EARTH PEOPLE TECHNOLOGY, Inc

USB-CPLD DEVELOPMENT SYSTEM FOR THE ARDUINO MEGA
User Manual

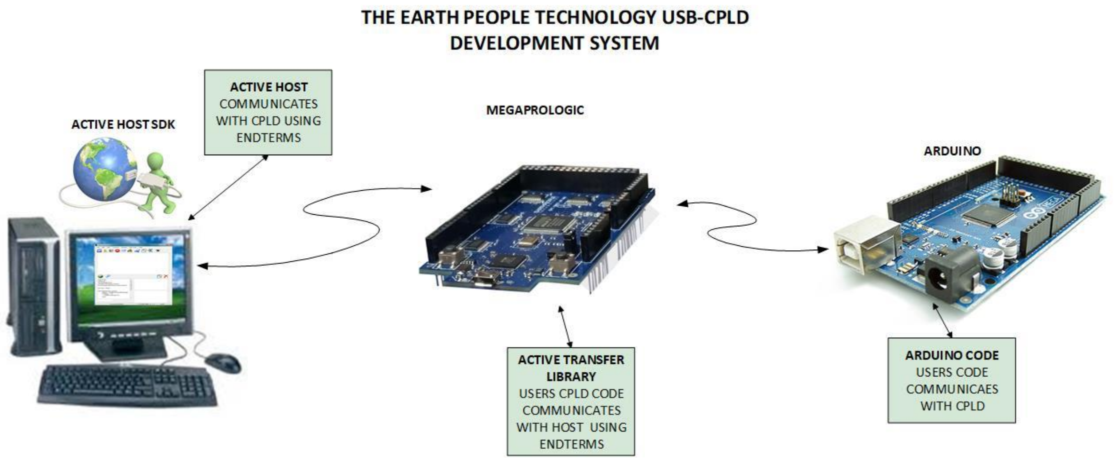
The EPT MegaProLogic development system provides an innovative method of developing and debugging programmable logic code. It also provides a high speed data transfer mechanism between an Arduino board and a host PC. The MegaProLogic development system provides a convenient, user-friendly work flow by connecting seamlessly with Intel's Quartus Prime software. The user will develop the code in the Quartus Prime environment on a Windows Personal Computer. The programmable logic code is loaded into the CPLD using only the Quartus Prime Programmer tool and a standard USB cable. The Active Host SDK provides a highly configurable communications interface between Arduino and host. It 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. No scheduling USB transfers, USB driver interface or inf file changes are needed. The EPT USB-CPLD development system is a unique combination of hardware and software.

http://www.earthpeopletechnology.com/

Table of Contents

1 Introduction and General Description

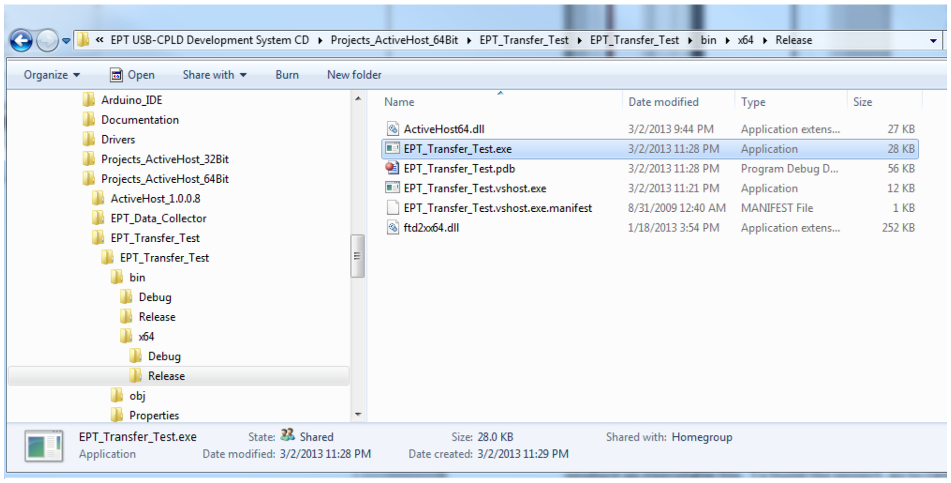
The Earth People Technology USB-CPLD development system hardware consists of a High Speed (480 Mb/s) USB to Serial bus chip and a CPLD. The USB interface provides both JTAG programming of the CPLD and a High Speed transfer path. The software consists of the Active Host SDK for the PC. The firmware includes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


THE EARTH PEOPLE TECHNOLOGY USB-CPLD DEVELOPMENT SYSTEM
ACTIVE HOST SDK
ACTIVE HOST COMMUNICATES WITH CPLD USING ENDTERMS
MEGAPROLOGIC
ARDUINO
ACTIVE TRANSFER LIBRARY USERS CPLD CODE COMMUNICATES WITH HOST USING ENDTERMS
ARDUINO CODE USERS CODE COMMUNICAES WITH CPLD

The user's Arduino code is developed to perform particular functions required by the user (such as reading a temperature sensor). The code is downloaded to the microcontroller using the Arduino IDE system provided as part of the microcontroller development system. The EPT USB-CPLD 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 MegaProLogic 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 MEGAPROLOGIC 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

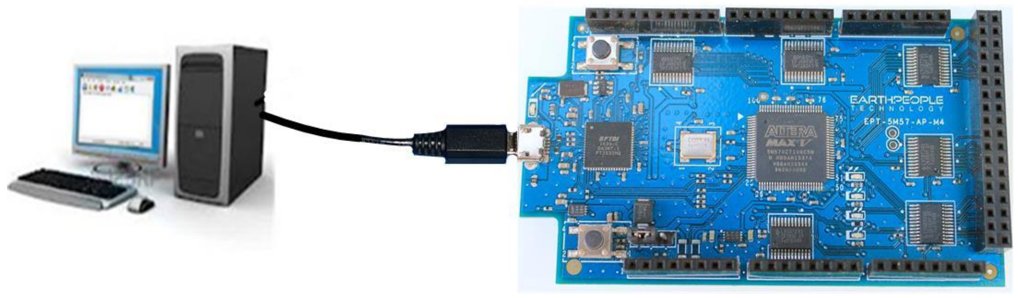
- Arduino_IDE
- Documentation
- Drivers
- Projects_ActiveHost_32Bit
- Projects_ActiveHost_64Bit
  - ActiveHost_1.0.0.8
  - EPT_Data_Collector
  - EPT_Transfer_Test
    - EPT_Transfer_Test
      - bin
        - Debug
        - Release
        - x64
          - Debug
          - Release
      - obj
  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

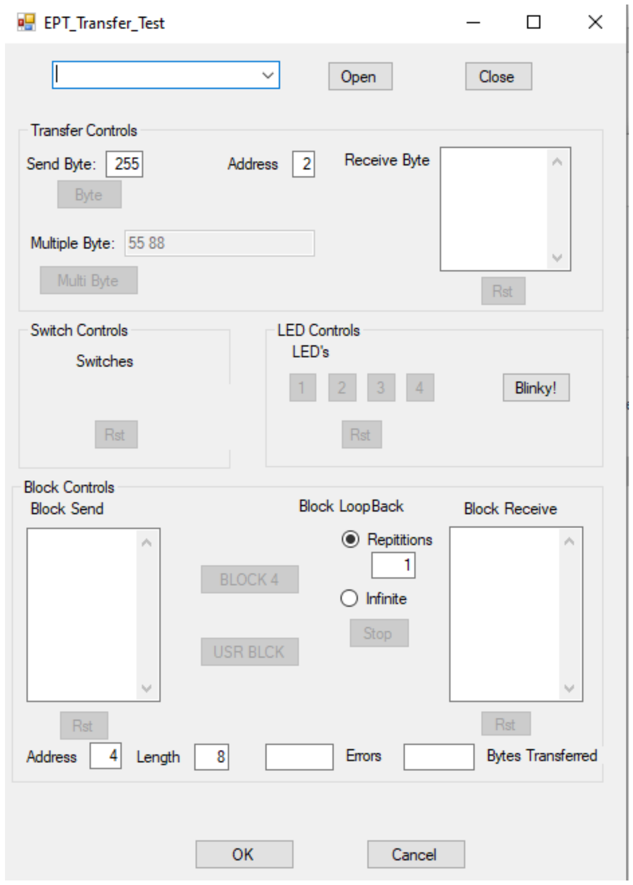
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	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

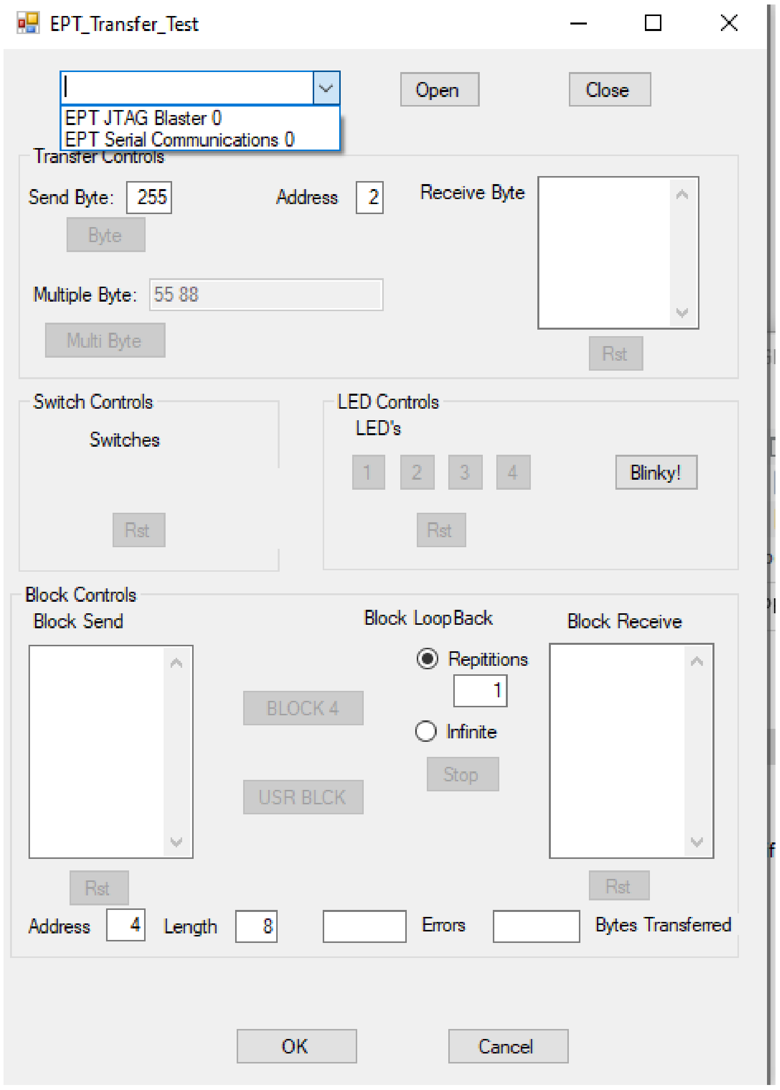
To test drive the application, connect the MEGAPROLOGIC to the Windows PC using Type A to Micro B USB cable. Load the driver for the board. See the section EPT Drivers for instructions on loading the MegaProLogic driver. If the USB driver fails to load, the Windows OS will indicate that no driver was loaded for the device.

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



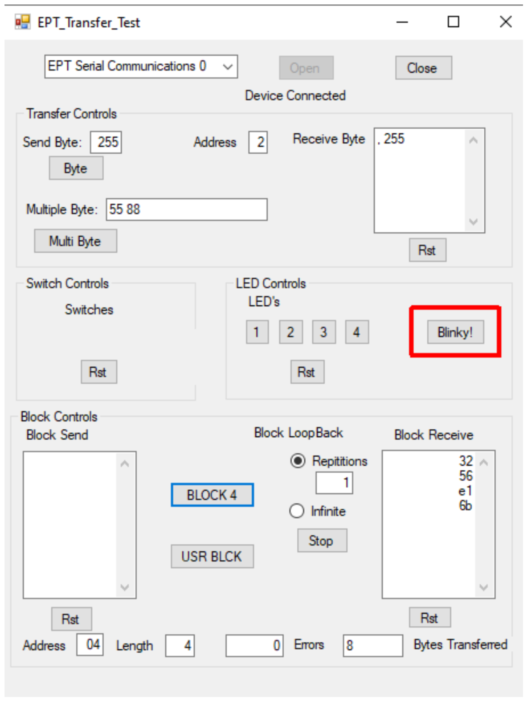


With the application loaded, select the EPT Serial Communications x board from the dropdown combo box and click on the "Open" button.

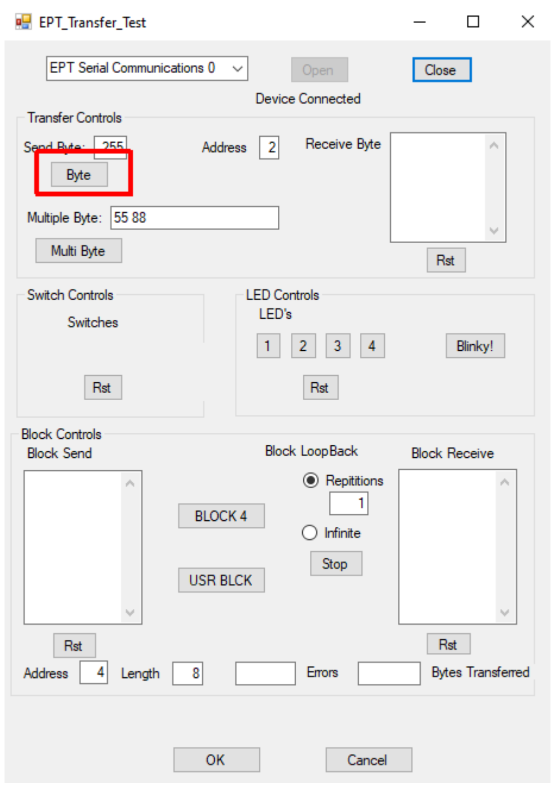


Leave the Address set at 2 for the Transfer Controls Group. And, leave the Address set at 4 for the Block Controls Group.

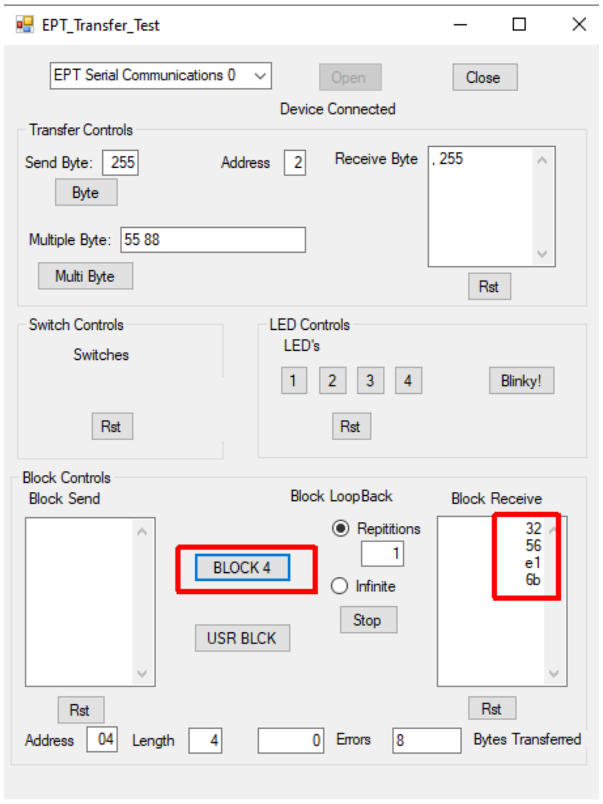
Click on one of the LED buttons in the middle of the window. The corresponding LED on the MegaProLogic board should light up. Clink on the Blinky button for a light show.



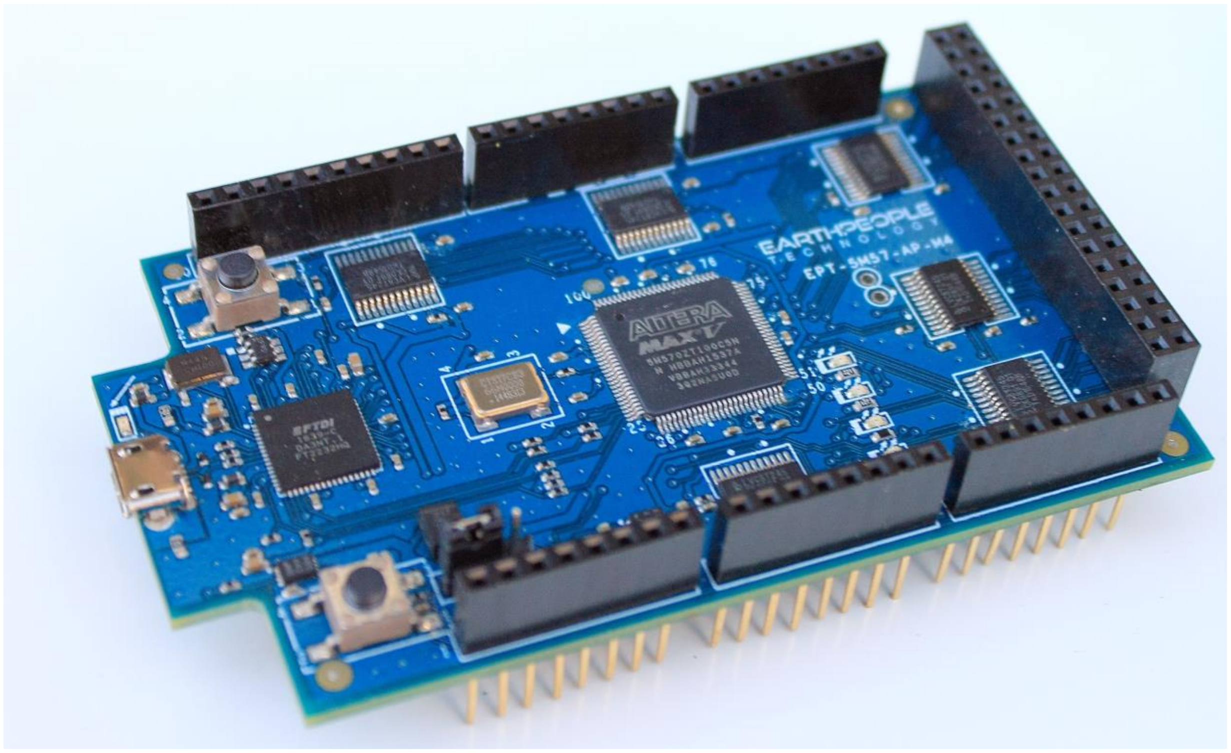
To exercise the Single Byte Transfer EndTerm, click the “Byte” button in the Transfer Controls group. Type in several numbers separated by a space and less 256 into the Multiple Byte textbox. Then hit the Multi Byte button. The numbers appear in the Receive Byte textbox.



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 MegaProLogic to view the Switch Controls in action.

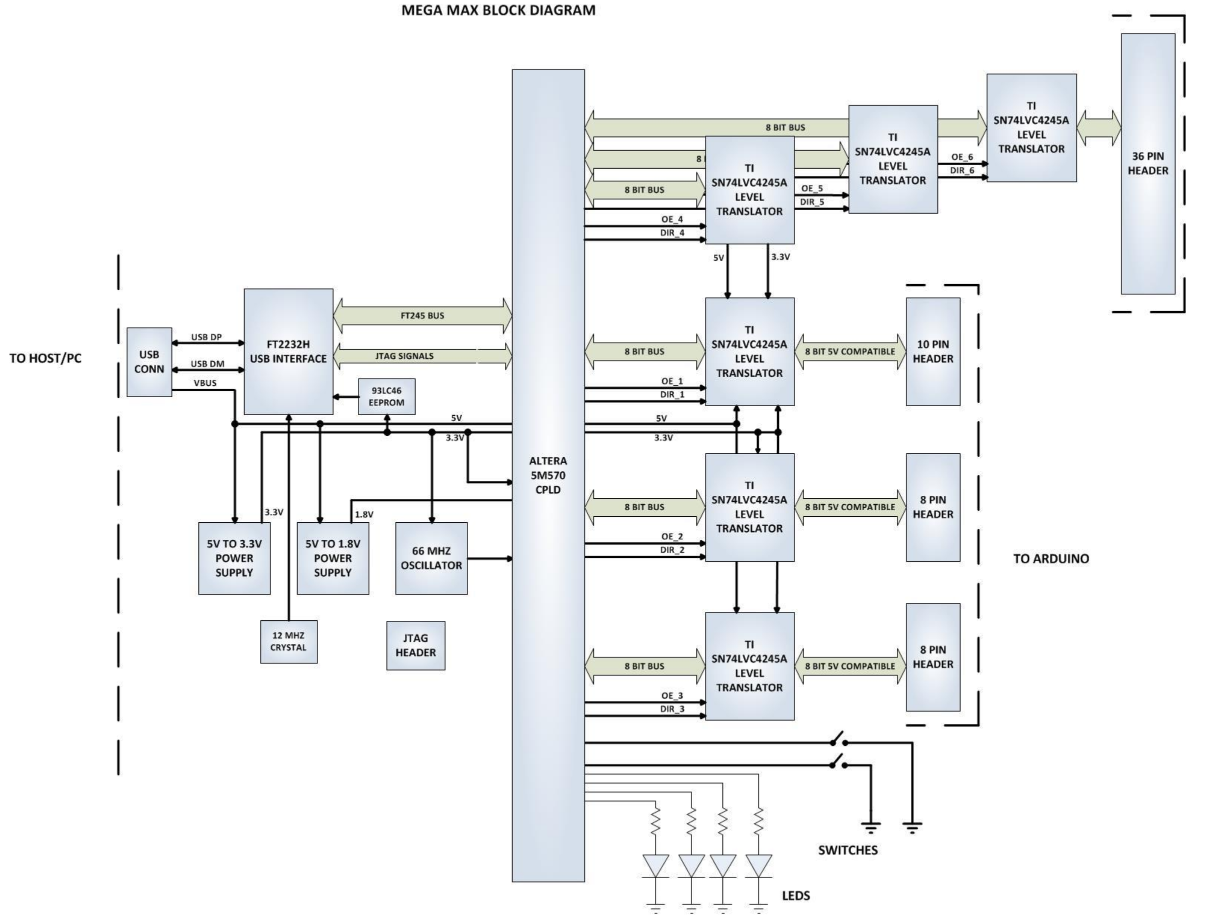


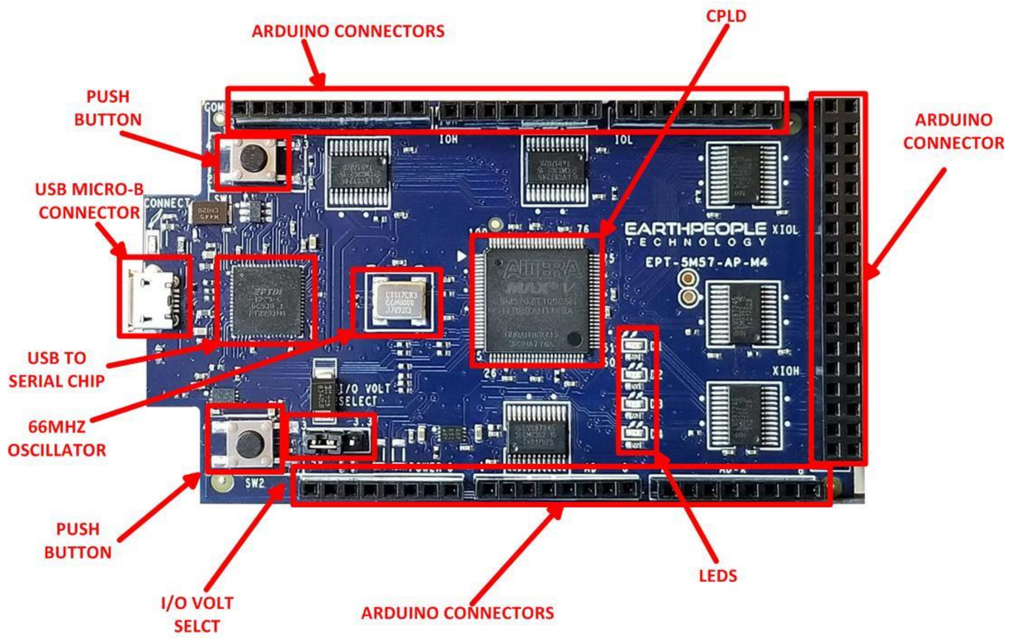
1.2 MEGAPROLOGIC

The MegaProLogic board is equipped with an Altera EPM570 CPLD; which is programmed using the Quartus Prime Lite 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. Twenty Four I/O's from the CPLD are attached to three 8 bit transceivers to provide 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 5M570 in the TQFP 100 pin package
- FT2232H USB to Serial Interface chip
- 66 MHz oscillator for driving USB data transfers and users code
- Five 74LVC245 bidirectional voltage translator/bus transceiver
- 32 user Input/Outputs
- Four Green LED's accessible by the user
- Two PCB switches accessible by the user
- All connectors to stack into the Arduino 2560 Mega

MEGA MAX BLOCK DIAGRAM





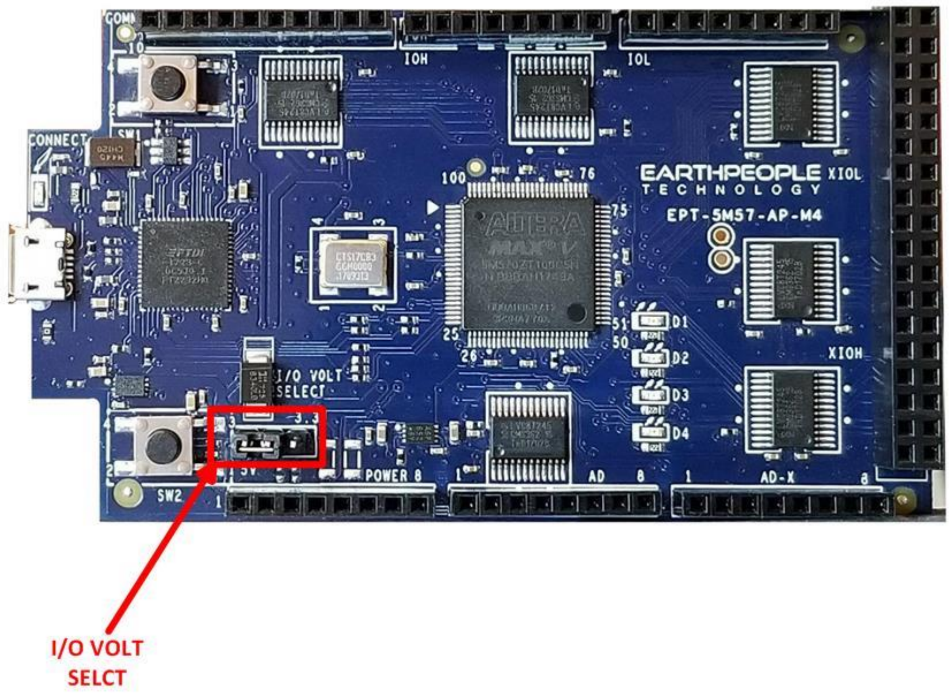
1.2.1 Serial USB Communications

The MegaProLogic USB-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 0.1 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 a single COM Port. 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 MegaProLogic for all of its +3.3V power budget.

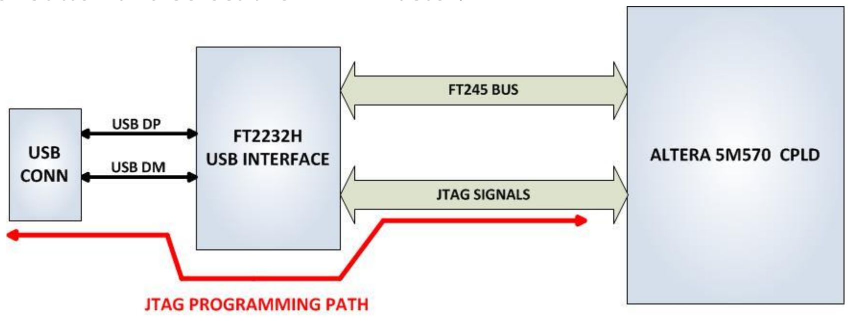
1.2.2 Inputs and Outputs

There are 32 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”.



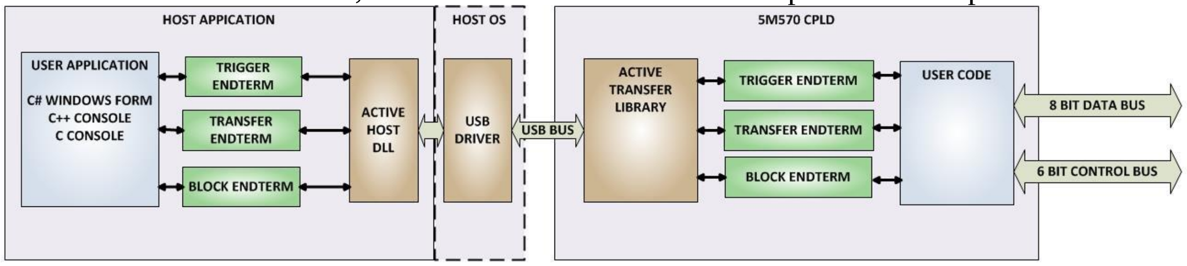
1.2.3 JTAG

The MegaProLogic 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.



1.3 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.

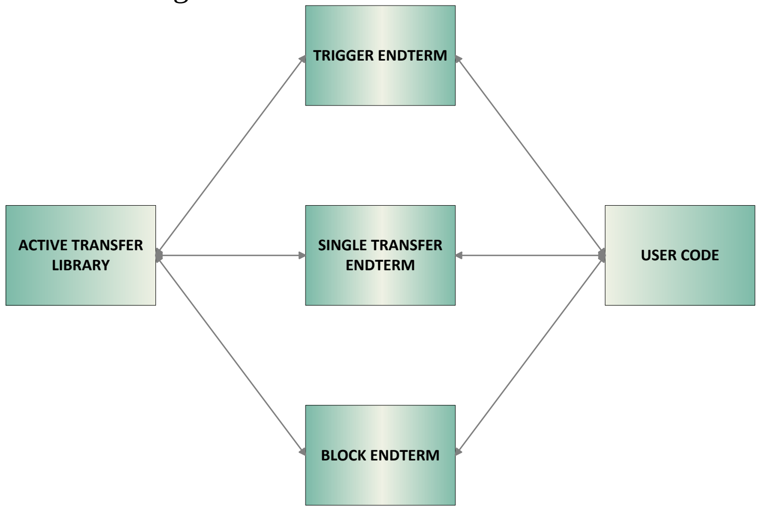
Receiving data from the CPLD is made simple by Active Host. Active Host transfers data from the CPLD as soon as it is available. It stores the transferred data into circular buffer. When the transfer is complete, Active Host invokes a callback function which is registered in the users application. This 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.

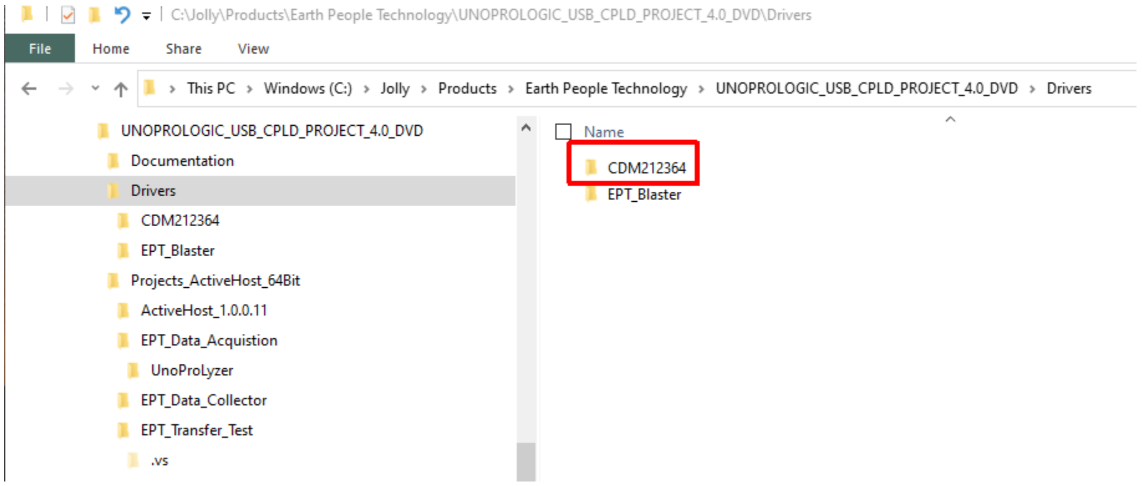
2 EPT Drivers

The MegaProLogic 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 MegaProLogic 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 MegaProLogic Development System CD using Windows Explorer.

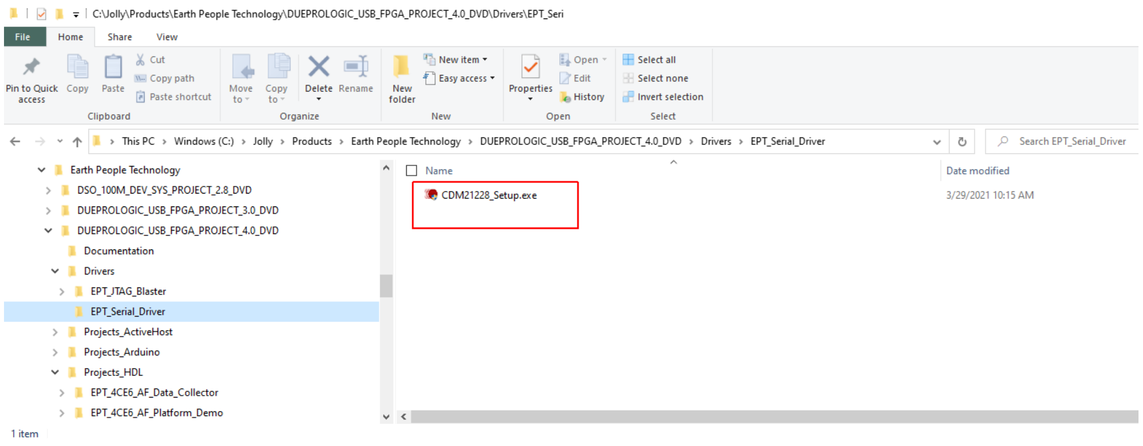


C:\Jolly\Products\Earth People Technology\UNOPROLOGIC_USB_CPLD_PROJECT_4.0_DVD\Drivers

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

CDM212364
EPT_Blaster



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

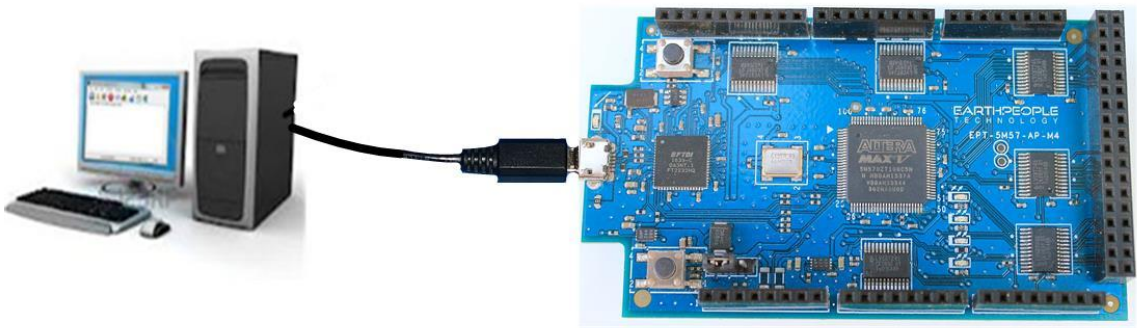
This PC › Windows (C:) › Jolly › Products › Earth People Technology › DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD › Drivers › EPT_Serial_Driver

- Earth People Technology
  - DSO_100M_DEV_SYS_PROJECT_2.8_DVD
  - DUEPROLOGIC_USB_FPGA_PROJECT_3.0_DVD
  - DUEPROLOGIC_USB_FPGA_PROJECT_4.0_DVD
    - Documentation
    - Drivers
      - EPT_JTAG_Blaster
      - EPT_Serial_Driver
    - Projects_ActiveHost
    - Projects_Arduino
    - Projects_HDL
      - EPT_4CE6_AF_Data_Collector
      - EPT_4CE6_AF_Platform_Demo

CDM21228_Setup.exe 3/29/2021 10:15 AM

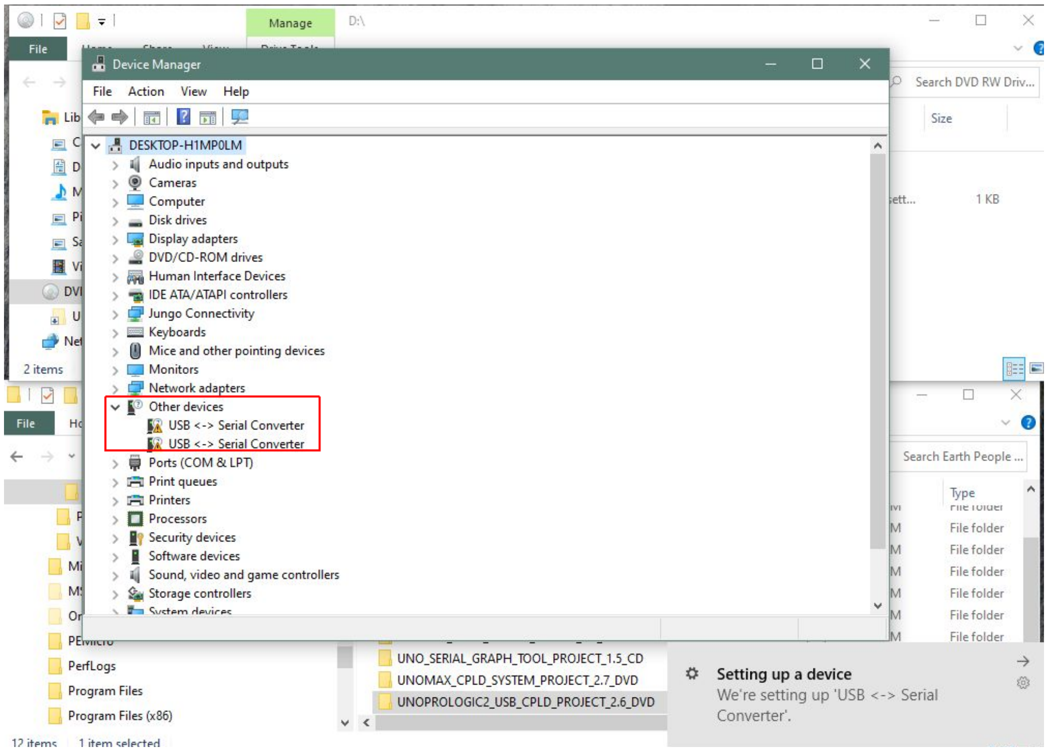
Double click on the *.exe file and select the default settings when the software tool queries the user.

Plug in the MegaProLogic device into an available USB port.



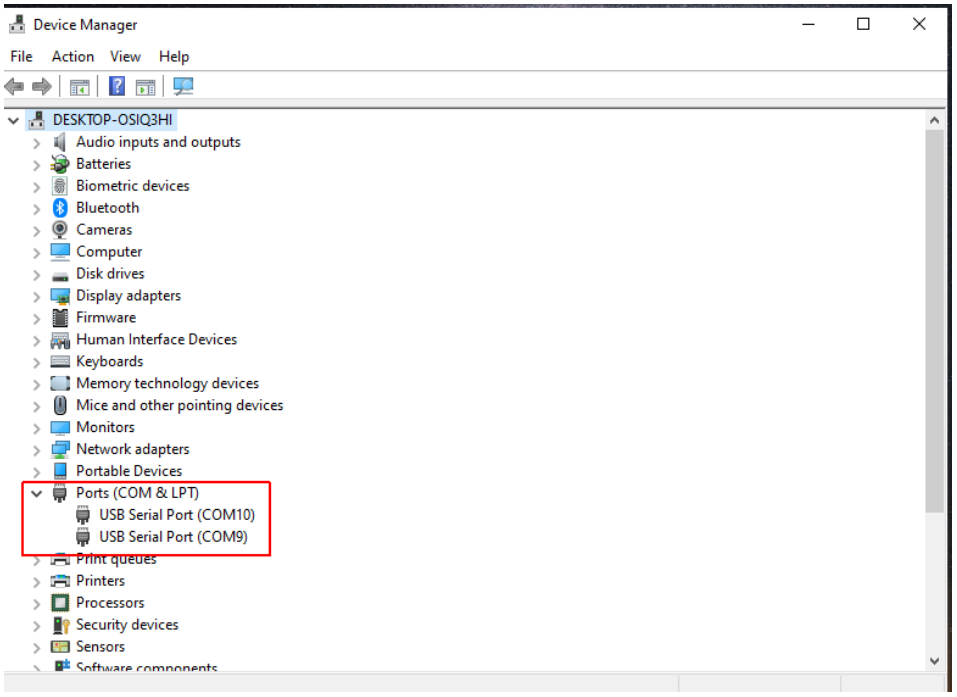
Windows will attempt to locate a driver for the USB device. When it does not find one, it will report a error, "Device driver software was not successfully installed". Ignore this error.

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 MegaProLogic. The MegaProLogic 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 MegaProLogic can be used for programming and USB data transfers.

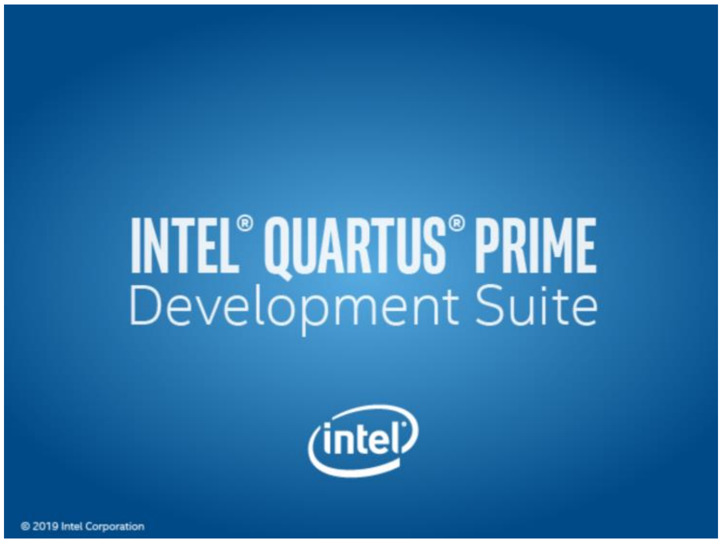
2.2 JTAG DLL Insert to Quartus Prime Lite

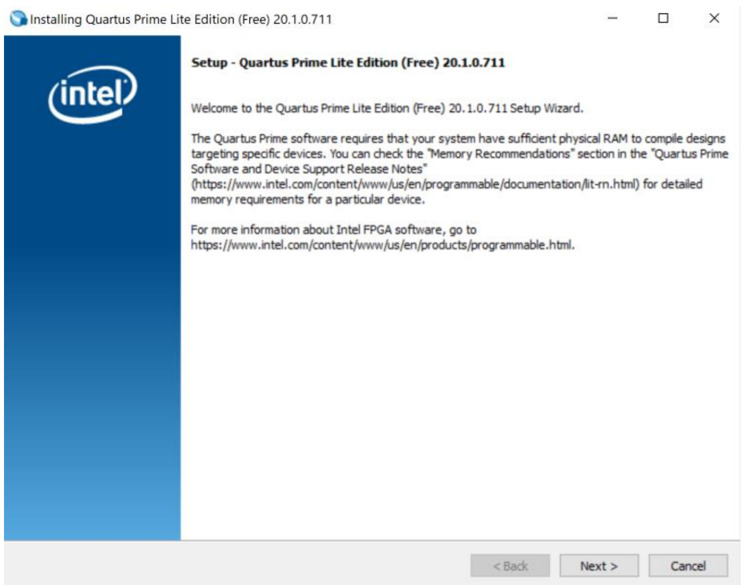
The JTAG DLL Insert to Quartus Prime Lite allows the Programmer Tool under Quartus to recognize the MegaProLogic. The MegaProLogic can then be selected and perform programming of the CPLD. The file, jtag_hw_mbftdi_blaster.dll must be placed into the folder that hosts the jtag_server for Quartus.

