	Visual C# Source f...	14 KB
SignalSetup.cs	3/28/2015 12:17 AM	Visual C# Source f...	30 KB

Go to the Release folder and locate the EPT_Data_Acquisition.exe file.



« UNOPROLOGIC2_USB_CPLD_PROJECT_1P4_DVD ▸ Projects_ActiveHost_64Bit ▸ EPT_Data_Acquisition ▸ EPT_Data_Acquistion ▸ bin ▸ x64 ▸ Release

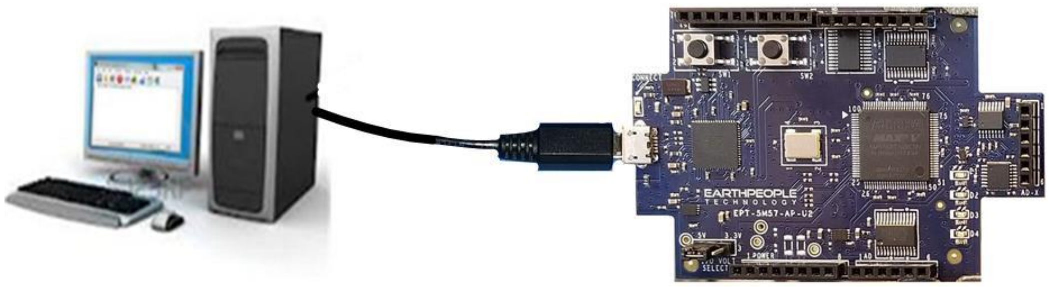
Organize ▾ Open Share with ▾ Burn New folder

Projects_ActiveHost_64Bit
ActiveHost_1.0.0.10
EPT_Data_Acquisition
EPT_Data_Acquistion
bin
Debug
x64
Debug
Release
obj
Properties
EPT_Data_Collector
EPT_Transfer_Test
EPTActiveHostConsoleClient
Projects_Arduino
Projects_HDL
Quartus_14_1
ESwitch

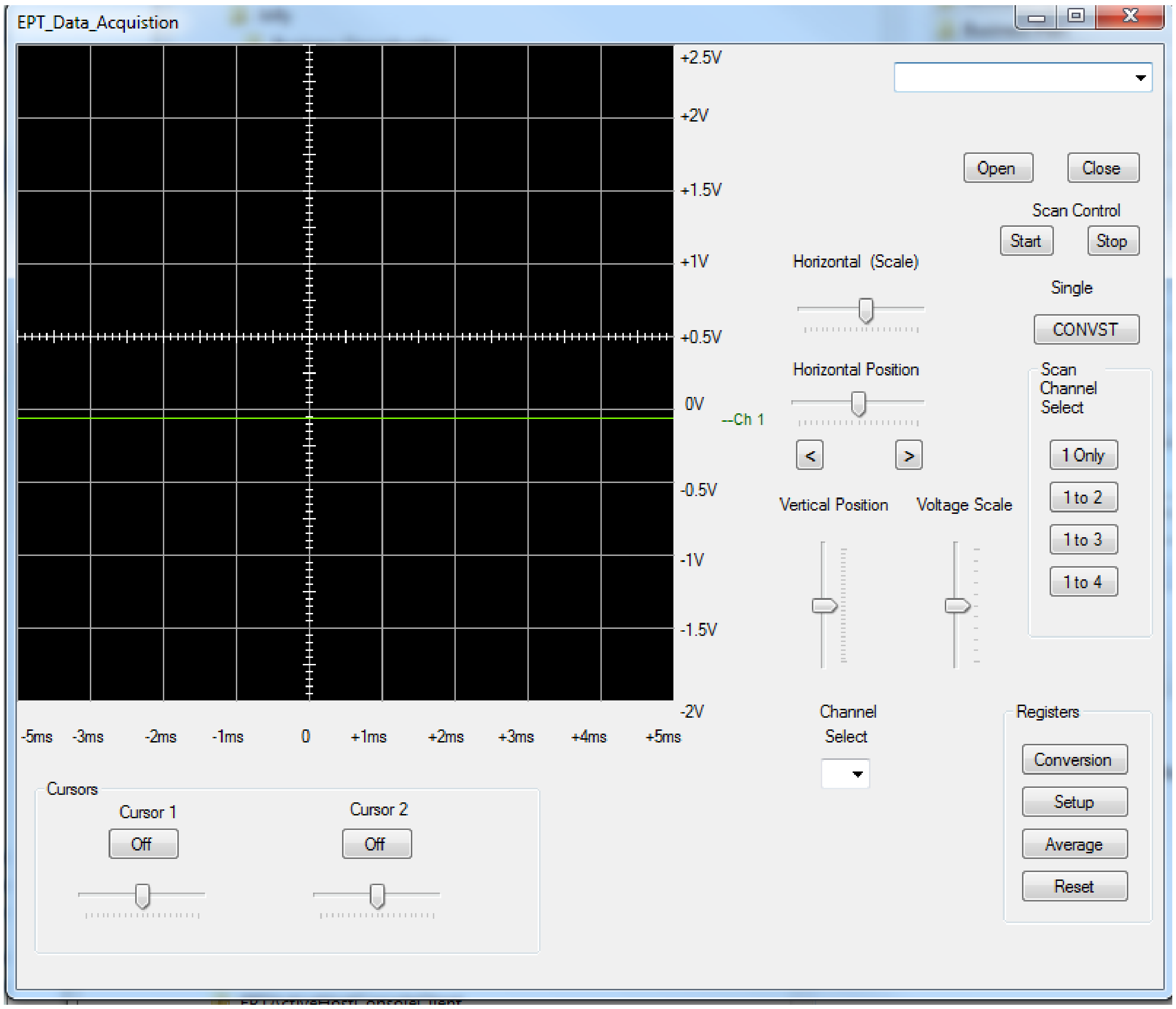
Name	Date modified	Type	Size
ActiveHost64.dll	5/4/2013 9:51 PM	Application extens...	35 KB
EPT_Data_Acquistion.exe	5/12/2015 11:22 PM	Application	48 KB
EPT_Data_Acquistion.pdb	5/12/2015 11:22 PM	Program Debug D...	80 KB
EPT_Data_Acquistion.vshost.exe	5/12/2015 11:18 PM	Application	12 KB
EPT_Data_Acquistion.vshost.exe.manifest	6/18/2013 5:28 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