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 Lite 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"

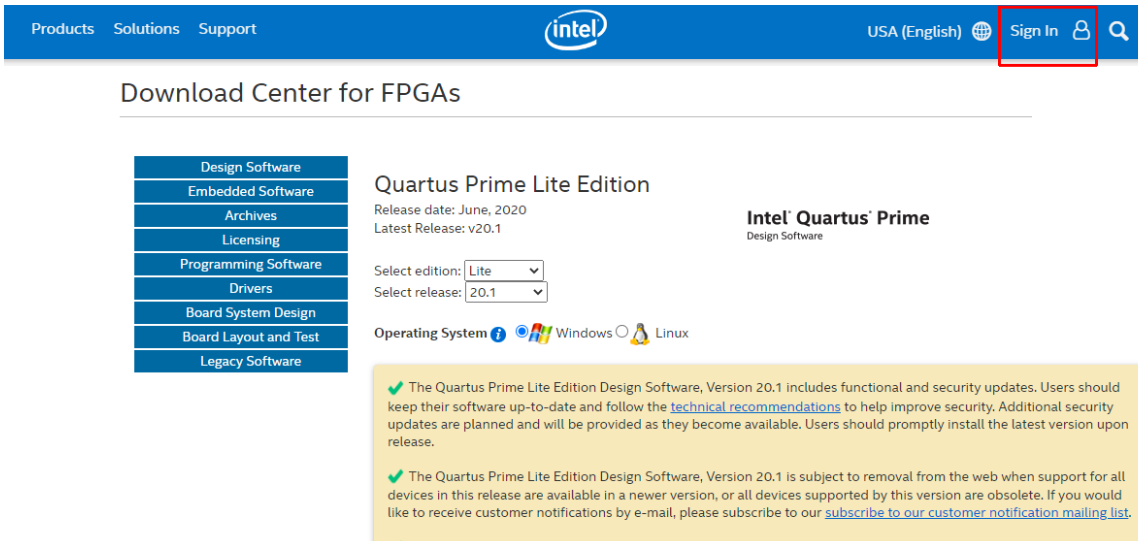
Next, skip the "Download Quartus" section. Go down to the "Quartus Installer" section to complete the Quartus installation.

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.



Download Center for FPGAs

Quartus Prime Lite Edition
Release date: June, 2020
Latest Release: v20.1

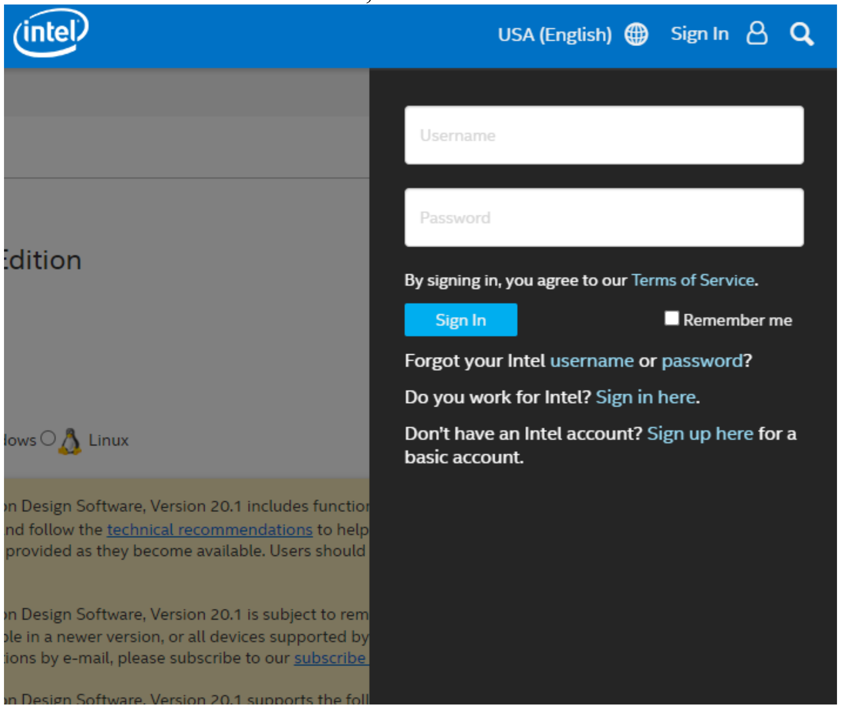
Select edition: Lite
Select release: 20.1

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 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?"



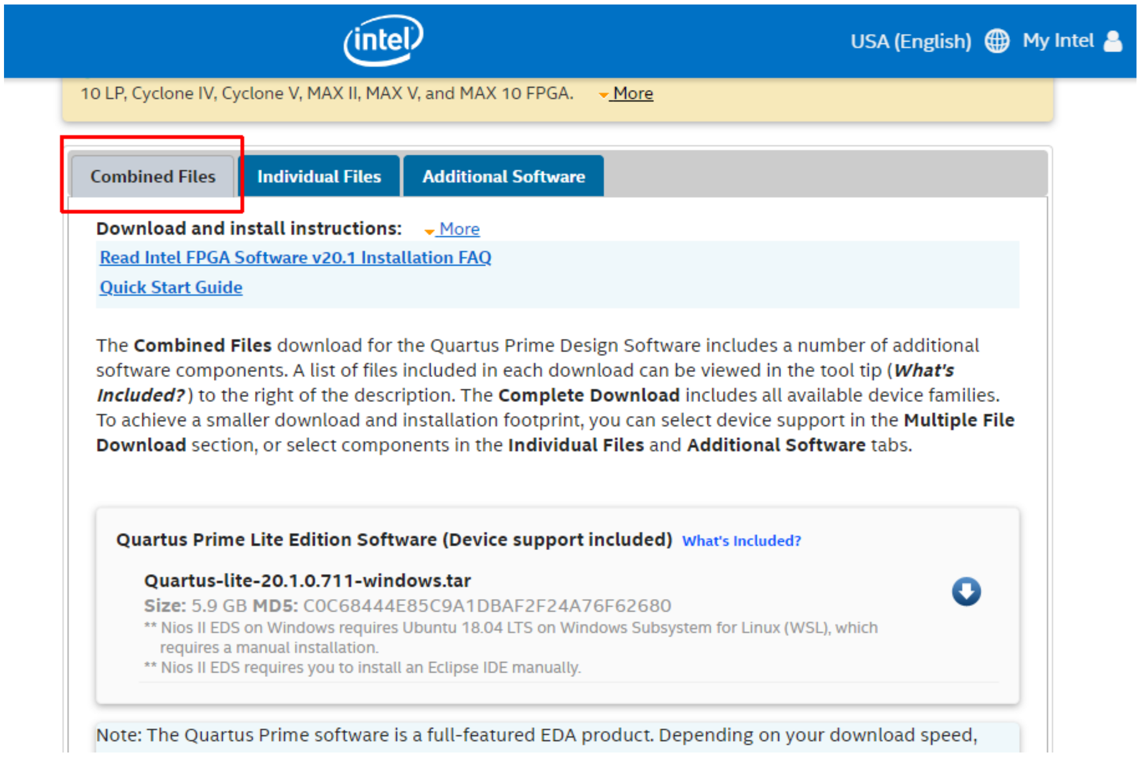
By signing in, you agree to our Terms of Service.

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.

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.



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

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

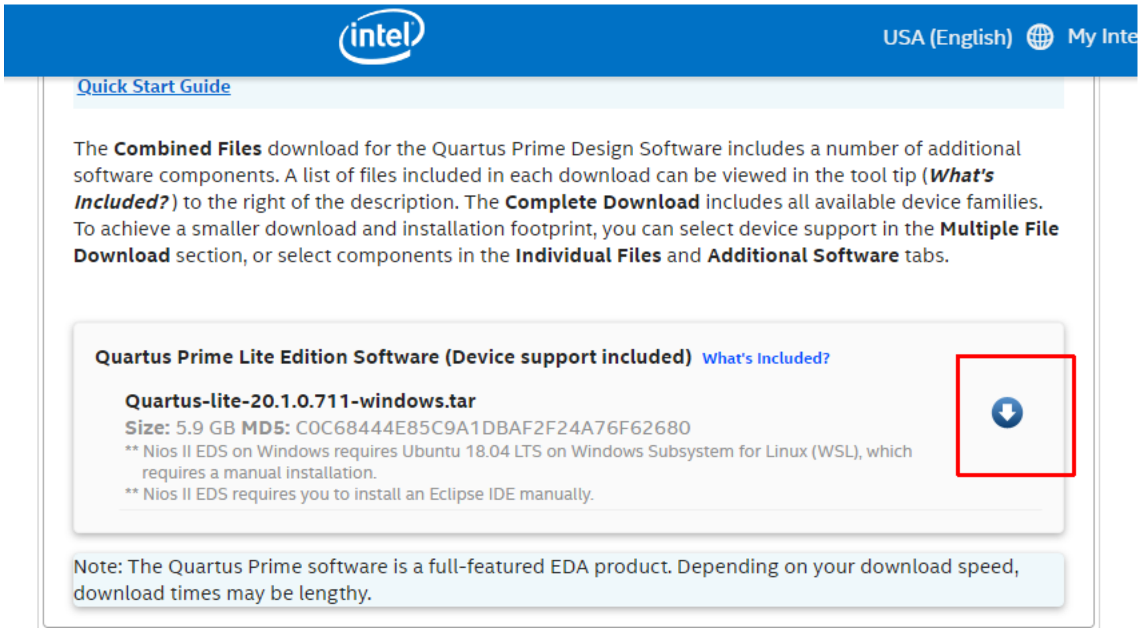
Size: 5.9 GB **MD5:** C0C68444E85C9A1DBAF2F24A76F62680

** 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.



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

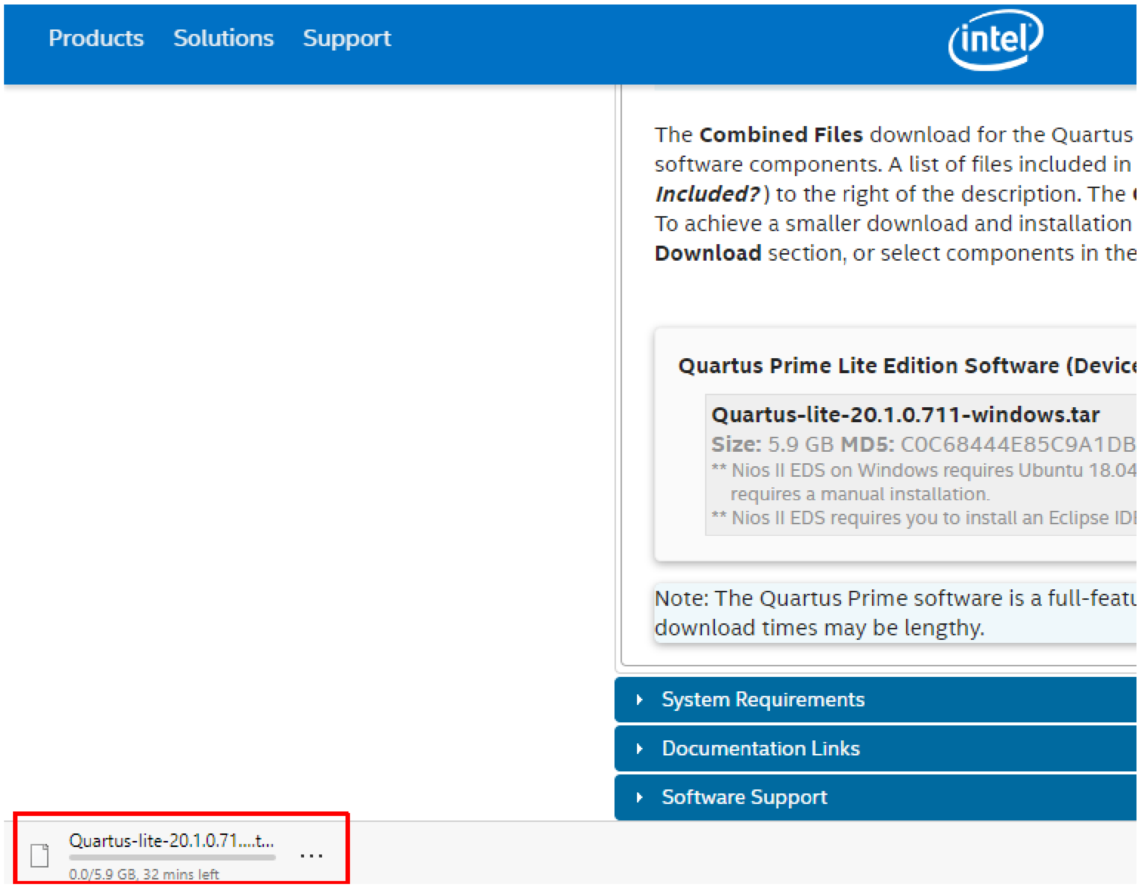
Size: 5.9 GB **MD5:** C0C68444E85C9A1DBAF2F24A76F62680

** 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.



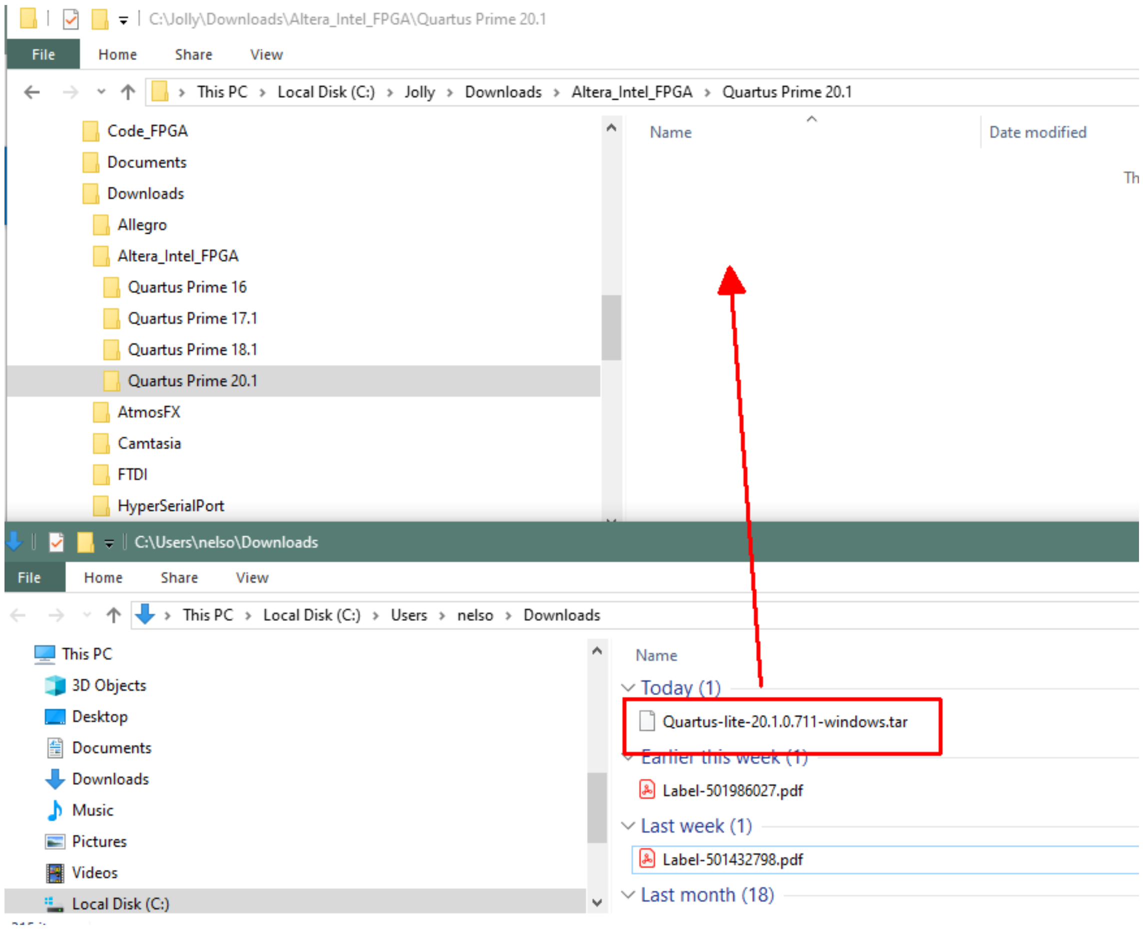
The **Combined Files** download for the Quartus software components. A list of files included in ***Included?***) to the right of the description. The To achieve a smaller download and installation **Download** section, or select components in the

Quartus Prime Lite Edition Software (Device

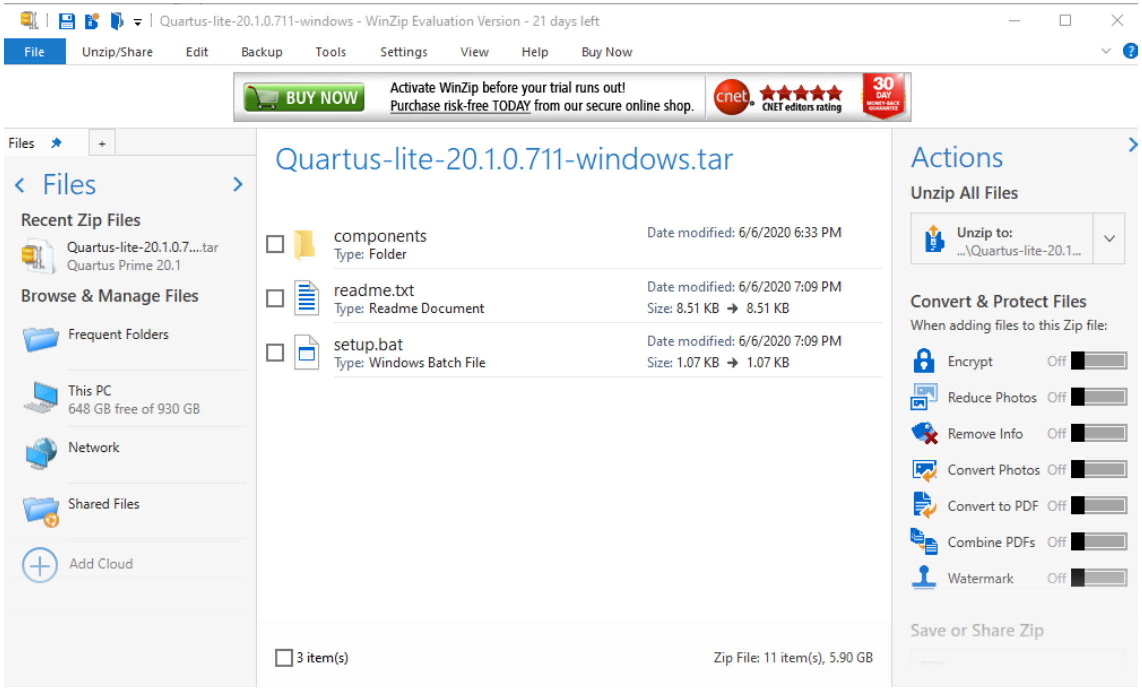
Quartus-lite-20.1.0.711-windows.tar
Size: 5.9 GB **MD5:** C0C68444E85C9A1DB
** Nios II EDS on Windows requires Ubuntu 18.04 requires a manual installation.
** Nios II EDS requires you to install an Eclipse IDI

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.



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

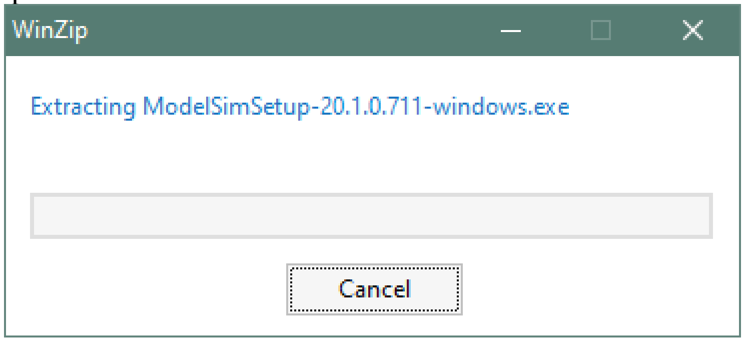


Quartus-lite-20.1.0.711-windows.tar

Name	Type	Date modified	Size
components	Folder	6/6/2020 6:33 PM	
readme.txt	Readme Document	6/6/2020 7:09 PM	8.51 KB → 8.51 KB
setup.bat	Windows Batch File	6/6/2020 7:09 PM	1.07 KB → 1.07 KB

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

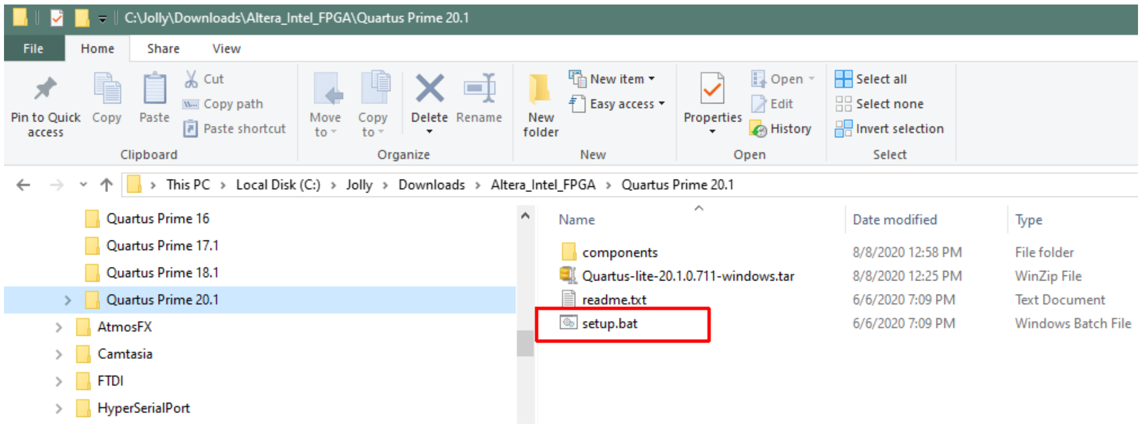
The tool will unpack all files.


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.

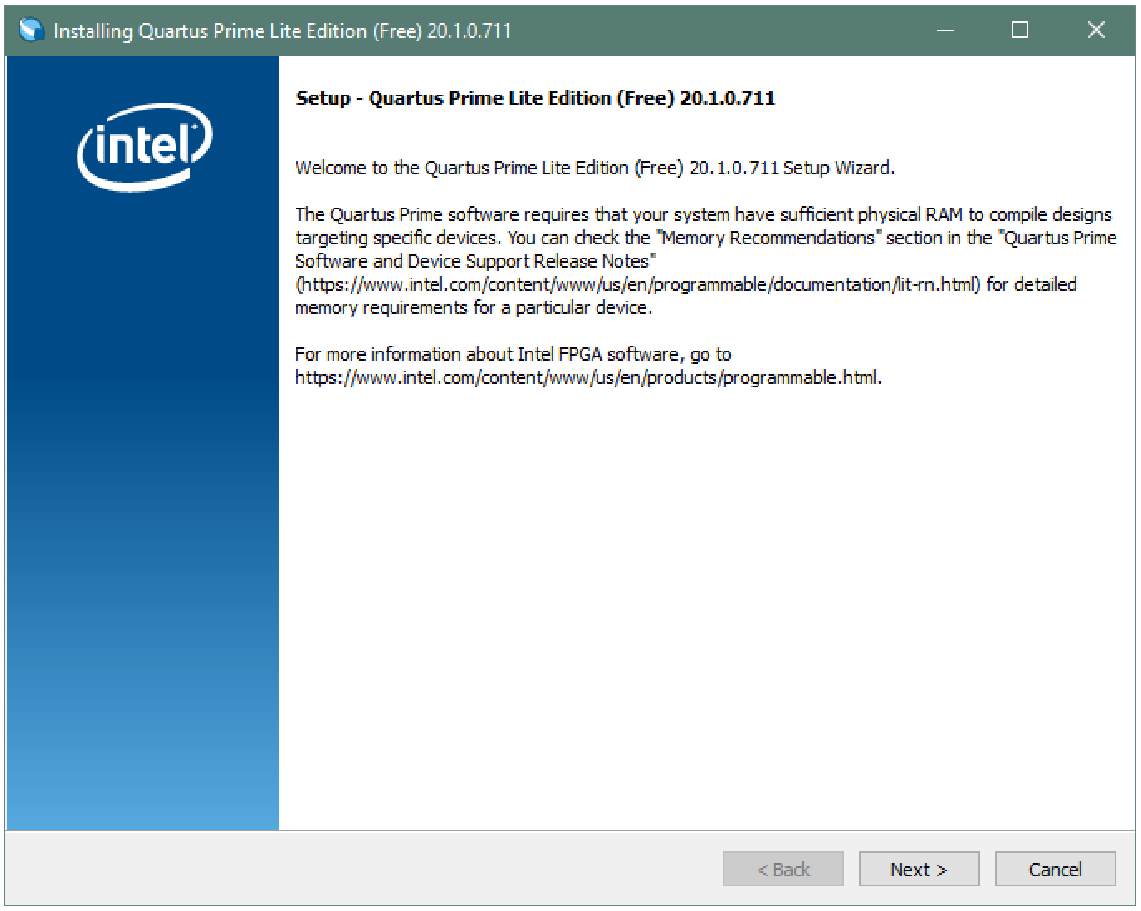


C:\Jolly\Downloads\Altera_Intel_FPGA\Quartus Prime 20.1

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

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.



Installing Quartus Prime Lite Edition (Free) 20.1.0.711

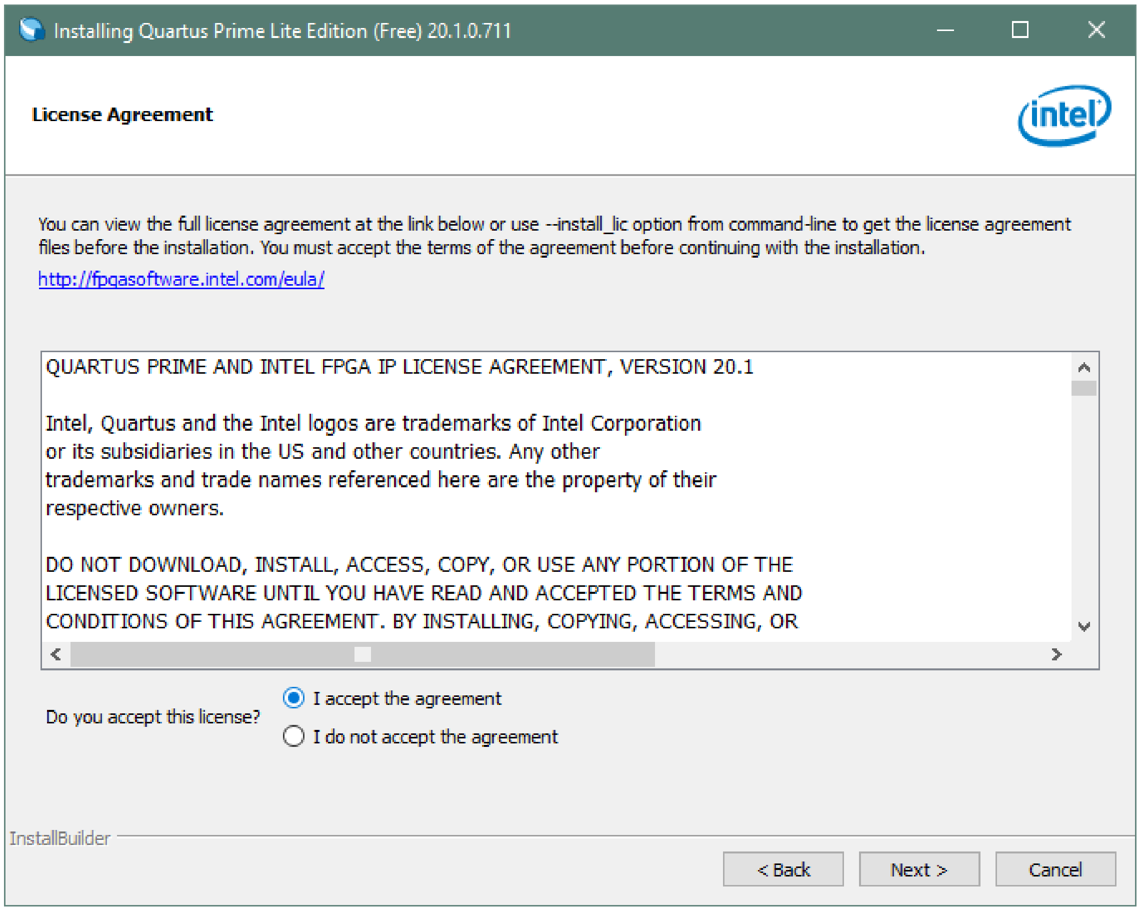
Setup - Quartus Prime Lite Edition (Free) 20.1.0.711

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

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

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

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


Installing Quartus Prime Lite Edition (Free) 20.1.0.711

License Agreement

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

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

QUARTUS PRIME AND INTEL FPGA IP LICENSE AGREEMENT, VERSION 20.1

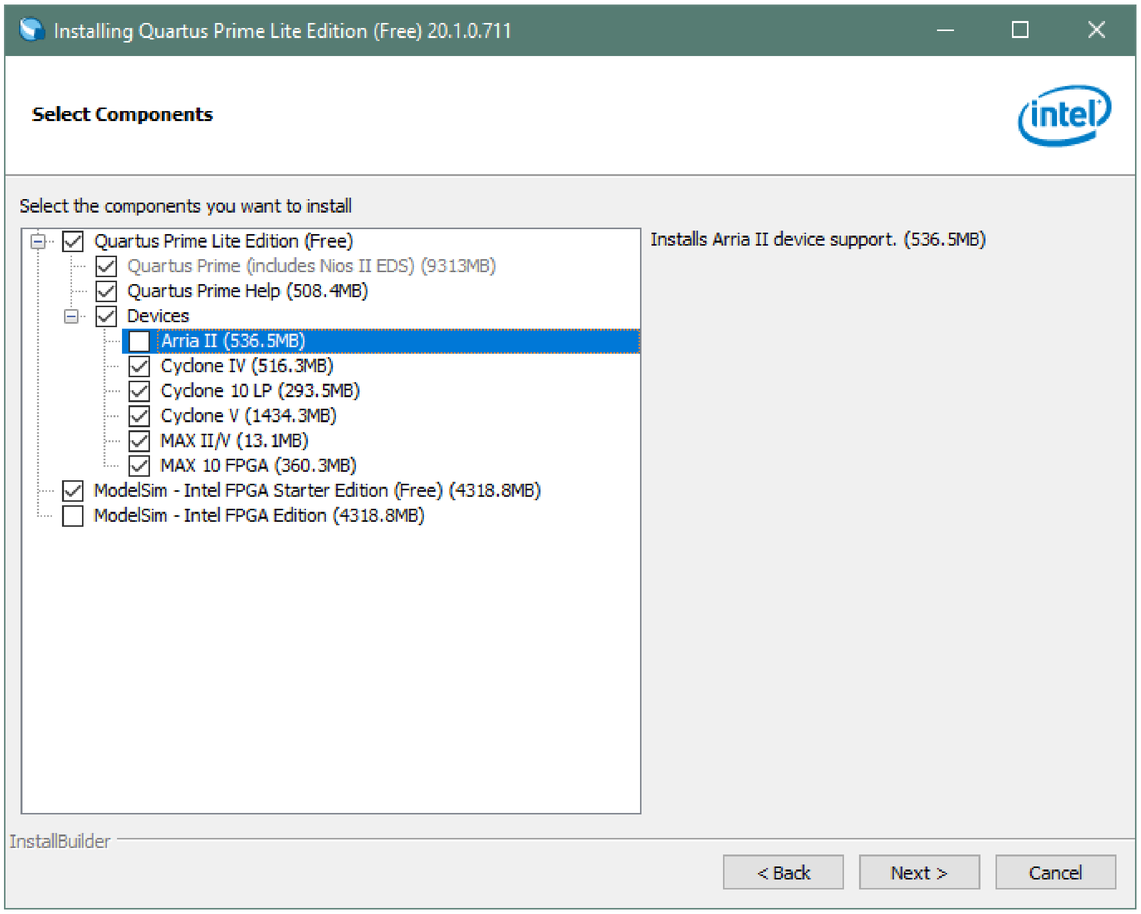
Intel, Quartus and the Intel logos are trademarks of Intel Corporation or its subsidiaries in the US and other countries. Any other trademarks and trade names referenced here are the property of their respective owners.

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

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

Click “Next” and accept the defaults.

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

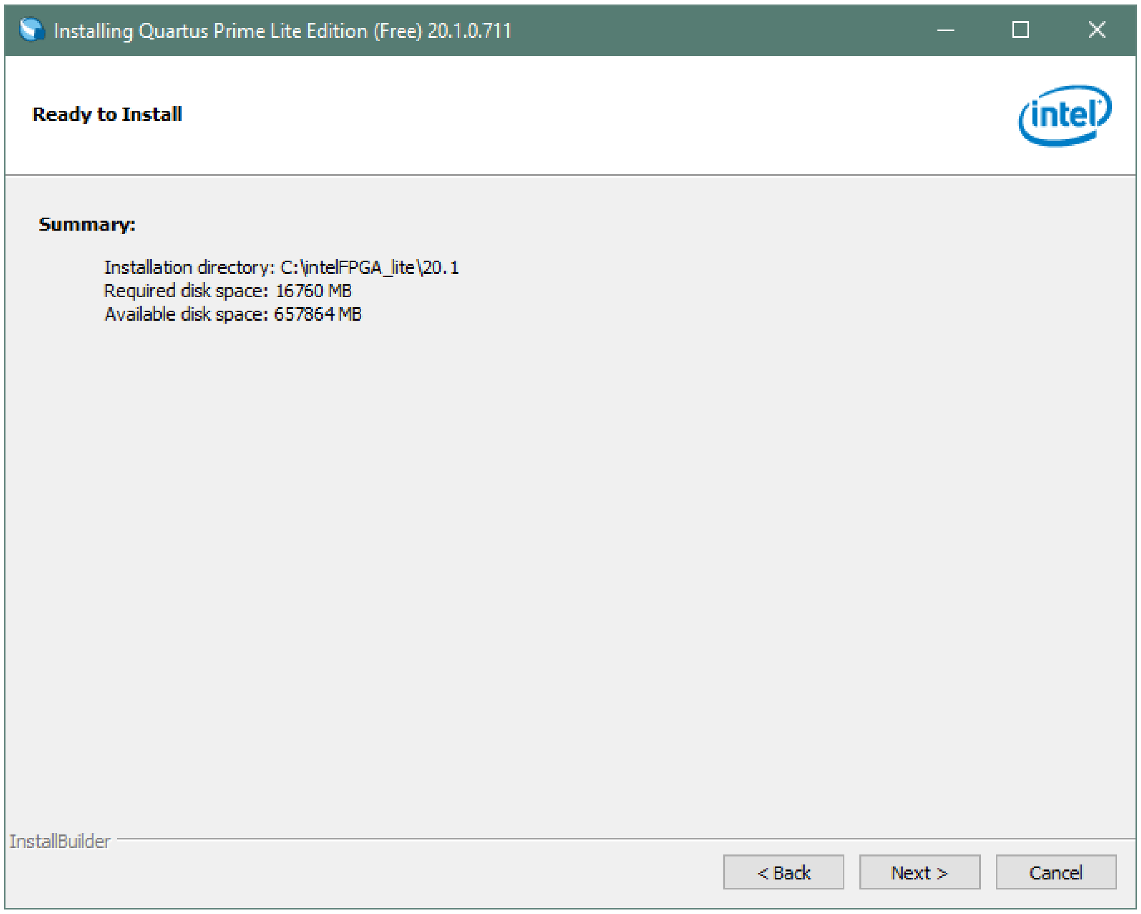


Select the components you want to install

- [✓] 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)

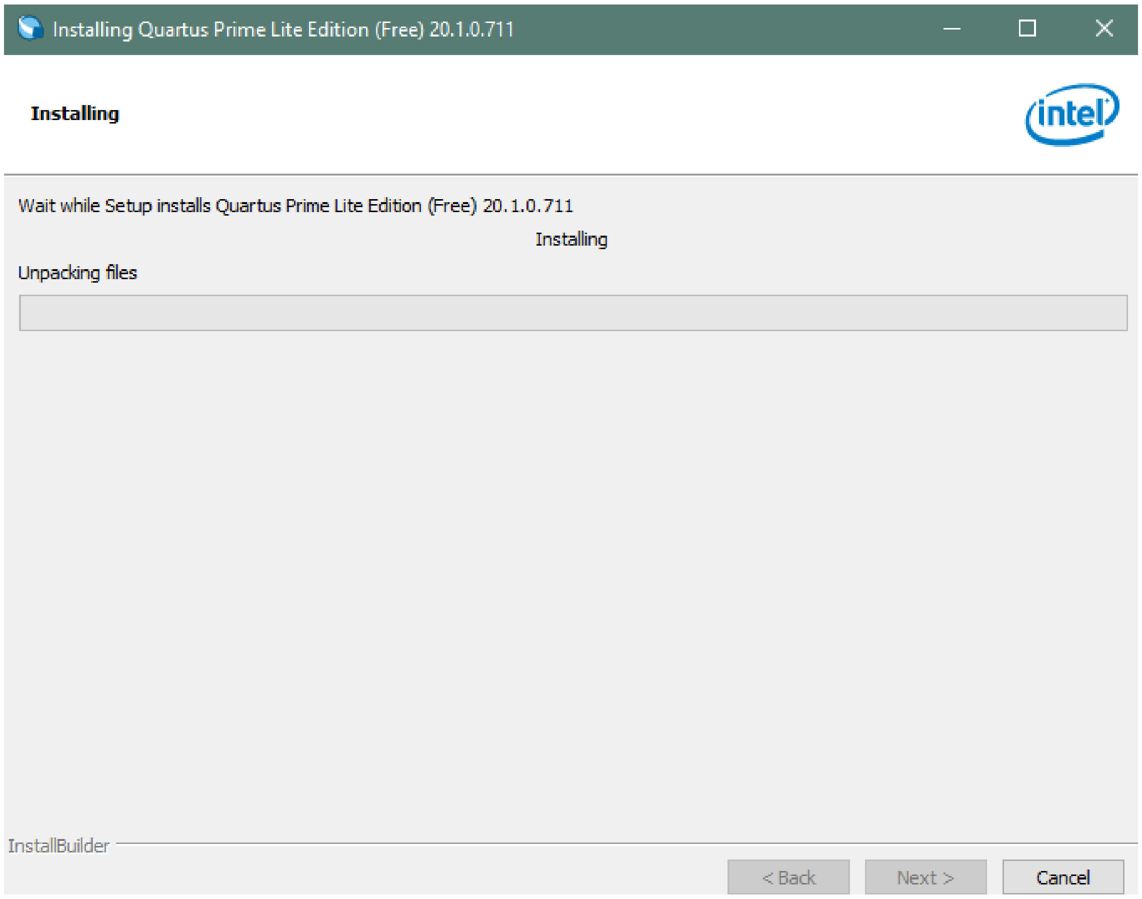
Click “Next” to accept the defaults



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

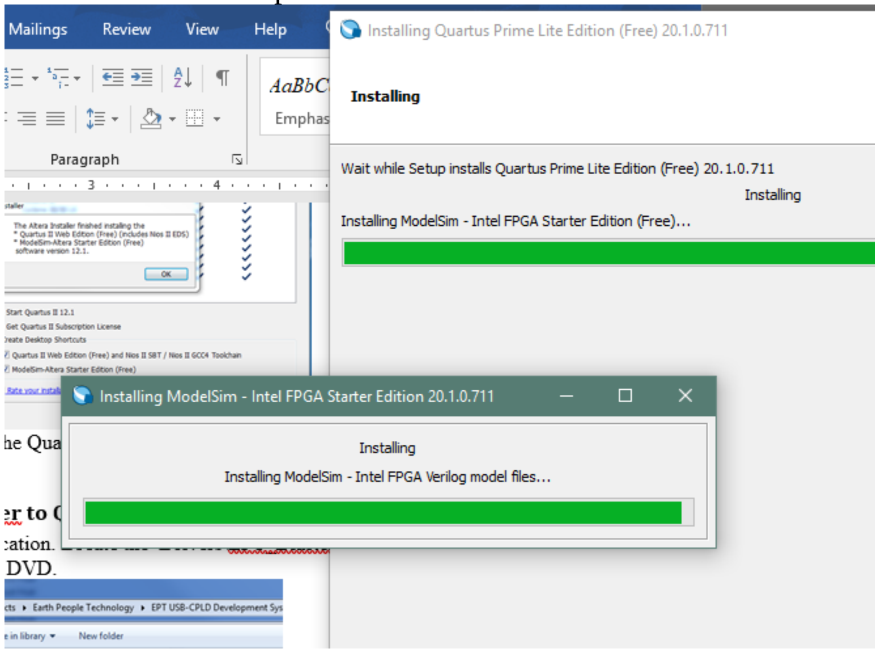
Installing

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

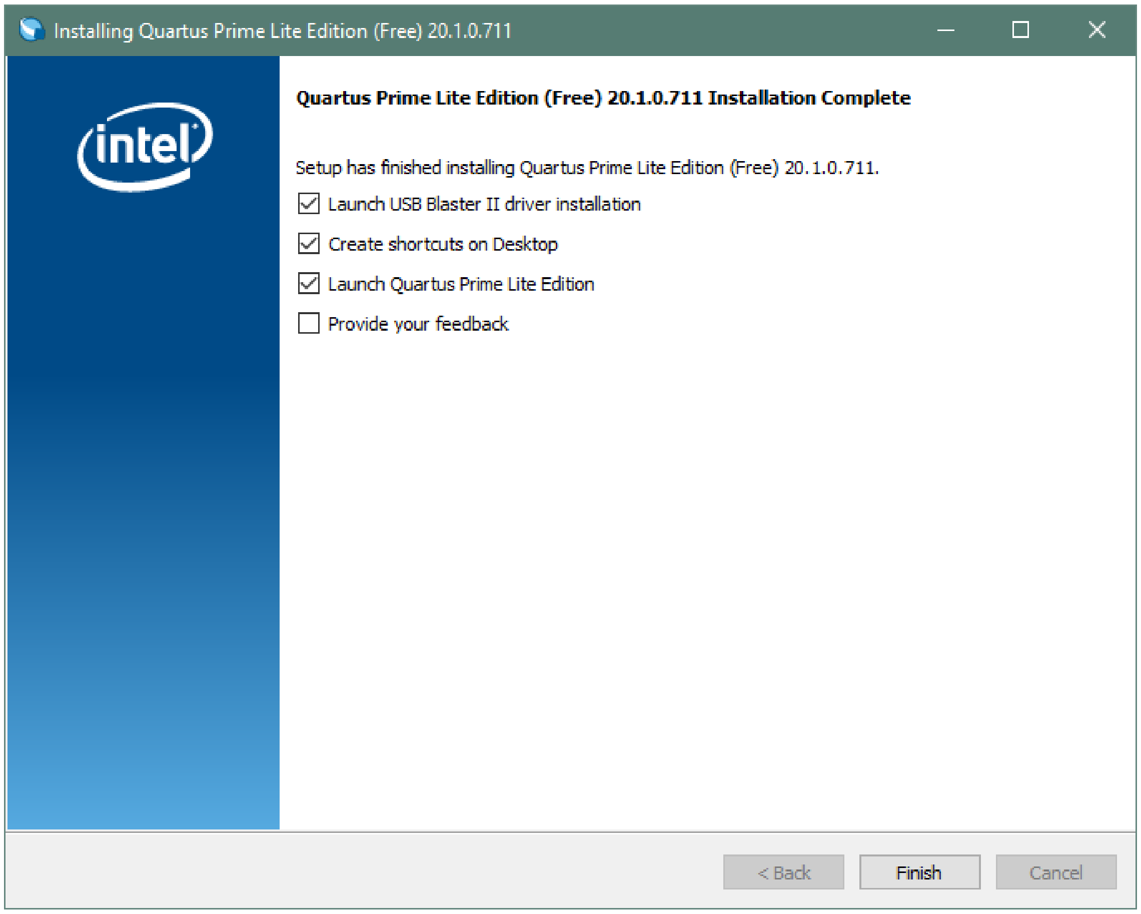
Installing

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...