EPT_Data_Acquistion.exe State: Shared Size: 48.0 KB Shared with: Homegroup
Application Date modified: 5/12/2015 11:22 PM Date created: 5/12/2015 11:34 PM

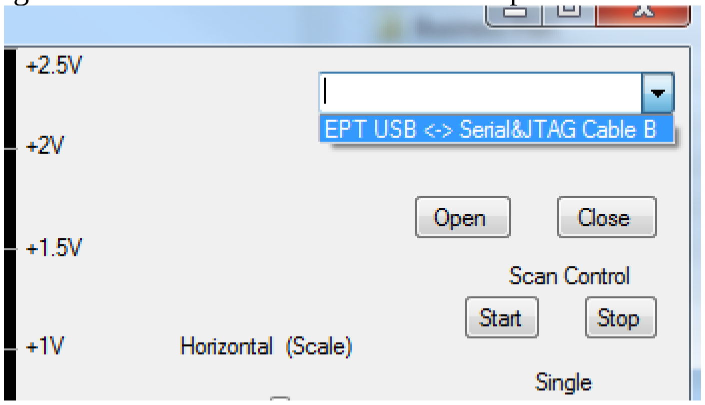
Make sure the UnoMax2 is installed and the USB driver has been loaded.



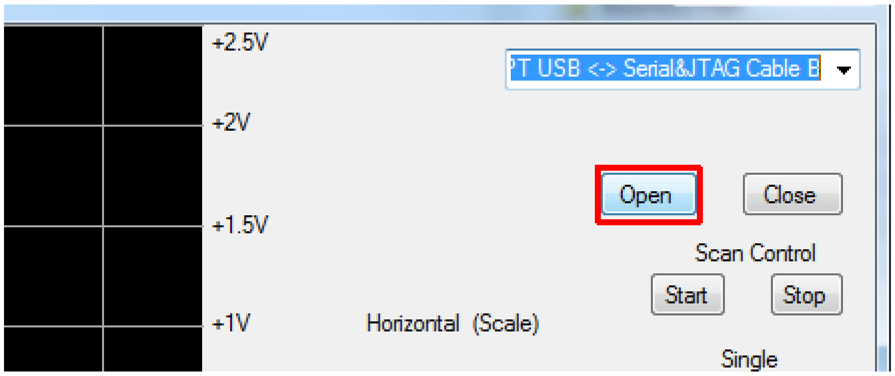
Double click on the EPT_Data_Acquisition.exe file and the UnoProLyzer Application will open.