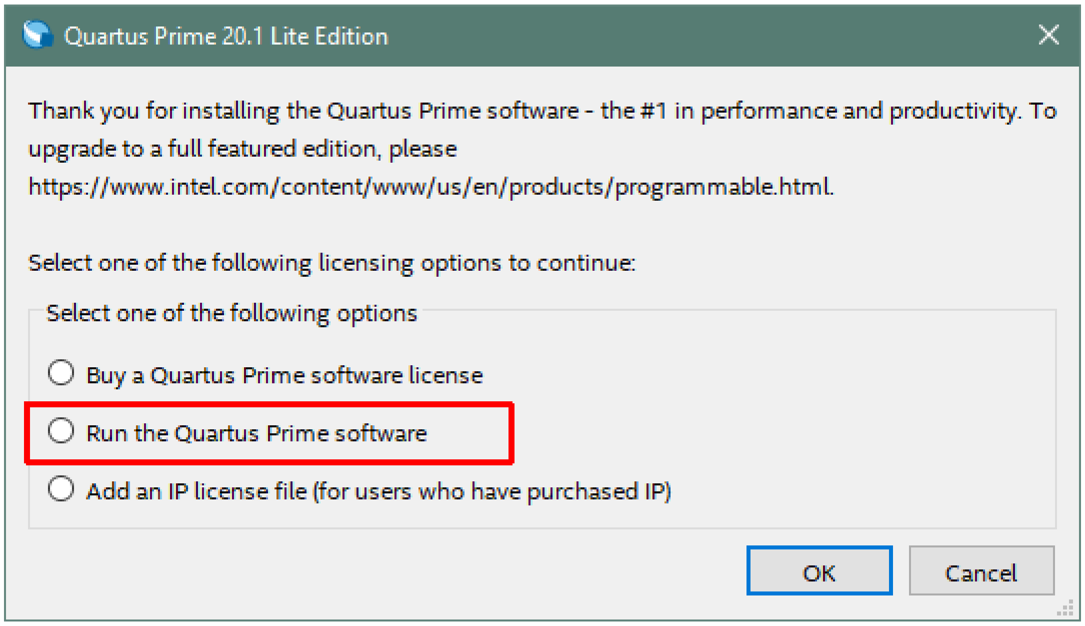

Installing Quartus Prime Lite Edition (Free) 20.1.0.711

Quartus Prime Lite Edition (Free) 20.1.0.711 Installation Complete

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

- [x] Launch USB Blaster II driver installation
- [x] Create shortcuts on Desktop
- [x] 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.

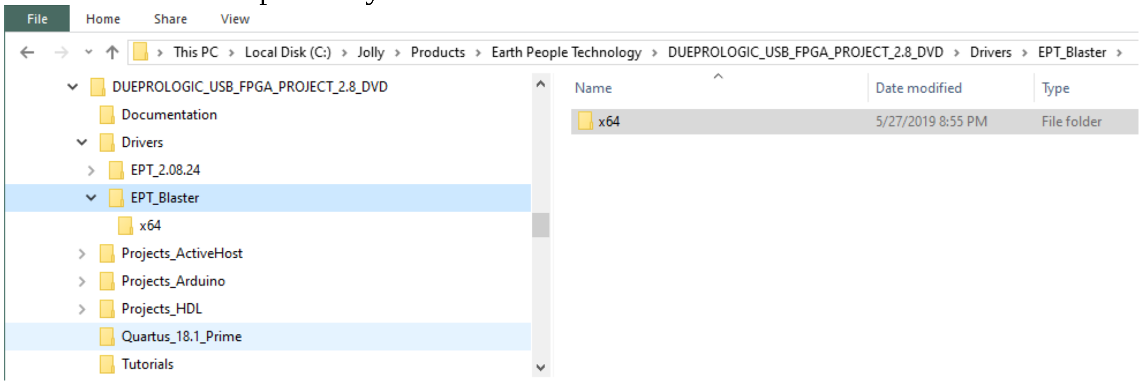
Select one of the following licensing options to continue:

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.

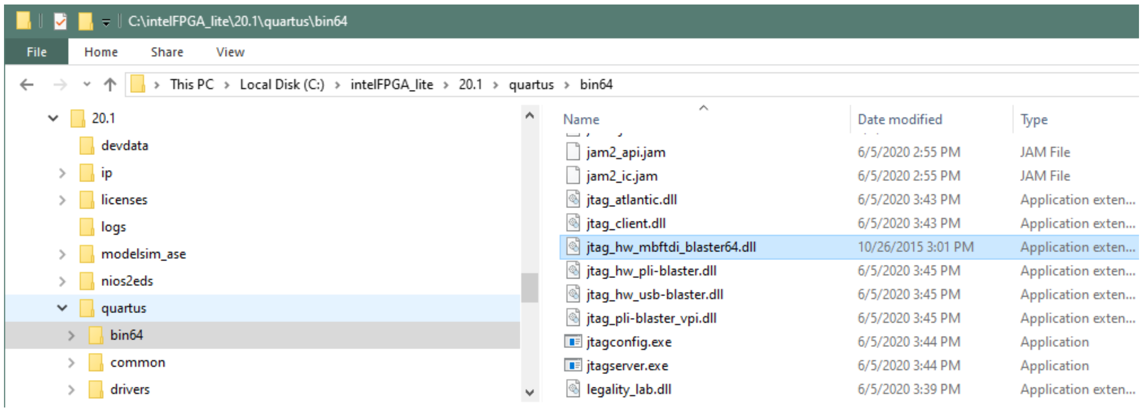


File
Home
Share
View
This PC › Local Disk (C:) › Jolly › Products › Earth People Technology › DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD › Drivers › EPT_Blaster ›
DUEPROLOGIC_USB_FPGA_PROJECT_2.8_DVD
Documentation
Drivers
EPT_2.08.24
EPT_Blaster
x64
Projects_ActiveHost
Projects_Arduino
Projects_HDL
Quartus_18.1_Prime
Tutorials

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

Follow these directions:

1. Open the C:\..\MEGAPROLOGIC_USB_CPLD_PROJECT_x.x_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

20.1
devdata
ip
licenses
logs
modelsim_ase
nios2eds
quartus
bin64
common
drivers

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...

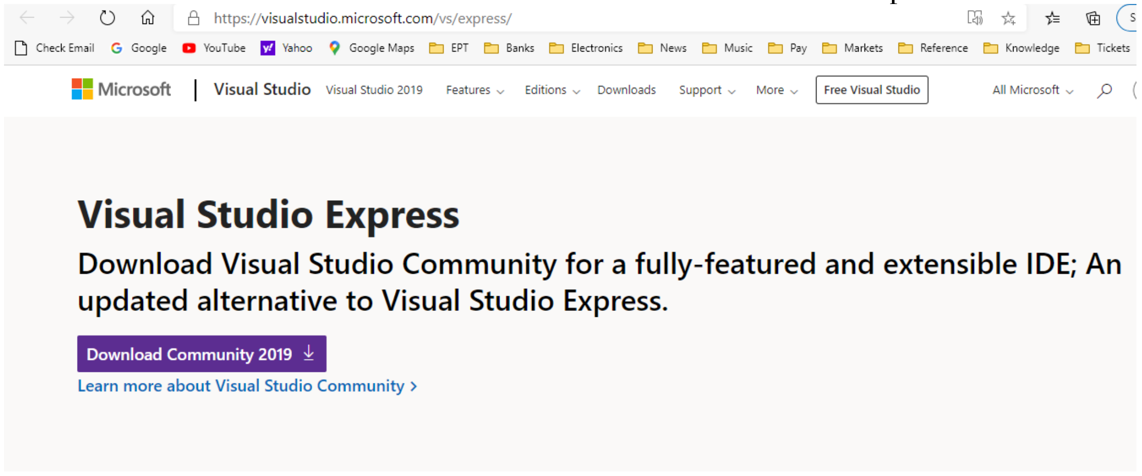
The DLL is installed and the JTAG server should recognize it. Go to the section "Programming the FPGA" of this manual for testing of the programming. If the driver is not found in the Programmer Tool->Hardware Setup box, see the JTAG DLL Insert to Quartus Prime Troubleshooting Guide.

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/

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

Microsoft | Visual Studio Visual Studio 2019 Features Editions Downloads Support More Free Visual Studio All Microsoft

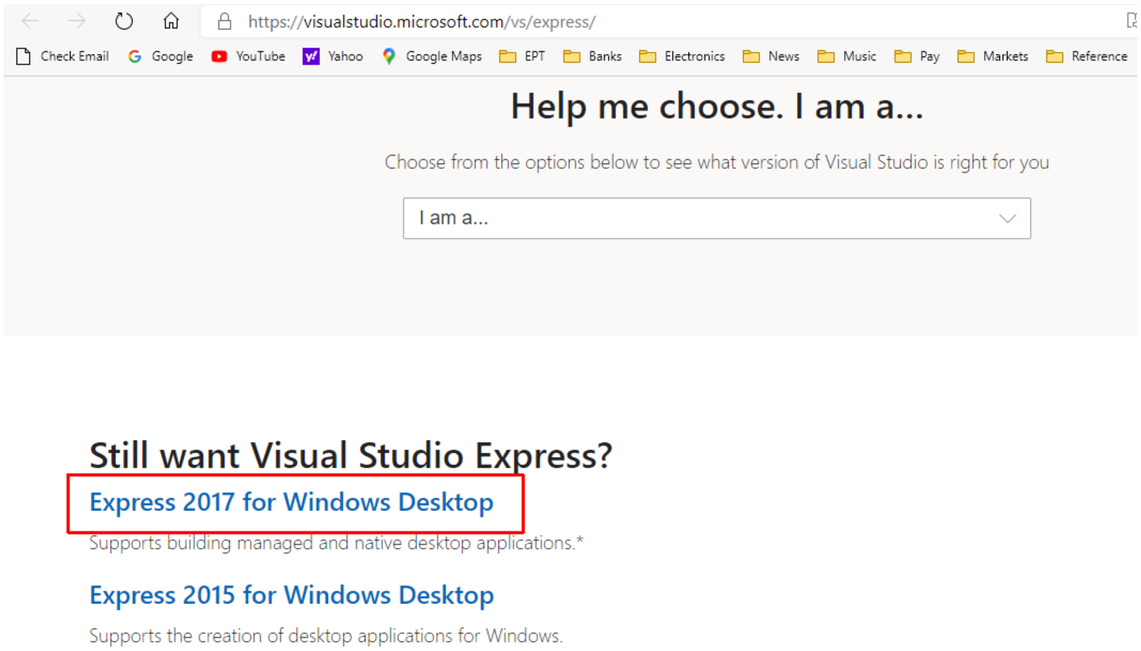
Visual Studio Express

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

Download Community 2019

Learn more about Visual Studio Community ›

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



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

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?

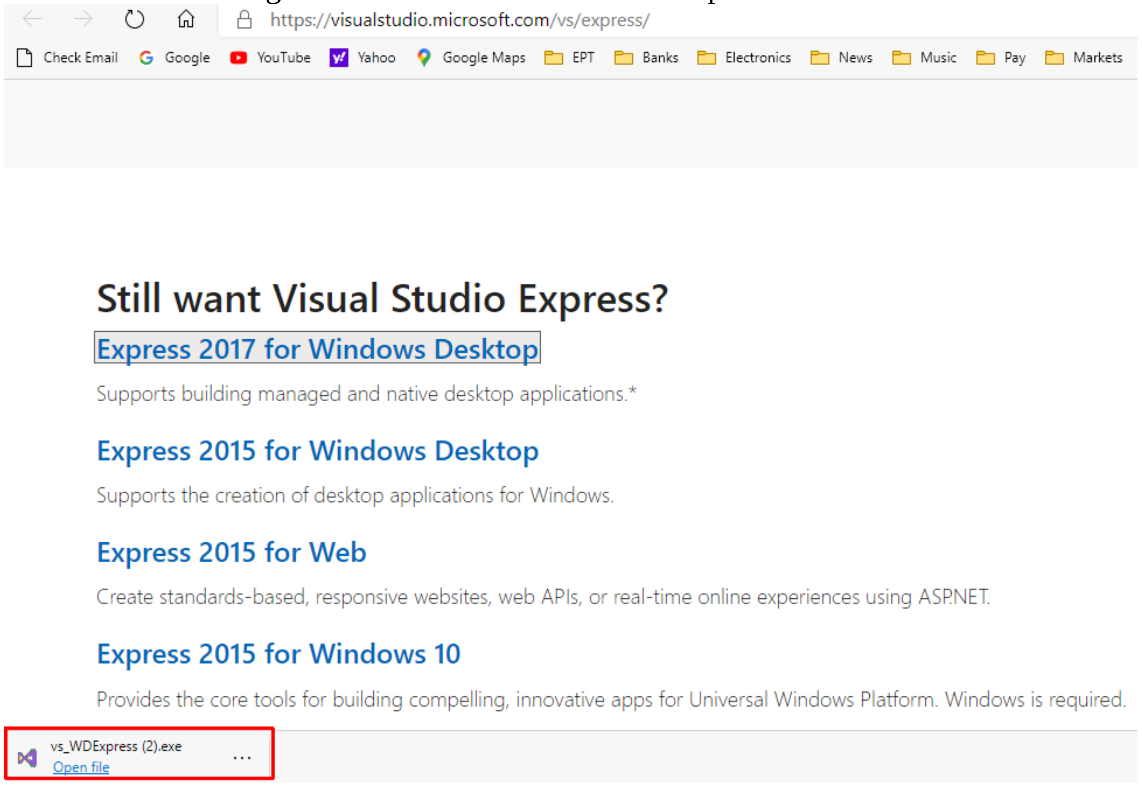
Express 2017 for Windows Desktop

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/

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

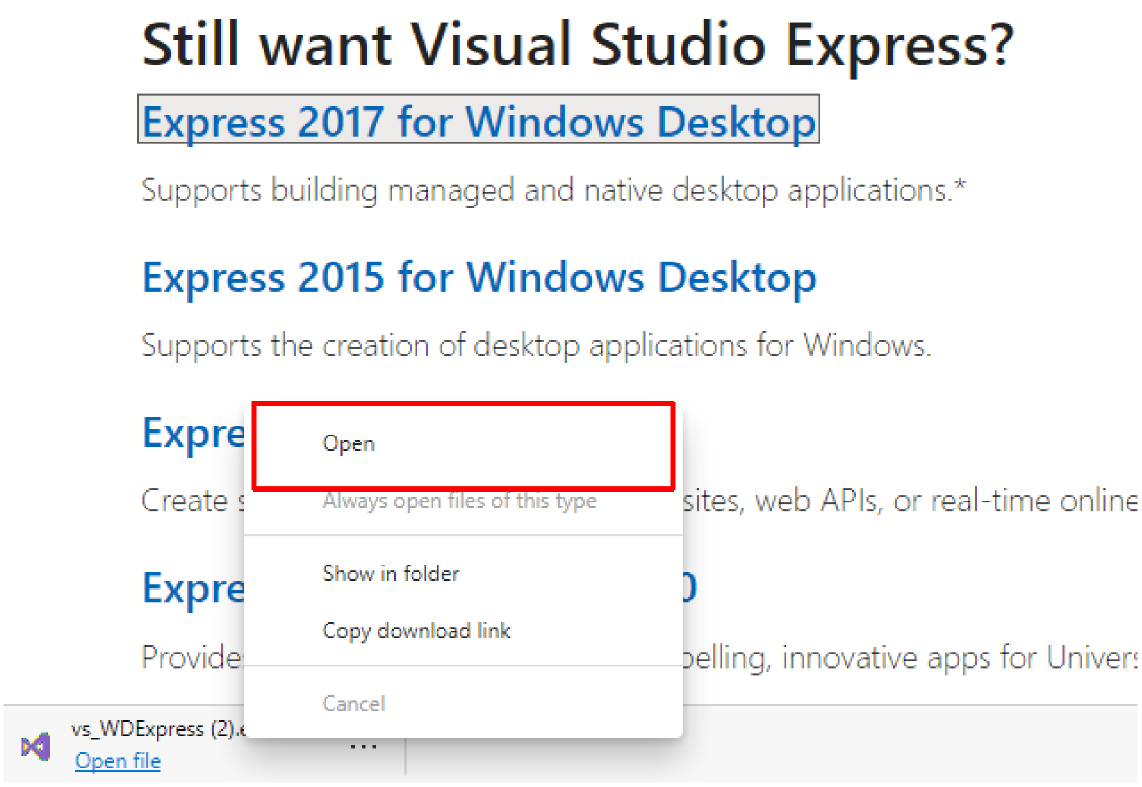
Create standards-based, responsive websites, web APIs, or real-time online experiences using ASP.NET.

Express 2015 for Windows 10

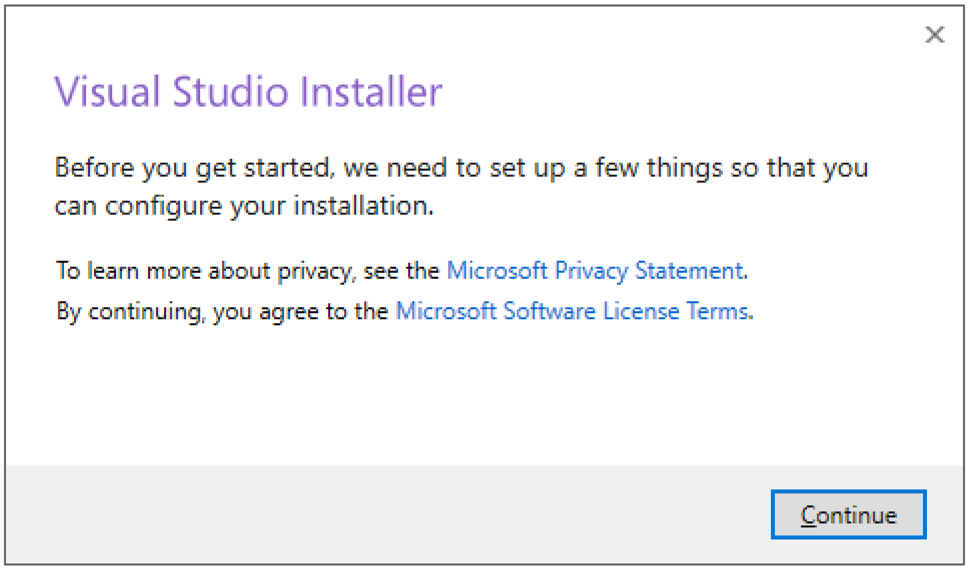
Provides the core tools for building compelling, innovative apps for Universal Windows Platform. Windows is required.

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.
Expre
Create s
sites, web APIs, or real-time online
Expre
Provide
pelling, innovative apps for Univer
Open
Always open files of this type
Show in folder
Copy download link
Cancel
vs_WDExpress (2).e
Open file

Click the “Continue” button.



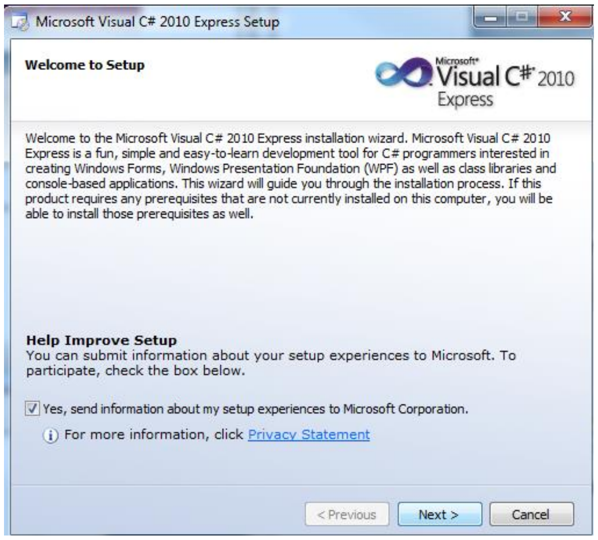
Visual Studio Installer

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.

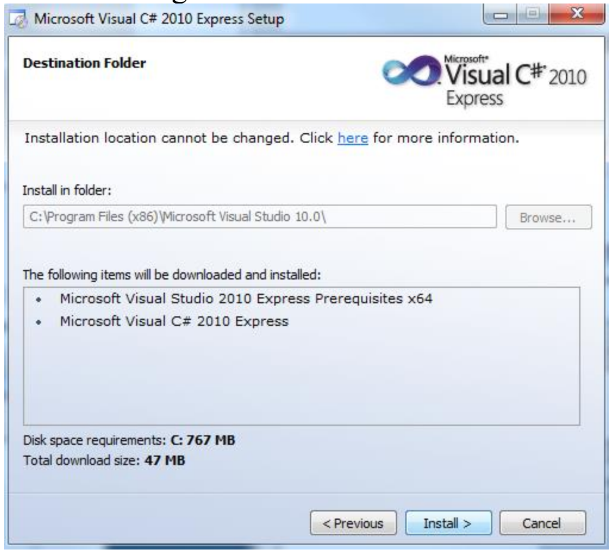
Help Improve Setup
You can submit information about your setup experiences to Microsoft. To participate, check the box below.

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

For more information, click Privacy Statement

< Previous | Next > | Cancel

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\ | Browse...

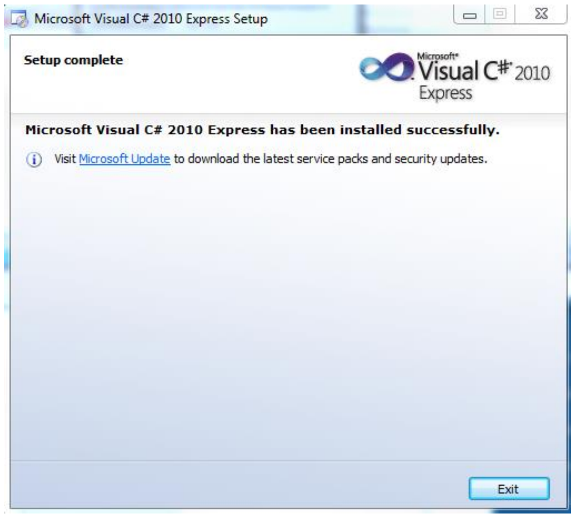
The following items will be downloaded and installed:

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

Disk space requirements: **C: 767 MB**
Total download size: **47 MB**

< Previous | Install > | Cancel

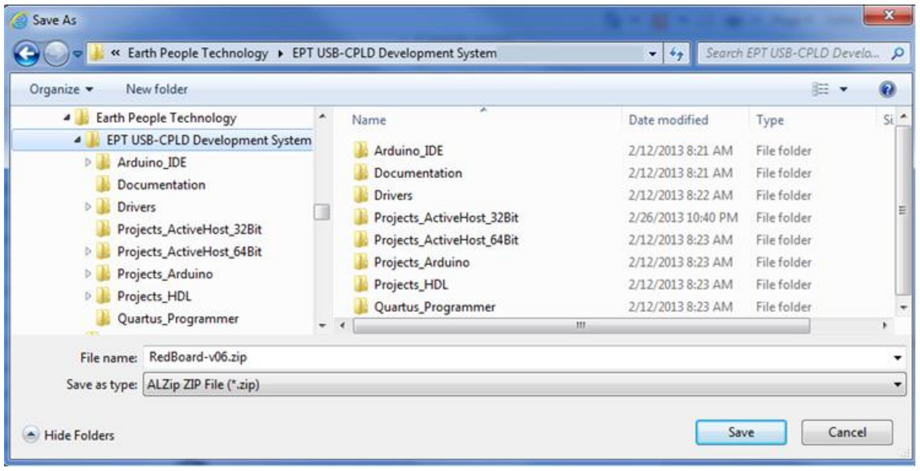
Visual C# 2010 Express will install. This may take up to twenty minutes depending on your internet connection.


Microsoft Visual C# 2010 Express Setup
Setup complete
Microsoft Visual C# 2010 Express
Exit
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 MegaProLogic. 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 MegaProLogic Development System CD.

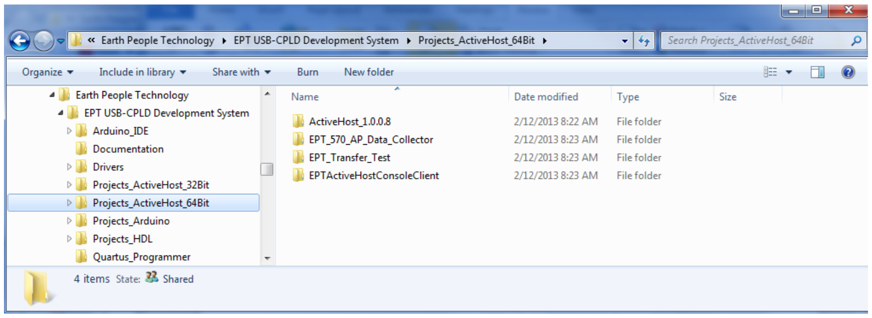

Save As
« Earth People Technology ▸ EPT USB-CPLD Development System
Search EPT USB-CPLD Develo...
Organize ▾
New folder
Earth People Technology
EPT USB-CPLD Development System
Arduino_IDE
Documentation
Drivers
Projects_ActiveHost_32Bit
Projects_ActiveHost_64Bit
Projects_Arduino
Projects_HDL
Quartus_Programmer
Hide Folders
Save
Cancel

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 MegaProLogic Development System using Windows Explorer.



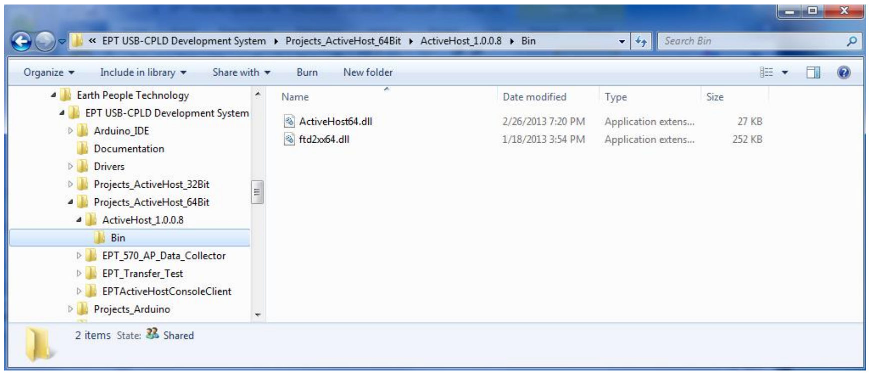
« Earth People Technology ▸ EPT USB-CPLD Development System ▸ Projects_ActiveHost_64Bit ▸

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.



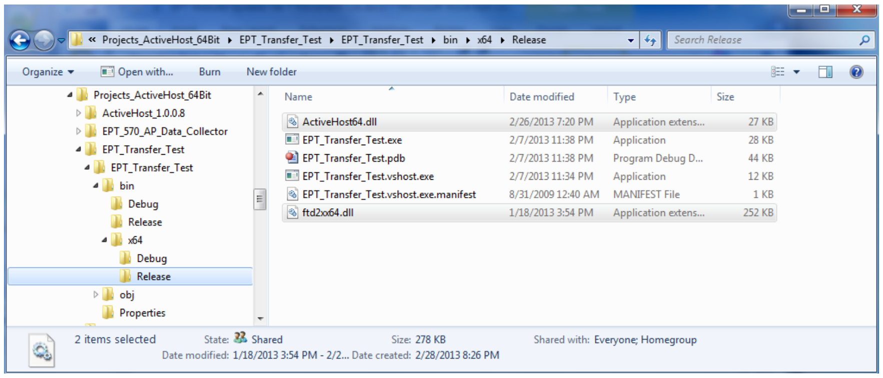
« EPT USB-CPLD Development System ▸ Projects_ActiveHost_64Bit ▸ ActiveHost_1.0.0.8 ▸ Bin

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 MegaProLogic 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

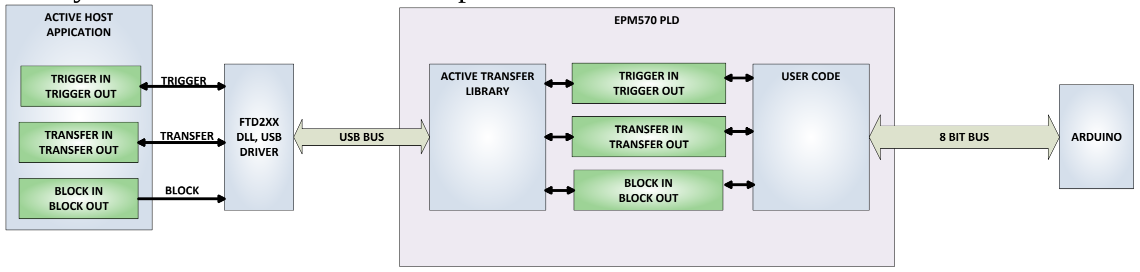
Projects_ActiveHost_64Bit
ActiveHost_1.0.0.8
EPT_570_AP_Data_Collector
EPT_Transfer_Test
EPT_Transfer_Test
bin
Debug
Release
x64
Debug
Release
obj
Properties

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



3.1 EPT Active Transfer System Overview

The Active Transfer System components consist of the following:

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

The Active_Serial_Library provides the communication to the USB hardware. While separate Input and Output buses provide bi-directional communications with the plug in modules. See Figure 6 for an overview of the EPT Active_Transfer system.

Figure 6 EPT Active Transfer Library Overview

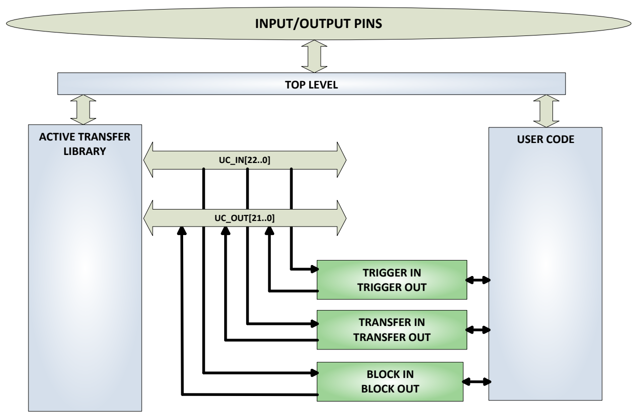
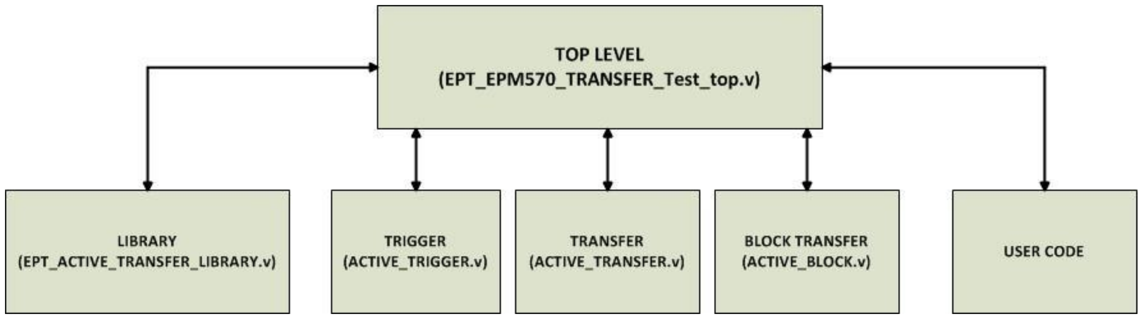
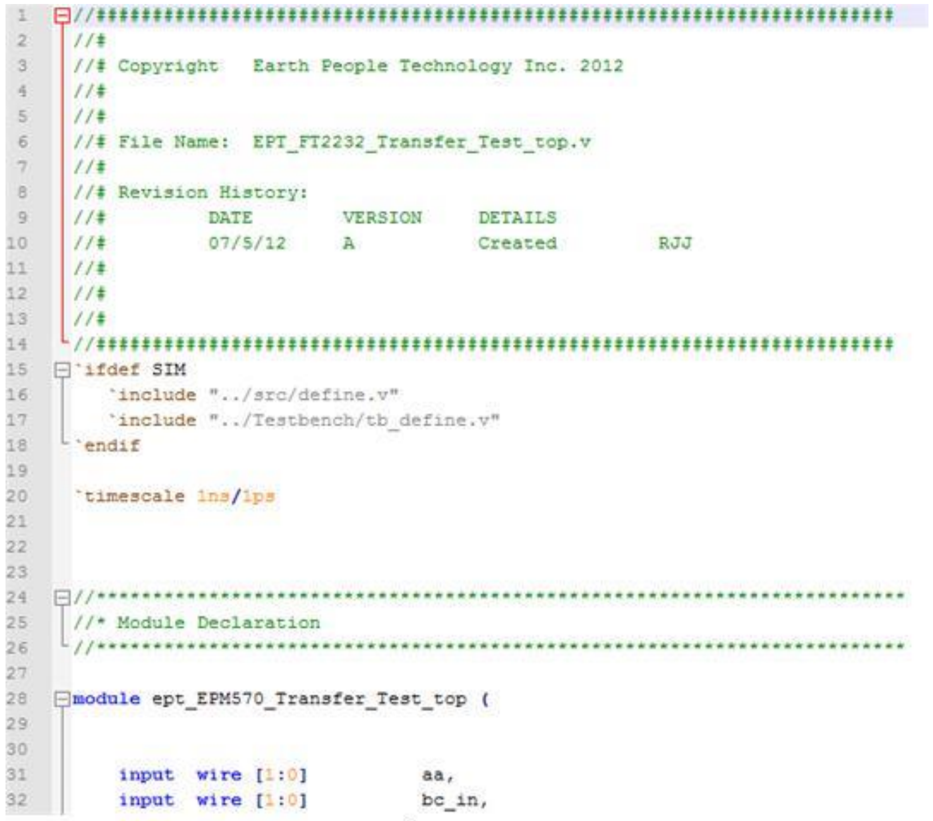


Figure 6 shows how the modules of the EPT Active Transfer Library attach to the overall user project. The EPT Active_Transfer_Library.vqm, Active_Trigger.v, Active_Transfer.v and Active_Block.v modules are instantiated in the top level of the user project. The User_Code.v module is also instantiated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



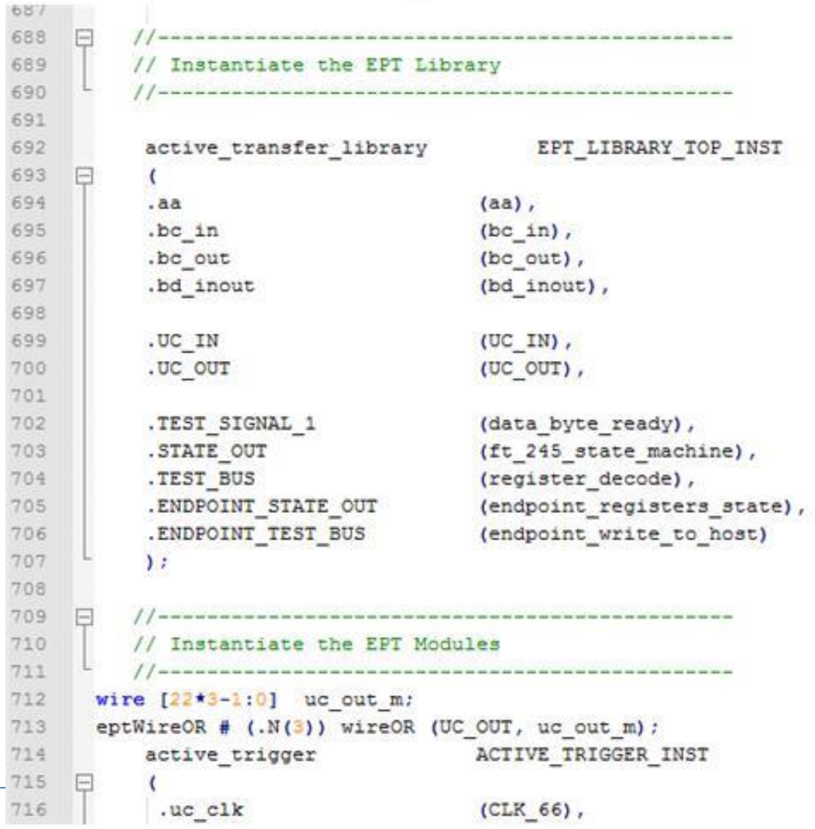


```
//#######################################################################
//#
//# Copyright   Earth People Technology Inc. 2012
//#
//#
//# File Name:  EPT_FT2232_Transfer_Test_top.v
//#
//# Revision History:
//#          DATE        VERSION     DETAILS
//#          07/5/12     A           Created         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,
```

●

●

●



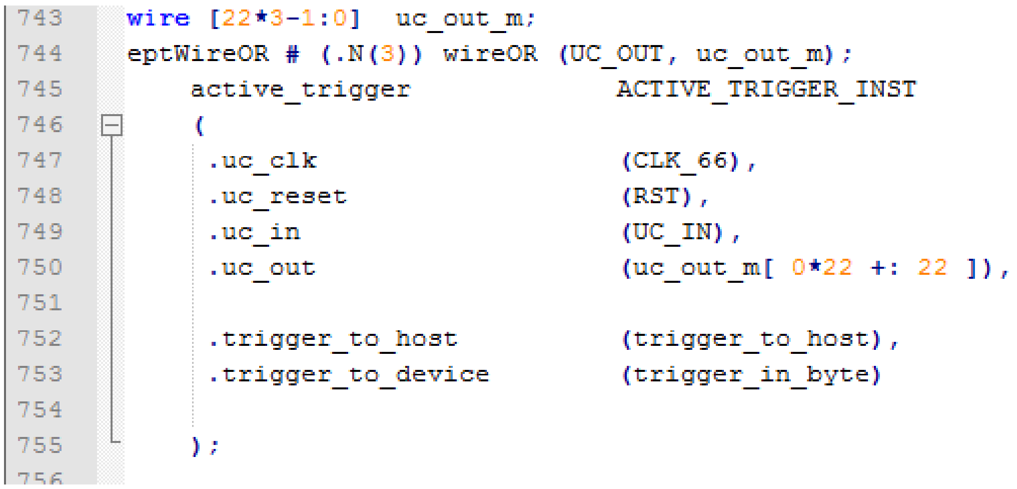
```
    //-----------------------------------------------
    // Instantiate the EPT Library
    //-----------------------------------------------

     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.

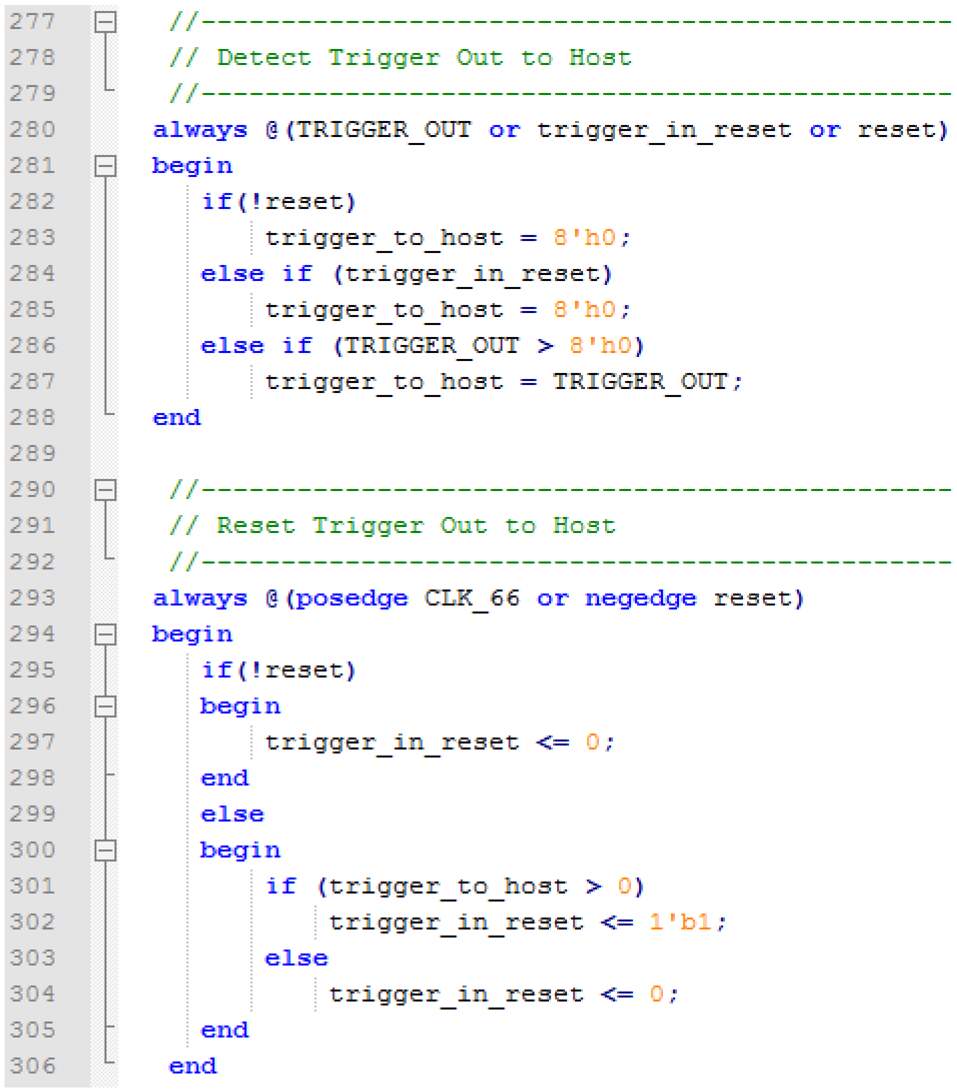
The control bus UART_IN and UART_OUT are used to channel data, and control signals to the USB interface chip. These signals are connected directly to input and output pins of the CPLD.

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)

    );
```