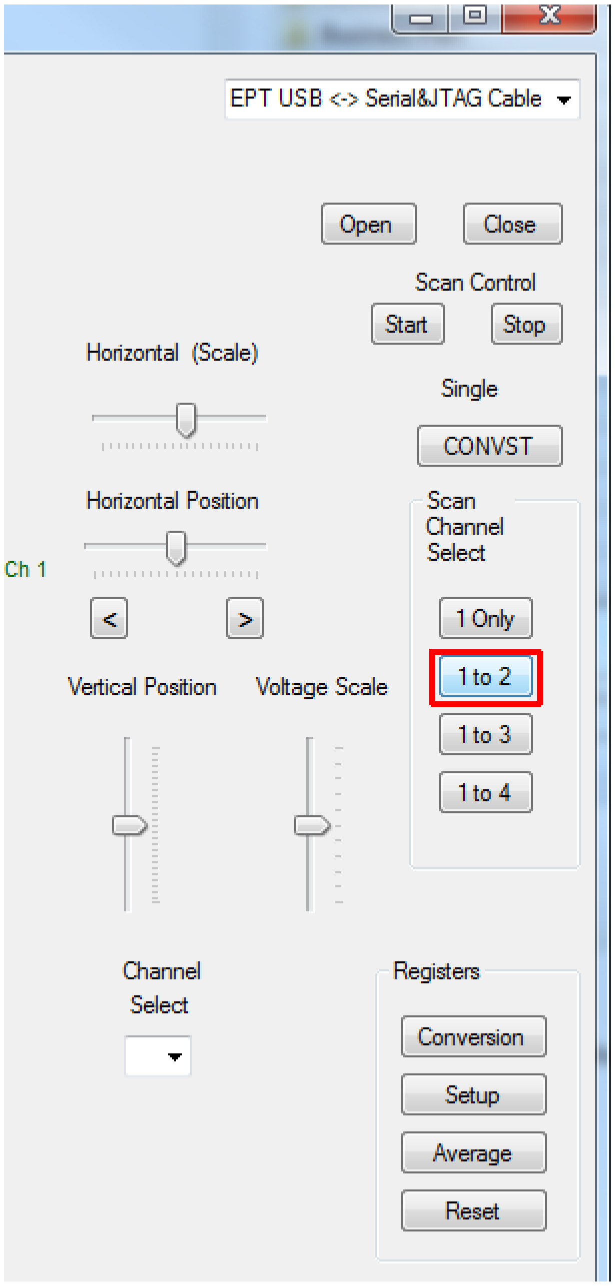
Go to the upper right of the window and click on the drop down box.