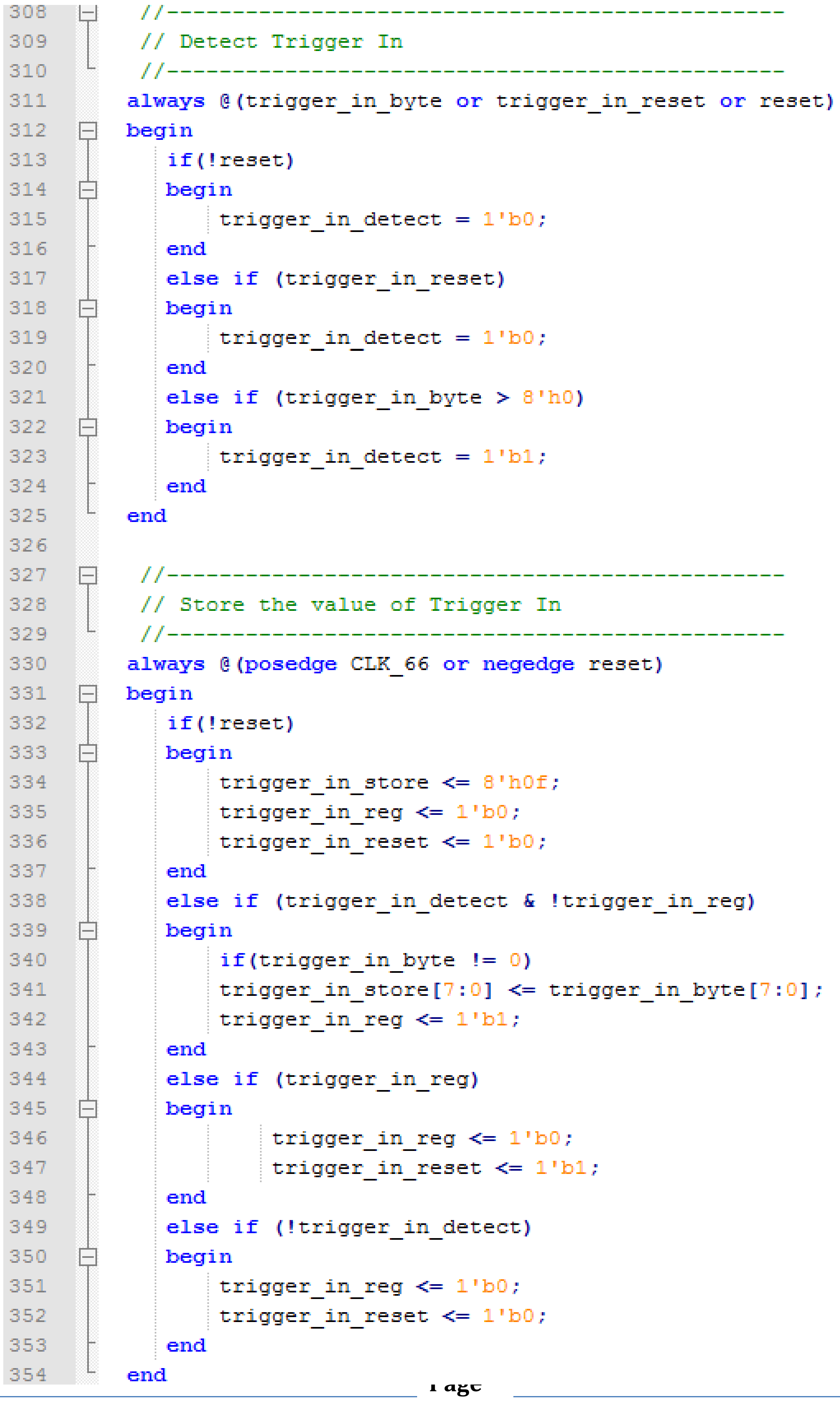
To send a trigger, decide which bit (or multiple bits) of the eight bits you want to send the trigger on. Then, set that bit (or bits) high. The Active Transfer Library will send a high on that trigger bit for one clock cycle (66 MHz), then reset itself to zero. The bit can stay high on the user code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

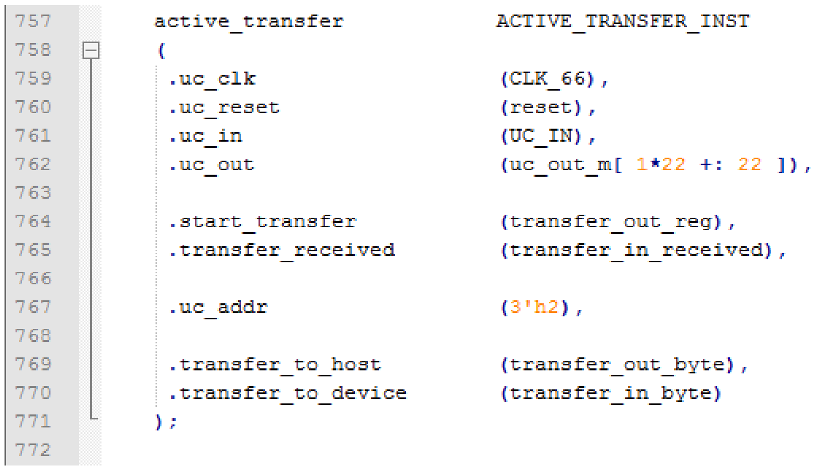
The user code must be setup to receive triggers from the Host. This can be done by using an asynchronous always block. Whenever a change occurs on a particular trigger bit (or bits), a conditional branch can detect if the trigger bit is for that block of code. Then, execute some code based on that trigger.



```
 //-----------------------------------------------
 // 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
```

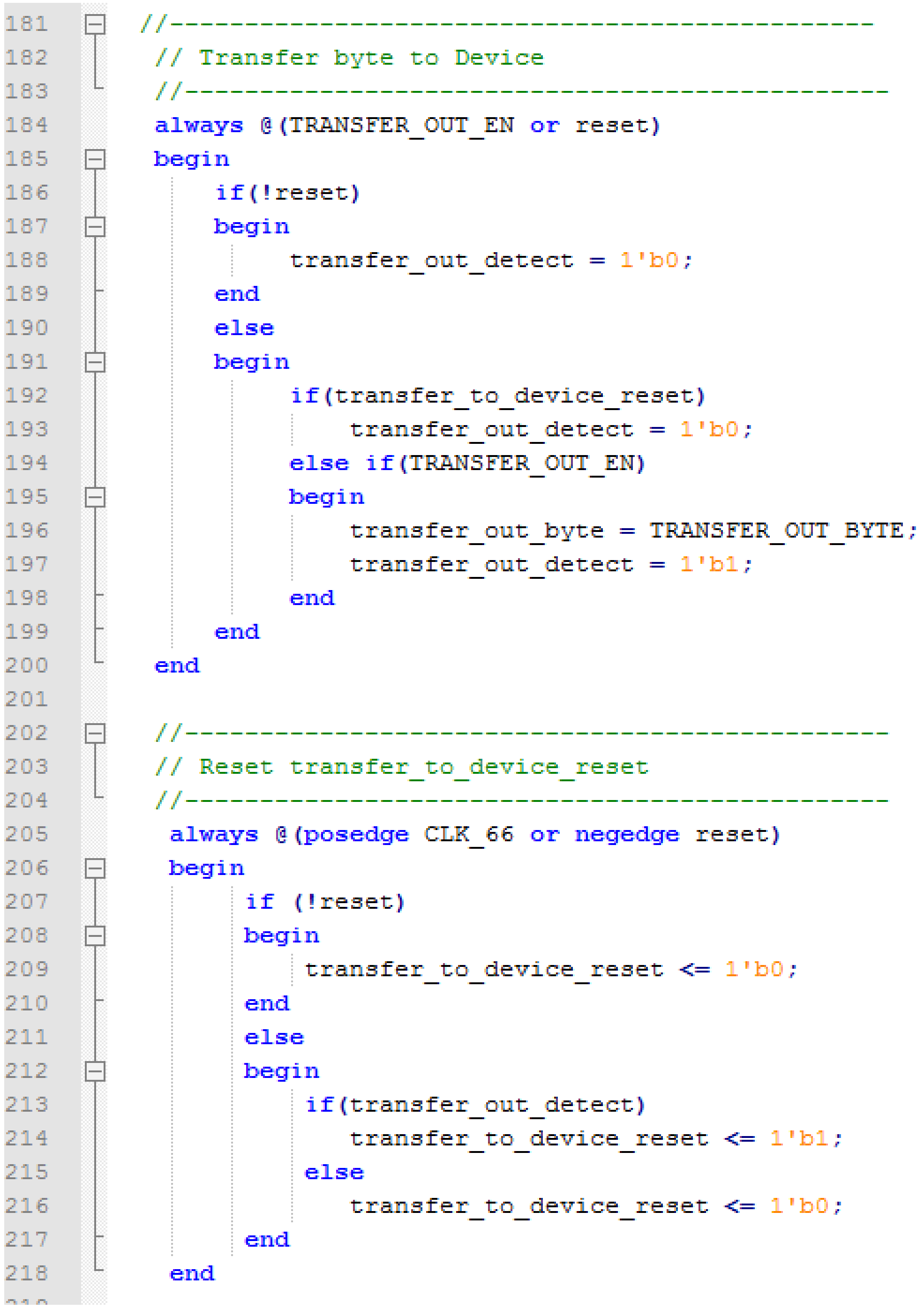
3.2.2 Active Transfer EndTerm

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.



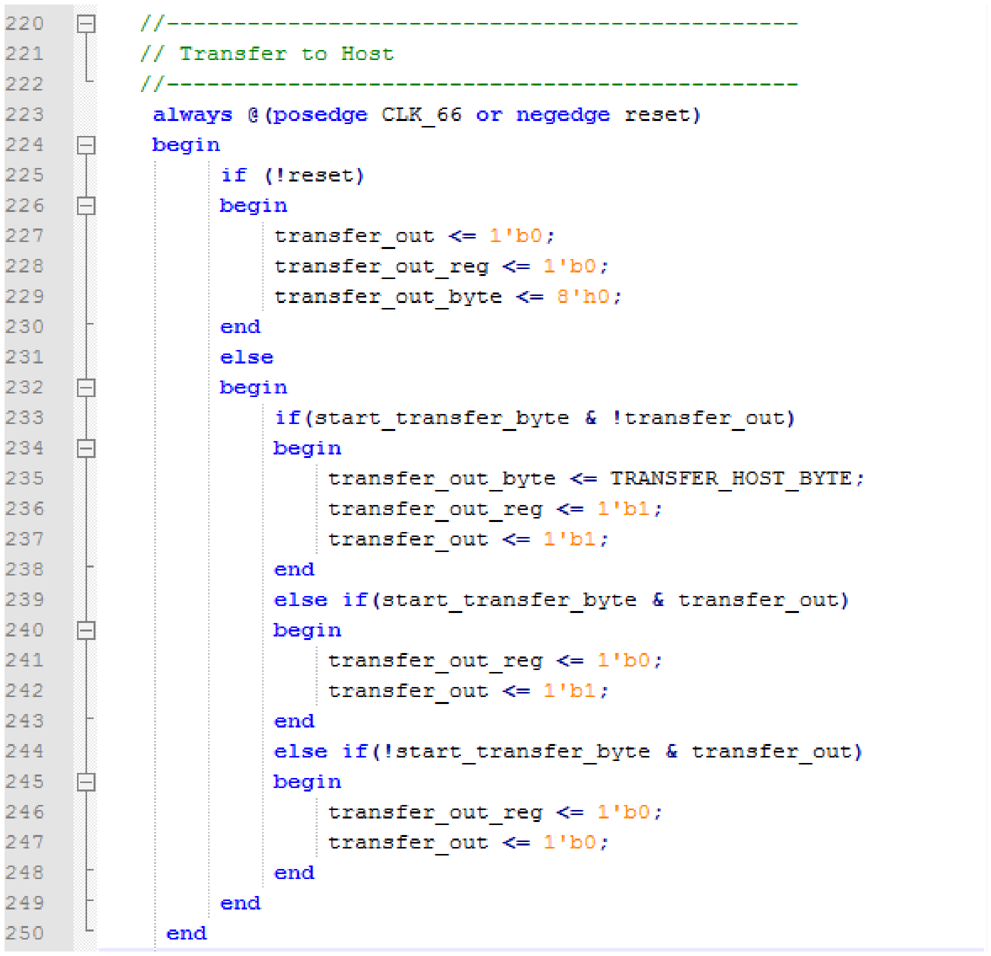
```
active_transfer              ACTIVE_TRANSFER_INST
(
 .uc_clk                     (CLK_66),
 .uc_reset                   (reset),
 .uc_in                      (UC_IN),
 .uc_out                     (uc_out_m[ 1*22 +: 22 ]),

 .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
```

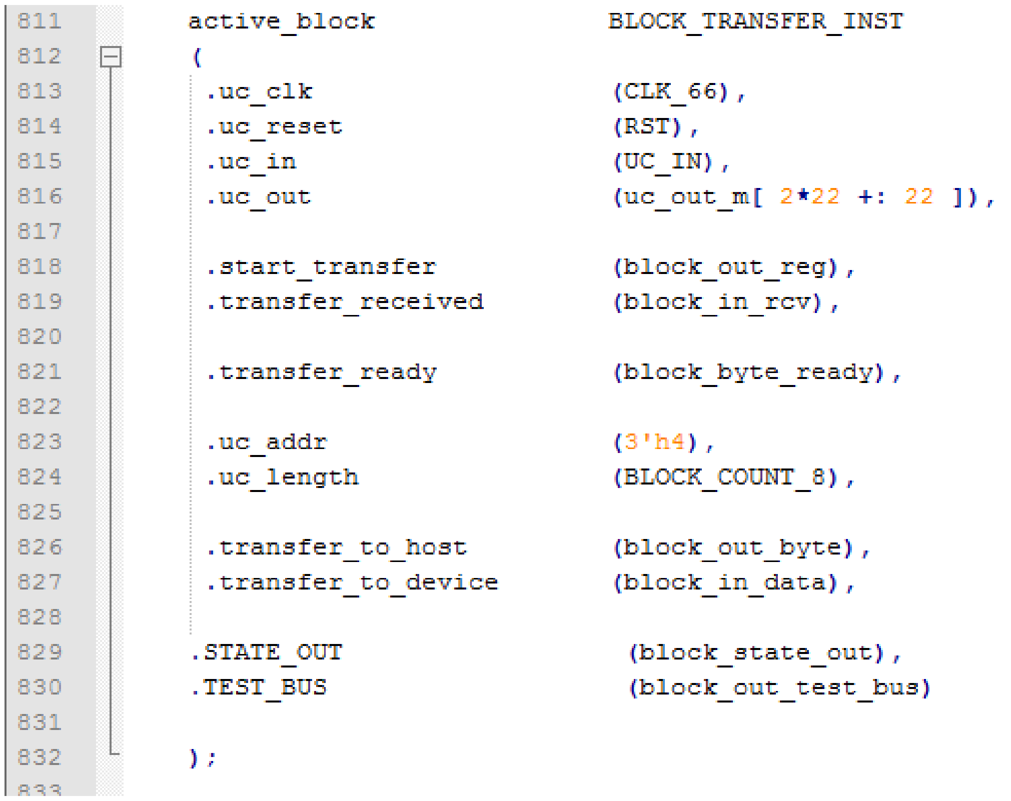
To receive a byte, the Active Host will send a byte using it's dll. The user code must monitor the transfer_received port. The transfer_received port will assert high for one clock cycle (66 MHz) when a byte is ready for reading on the transfer_to_device port. User code should use an asynchronous always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

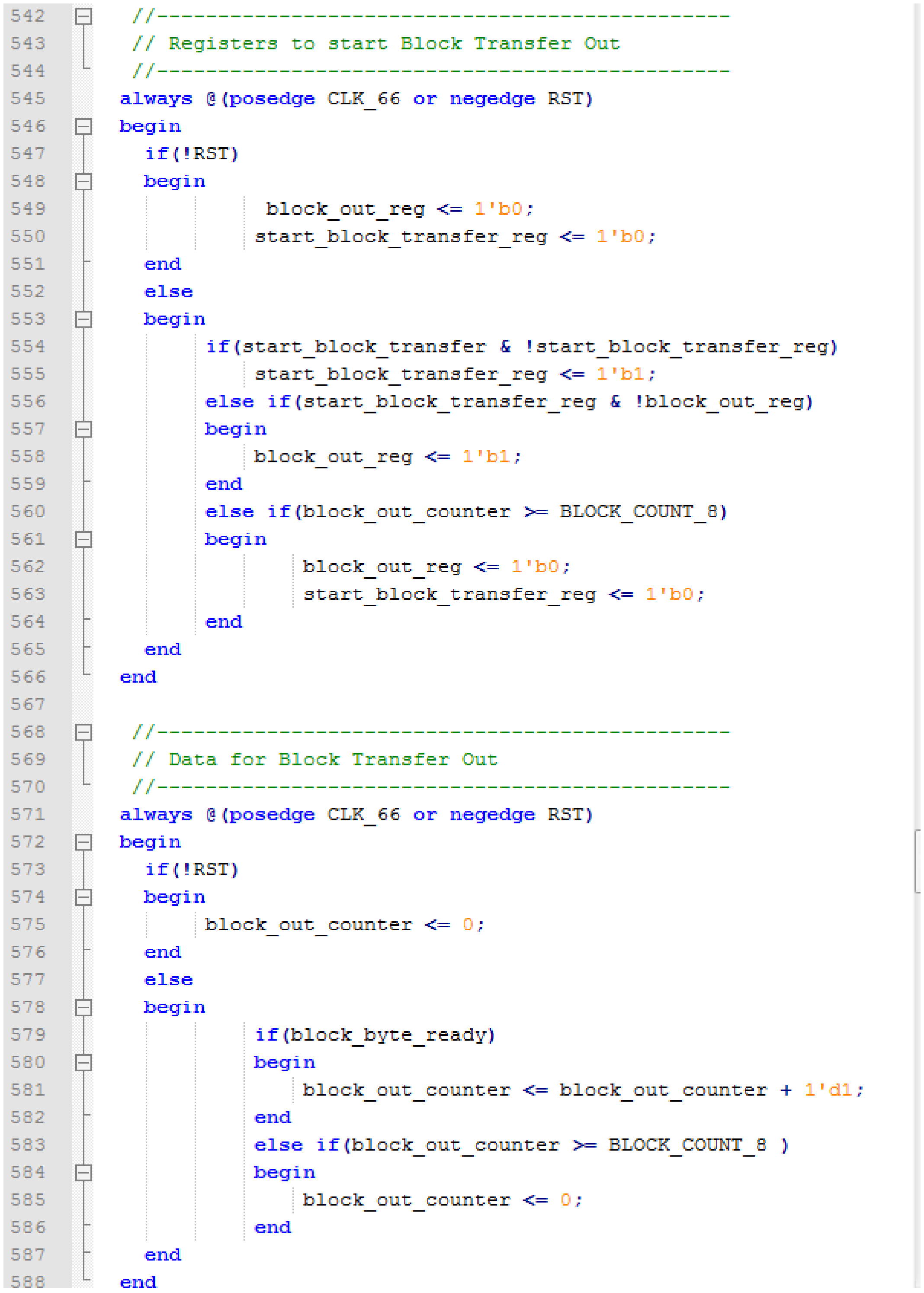
The Active Block module is designed to transfer blocks of data between Host and User Code and vice versa. This allows buffers of data to be transferred with a minimal amount of code. The Active Block module is addressable, so up to eight individual modules can be instantiated and separately addressed.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

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

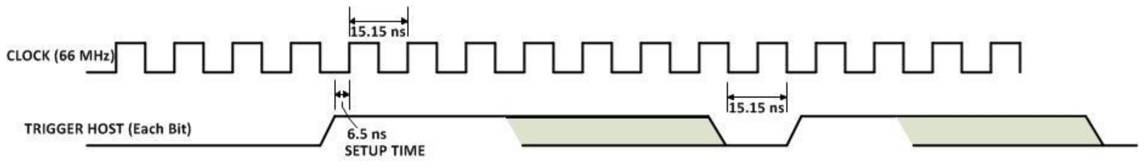


Figure xx Active Trigger to Host Timing

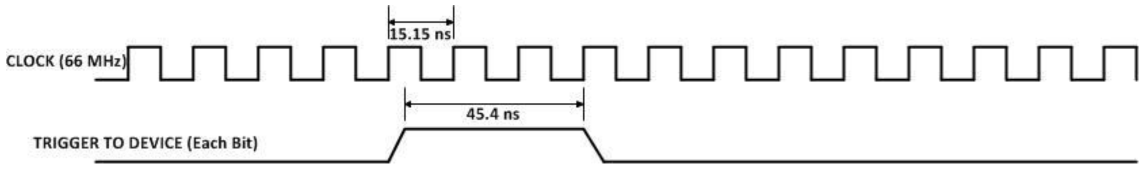


Figure xx Active Trigger to Device Timing

3.3.2 Active Transfer EndTerm Timing

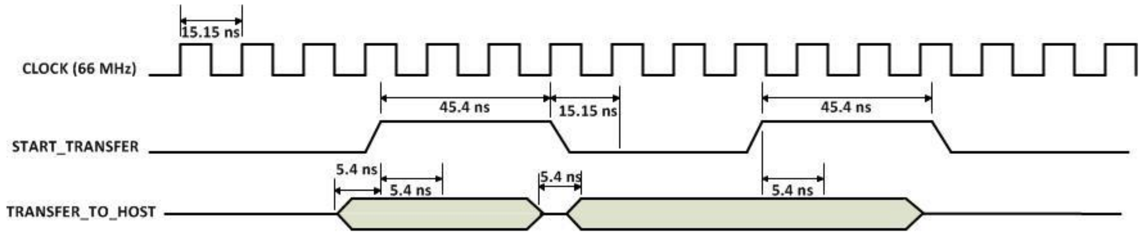


Figure xx Active Transfer To Host Timing

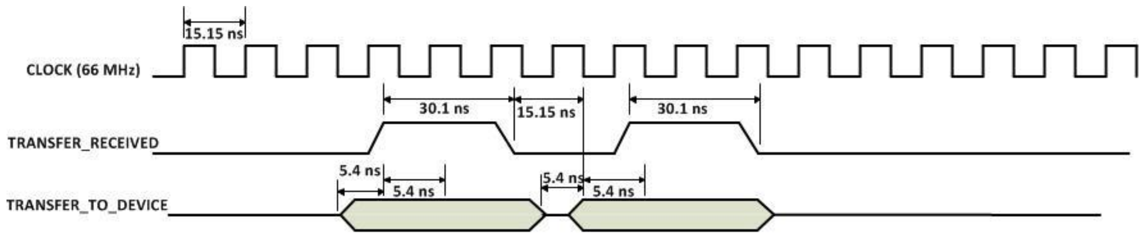


Figure xx Active Transfer To Device Timing

3.3.3 Active Block EndTerm Timing

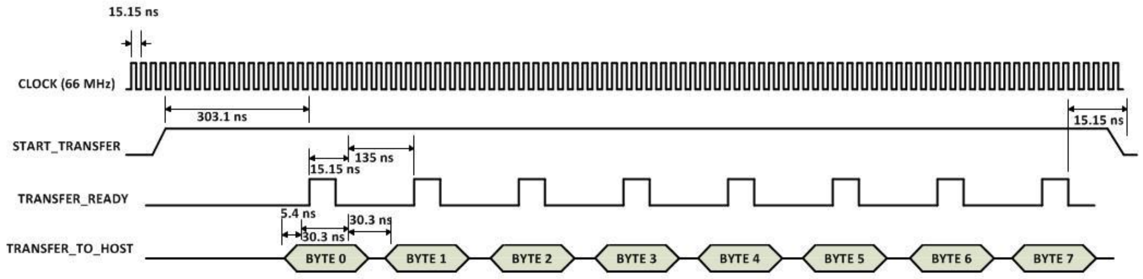


Figure xx Active Block To Host Timing

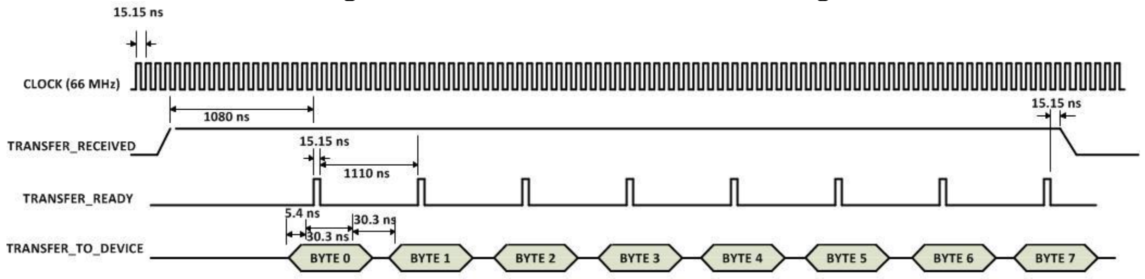
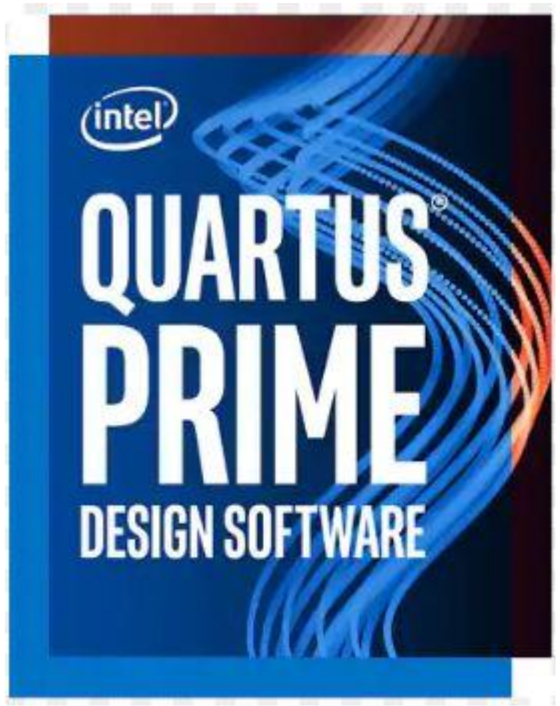


Figure xx Active Block To Device Timing

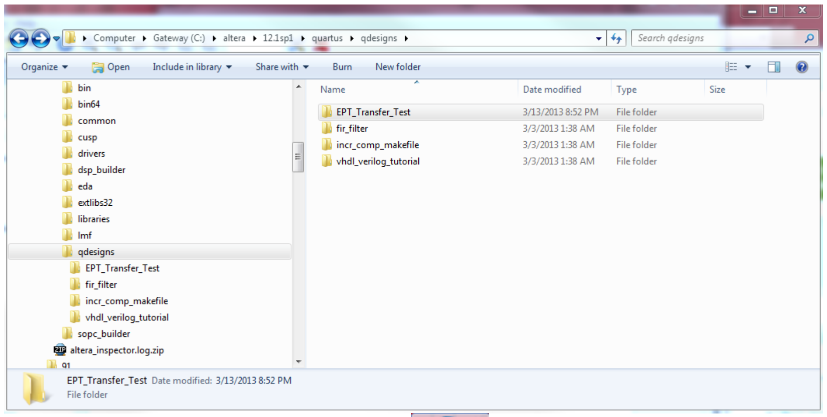
4 Compiling, Synthesizing, and Programming CPLD



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

4.1 Setting up the Project and Compiling

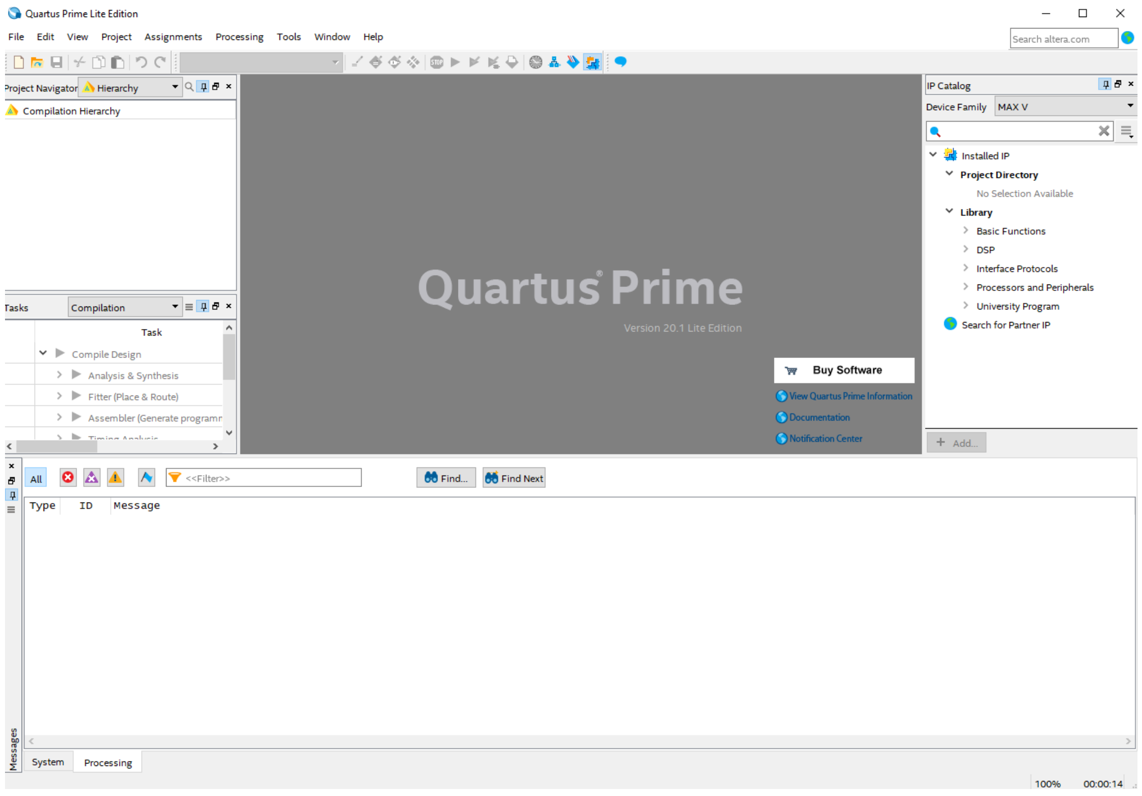
Once the HDL code (Verilog or VHDL) is written and verified using a simulator, a project can be created using Quartus Prime. Writing the HDL code and simulating it will be covered in later sections. Bring up Quartus Prime, then use Windows Explorer to browse to C:\intelFPGA_lite\xxx.x\quartus\qdesignscreate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



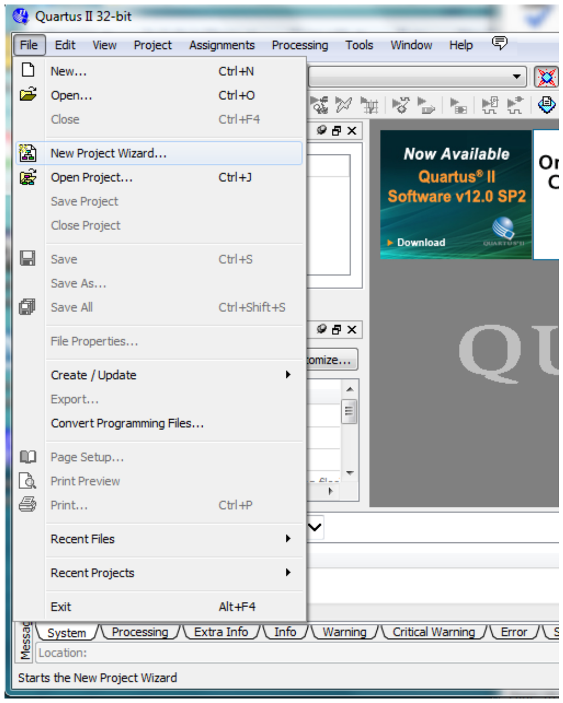
Name	Date modified	Type	Size
EPT_Transfer_Test	3/13/2013 8:52 PM	File folder	
fir_filter	3/3/2013 1:38 AM	File folder	
incr_comp_makefile	3/3/2013 1:38 AM	File folder	
vhdl_verilog_tutorial	3/3/2013 1:38 AM	File folder	

EPT_Transfer_Test Date modified: 3/13/2013 8:52 PM
File folder

Open Quartus Prime by clicking on the icon .

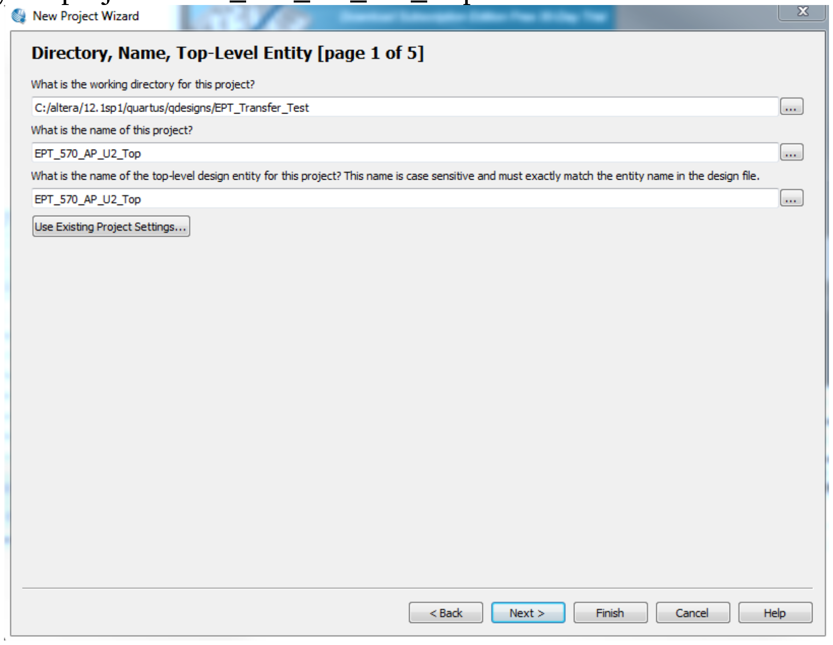


Under Quartus, Select File->New Project Wizard. The Wizard will walk you through setting up files and directories for your project.



Starts the New Project Wizard

At the Top-Level Entity page, browse to the C:\intelFPGA_lite\xxx.x\quartus\qdesignscreate directory to store your project. Type in a name for your project "EPT_570_AP_M4_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?

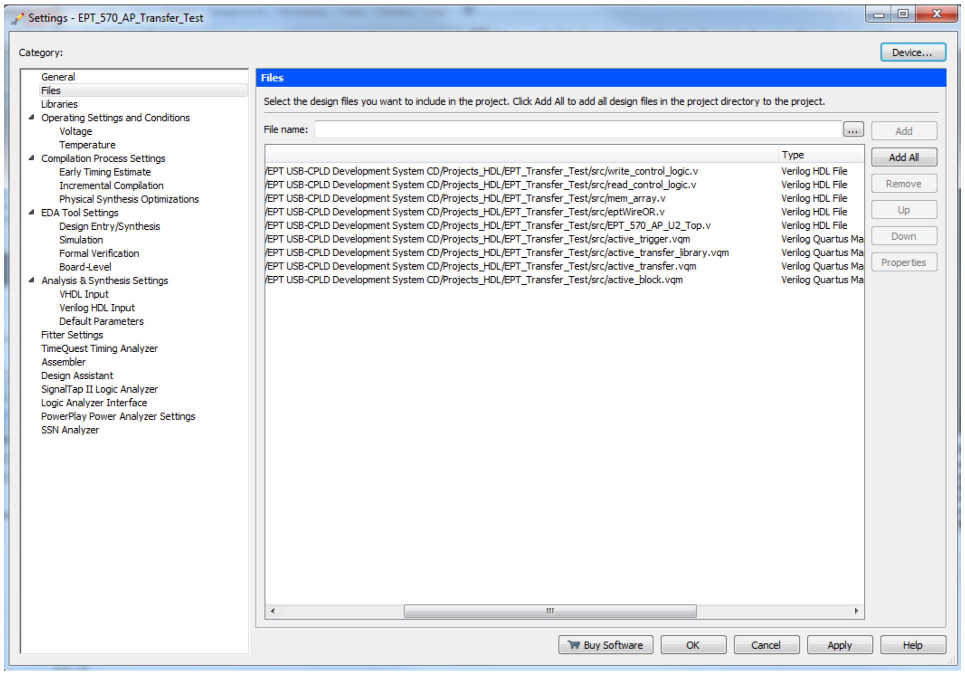
EPT_570_AP_U2_Top

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 MegaProLogic Development System DVD. Copy the files from the \src directory.

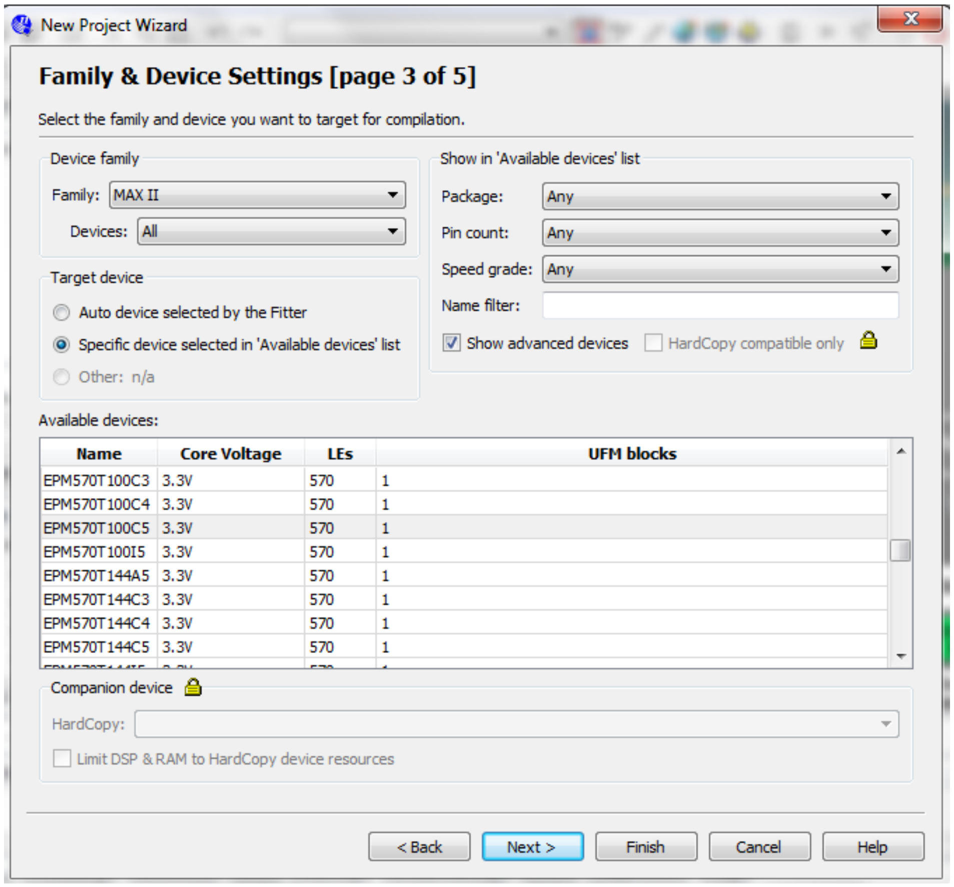
- Active_block.v
- Active_transfer.v
- Active_trigger.v
- Active_Serial_library.v
- eptWireOr.v
- mem_array.v
- read_control_logic.v
- write_control_logic.v
- EPT_570_AP_M4_Top.v



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

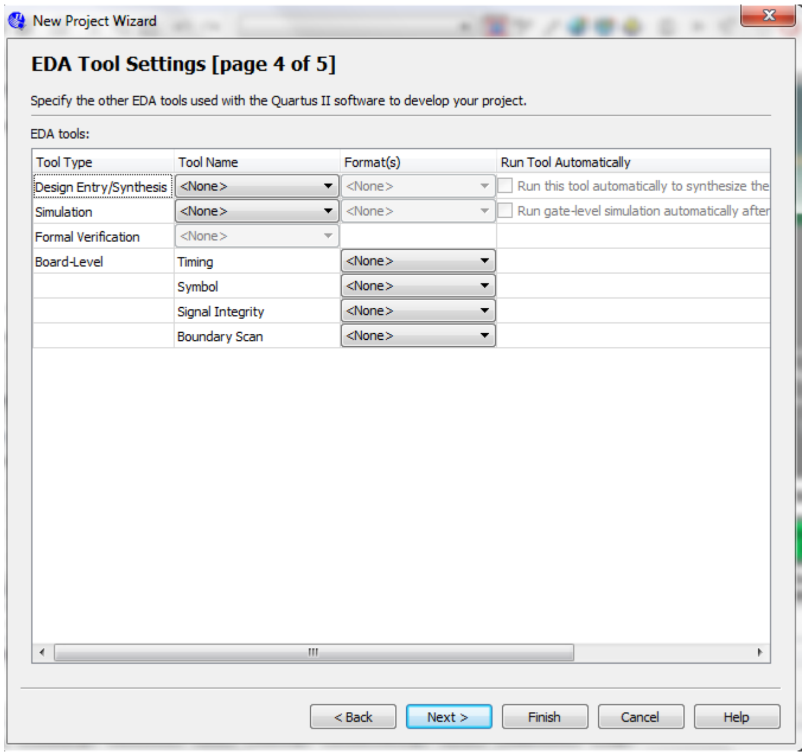
Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
Name filter:
[✓] Show advanced devices
[ ] HardCopy compatible only

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

Select Next, leave defaults for the EDA Tool Settings.