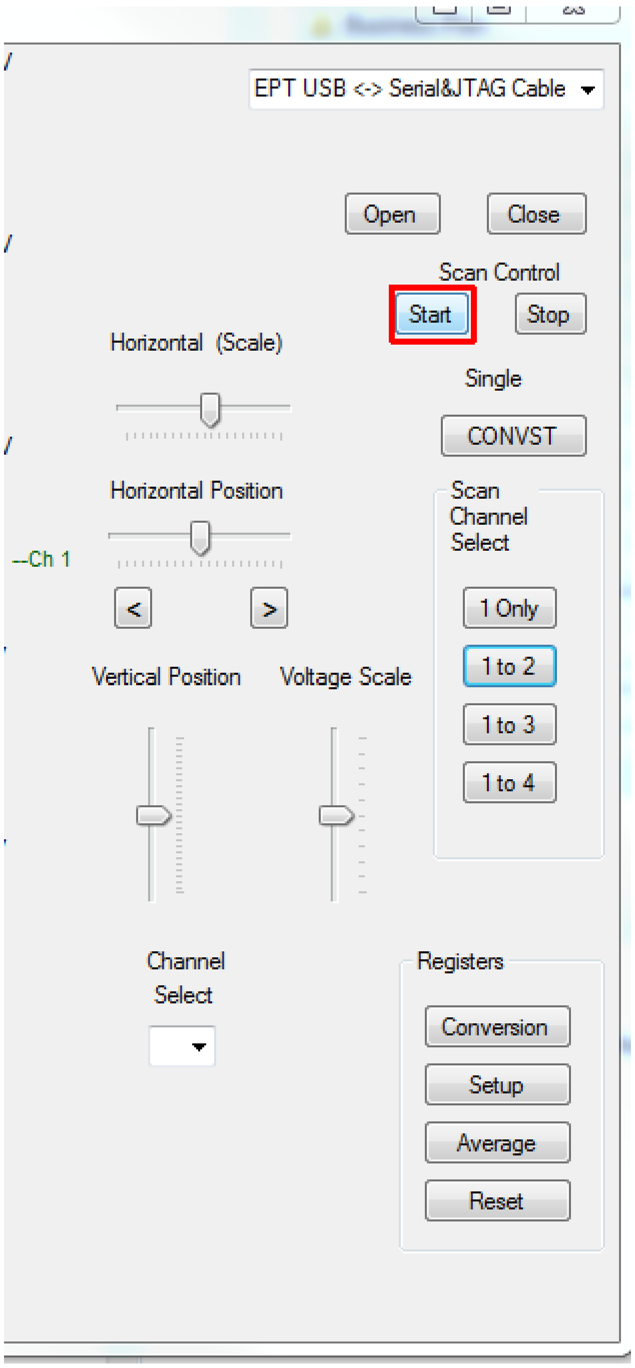
Select the "EPT USB<-> Serial&JTAG Cable B. Then click on the Open button



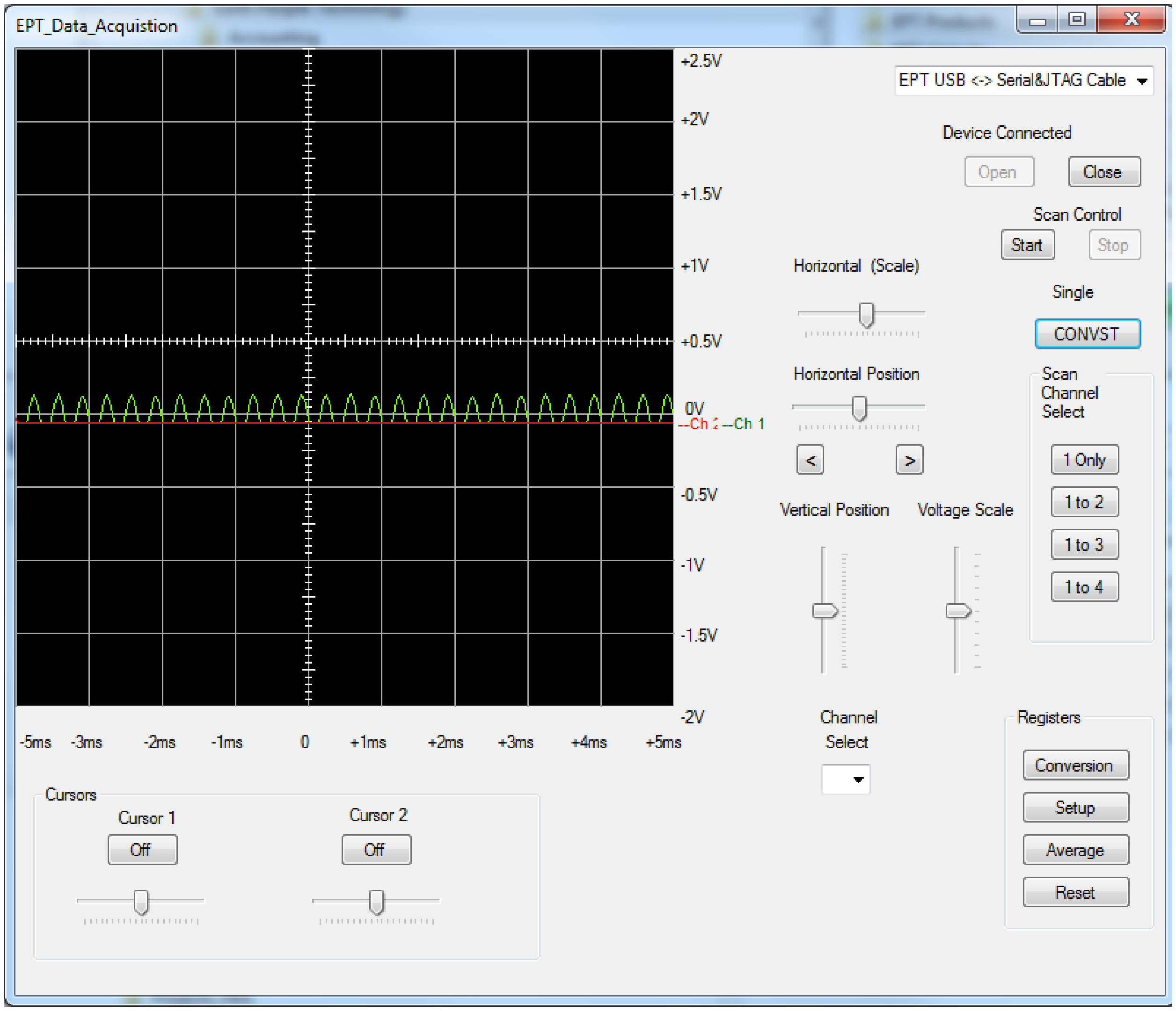
Next, select the number of channels to display. The channels have to be selected in sequential order, you cannot pick out single channel (except for channel 1). So for two channels, click on the “1 to 2” button. This will display the data from both channel 1 and 2.



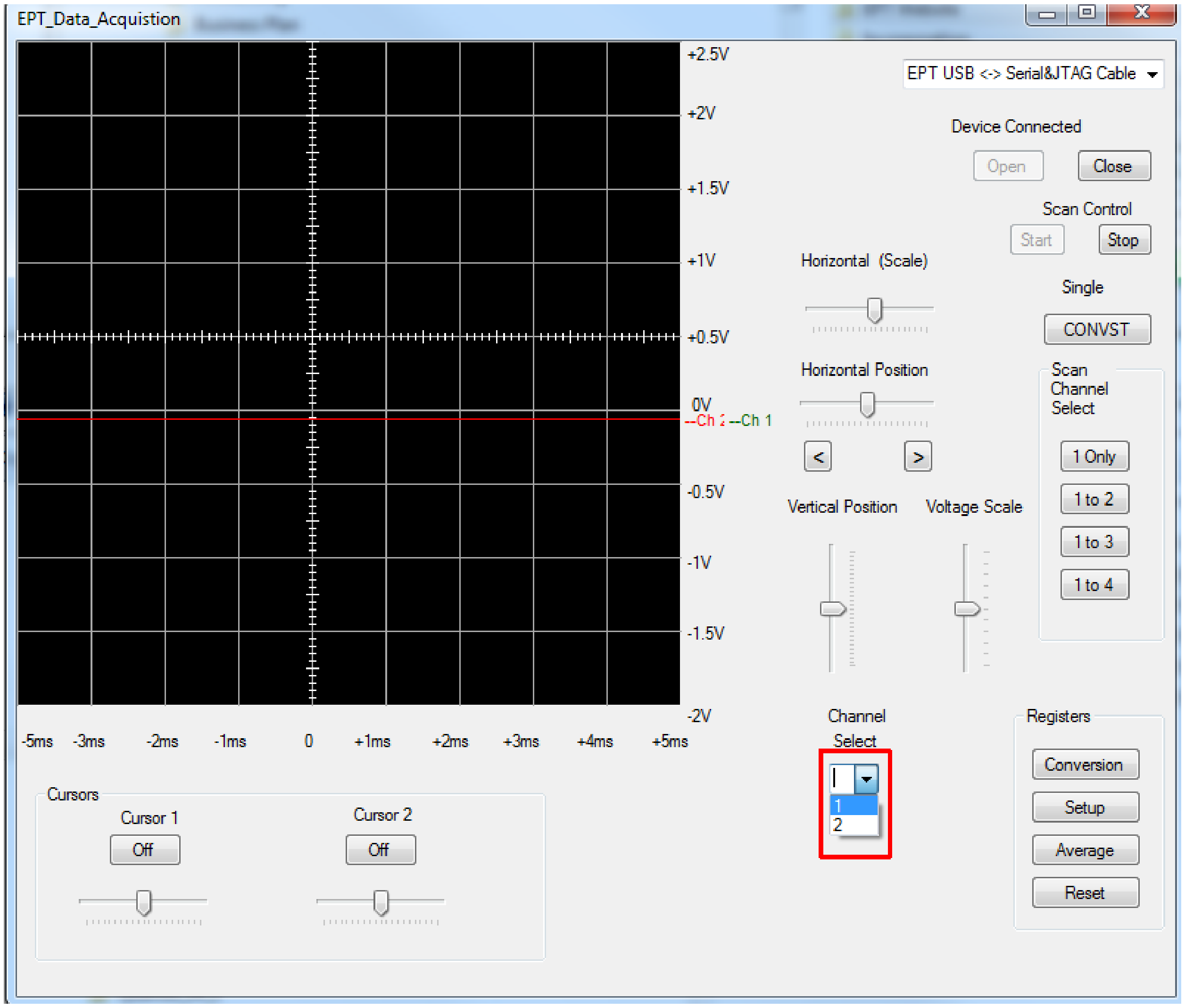
Next, click on the Start button.