New Project Wizard

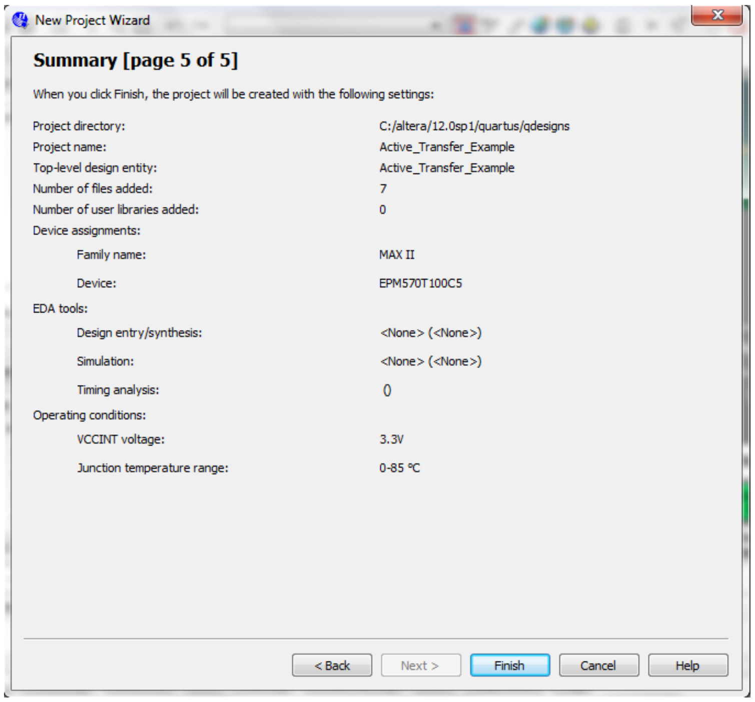
EDA Tool Settings [page 4 of 5]

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>	

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


New Project Wizard

Summary [page 5 of 5]

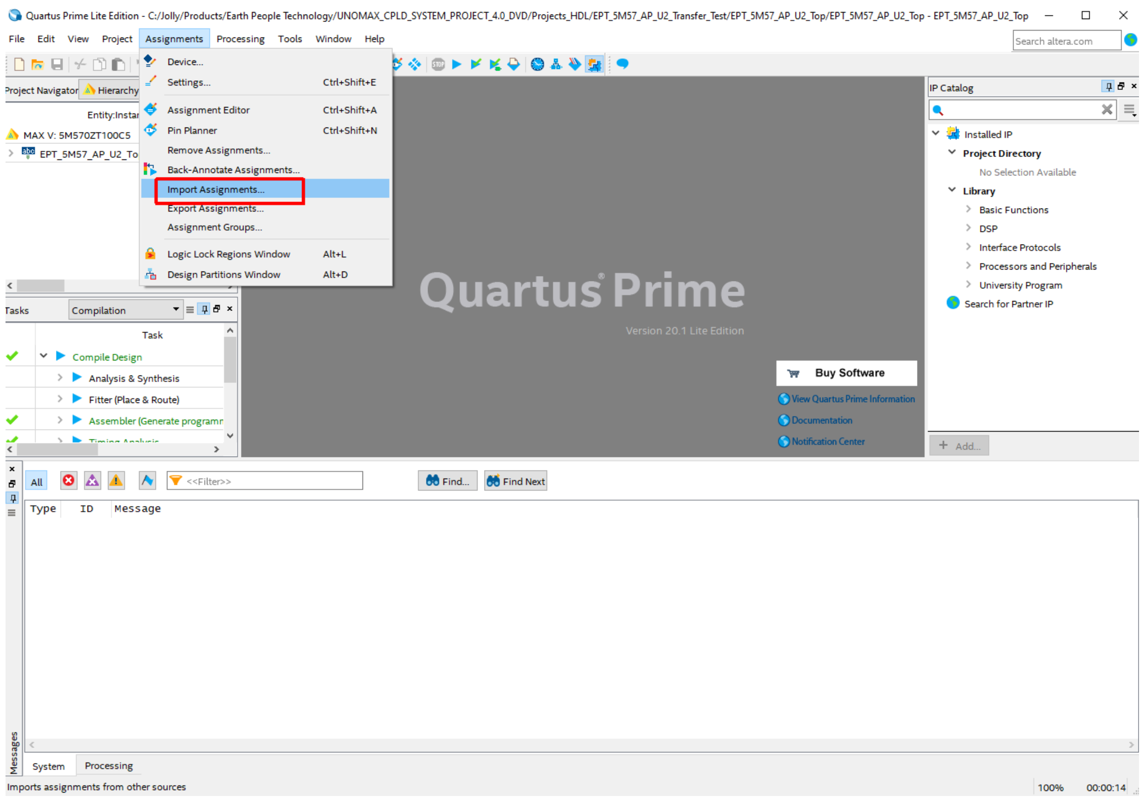
When you click Finish, the project will be created with the following settings:

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 ℃

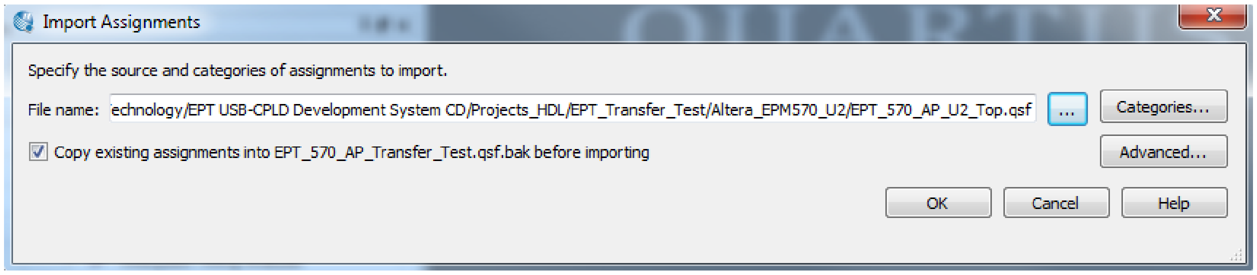
Next, we will select the pins and synthesize the project.

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_M4_Top.v) will connect directly to pins on the CPLD. The Pin Planner Tool from Quartus Prime will add the pins and check to verify that our pin selections do not violate any restrictions of the device. In the case of this example we will import pin assignments that created at an earlier time. Under Assignments, Select Import Assignments.



At the Import Assignment dialog box, Browse to the \Projects_HDL\EPT_Transfer_Test \ EPT_MegaProLogic_TOP folder of the MegaProLogic Development System DVD. Select the “EPT_570_AP_M4_Top.qsf” file.

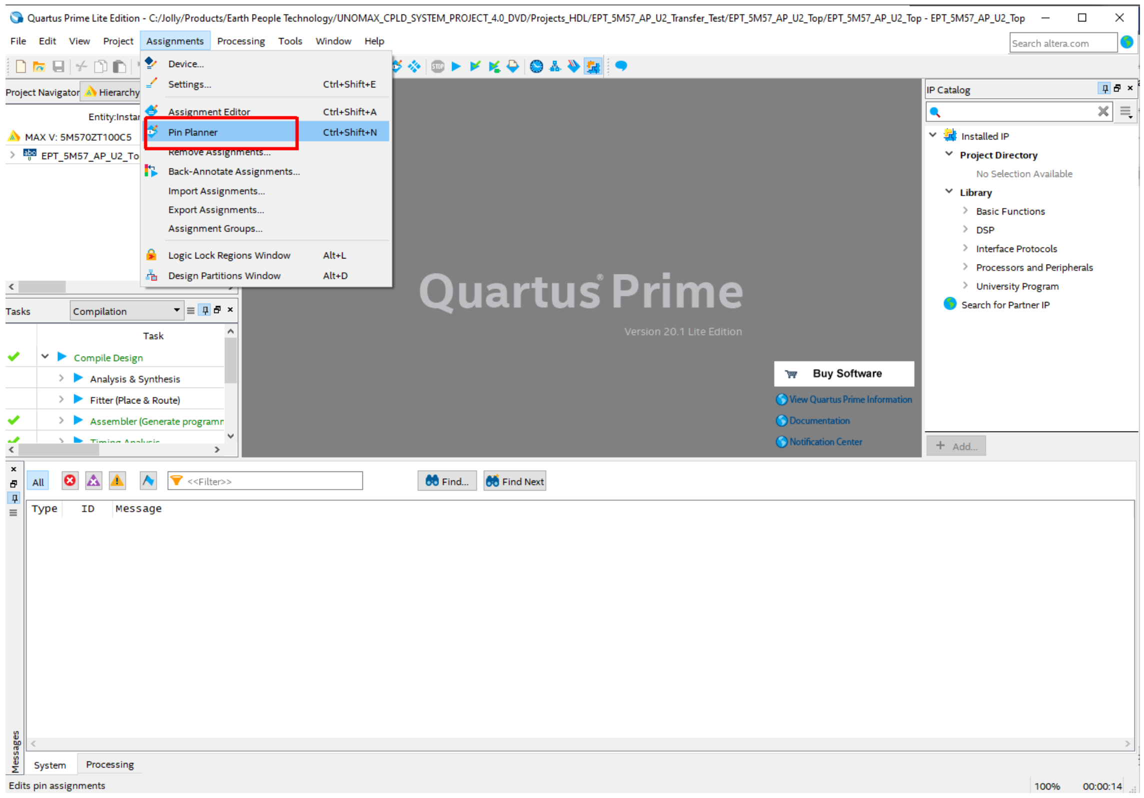

Import Assignments

Specify the source and categories of assignments to import.

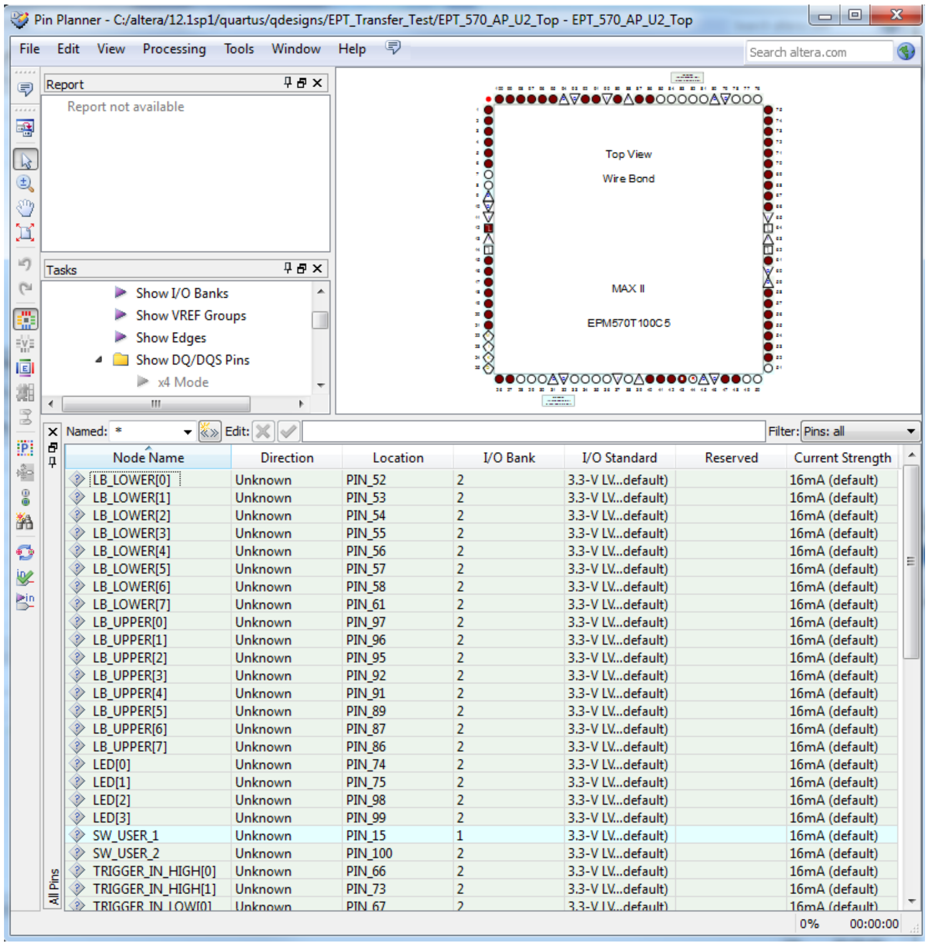
File name: echnology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/Altera_EPM570_U2/EPT_570_AP_U2_Top.qsf

[✓] Copy existing assignments into EPT_570_AP_Transfer_Test.qsf.bak before importing

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


Quartus Prime Lite Edition - C:/Jolly/Products/Earth People Technology/UNOMAX_CPLD_SYSTEM_PROJECT_4.0_DVD/Projects_HDL/EPT_5M57_AP_U2_Transfer_Test/EPT_5M57_AP_U2_Top/EPT_5M57_AP_U2_Top - EPT_5M57_AP_U2_Top

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.



Pin Planner - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top

Report not available

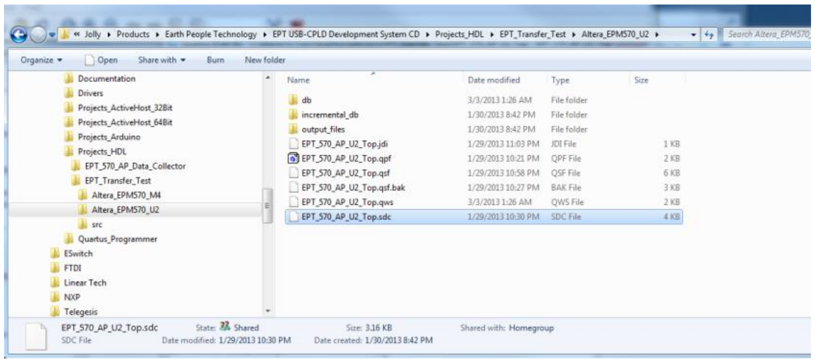
Named: * Filter: Pins: all

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

https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/VHDL/Timequest.pdf

Browse to the \Projects_HDL\EPT_Transfer_Test \ EPT_MegaProLogic_TOP folder of the MegaProLogic Development System DVD. Select the "EPT_570_AP_M4_Top.sdc" file.

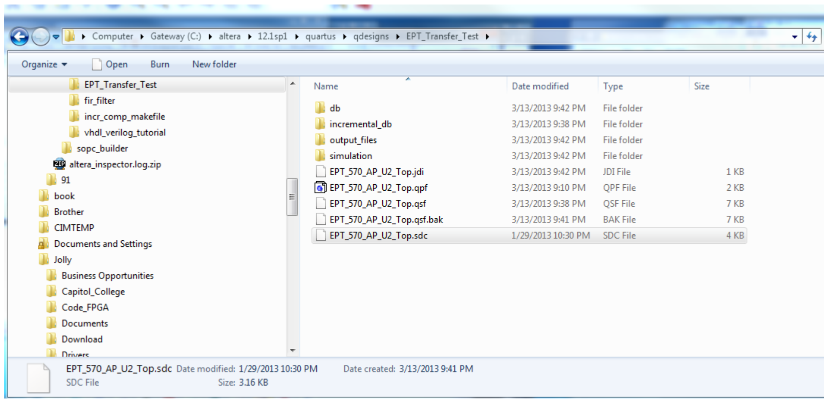


« Jolly ▸ Products ▸ Earth People Technology ▸ EPT USB-CPLD Development System CD ▸ Projects_HDL ▸ EPT_Transfer_Test ▸ Altera_EPM570_U2 ▸

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.x\quartus\qdesignscreate \EPT_Transfer_Test directory. Paste the file.

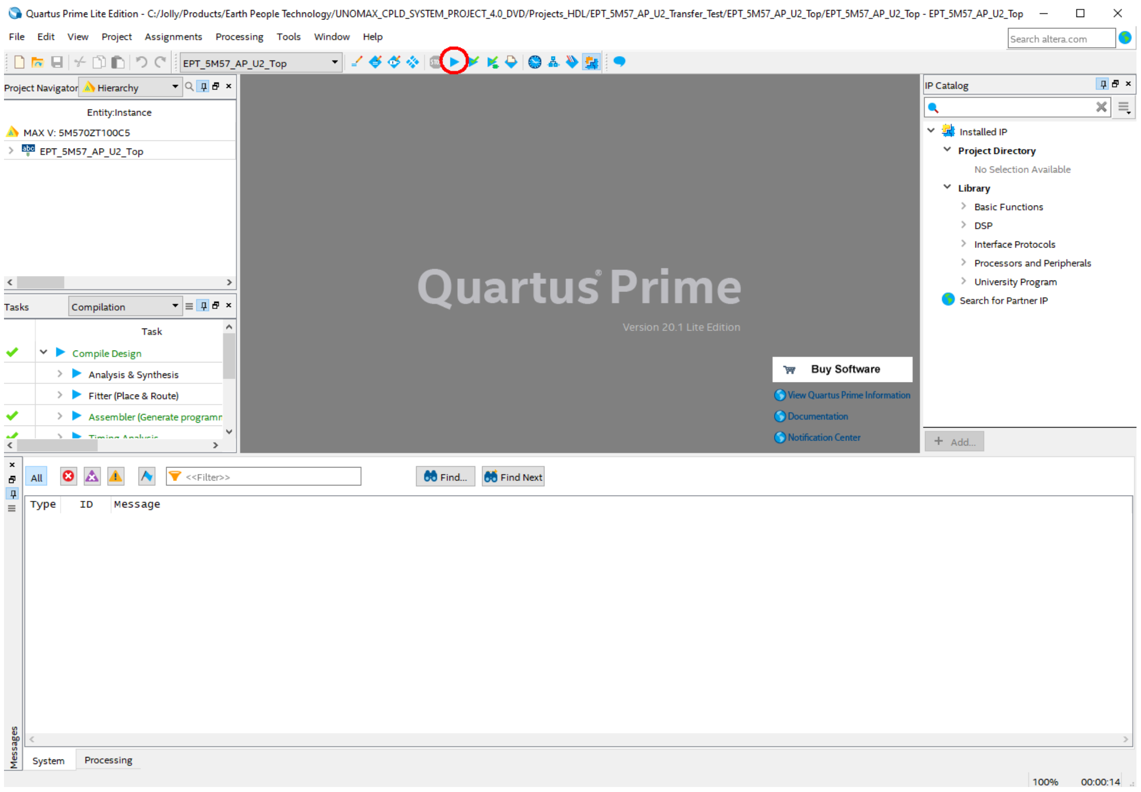


▸ Computer ▸ Gateway (C:) ▸ altera ▸ 12.1sp1 ▸ quartus ▸ qdesigns ▸ EPT_Transfer_Test ▸

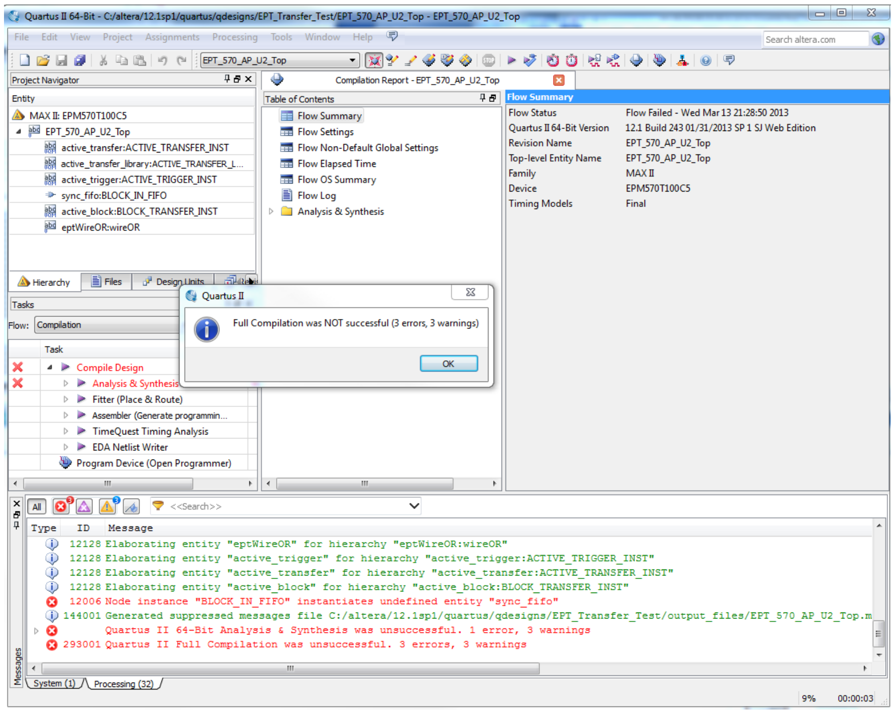
Name	Date modified	Type	Size
db	3/13/2013 9:42 PM	File folder	
incremental_db	3/13/2013 9:38 PM	File folder	
output_files	3/13/2013 9:42 PM	File folder	
simulation	3/13/2013 9:42 PM	File folder	
EPT_570_AP_U2_Top.jdi	3/13/2013 9:42 PM	JDI File	1 KB
EPT_570_AP_U2_Top.qpf	3/13/2013 9:10 PM	QPF File	2 KB
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 64-Bit - C:/altera/12.1sp1/quartus/qdesigns/EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2_Top

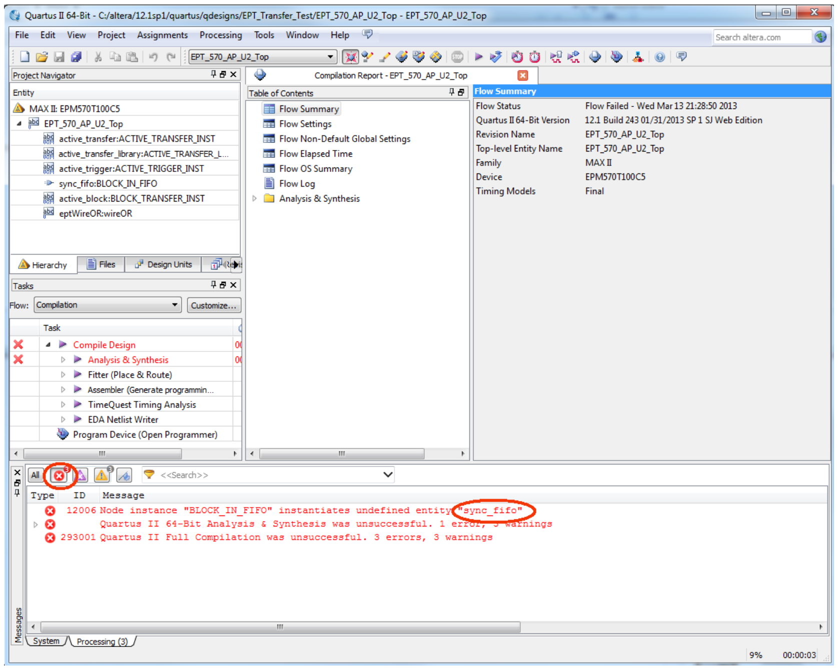
Flow Summary

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

Quartus II: Full Compilation was NOT successful (3 errors, 3 warnings)

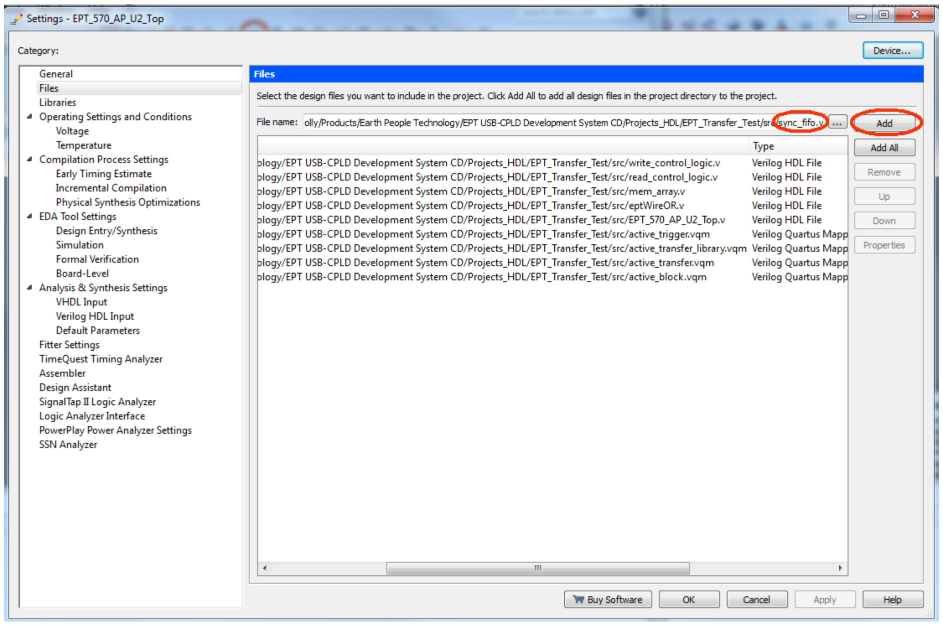
```
Type   ID     Message
       12128 Elaborating entity "eptWireOR" for hierarchy "eptWireOR:wireOR"
       12128 Elaborating entity "active_trigger" for hierarchy "active_trigger:ACTIVE_TRIGGER_INST"
       12128 Elaborating entity "active_transfer" for hierarchy "active_transfer:ACTIVE_TRANSFER_INST"
       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

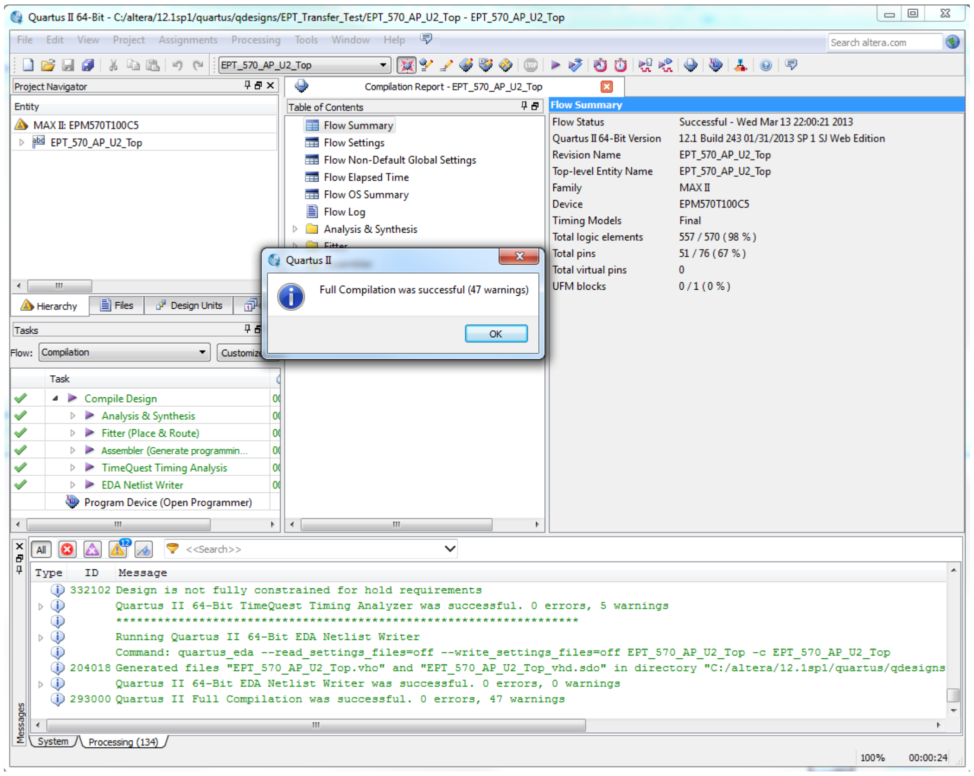
Files

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

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 %)

Quartus II

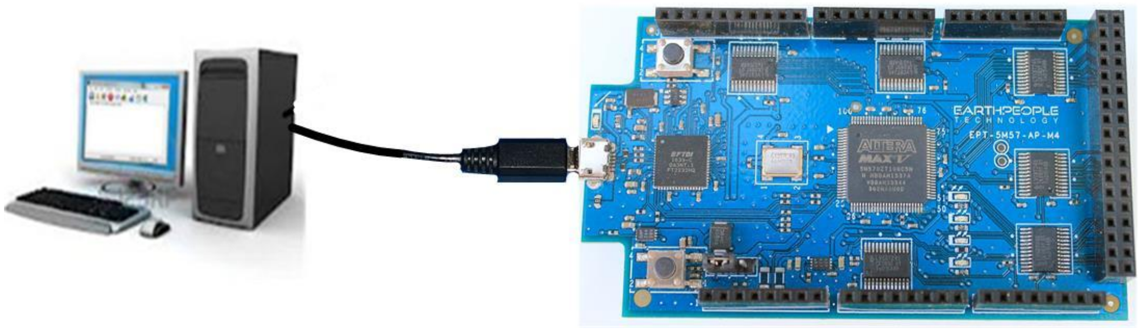
Full Compilation was successful (47 warnings)

```
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
```

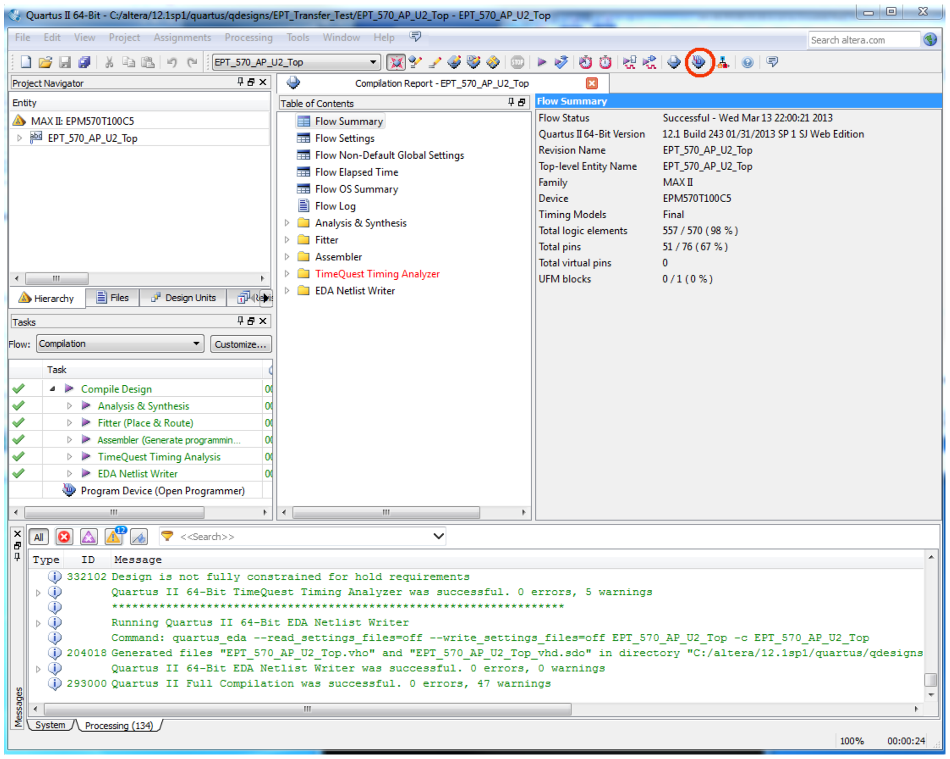
At this point the project has been successfully compiled, synthesized and a programming file has been produce. See the next section on how to program the CPLD.

4.1.2 Programming the CPLD

Programming the CPLD is quick and easy. All that is required is a standard USB cable with a Mini Type B connector on one end and the EPT_Blaster Driver DLL. Connect the MegaProLogic to the PC, open up Quartus Prime, 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.



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

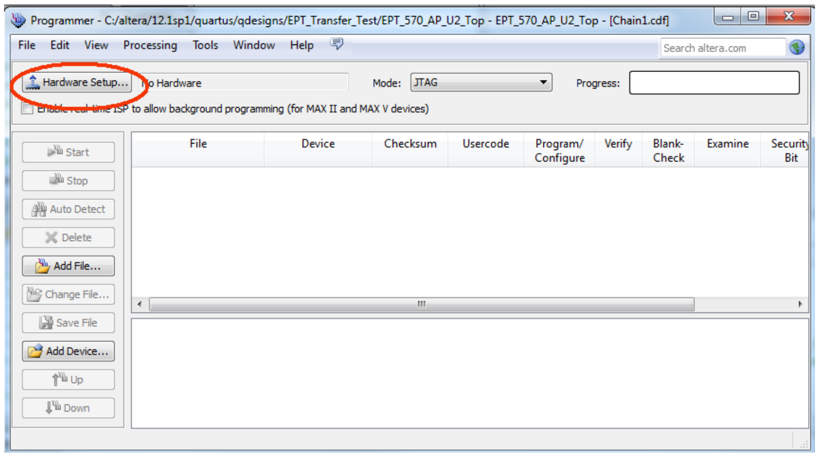


Flow Summary

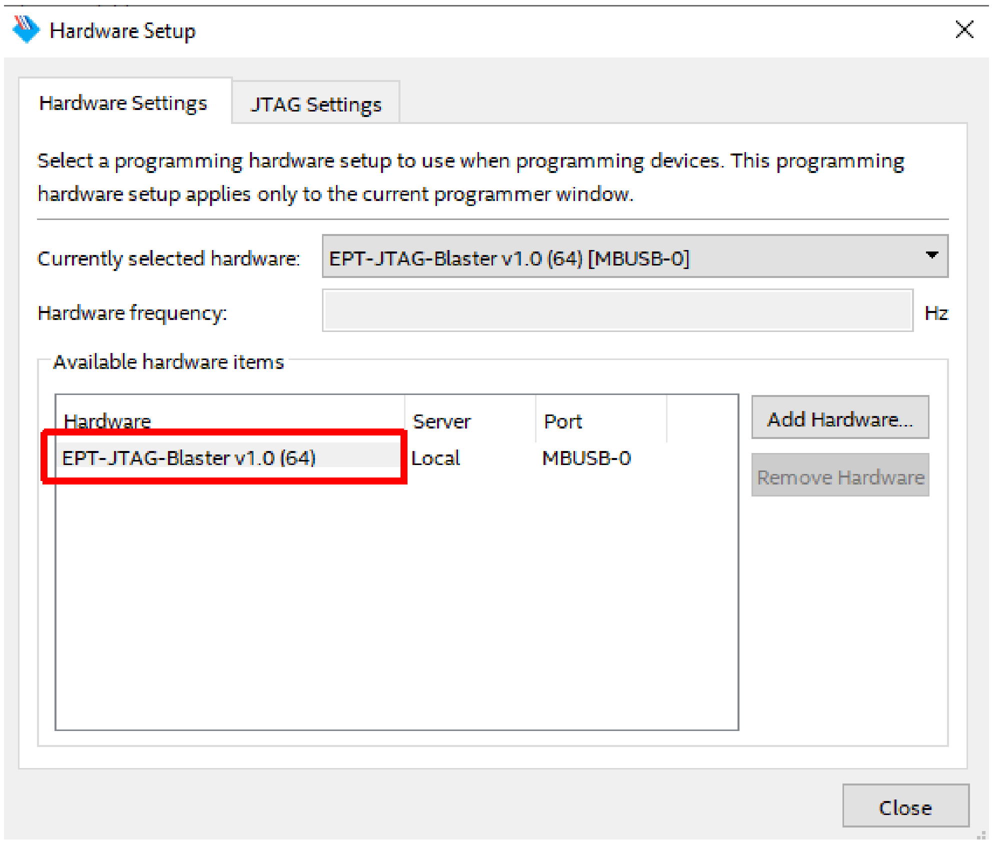
Flow Status	Successful - Wed Mar 13 22:00:21 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	EPT_570_AP_U2_Top
Top-level Entity Name	EPT_570_AP_U2_Top
Family	MAX II
Device	EPM570T100C5
Timing Models	Final
Total logic elements	557 / 570 (98 %)
Total pins	51 /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

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



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


Hardware Setup

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

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

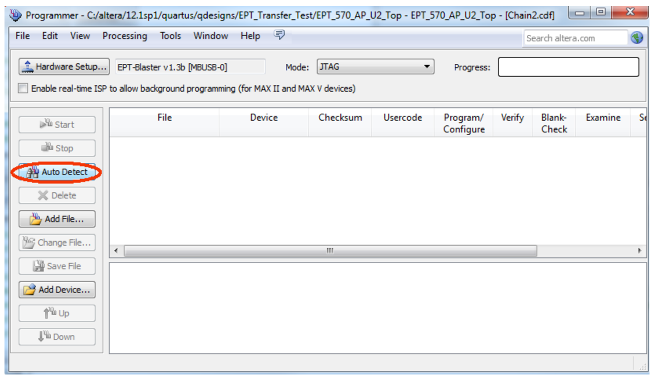
Hardware frequency: Hz

Available hardware items

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

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

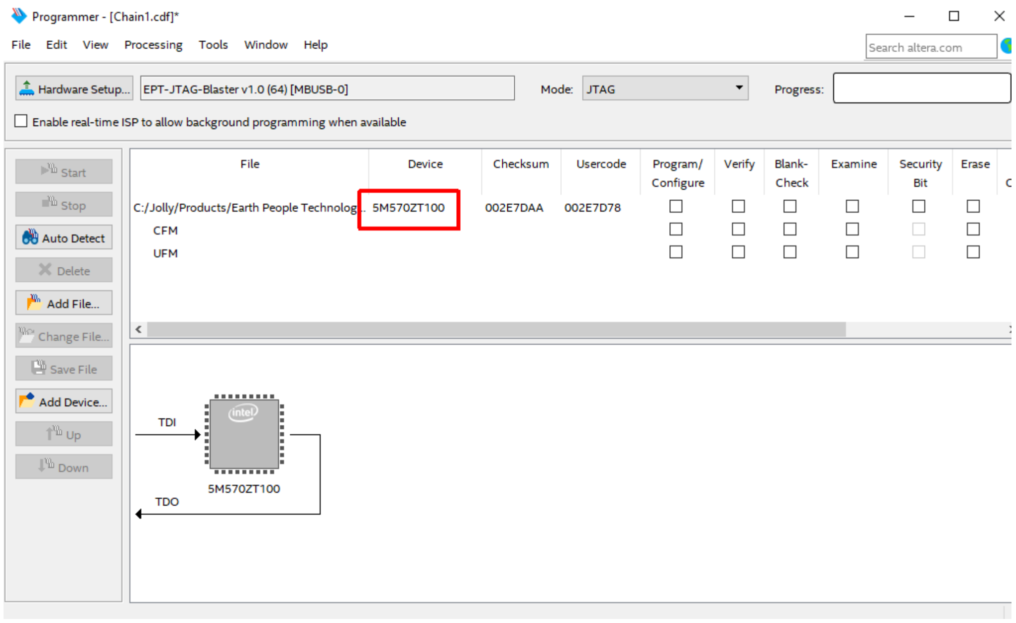
Click on the Auto-Detect button. This will verify that the EPT-Blaster driver can connect with the MegaProLogic device.


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 Progress:

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

Select the 5M570 under "Device".

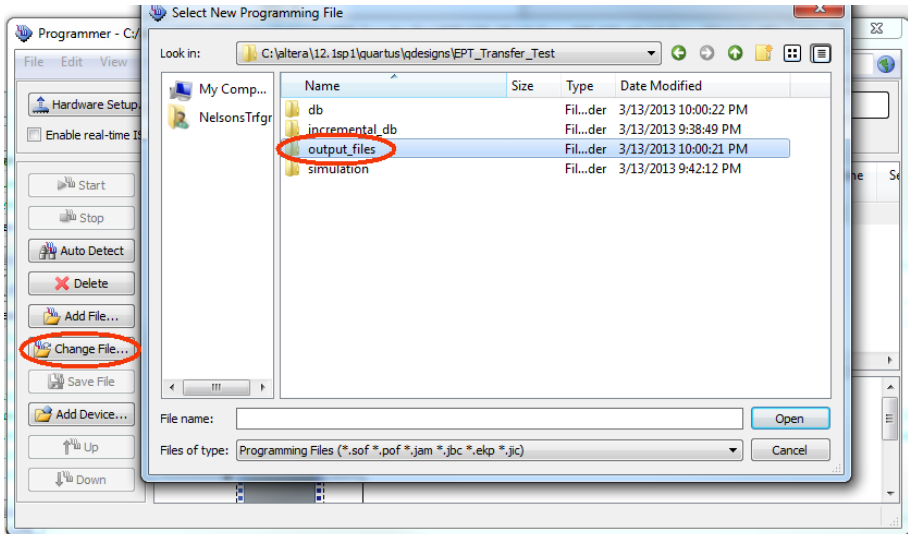


Hardware: EPT-JTAG-Blaster v1.0 (64) [MBUSB-0] Mode: JTAG Progress:

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
C:/Jolly/Products/Earth People Technolog...	5M570ZT100	002E7DAA	002E7D78	☐	☐	☐	☐	☐	☐
CFM				☐	☐	☐	☐	☐	☐
UFM				☐	☐	☐	☐	☐	☐

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



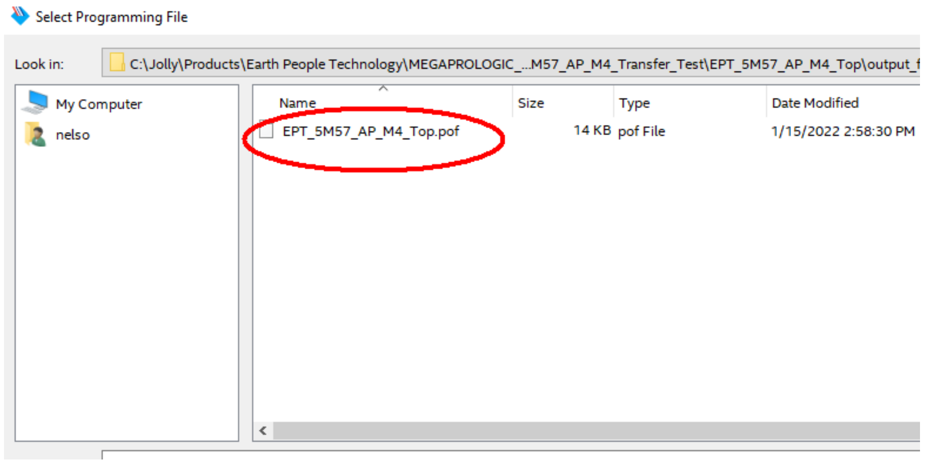
Look in: C:\altera\12.1sp1\quartus\qdesigns\EPT_Transfer_Test

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

File name:

Files of type: Programming Files (*.sof *.pof *.jam *.jbc *.ekp *.jic)

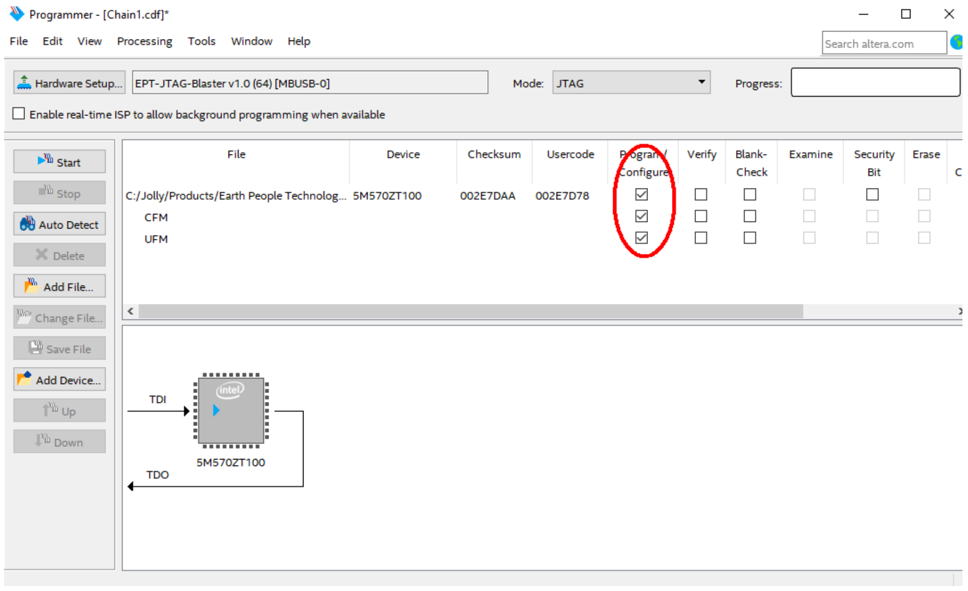
Click on the EPT_5M57_AP_M4_Top.pof file to select it.


Select Programming File
Look in: C:\Jolly\Products\Earth People Technology\MEGAPROLOGIC_...M57_AP_M4_Transfer_Test\EPT_5M57_AP_M4_Top\output_f
My Computer
nelso
Name
Size
Type
Date Modified

Name	Size	Type	Date Modified
EPT_5M57_AP_M4_Top.pof	14 KB	pof File	1/15/2022 2:58:30 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.

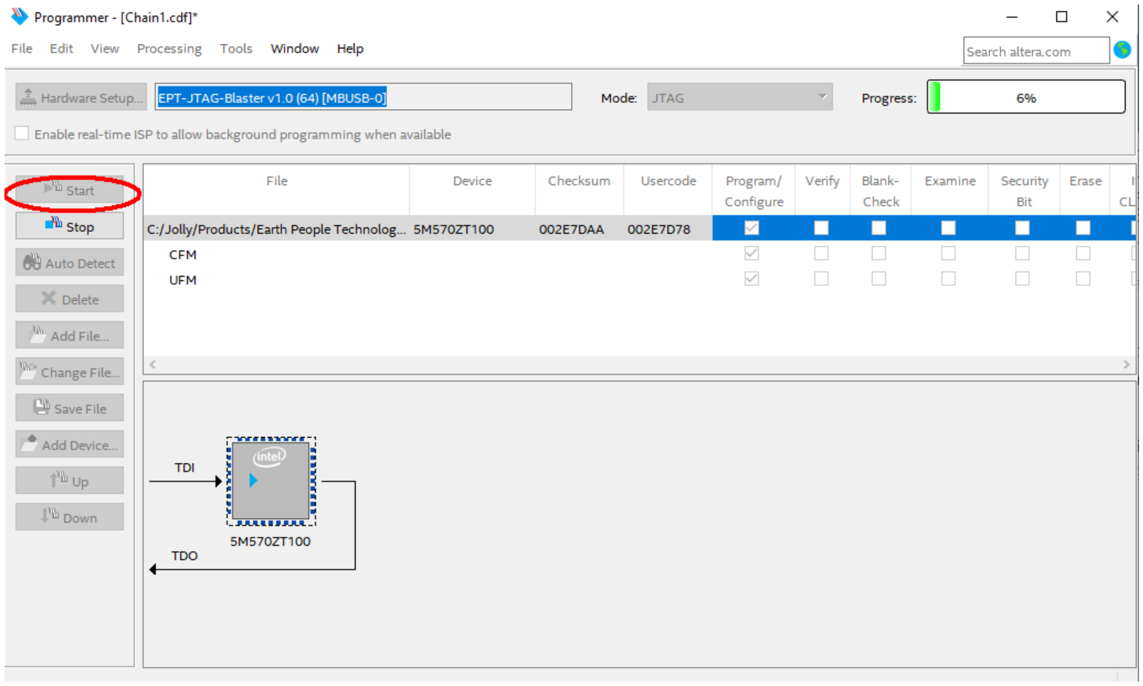


Hardware: EPT-JTAG-Blaster v1.0 (64) [MBUSB-0] Mode: JTAG Progress:

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

File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine	Security Bit	Erase
C:/Jolly/Products/Earth People Technolog...	5M570ZT100	002E7DAA	002E7D78	☑	☐	☐	☐	☐	☐
CFM				☑	☐	☐	☐	☐	☐
UFM				☑	☐	☐	☐	☐	☐

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

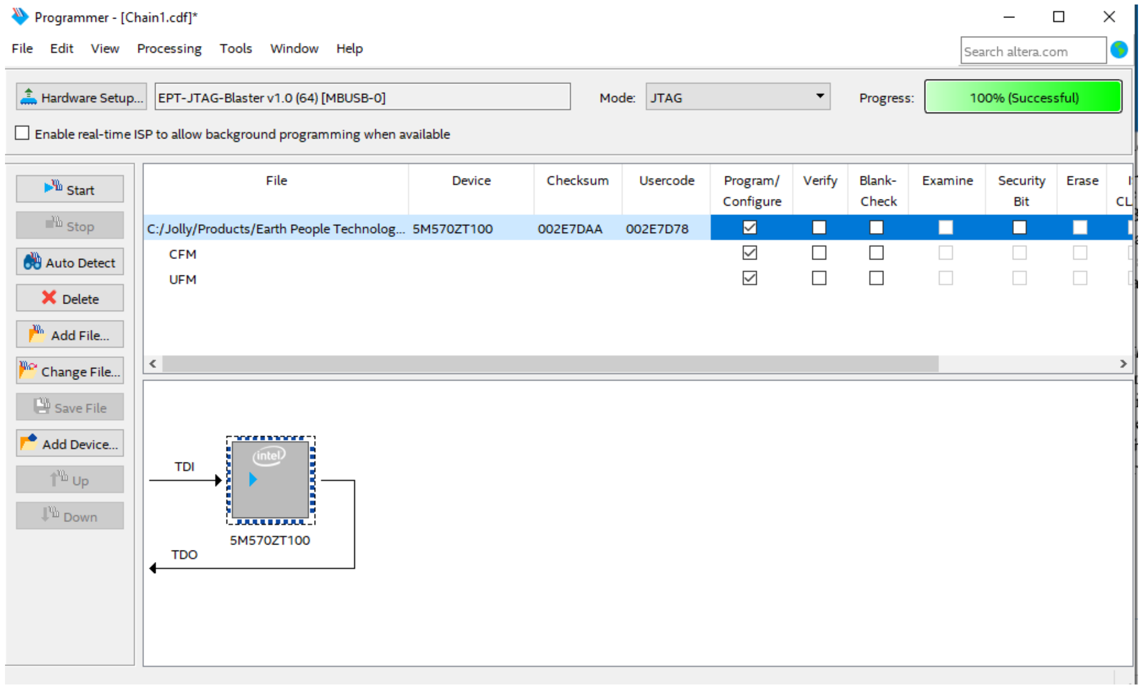


Hardware: EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]   Mode: JTAG   Progress: 6%

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
C:/Jolly/Products/Earth People Technolog...	5M570ZT100	002E7DAA	002E7D78	☑	☐	☐	☐	☐	☐
CFM				☑	☐	☐	☐	☐	☐
UFM				☑	☐	☐	☐	☐	☐

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



Hardware: EPT-JTAG-Blaster v1.0 (64) [MBUSB-0]   Mode: JTAG   Progress: 100% (Successful)

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
C:/Jolly/Products/Earth People Technolog...	5M570ZT100	002E7DAA	002E7D78	☑	☐	☐	☐	☐	☐
CFM				☑	☐	☐	☐	☐	☐
UFM				☑	☐	☐	☐	☐	☐

At this point, the MegaProLogic 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

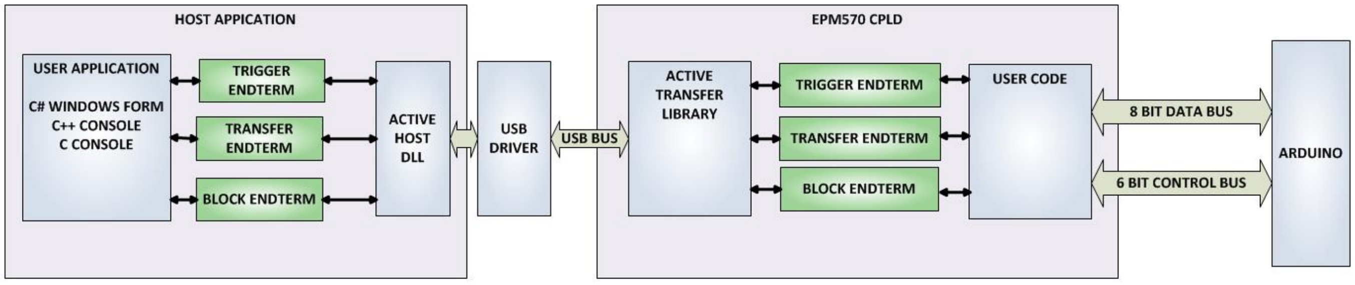
LED's and verify that the LED selected lights up. Press one of the switches on the board and ensure that the switch is captured on the Active Host Test Tool. Now you are ready to connect to the Arduino Uno and write some code to transfer data between microcontroller and PC.

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

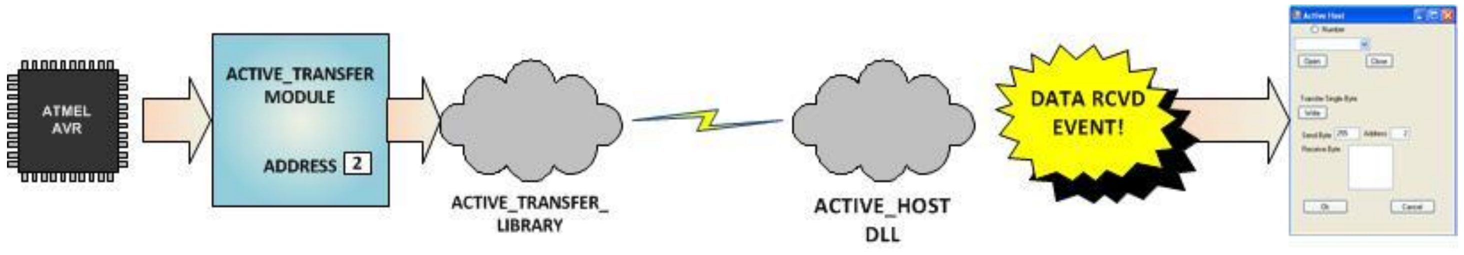
The Transfer EndTerm is a software component that provides a direct path from the users application to the commensurate Transfer EndTerm in the CPLD. It is used to transfer a byte to and from the CPLD. Eight separate Transfer EndTerm modules can be instantiated in the CPLD. Each module is addressed by 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.



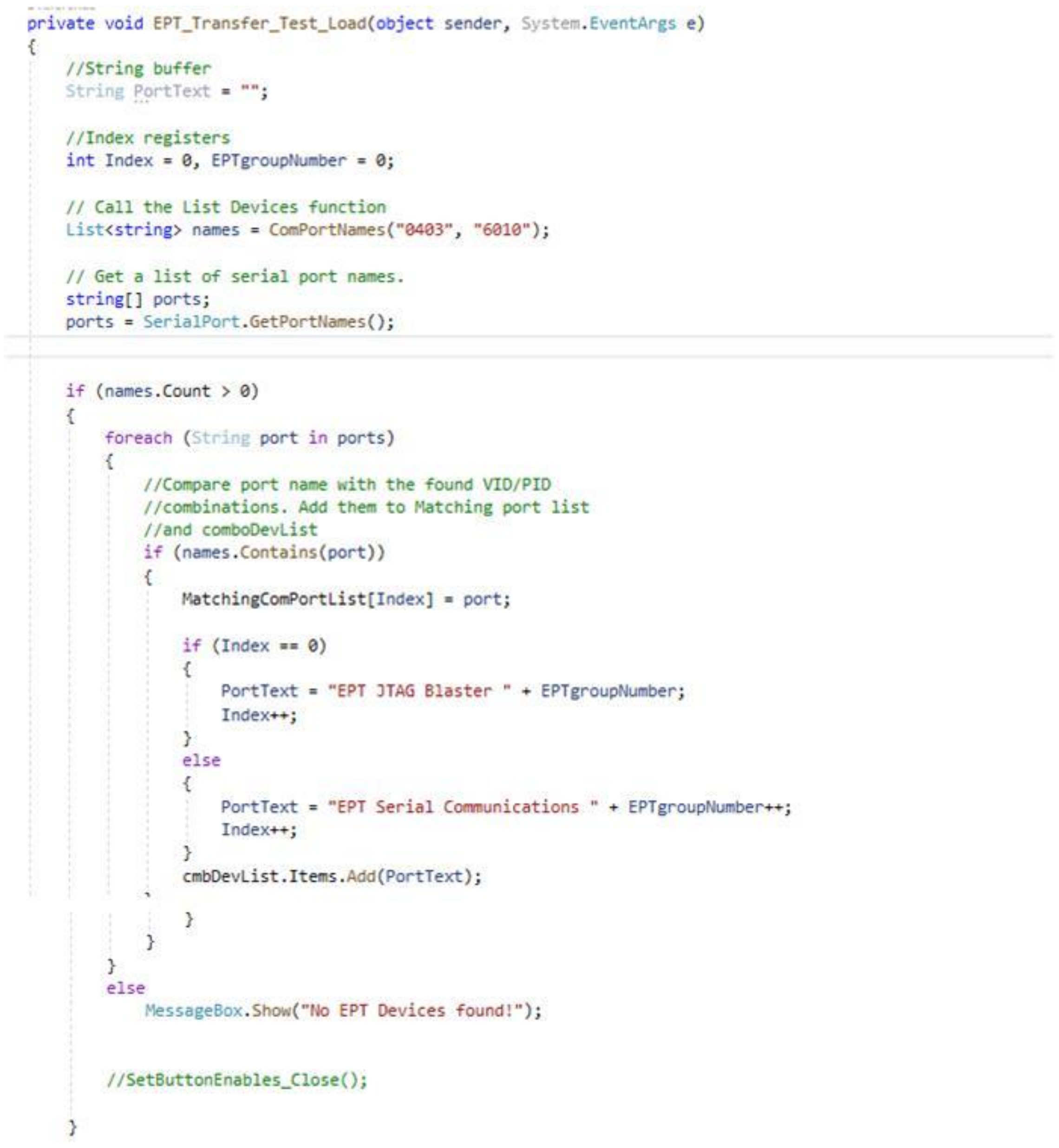
The data seamlessly appears in Host PC memory from the Arduino. The user code will direct the data to a control such as a textbox on a Windows Form. The transparent receive transfer path is made possible by a Callback mechanism in the Active Host dll. The dll calls a registered callback function in the 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_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();
```

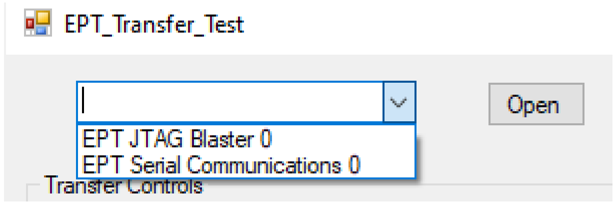
to determine the Serial devices attached to the PC. Next,

```
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);
```

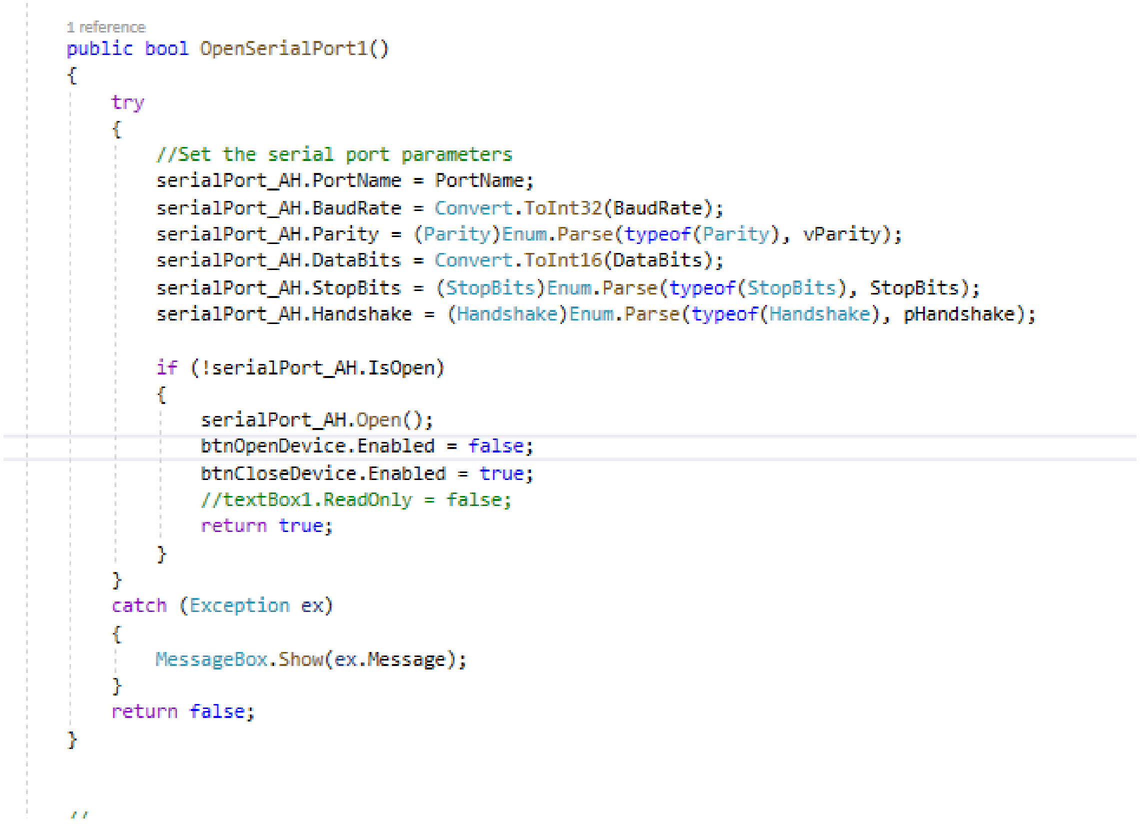
The user will select the device from the drop down combo box. This can be seen when the Windows Form is opened and the cmbDevList combo box is populated with all the devices. The selected device will be stored as an index number in the variable device_index.



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

```
OpenSerialPort1()
```

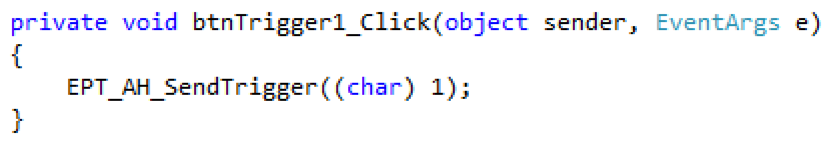
function. The device_index is passed into the function. If the function is successful, the device name is displayed in the label, labelDeviceCnt. Next, the Open button is grayed out and the Close button is made active.



```
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

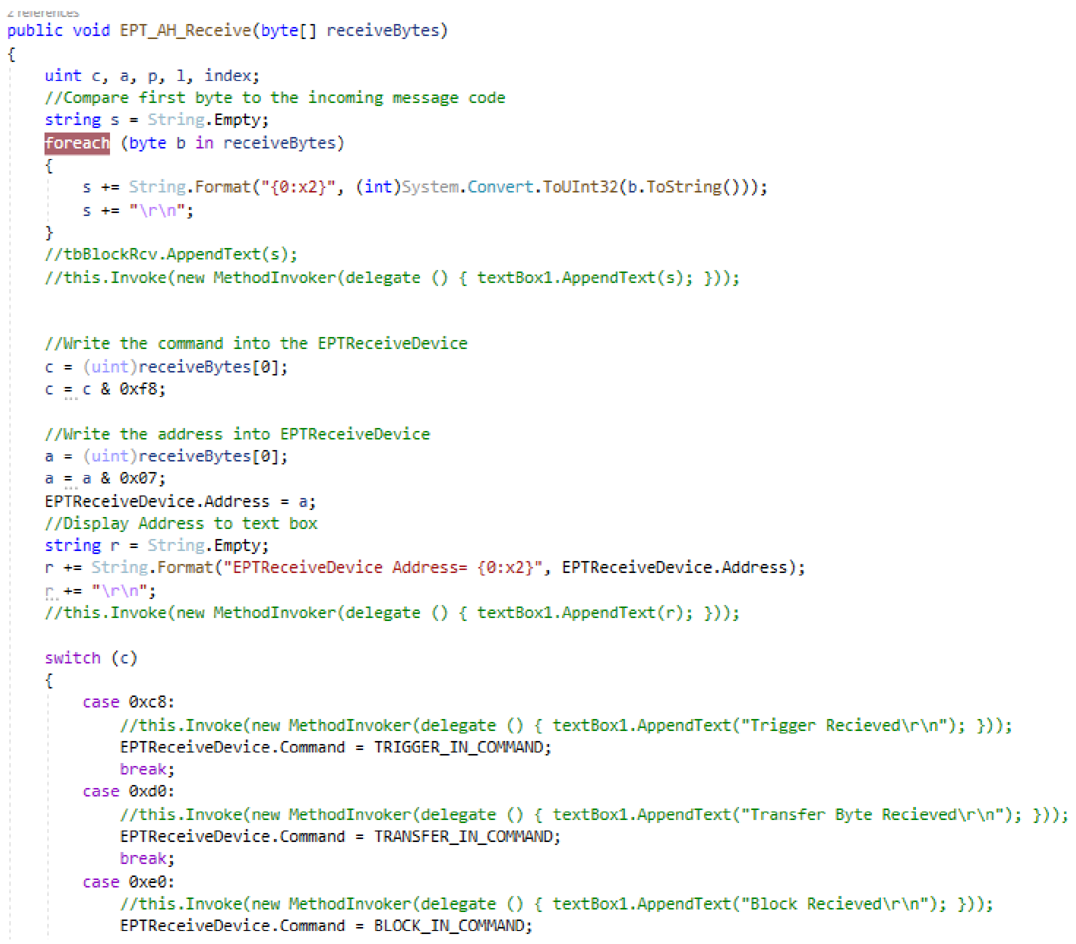
The user application can send a trigger to the CPLD by using the EPT_AH_SendTrigger() function. First, open the EPT device to be used with OpenSerialPort1 (). Call the function with the bit or bits to assert high on the trigger byte as the parameter. Then execute the function, the trigger bit or bits will momentarily assert high in the user code on the CPLD.



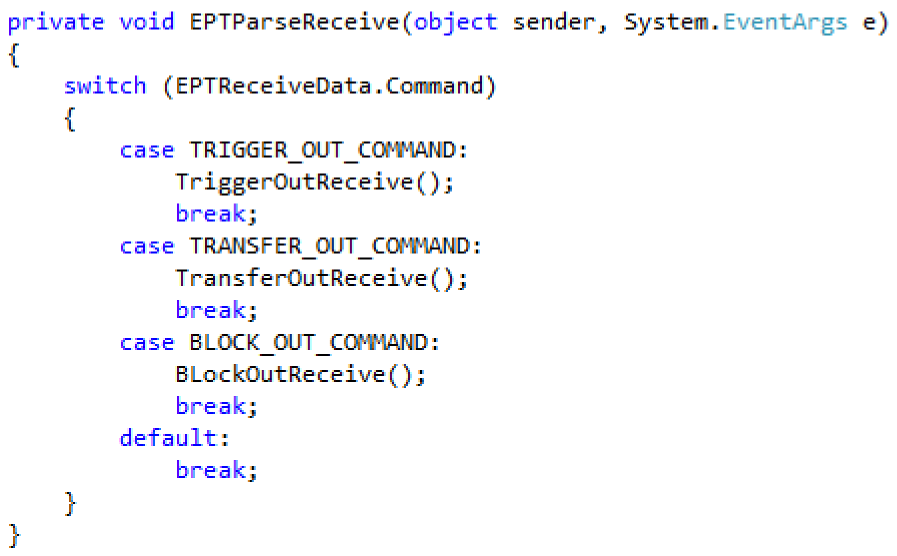
```
private void btnTrigger1_Click(object sender, EventArgs e)
{
    EPT_AH_SendTrigger((char) 1);
}
```

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

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

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

5.4.4 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.

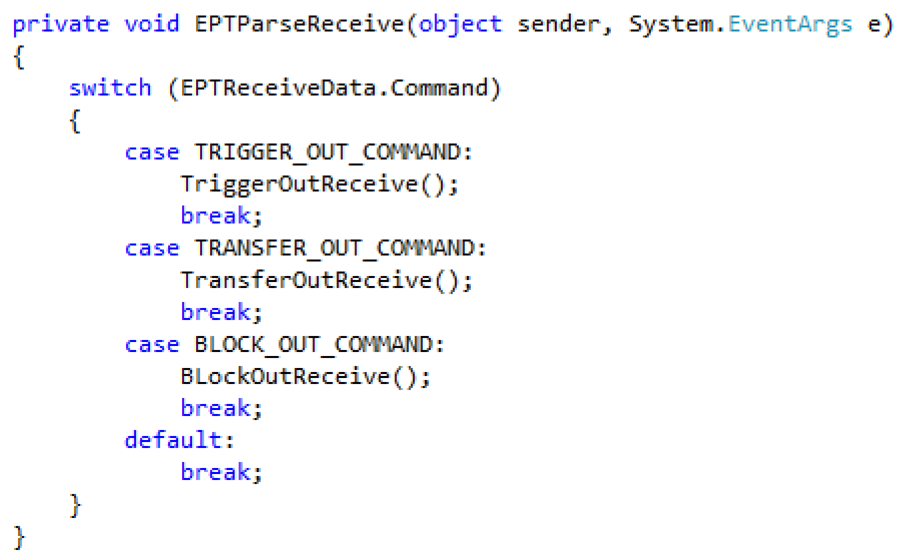
To send a block, first, open the EPT device to be used with EPT_AH_OpenDeviceByIndex(). Next, use the EPT_AH_SendBlock() function to send the block. Add the address of the transfer module as the first parameter. Next, place the pointer to the buffer in the second parameter of EPT_AH_SendBlock(). Add 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 EventHandler function EPTParseReceive() is called by the Read Callback function. The EPTParseReceive() function will examine the command of the incoming byte transfer and determine which receive function to call.

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

For our example project, the Receive_Block_In() function writes the Transfer block received to a text block.

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

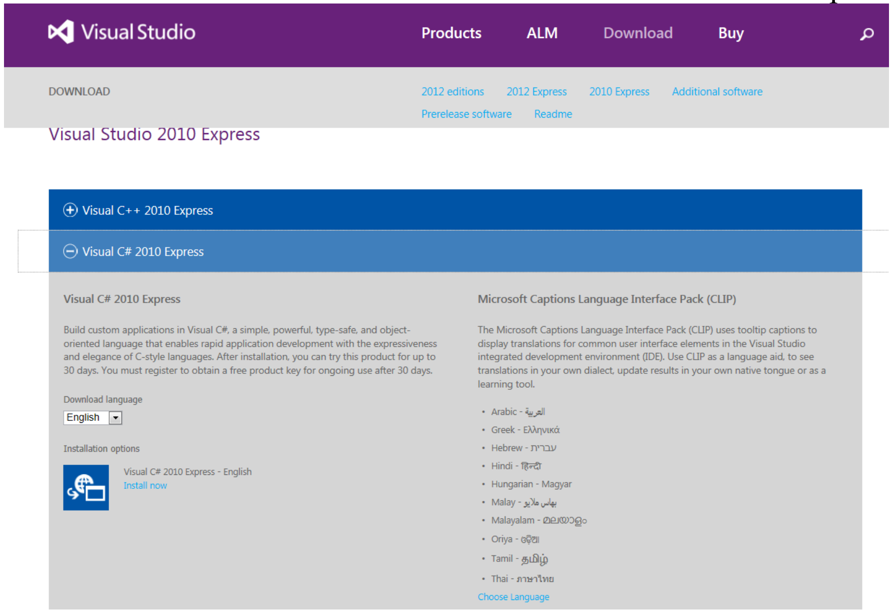
The Active Host Application DLL is used to build a custom standalone executable on the PC that can perform Triggers and Transfer data to/from the MEGAPROLOGIC. A standalone project can be range from a simple program to display and send data from the user to/from the Arduino MEGA. Or it can more complex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

6.1 Creating a Project

Download the latest version of Microsoft Visual C# 2010 Express environment from Microsoft. It's a free download.

http://www.microsoft.com/visualstudio/eng/downloads#d-2010-express

Go to the website and click on the "+" icon next to the Visual C# 2010 Express.



Visual Studio 2010 Express

Visual C# 2010 Express

Build custom applications in Visual C#, a simple, powerful, type-safe, and object-oriented language that enables rapid application development with the expressiveness and elegance of C-style languages. After installation, you can try this product for up to 30 days. You must register to obtain a free product key for ongoing use after 30 days.

Download language: English

Installation options

Visual C# 2010 Express - English
Install now

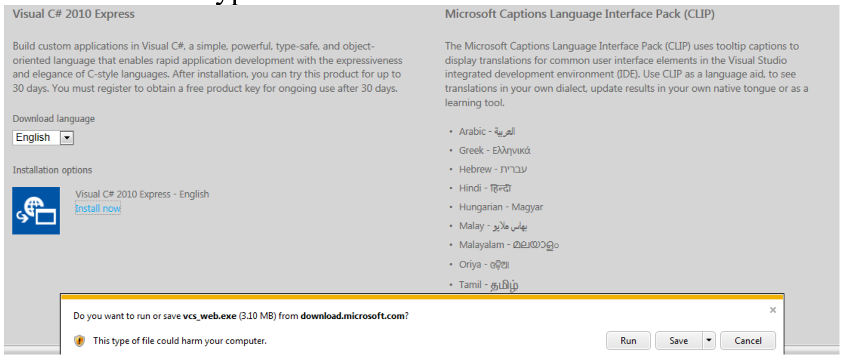
Microsoft Captions Language Interface Pack (CLIP)

The Microsoft Captions Language Interface Pack (CLIP) uses tooltip captions to display translations for common user interface elements in the Visual Studio integrated development environment (IDE). Use CLIP as a language aid, to see translations in your own dialect, update results in your own native tongue or as a learning tool.

- Arabic - العربية
- Greek - Ελληνικά
- Hebrew - עברית
- Hindi - हिन्दी
- Hungarian - Magyar
- Malay - بهاس ملايو
- Malayalam - മലയാളം
- Oriya - ଓଡ଼ିଆ
- Tamil - தமிழ்
- Thai - ภาษาไทย

Choose Language

Click on the "Install now" hypertext.



Visual C# 2010 Express

Build custom applications in Visual C#, a simple, powerful, type-safe, and object-oriented language that enables rapid application development with the expressiveness and elegance of C-style languages. After installation, you can try this product for up to 30 days. You must register to obtain a free product key for ongoing use after 30 days.

Download language: English

Installation options

Visual C# 2010 Express - English
Install now

Microsoft Captions Language Interface Pack (CLIP)

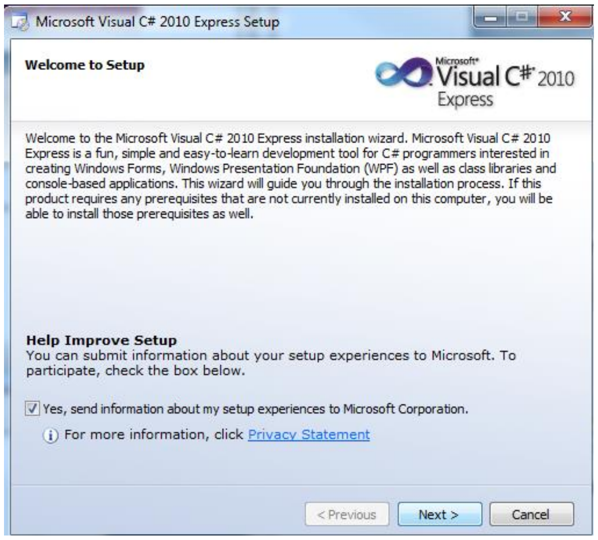
The Microsoft Captions Language Interface Pack (CLIP) uses tooltip captions to display translations for common user interface elements in the Visual Studio integrated development environment (IDE). Use CLIP as a language aid, to see translations in your own dialect, update results in your own native tongue or as a learning tool.

- Arabic - العربية
- Greek - Ελληνικά
- Hebrew - עברית
- Hindi - हिन्दी
- Hungarian - Magyar
- Malay - بهاس ملايو
- Malayalam - മലയാളം
- Oriya - ଓଡ଼ିଆ
- Tamil - தமிழ்

Do you want to run or save **vcs_web.exe** (3.10 MB) from **download.microsoft.com**?

This type of file could harm your computer.

Click the "Run" button.



Microsoft Visual C# 2010 Express Setup

Welcome to Setup

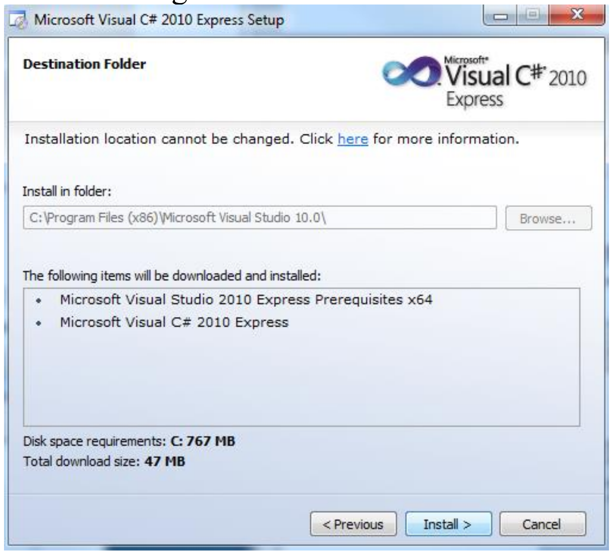
Welcome to the Microsoft Visual C# 2010 Express installation wizard. Microsoft Visual C# 2010 Express is a fun, simple and easy-to-learn development tool for C# programmers interested in creating Windows Forms, Windows Presentation Foundation (WPF) as well as class libraries and console-based applications.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

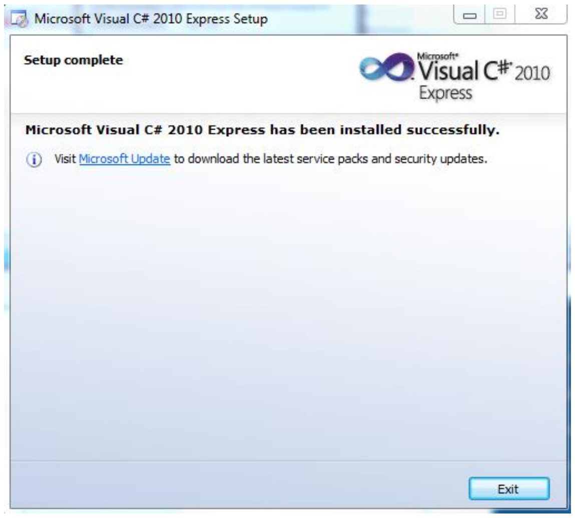
Installation location cannot be changed. Click here for more information.

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

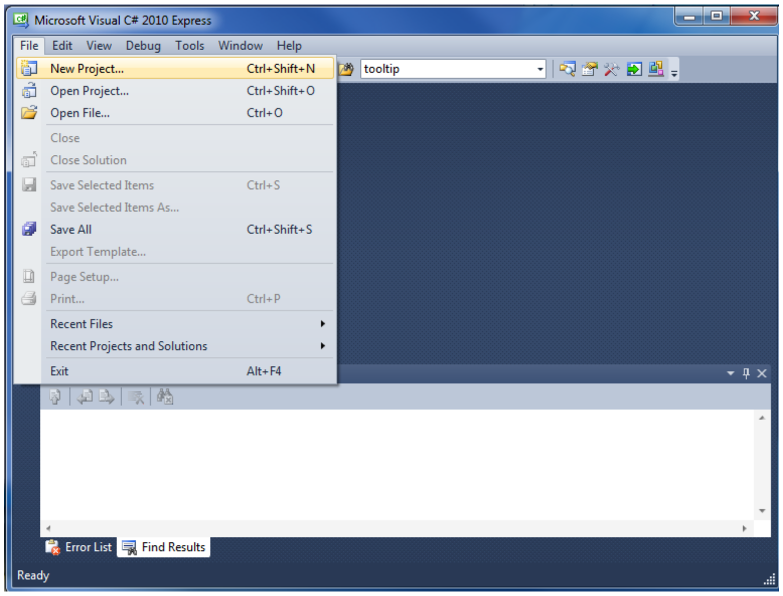
Setup complete

Microsoft Visual C# 2010 Express has been installed successfully.

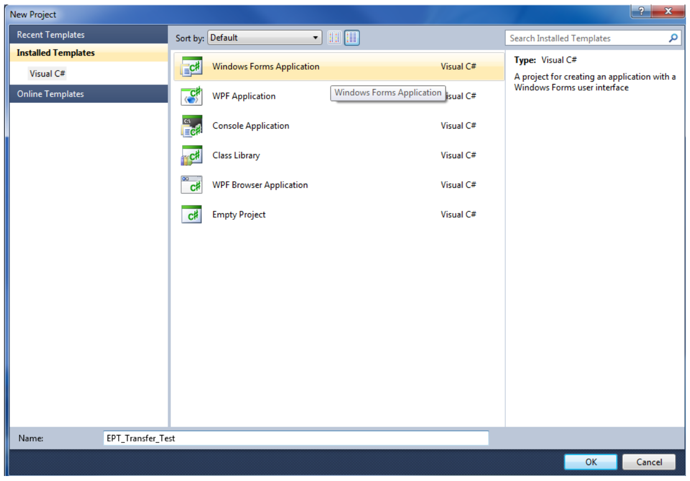
Visit Microsoft Update to download the latest service packs and security updates.

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

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

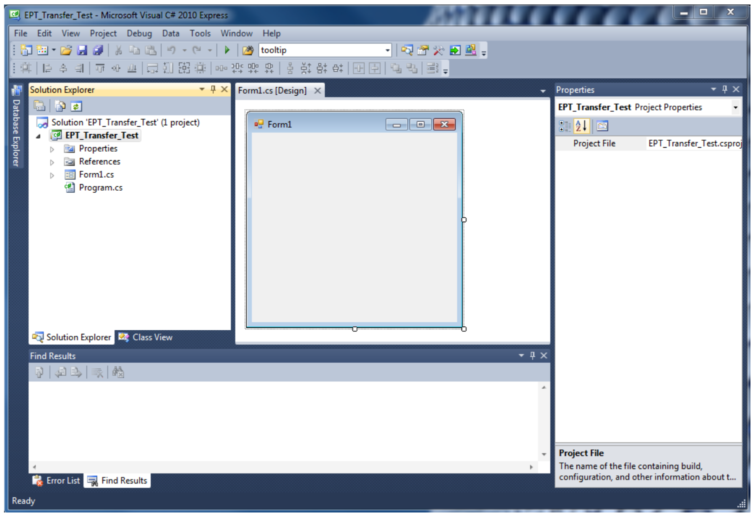

New Project
Recent Templates
Installed Templates
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
OK
Cancel

Type: Visual C#

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

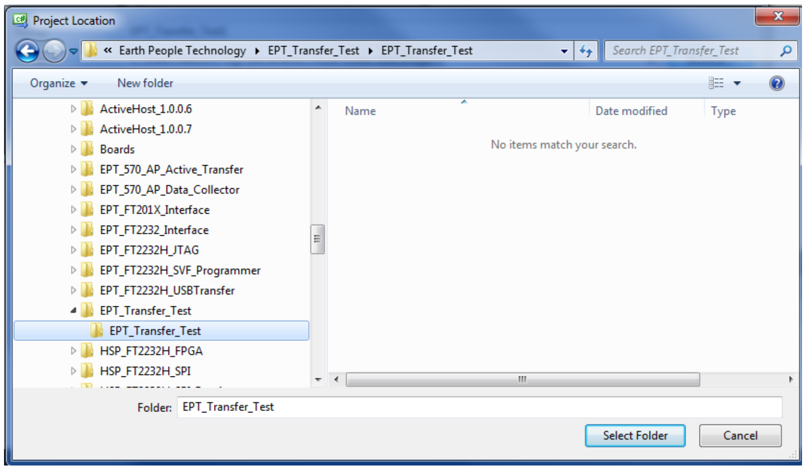
Name: EPT_Transfer_Test

The project creation is complete.


EPT_Transfer_Test - Microsoft Visual C# 2010 Express
File Edit View Project Debug Data Tools Window Help
Solution Explorer
Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs
Form1.cs [Design]
Form1
Properties
EPT_Transfer_Test Project Properties
Project File EPT_Transfer_Test.csproj
Find Results
Error List
Find Results
Ready

Project File
The name of the file containing build, configuration, and other information about t...

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.

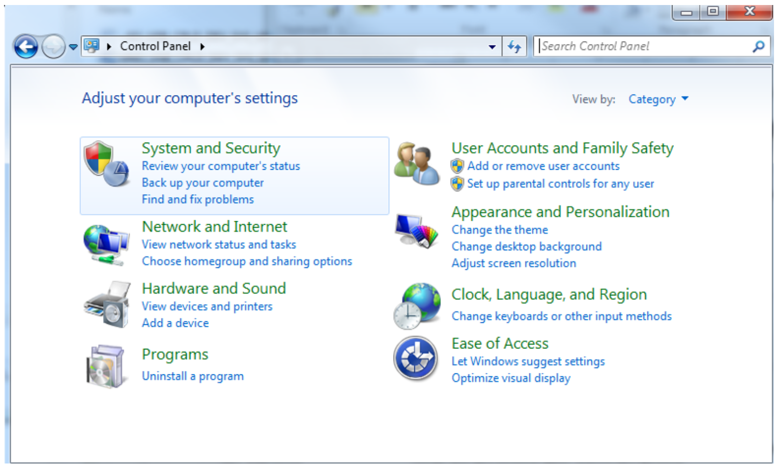


No items match your search.

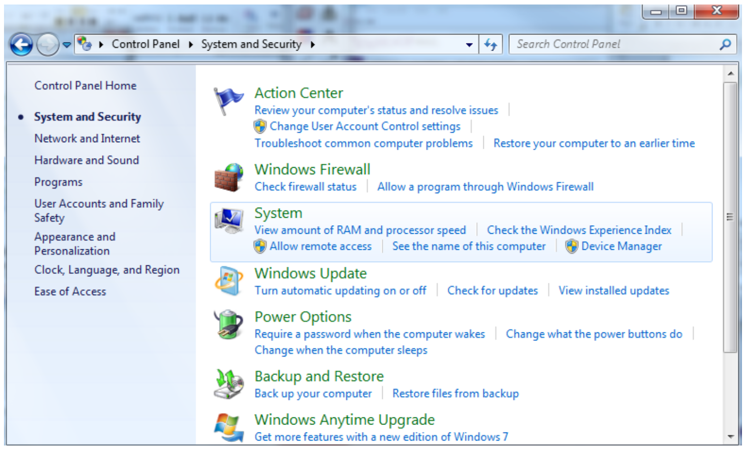
Folder: EPT_Transfer_Test

6.1.1 Setting up the C# Express Environment for 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

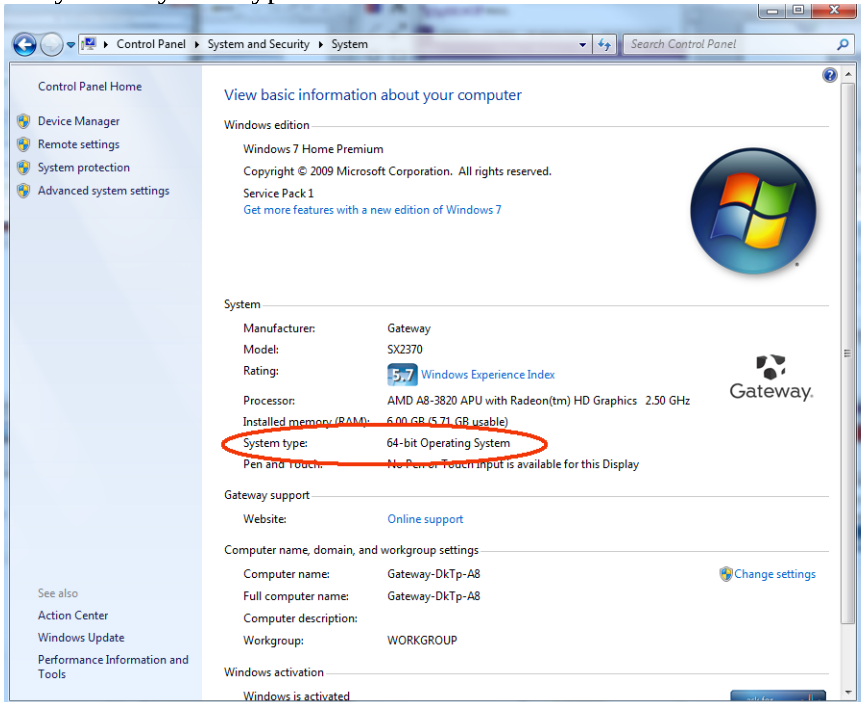
Windows Update
Turn automatic updating on or off | Check for updates | View installed updates

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

Search Control Panel

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

See also
- Action Center
- Windows Update
- Performance Information and Tools

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

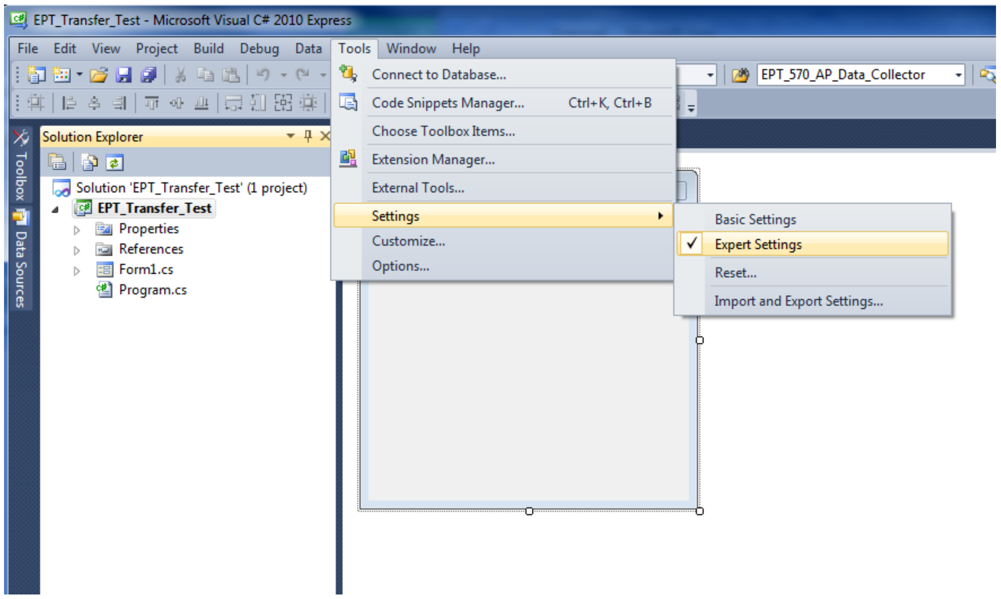
Gateway support
Website: Online support

Computer name, domain, and workgroup settings

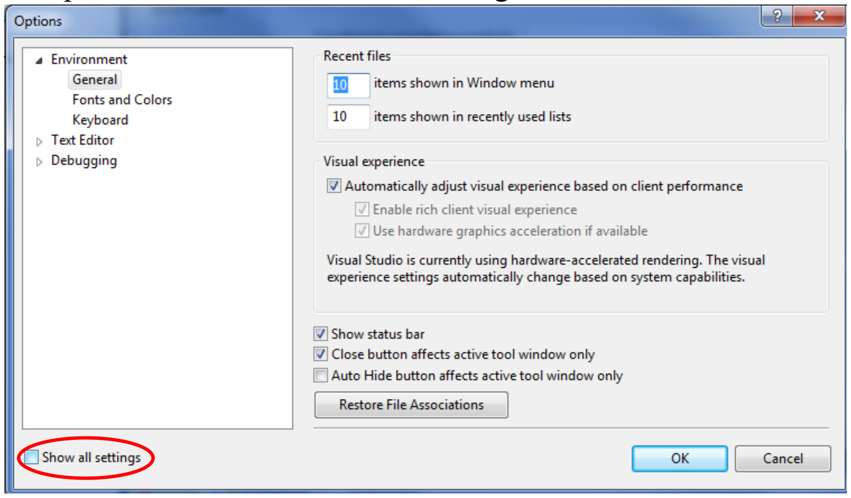
Computer name:	Gateway-DkTp-A8	Change settings
Full computer name:	Gateway-DkTp-A8	
Computer description:		
Workgroup:	WORKGROUP	

Windows activation
Windows is activated

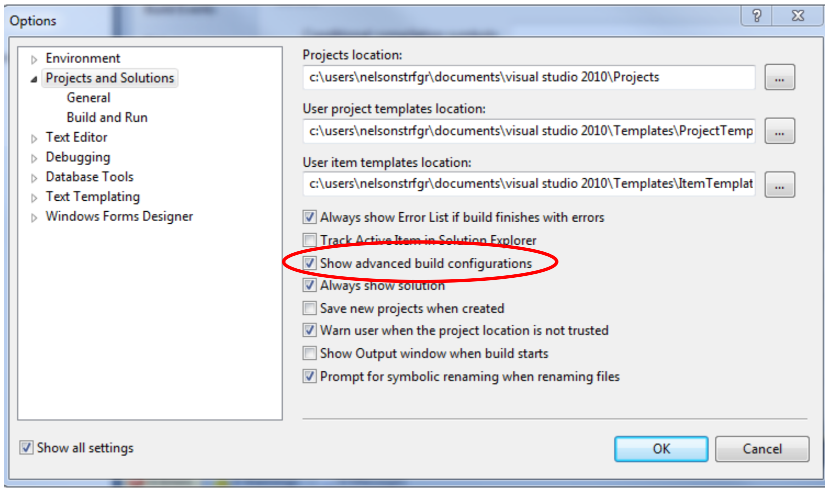
First, we need tell C# Express to produce 64 bit code if we are running on a x64 platform. Go to Tools->Settings and select Expert Settings


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

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


Options
Environment
General
Fonts and Colors
Keyboard
Text Editor
Debugging
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
Restore File Associations
Show all settings
OK
Cancel

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


Options

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

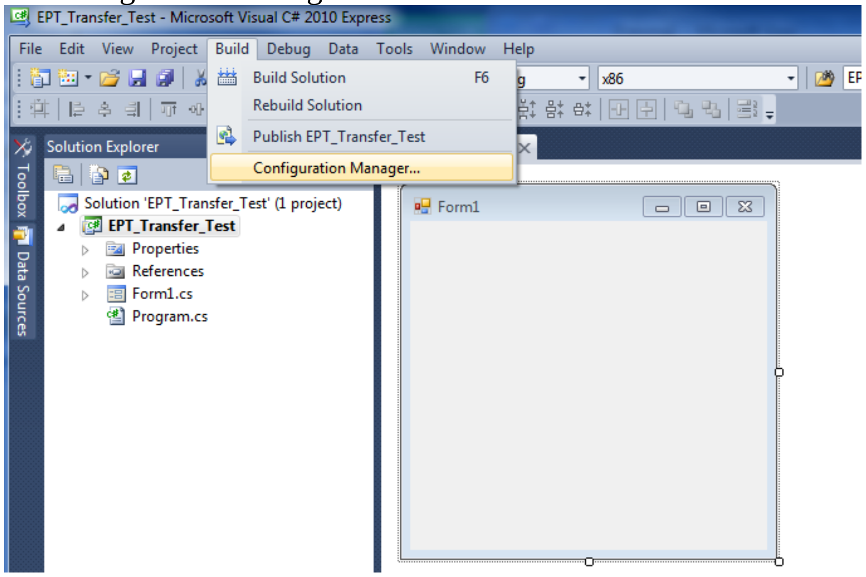
User project templates location:
c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ProjectTemp

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

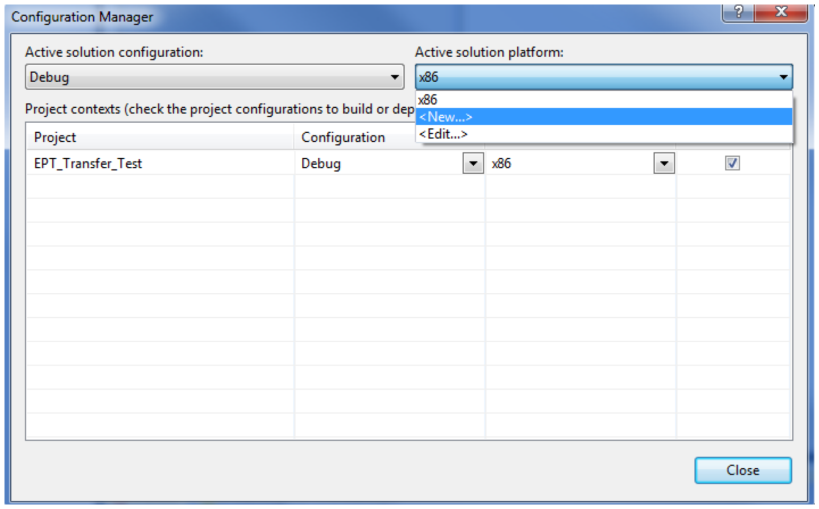
- [✓] Always show Error List if build finishes with errors
- [ ] Track Active Item in Solution Explorer
- [✓] Show advanced build configurations
- [✓] Always show solution
- [ ] Save new projects when created
- [✓] Warn user when the project location is not trusted
- [ ] Show Output window when build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

[✓] Show all settings

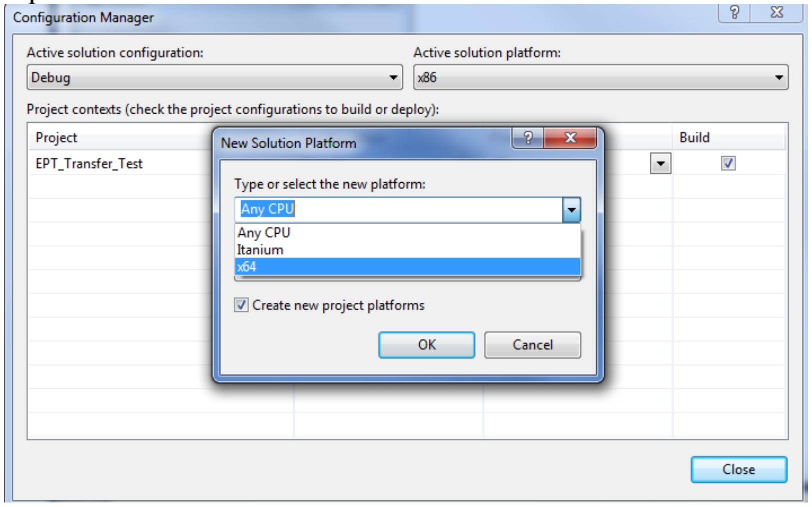
Go to Build->Configuration Manager.


EPT_Transfer_Test - Microsoft Visual C# 2010 Express

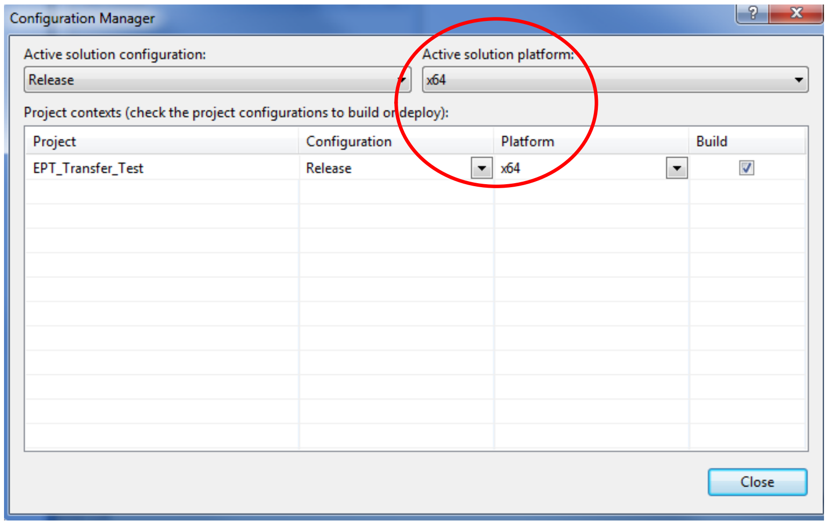
In the Configuration Manager window, locate the “Active solution platform:” label, select “New” from the drop down box.


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
OK
Cancel
Close

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


Configuration Manager

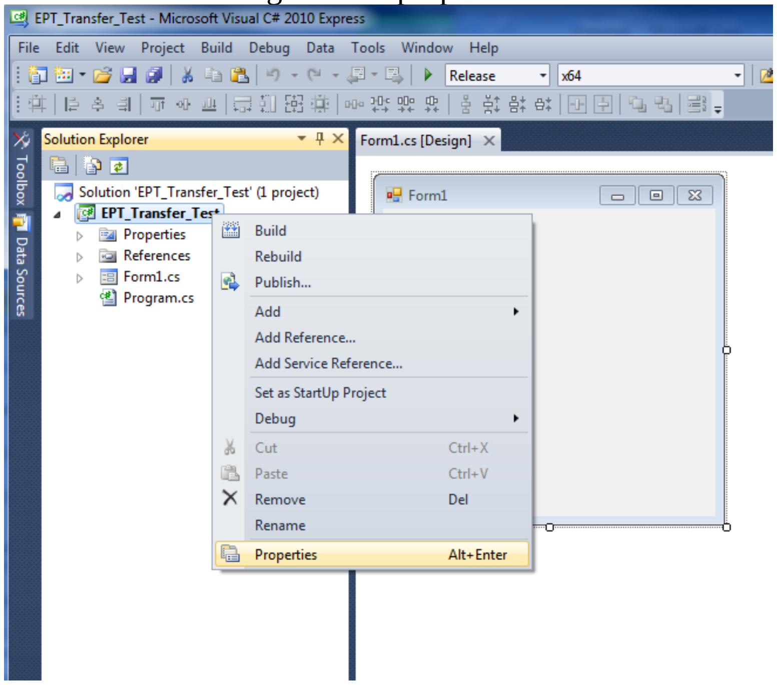
Active solution configuration: Release

Active solution platform: x64

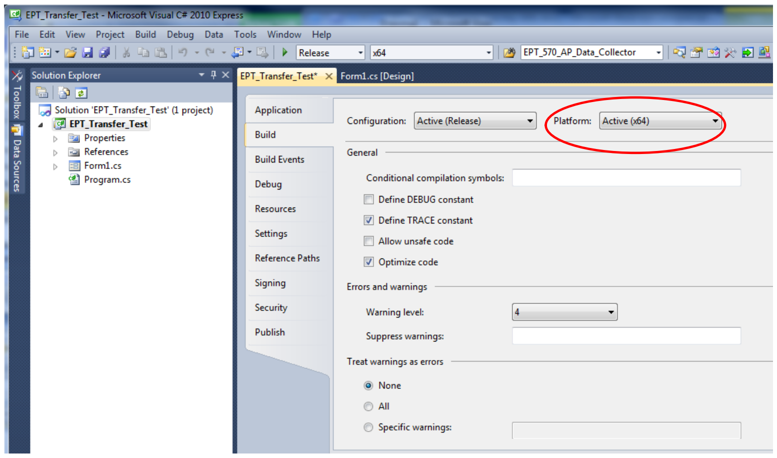
Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build
EPT_Transfer_Test	Release	x64	✓

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.



EPT_Transfer_Test - Microsoft Visual C# 2010 Express

Release | x64 | EPT_570_AP_Data_Collector

Solution 'EPT_Transfer_Test' (1 project)
EPT_Transfer_Test
Properties
References
Form1.cs
Program.cs

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

General
Conditional compilation symbols:
☐ Define DEBUG constant
☑ Define TRACE constant
☐ Allow unsafe code
☑ Optimize code

Errors and warnings
Warning level: 4
Suppress warnings:

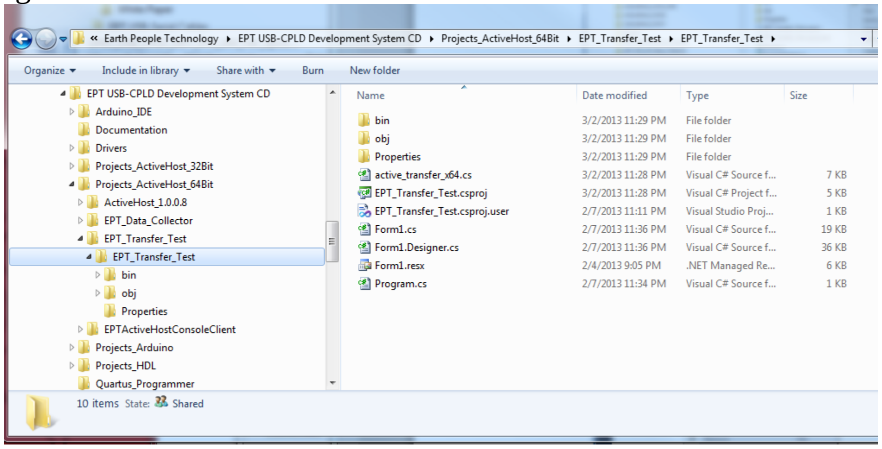
Treat warnings as errors
● None
○ All
○ Specific warnings:

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

6.2 *Assembling Files into the Project*

Locate the EPT 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. These files are:

- Active_transfer_xxx.cs
- Form1.cs
- Form1.Designer.cs
- Program.cs



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

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

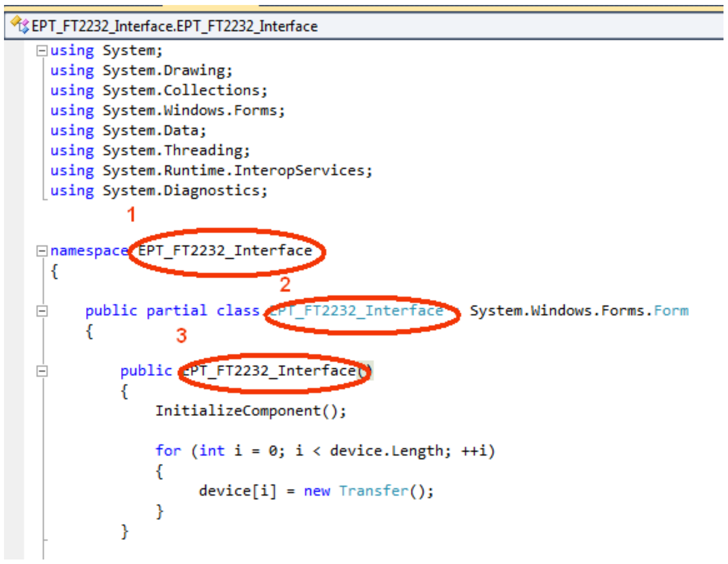
10 items State: Shared

6.2.1 Changing Project Name

NOTE

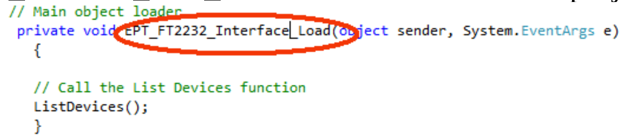
If you named your project something other than EPT_Transfer_Test, you will have to make changes to the *.cs files above. This is because Visual C# Express links the project files and program files together. These chages can be made by modifying the following:

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.



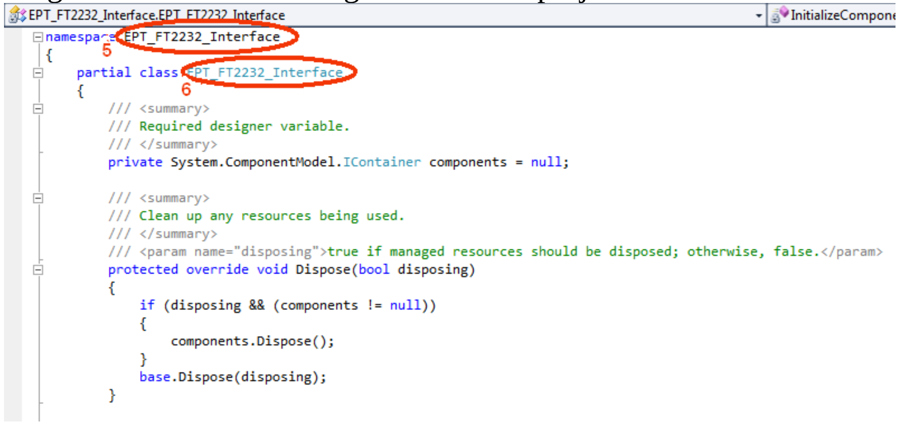
```
EPT_FT2232_Interface.EPT_FT2232_Interface
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;
                    1
namespace EPT_FT2232_Interface
{                              2
    public partial class EPT_FT2232_Interface : System.Windows.Forms.Form
    {         3
        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