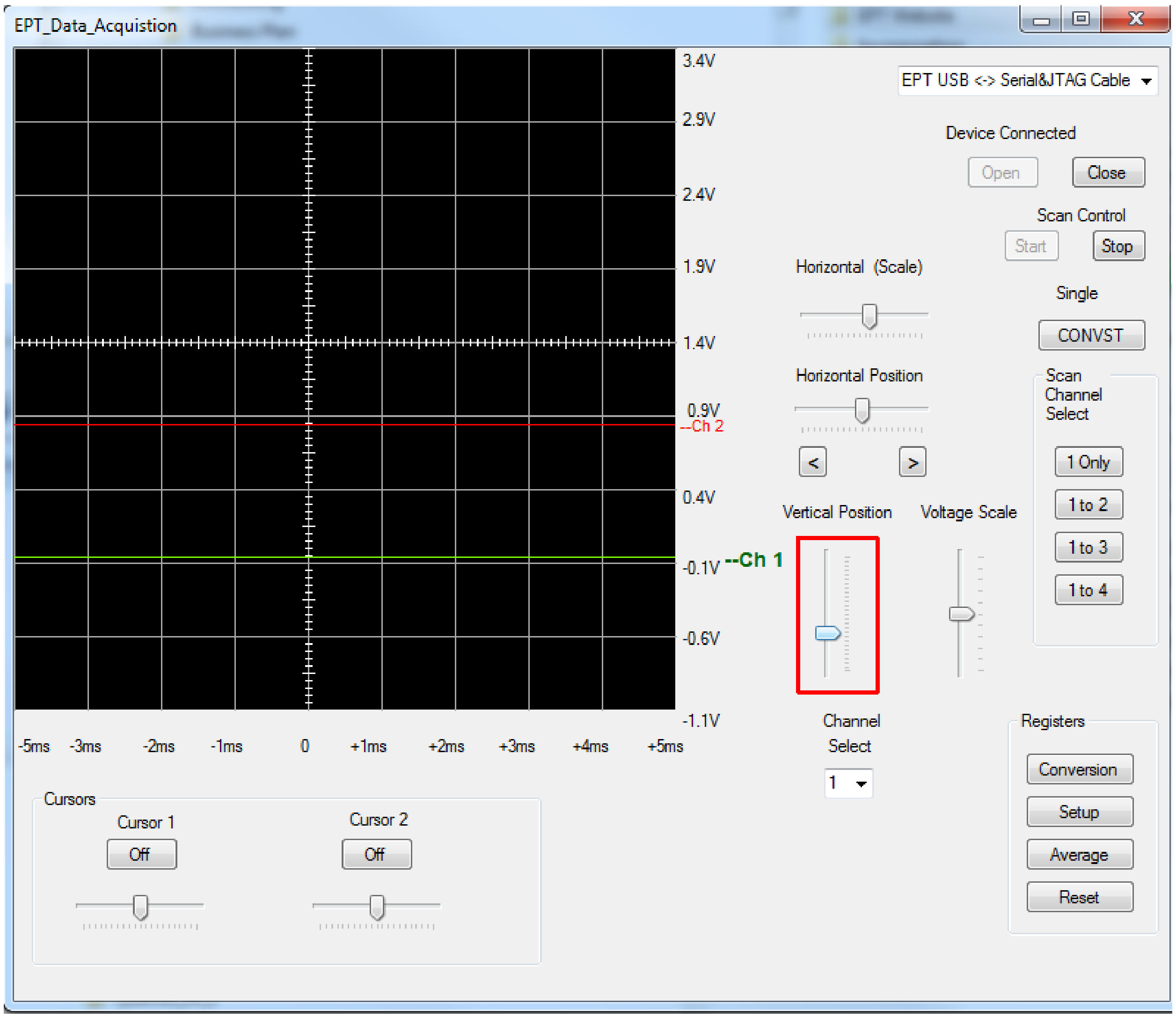
The data from the two channels will appear at the same latitude on the graph.