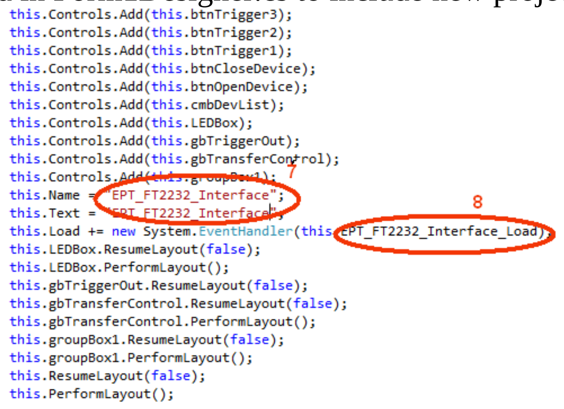
```
// Main object loader
private void EPT_FT2232_Interface_Load(object sender, System.EventArgs e)
  {

  // 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.



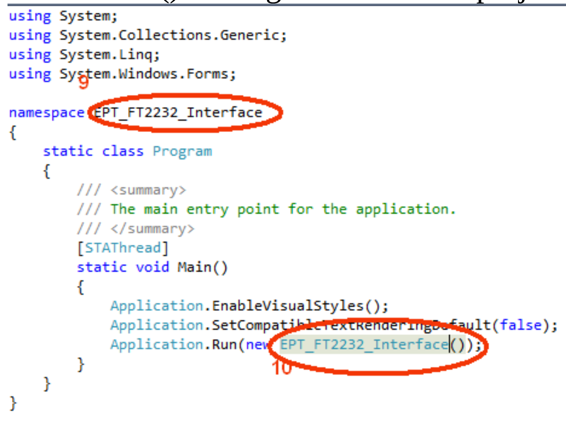
```
EPT_FT2232_Interface.EPT_FT2232_Interface                    InitializeCompone
namespace EPT_FT2232_Interface
{          5
    partial class EPT_FT2232_Interface
    {          6
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer 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();
```

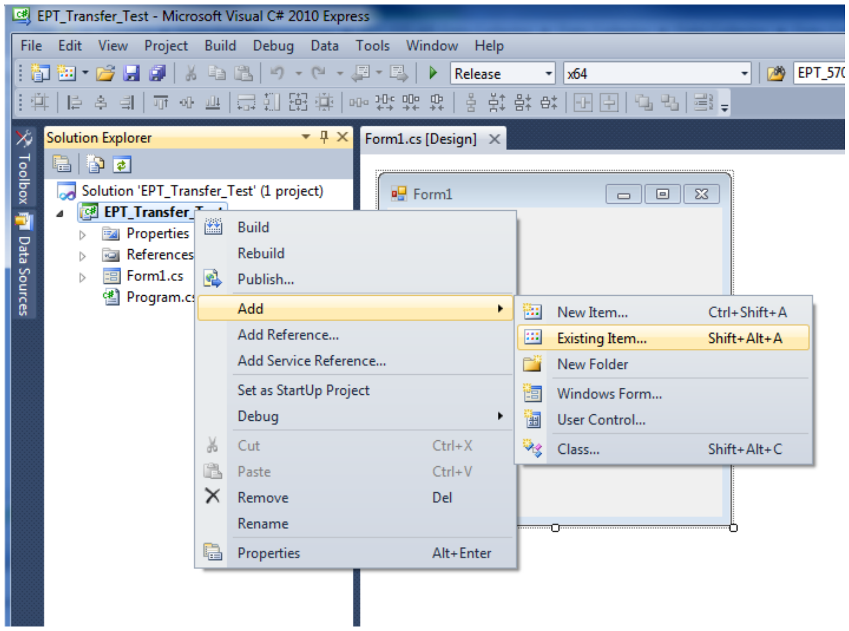
9. Change namespace in Program.cs to new project name
10. Change Application.Run() in Program .cs to new projectname.



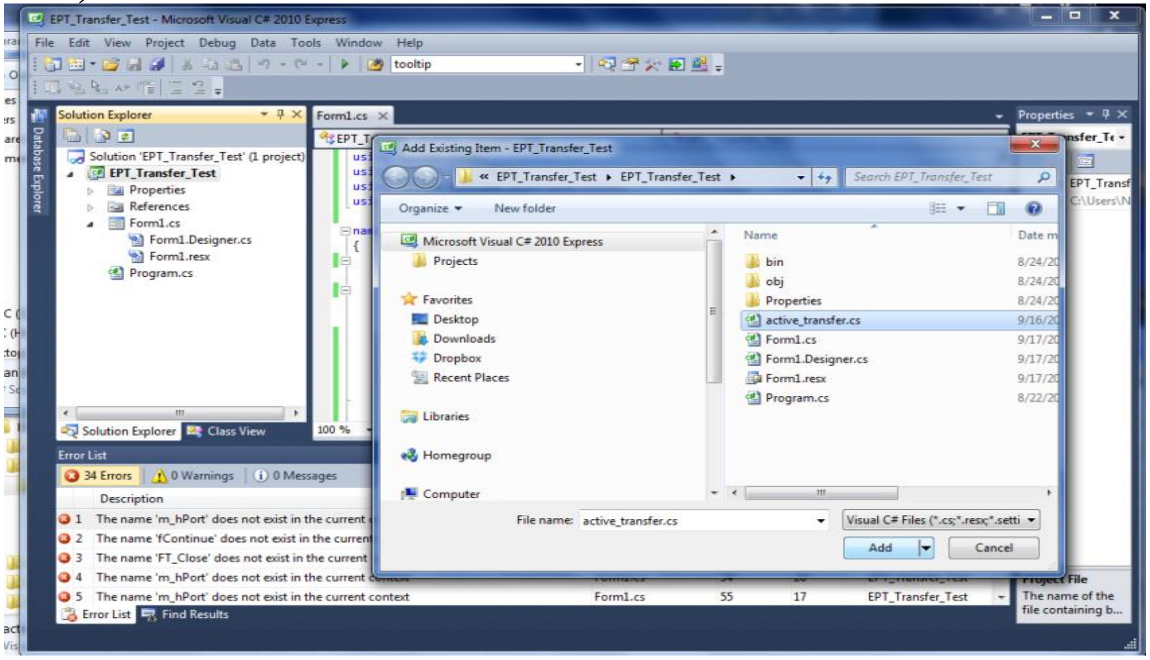
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

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());
        }
    }
}
```

6.2.2 Add Files to Project

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

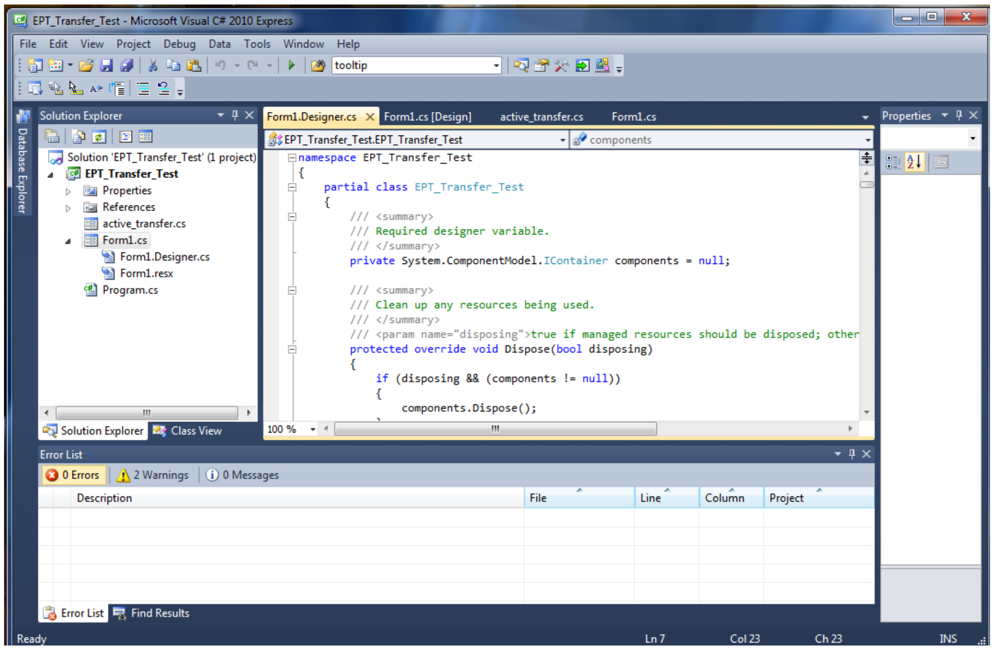


Browse to the EPT_Transfer_Test project folder and select the active_transfer_xx.cs file (choose either the 32 bit or 64 bit version, depending on whether your OS is 32 or 64 bit). Click Add.



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 context
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.



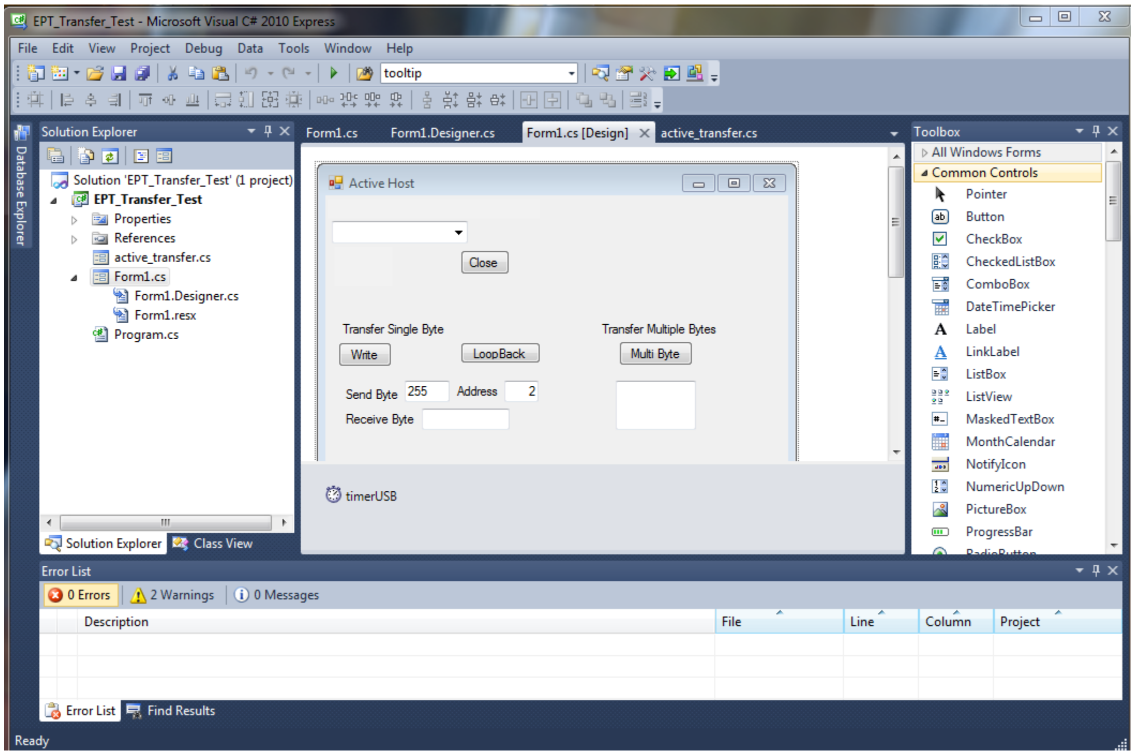
```
namespace EPT_Transfer_Test
{
    partial class EPT_Transfer_Test
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <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();
```

6.2.3 Adding Controls to the Project

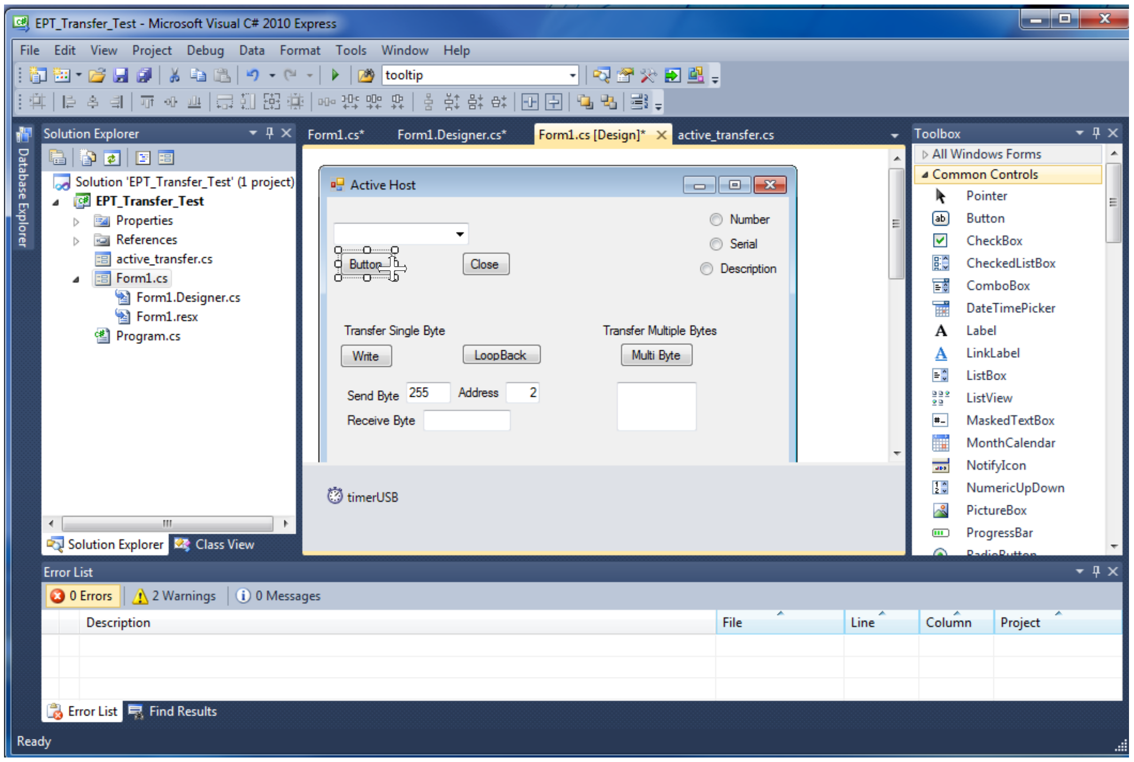
Although, the C# language is very similar to C Code, there are a few major differences. The first is C# .NET environment is event based. A second is C# utilizes classes. This guide will keep the details of these items hidden to keep things simple. However, a brief introduction to events and classes will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

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

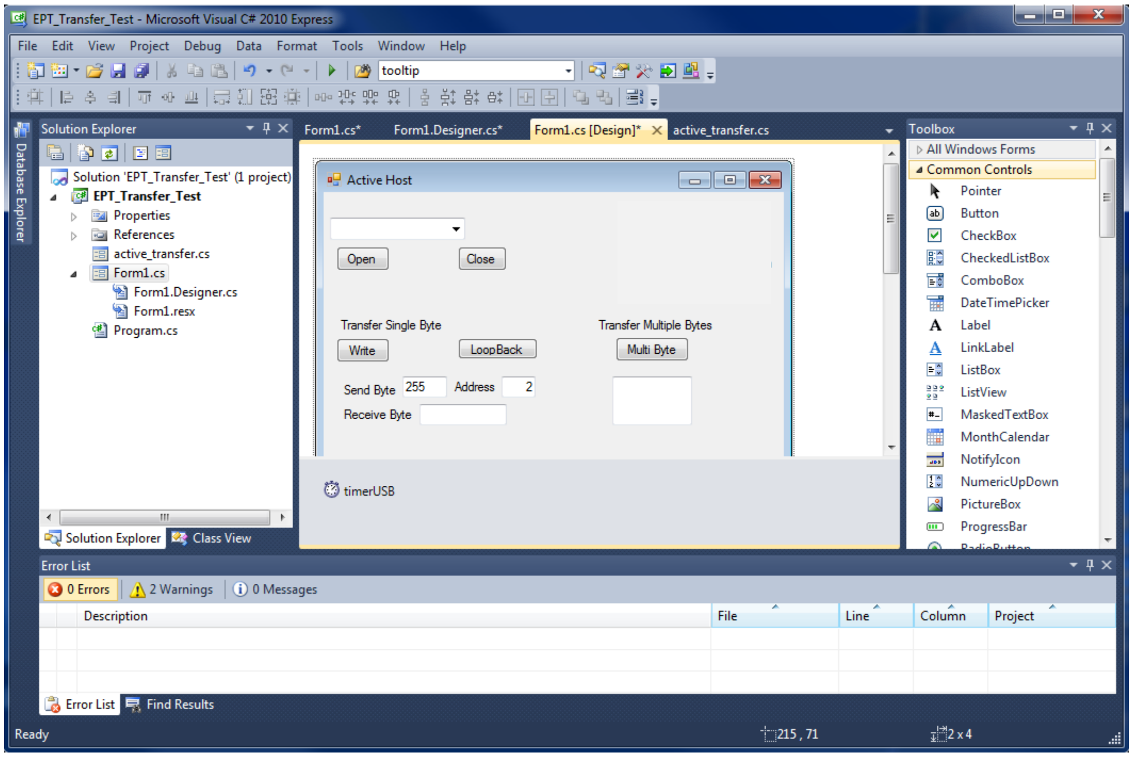
The easiest way to add a button to a form is to double click the Form1.cs in the Solution Explorer. Click on the button to launch the Toolbox.



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



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



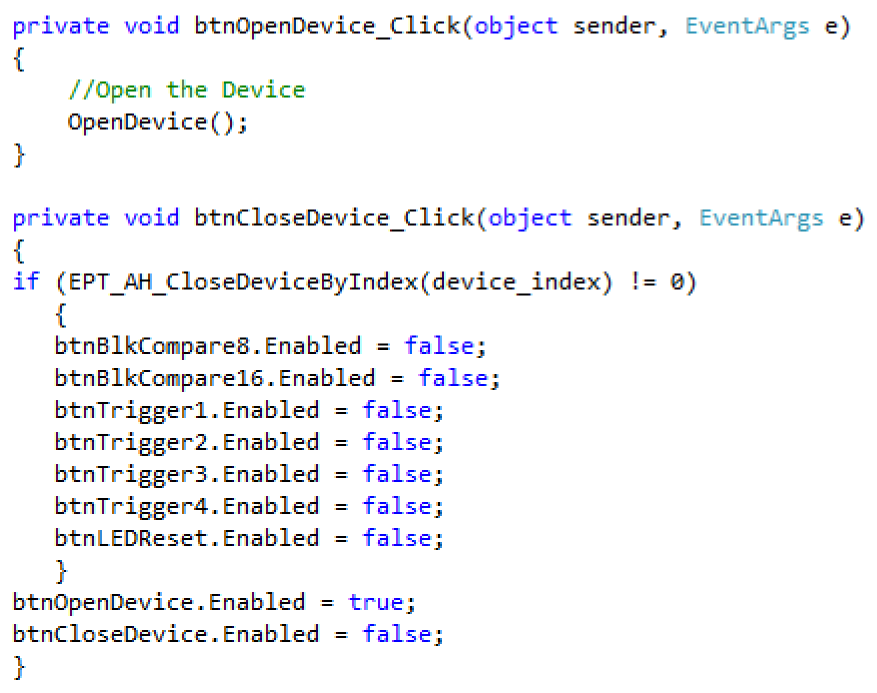
Double click on the Open button. The C# Explorer will automatically switch to the Form1.cs code view. The callback function will be inserted with the name of the button along with "_click" appended to it. The parameter list includes (object sender, System.EventArgs e). These two additions are required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

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

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

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

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



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

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

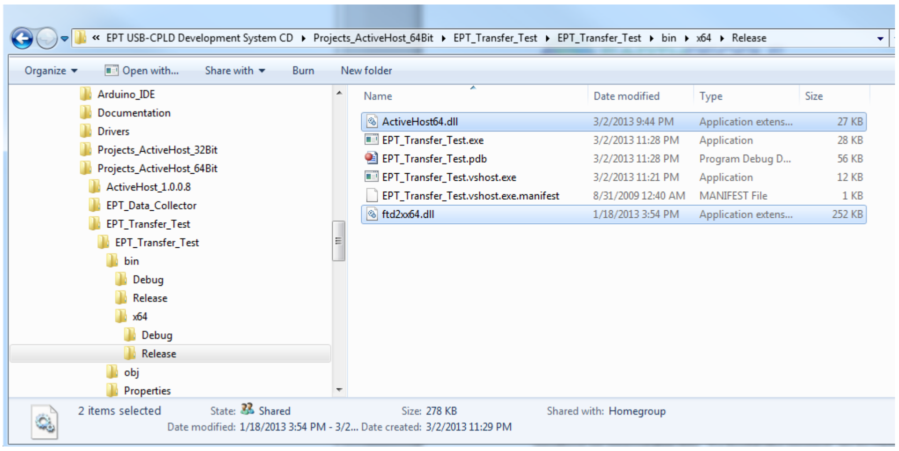
When btnOpenDevice_Click is called, it calls the function "OpenDevice()". This function is defined in the dll and will connect to the device selected in the combo box. This is a quick view of how to create, add files, and add controls to a C# project. The user is encouraged to spend some time reviewing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

6.2.4 Adding the DLL's to the Project

Locate the EPT USB-CPLD Development System CD installed on your PC. Browse to the Projects_ActiveHost 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.



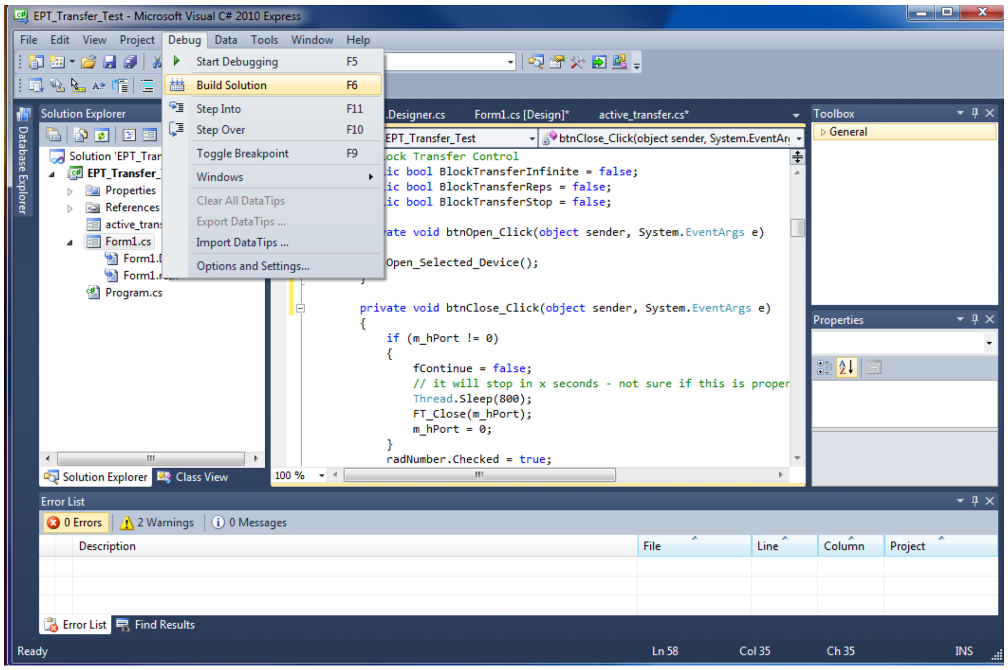
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	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.