Next, locate the Channel Select drop down box and click on channel 1.

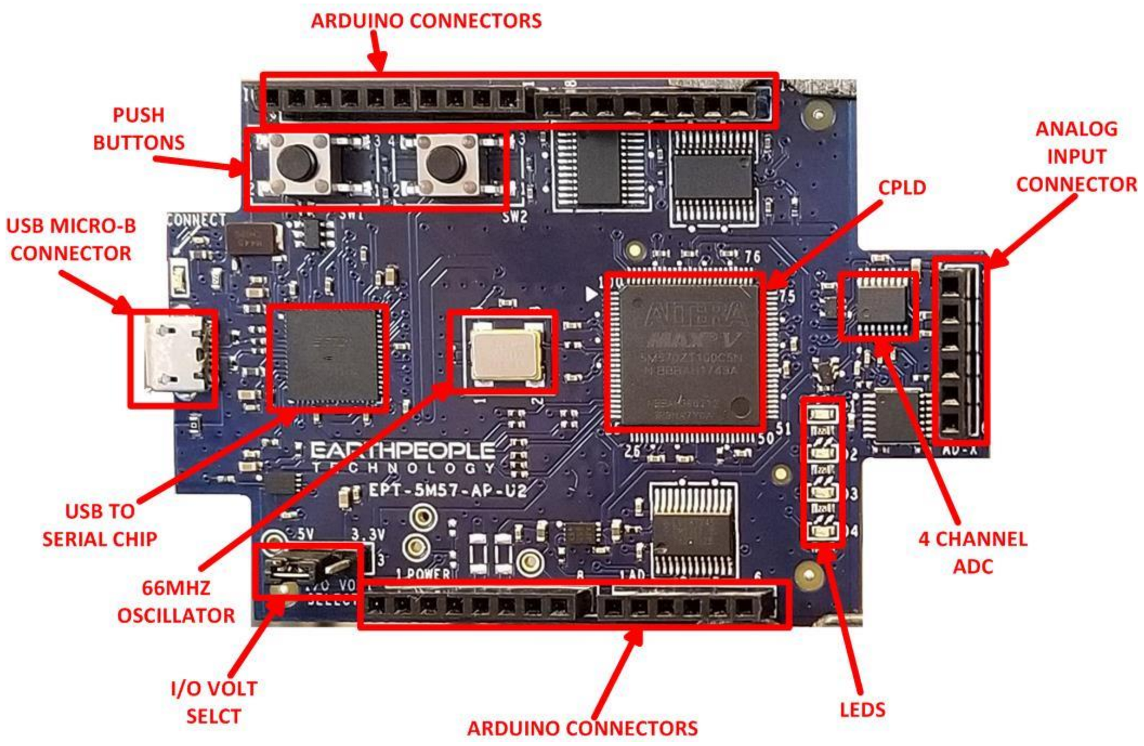


Locate the Vertical Position slider and pull it down. The channel 1 data will change position in the graph depending on where you move the slider. The voltage magnitude data also adjusts to indicate the magnitude of the data relative to the position of channel 1 data.

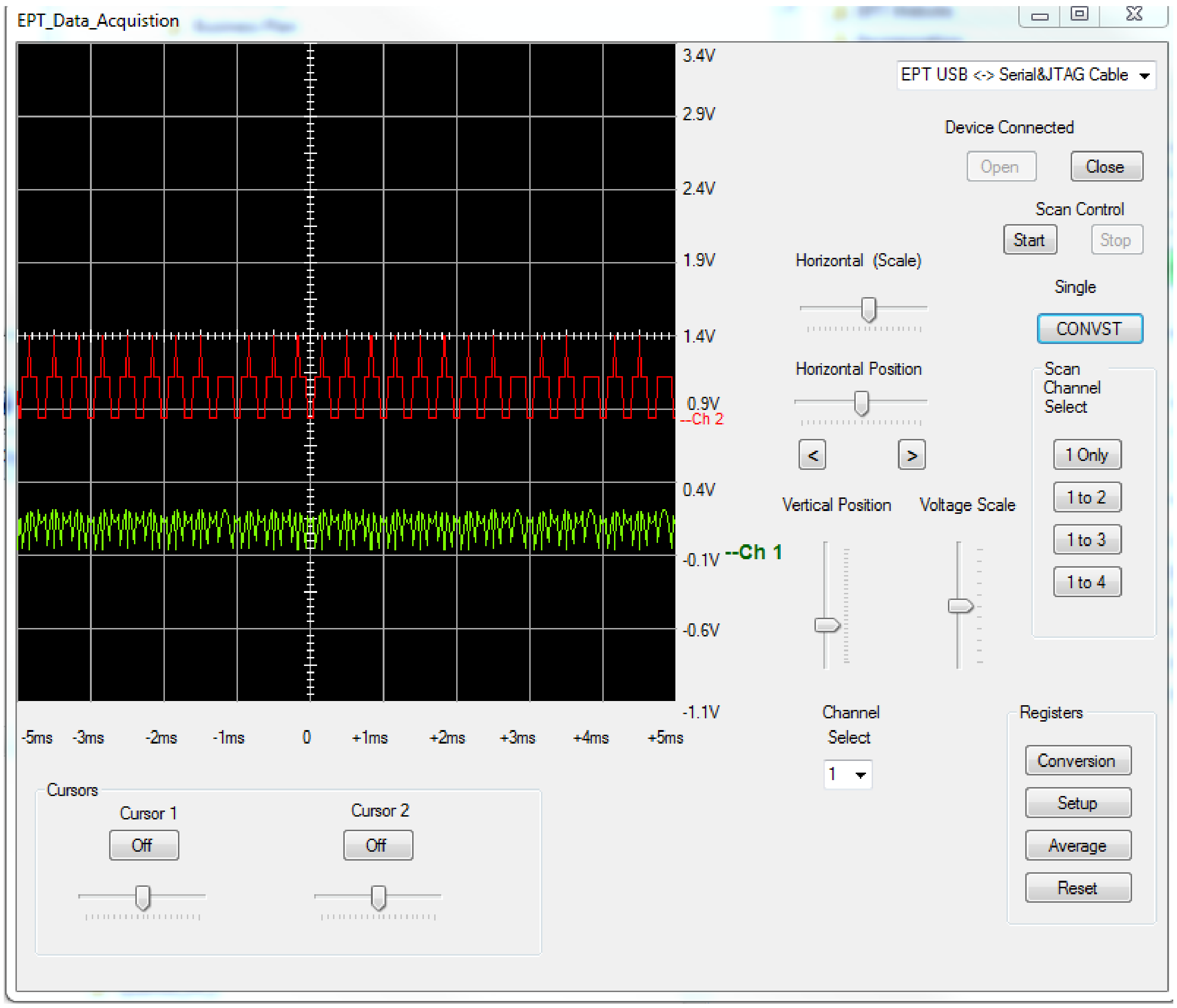


The selected channel will show up as a large icon. Its position indicates the zero position of the data. The magnitude information along the y-axis is only for the selected channel.

Then connect a signal to the channel 1 input on the UnoMax2.



If you don’t have a 0-5 Volt signal to connect to the UnoMax2, you can use your finger and touch it to the bottom of the Analog Input Connector. The ambient electricity from your body has just enough current to give the Analog inputs a deflection from zero.



Now the UnoMax and UnoProLyzer are ready to measure an 0-5VDC signals.

8 Developing Verilog Applications

The UnoMax CPLD development system includes tutorials for getting started with Verilog. The DVD includes a Verilog tutorial and a Modelsim tutorial. These two tutorials will assist with learning Verilog and getting up to speed with simulating Verilog designs in software. The Verilog simulation is

9 APPENDIX I

List of Abbreviations and Acronyms

EPT	Earth People Technology
FIFO	First In – First Out
FTDI	Future Technology Device International
HSP	Hyper Serial Port
I2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
PC	Personal Computer
CPLD	Complex Programmable Logic Device
USB	Universal Serial Bus

APPENDIX II

Details of the Altera EPM570 CPLD