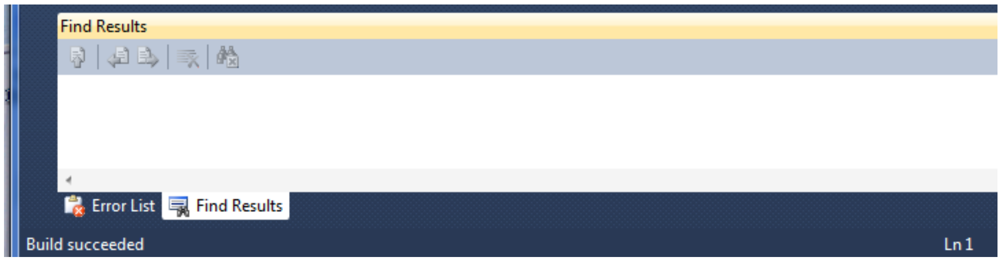
6.2.5 Building the Project

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;
```

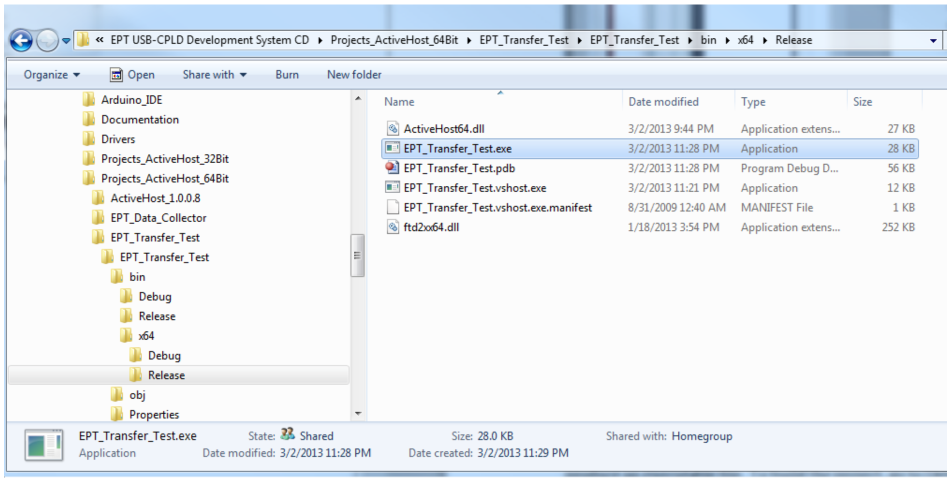
The C# Express compiler will start the building process. If there are no errors with code syntax, function usage, or linking, then the environment responds with “Build Succeeded”.



Build succeeded Ln 1

6.2.6 Testing the Project

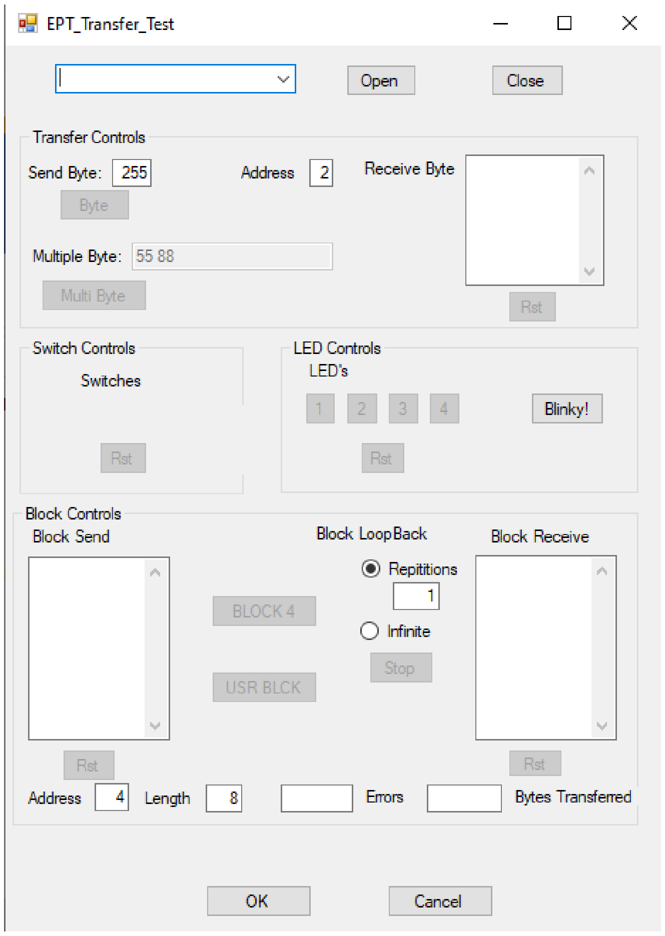
Once the project has been successfully built, it produces an *.exe file. The file will be saved in the Release or Debug folders.



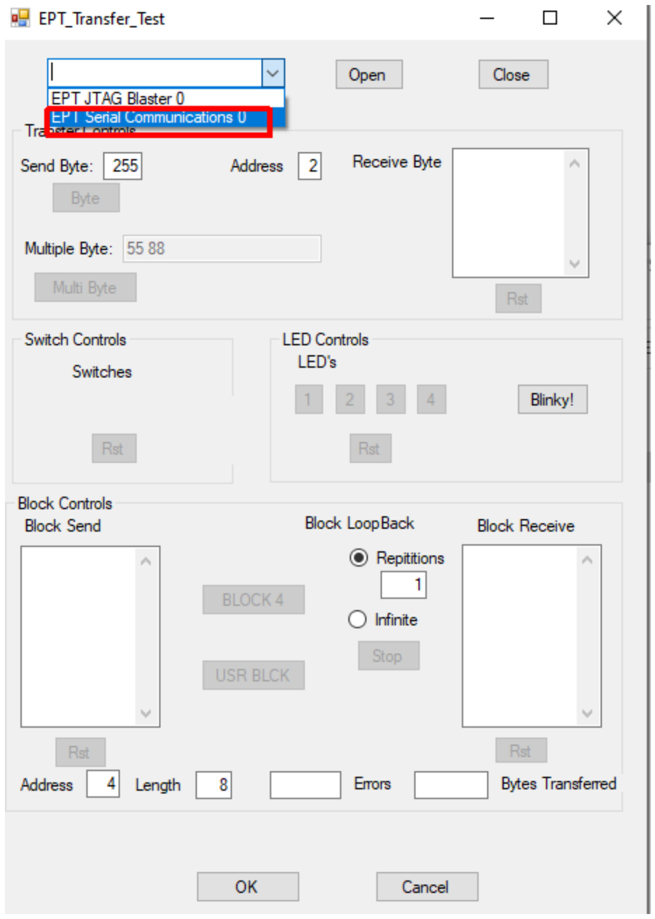
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
EPT_Transfer_Test.exe	3/2/2013 11:28 PM	Application	28 KB
EPT_Transfer_Test.pdb	3/2/2013 11:28 PM	Program Debug D...	56 KB
EPT_Transfer_Test.vshost.exe	3/2/2013 11:21 PM	Application	12 KB
EPT_Transfer_Test.vshost.exe.manifest	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

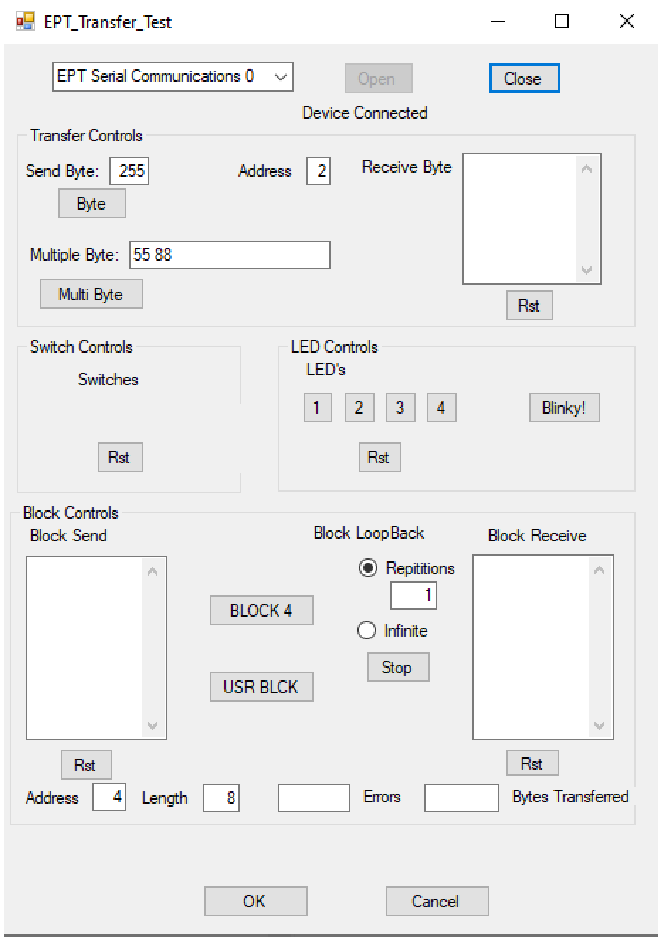
The EPT_Transfer_Text.exe file can now be tested using the MEGAPROLOGIC board. To test the file, connect the MEGAPROLOGIC to the Windows PC using Type A to Type Micro B USB cable. Make sure the driver for the board loads. If the USB driver fails to load, the Windows OS will indicate that no driver was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



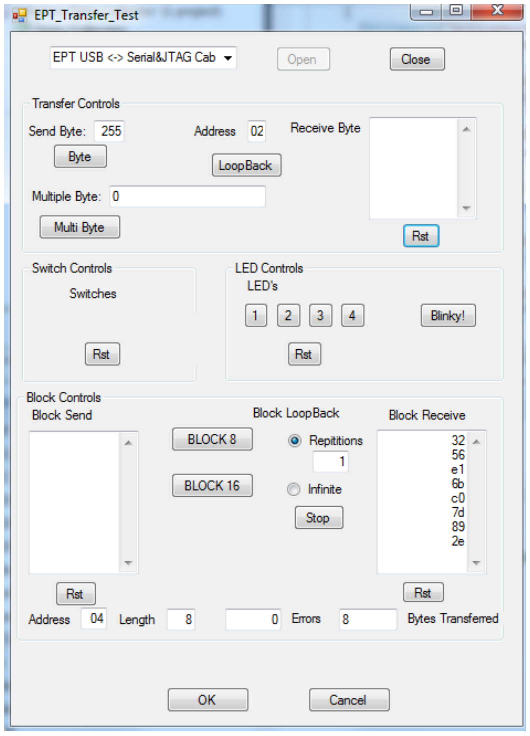
With the application loaded, select the USB-CPLD board from the dropdown combo box and click on the “Open” button.



Click on one of the LED buttons in the middle of the window. The corresponding LED on the MegaProLogic-U2 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.



EPT_Transfer_Test

EPT USB <-> Serial&JTAG Cab | Open | Close

Transfer Controls

Send Byte: 255 | Address 02 | Receive Byte

Byte | LoopBack

Multiple Byte: 0

Multi Byte | Rst

Switch Controls

Switches

Rst

LED Controls

LED's

1 | 2 | 3 | 4 | Blinky!

Rst

Block Controls

Block Send | Block LoopBack | Block Receive

BLOCK 8 | Repititions 1

BLOCK 16 | Infinite

Stop

32
56
e1
6b
c0
7d
89
2e

Rst | Rst

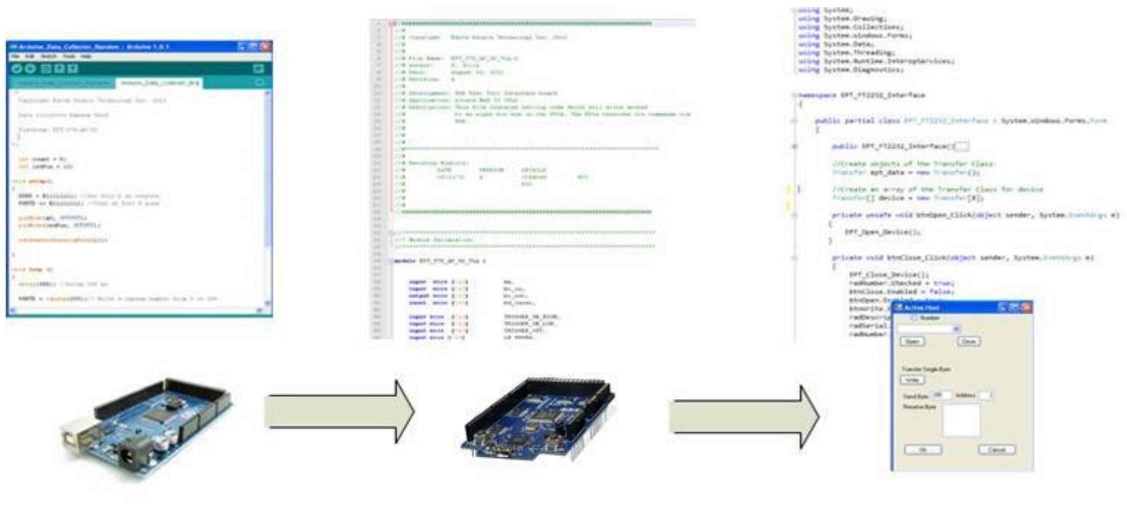
Address 04 | Length 8 | 0 | Errors | 8 | Bytes Transferred

OK | Cancel

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

6.3 Designing a Simple Data Collection Sampler

The Data Collection Sampler is a very simple introductory project that will guide the user in the creation of an overall design using the Arduino Programming Language, Verilog HDL, and C# Language. These elements will run on the Arduino Platform, MEGAPROLOGIC-M4 CPLD, and a Windows PC respectively.



The first order of business is to layout the design. Start with the Arduino, and create a simple bit output using a random number generator. Next, use the EPT Active Transfer Library to create a byte transfer module to read the byte from the Arduino and send it to the Host PC. Finally, use EPT Active Host to accept the byte transfer from EPT Active Transfer, and display in a textbox. This is just the hierarchical system level design. In the following sections, we will fill in the above blocks.

6.3.1 The Arduino Microcontroller Board

Using the features and capabilities of the Arduino development system, the user will develop the source code using the "Wiring" programming language and download the resulting binary code from the Processing development environment to the Flash memory of the microcontroller.

6.3.2 Create Data Generator

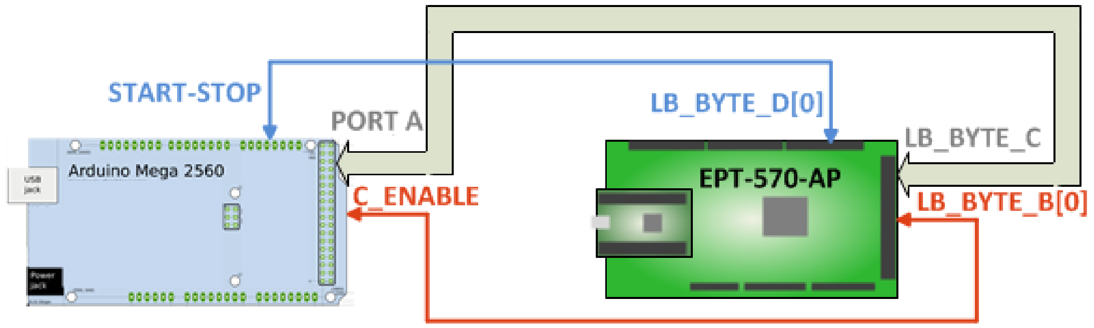
To keep the design simple, no external data source will be used. We will create a data source using the Arduino, then transmit this data to the MegaProLogic board. To create the data source, we will use the random() function. This function generates pseudo random numbers from a seed value. We will give the randomSeed() function a fairly random input using the value from the analogRead(). This will give different values every time the random() function is called. We will limit the random number output from the function to 8 bits. The random() function will be called once per iteration of the loop() function.

The randomSeed() function must be called during the setup() function. It takes as input parameter the output of the Analog Pin 1. The output of this Pin 1 will have a small amount of random noise on it. Because of this noise, the randomSeed() function will produce a different seed every time the sketch is initialized.

```
void setup()
{
  randomSeed(analogRead(1));
}
```

6.3.3 Select I/O's for Fast Throughput on Arduino

An 8 bit port is used to connect the 8 bit byte from the random function output to the input of the MegaProLogic. There is also a one bit control line which will be used to inform the CPLD that a byte is ready to be written to the USB.



Each port is controlled by three registers, which are also defined variables in the Arduino language. The DDR register, determines whether the pin is an INPUT or OUTPUT. The PORT register controls whether the pin is HIGH or LOW, and the PIN register reads the state of INPUT pins set to input with pinMode(). The maps of the ATmega328 chips show the ports.

DDR and PORT registers may be both written to, and read. PIN registers correspond to the state of inputs and may only be read.

PORTA maps to Arduino digital pins 0 to 7

DDRA - The Port A Data Direction Register - read/write
PORTA - The Port A Data Register - read/write
PINA - The Port A Input Pins Register - read only

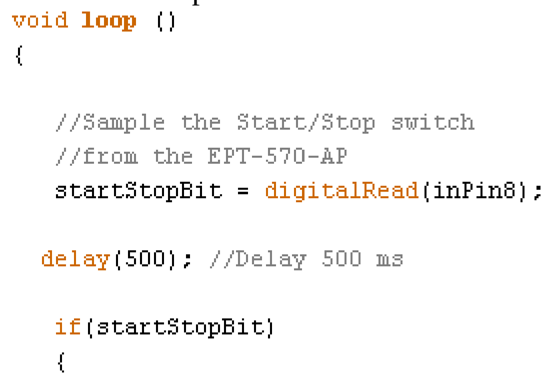
The ports and pins for the Data Collection Sampler project must be initialized in the setup() function. The setup function will only run once, after each powerup or reset of the Arduino board.

```
void setup()
{
  DDRA = B11111111; //Set Port A as outputs
  PORTA &= B11111111; //Turn on Port A pins
```

After the setup() function executes, the PORTA is ready to be assigned the results of our random() function. And the A0 pin will be used to latch the value on PORTA pins into the CPLD.

6.3.4 Coding the Arduino Data Sampler

Now that we have the data generator and the ports defined, we can add some delays in the loop() function and make a simulated data collector. Because Start and Stop buttons will be added to the C# Windows Form, the Data Collector code will need to monitor a single pin output from the MEGAPROLOGIC. This output pin (from the MEGAPROLOGIC) becomes an input to the Arduino and is used in conditional switch.



```
void loop ()
{

   //Sample the Start/Stop switch
   //from the EPT-570-AP
   startStopBit = digitalRead(inPin8);

  delay(500); //Delay 500 ms

   if(startStopBit)
   {
```

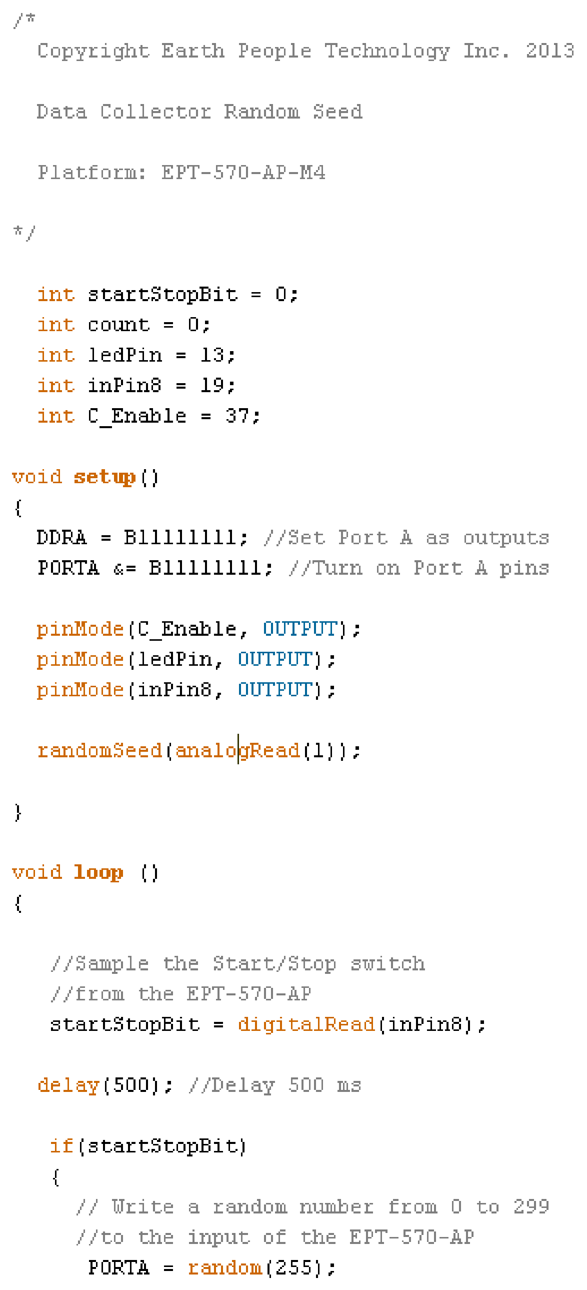
This code will sample the Start/Stop switch which is an output from the MEGAPROLOGIC on J5 PIN 3. On the Arduino, this is PIN 19 of the Digital pins. Each iteration of the loop() function, the startStopBit variable stores the state of DigitalPin19. Then, a delay of 500 milliseconds is added. The delay() function pauses the program for the amount of time (in milliseconds) specified as parameter. Next, the startStopBit is checked with a conditional switch. If the bit is set, the conditional branch is entered and the random number is sent to the MEGAPROLOGIC. If the bit is not set, the end of the loop() function is reached and it branches to the top of the loop().

We will also add an LED Pin that will blink so that we can have a visual indication that the project is working.

We want to add a delay so that the data from the generated displays on the Windows PC long enough for our eyes to verify that the data is updating correctly. This delay should be one second in total. So, the data will change then stay stable in the textbox for one second before changing again.

For the LED to blink correctly, it should turn on, delay for half a second then turn off and delay for half a second. If we don't use half second intervals for the LED blink, the LED will appear to not change at all. It will look like it stays on all the time or off all the time.

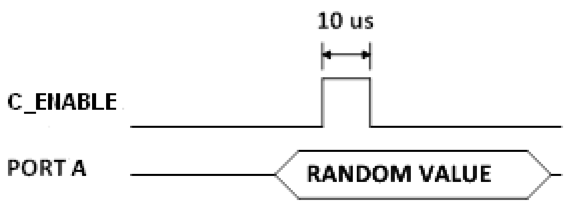
So, the code looks like this:



```
/*
  Copyright Earth People Technology Inc. 2013

  Data Collector Random Seed

  Platform: EPT-570-AP-M4

*/

  int startStopBit = 0;
  int count = 0;
  int ledPin = 13;
  int inPin8 = 19;
  int C_Enable = 37;

void setup()
{
  DDRA = B11111111; //Set Port A as outputs
  PORTA &= B11111111; //Turn on Port A pins

  pinMode(C_Enable, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(inPin8, OUTPUT);

  randomSeed(analogRead(1));

}

void loop ()
{

   //Sample the Start/Stop switch
   //from the EPT-570-AP
   startStopBit = digitalRead(inPin8);

  delay(500); //Delay 500 ms

   if(startStopBit)
   {
     // Write a random number from 0 to 299
     //to the input of the EPT-570-AP
      PORTA = random(255);
```

```
            //Set the Write Enable Pin High
             digitalWrite(C_Enable, HIGH);

            //Set the LED Pin High
            digitalWrite(ledPin, HIGH);

            delay(500); //Delay 500 ms

            //Set the LED Pin Low
            digitalWrite(ledPin, LOW);
            //Set the Write Enable Pin Low
            digitalWrite(C_Enable, LOW);
         }

      }
```

Notice that PORTA equals the return of random(255). The parameter passed to the random() function is the maximum decimal value of the return value. In our case we want the maximum value to be an 8 bit value, B11111111 = 0xff = 255(decimal). Also, note that the C_Enable write enable signal for the CPLD has back to back instructions turning it on then off immediately. Because the ATMega328 chip takes approximately 160 clock cycles to execute the digitalWrite() function and affect the Pin at C_Enable, this produces a write enable pulse of 10 microseconds.



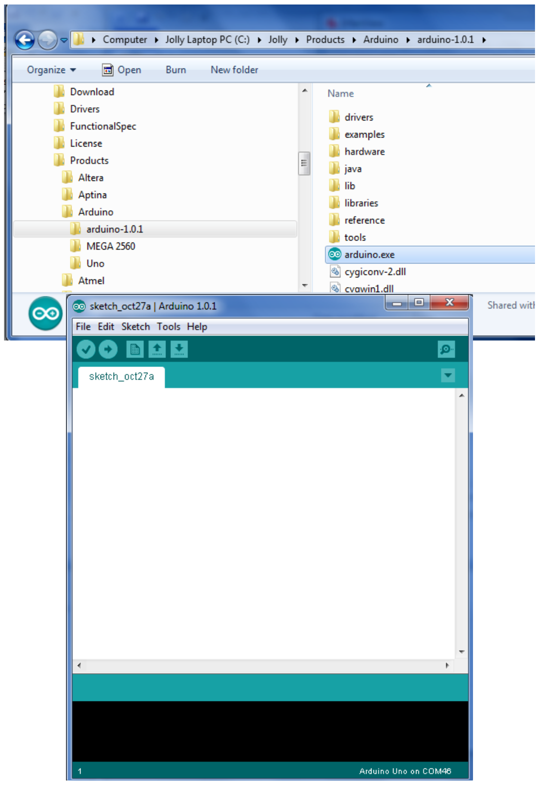
The RANDOM VALUE will be stable before the C_Enable asserts thus guaranteeing a successful transfer of data from Arduino to CPLD.

6.3.5 Building Arduino Project

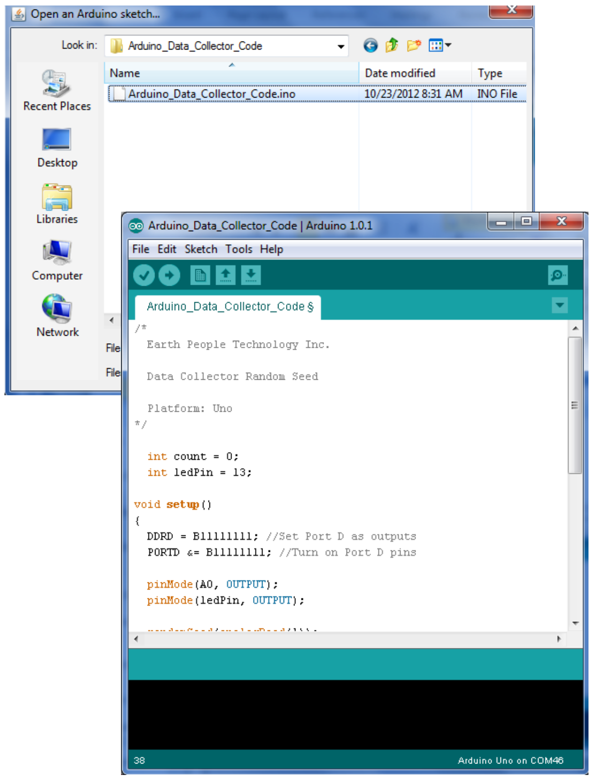
Building the Arduino project is the process of converting (compiling) the code you just wrote into machine level code that the processor can understand. The Arduino IDE is the software tool that does the compiling. The machine level code is a set of basic instructions that the processor uses to perform the functions the user code. Browse to the \Projects_Arduino\Arduino_Data_Collector_Code\ folder of the EPT USB-CPLD Development System CD. Copy Arduino_Data_Collector_Code_M4.ino .

To compile your code,

- Open up the Arduino IDE



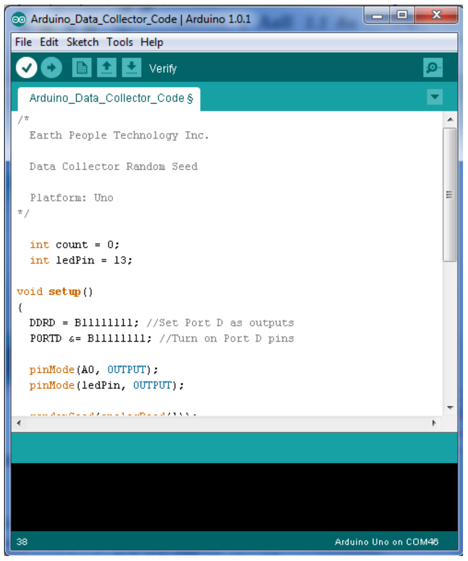
- Paste your code into the Sketch.


Open an Arduino sketch...
Look in: Arduino_Data_Collector_Code
Recent Places
Desktop
Libraries
Computer
Network
Arduino_Data_Collector_Code | Arduino 1.0.1
File Edit Sketch Tools Help
Arduino_Data_Collector_Code §
38
Arduino Uno on COM46

Name	Date modified	Type
Arduino_Data_Collector_Code.ino	10/23/2012 8:31 AM	INO File

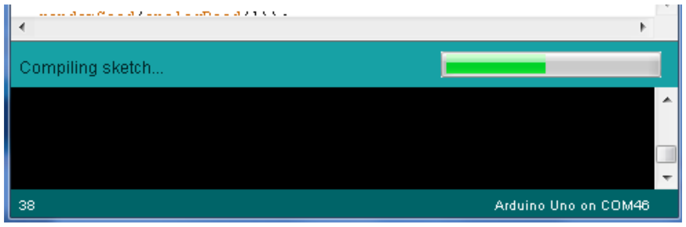
```
/*
  Earth People Technology Inc.

  Data Collector Random Seed

  Platform: Uno
*/

  int count = 0;
  int ledPin = 13;

void setup()
{
  DDRD = B11111111; //Set Port D as outputs
  PORTD &= B11111111; //Turn on Port D pins

  pinMode(A0, OUTPUT);
  pinMode(ledPin, OUTPUT);
```

- Click the Verify button

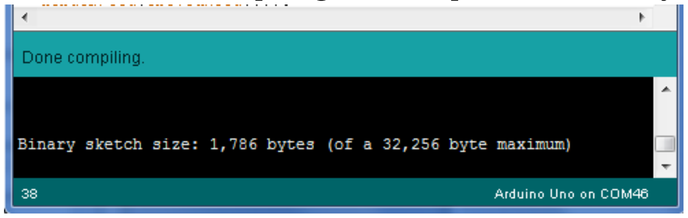

Arduino_Data_Collector_Code | Arduino 1.0.1
File Edit Sketch Tools Help
Verify
Arduino_Data_Collector_Code §
38
Arduino Uno on COM46

```
/*
  Earth People Technology Inc.

  Data Collector Random Seed

  Platform: Uno
*/

  int count = 0;
  int ledPin = 13;

void setup()
{
  DDRD = B11111111; //Set Port D as outputs
  PORTD &= B11111111; //Turn on Port D pins

  pinMode(A0, OUTPUT);
  pinMode(ledPin, OUTPUT);
```

- The sketch will compile



- If there are no errors, the compiling will complete successfully



```
Binary sketch size: 1,786 bytes (of a 32,256 byte maximum)
```

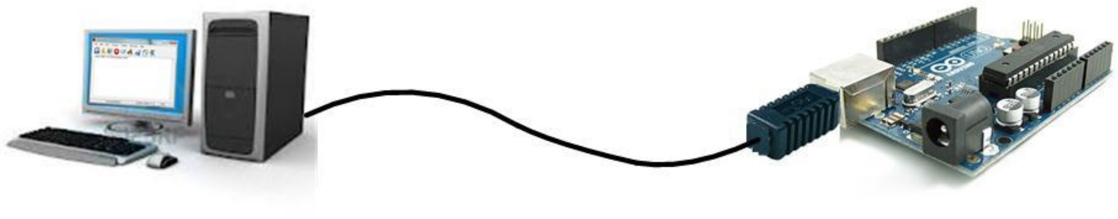
Now we are done with compiling and ready to program the Arduino

6.3.6 Programming the Arduino

Programming the Arduino is the process of downloading the user's compiled code into the Flash memory of the Atmel ATMega328 chip. Once the code is downloaded, the Arduino IDE resets the chip and the processor starts executing out of Flash memory.

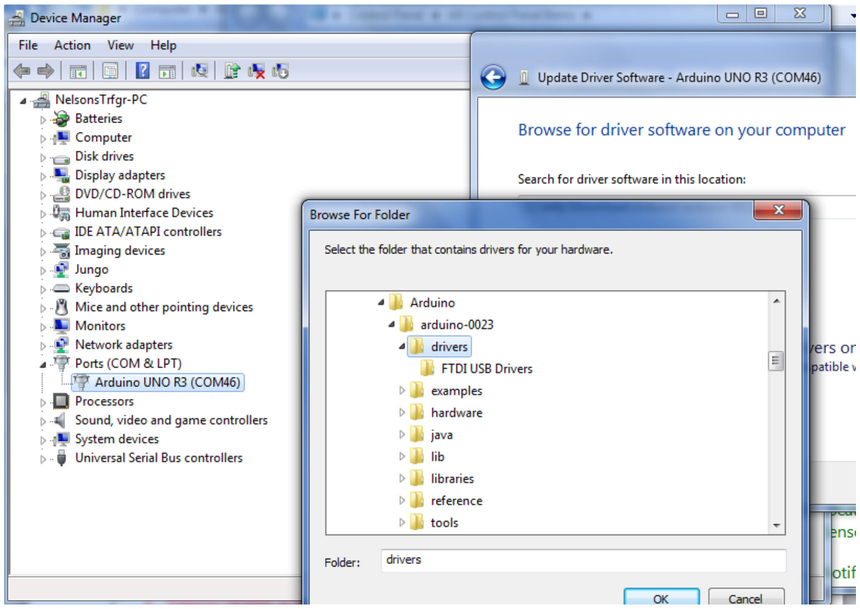
To program the Arduino

- Connect the USB cable from PC to Arduino

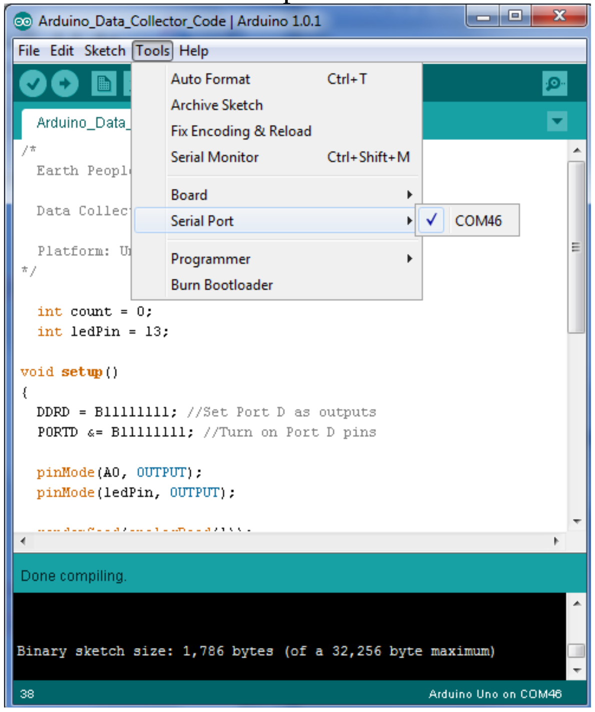


- Load the Arduino USB driver according to the manual
- Plug in your board and wait for Windows to begin it's driver installation process. After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino MEGA (COMxx)"
- Right click on the "Arduino MEGA (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.

- Finally, navigate to and select the MEGA's driver file, named **"ArduinoMEGA.inf"**, located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory).
- Windows will finish up the driver installation from there.



- Once the driver is loaded, we can set the COM Port. Click on Tools and select Serial Port, then click the available port.

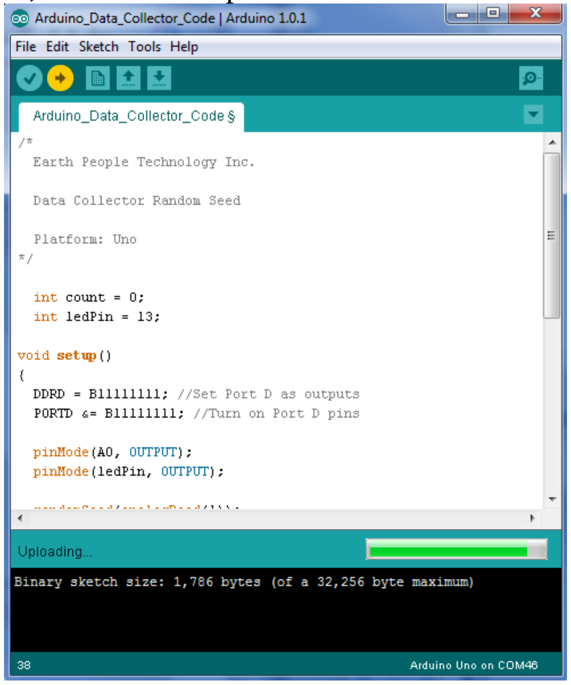


```
/*
  Earth Peopl
  Data Collec
  Platform: U
*/

  int count = 0;
  int ledPin = 13;

void setup()
{
  DDRD = B11111111; //Set Port D as outputs
  PORTD &= B11111111; //Turn on Port D pins

  pinMode(A0, OUTPUT);
  pinMode(ledPin, OUTPUT);
```

Done compiling.

```
Binary sketch size: 1,786 bytes (of a 32,256 byte maximum)
```

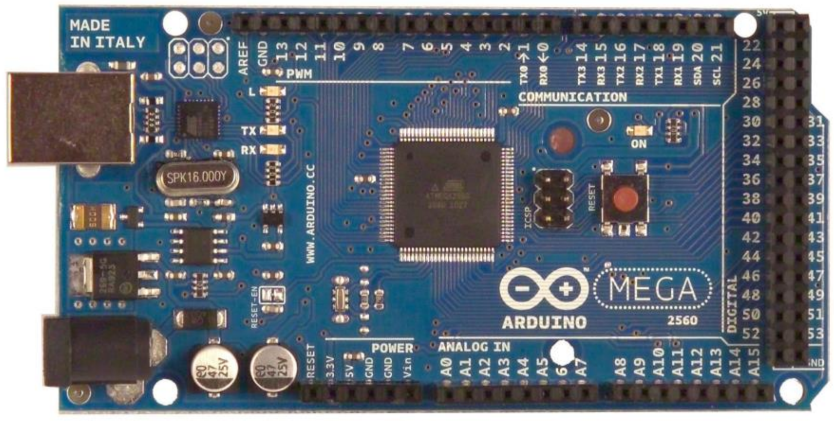
- To load the code, click on the Upload button.


Arduino_Data_Collector_Code | Arduino 1.0.1
File Edit Sketch Tools Help
Arduino_Data_Collector_Code §
Uploading...
38
Arduino Uno on COM46

```
/*
  Earth People Technology Inc.

  Data Collector Random Seed

  Platform: Uno
*/

  int count = 0;
  int ledPin = 13;

void setup()
{
  DDRD = B11111111; //Set Port D as outputs
  PORTD &= B11111111; //Turn on Port D pins

  pinMode(A0, OUTPUT);
  pinMode(ledPin, OUTPUT);
```

```
Binary sketch size: 1,786 bytes (of a 32,256 byte maximum)
```

When the code has completed loading, the Arduino IDE will automatically command the processor to start executing the code. The L LED will blink at one second intervals.

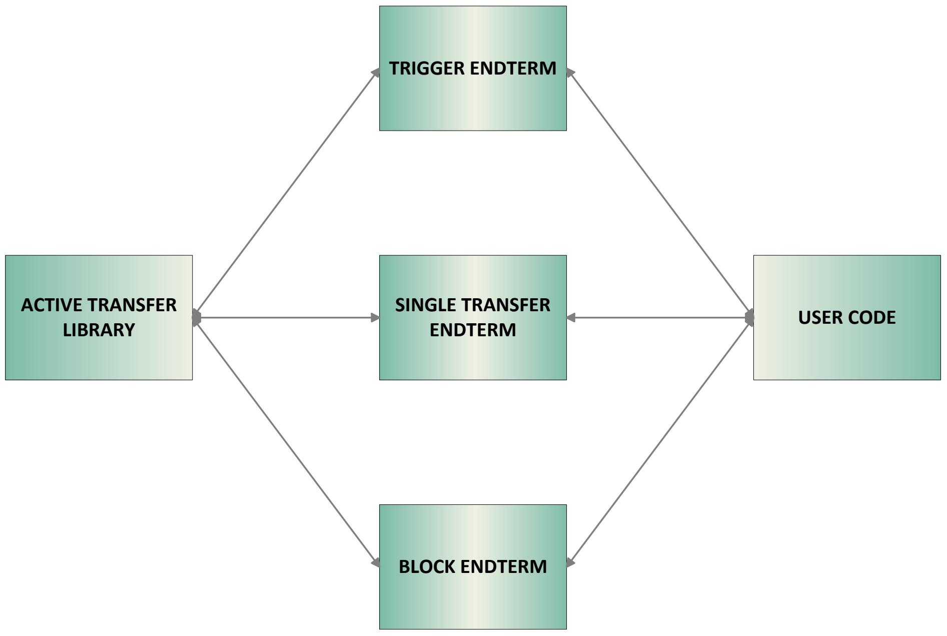


6.3.7 CPLD Active Transfer Coding and Initiation

The MEGAPROLOGIC will accept the data collected by the Arduino and transfer it to the PC. It is designed to plug directly into the Arduino MEGA and there is no need for external wires to be added. The Active Transfer Library can be used to send the data to the PC. This library has been designed to make it easy to transfer data to and from the PC via the USB. The user must create some interface code between the incoming data and the Active Transfer Library. We will now go through exercise of creating the CPLD code for the Data Collector Sampler.

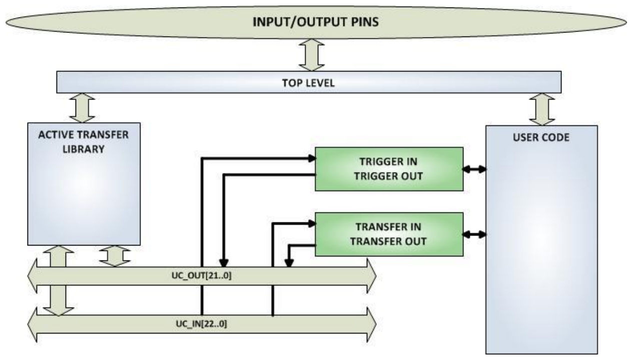
6.3.8 CPLD: Define the User Design.

In this step we will define the user's code and include modules from the EPT Active Transfer Library. The Active Transfer Library contains a set of files with a ".vqm" name extension which select particular operations to perform (e.g., byte transfer, block transfer, trigger). The active_transfer_library.vqm file must be included in the top level file of the project. The function modules will connect to the active_transfer_library and provide a path to connect user code to the library. All of these files are available on the Earth People Technology website.



We will build our CPLD project using Quartus Prime software from Altera. The primary file defining the user's CPLD project is named "EPT_570_AP_M4_Top.v". It defines the user code and connects the active_transfer_library and active_transfer logic functions. In order to route the pins of the Arduino to the CPLD, the Pin Planner tool is used. This tool allows the user to match internal net names to the pins of the CPLD.

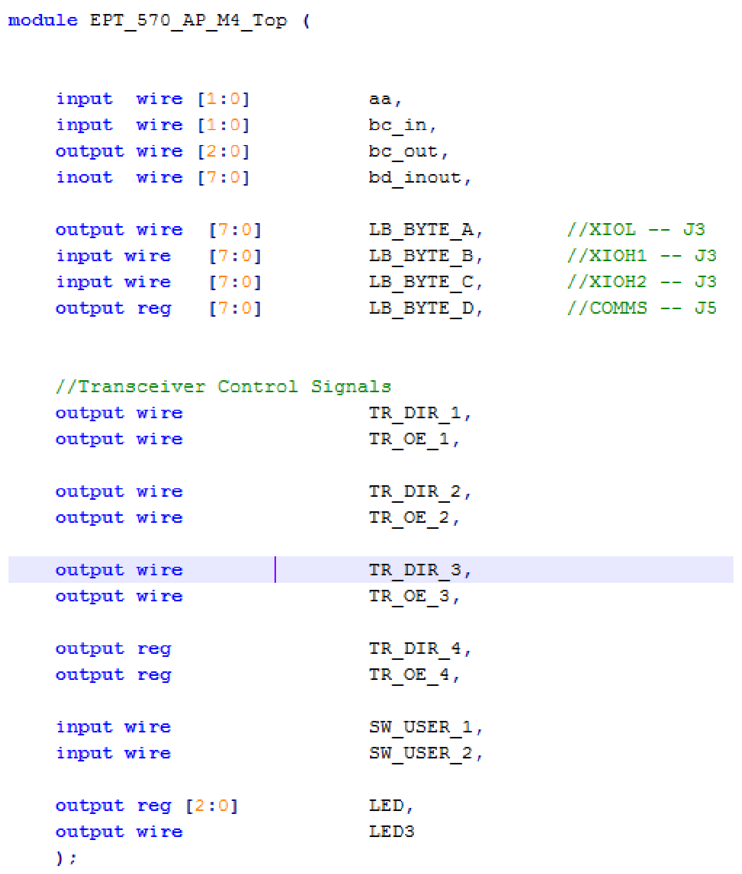
Our project needs to accept an 8 bit value on the J3 connector and a write enable on Pin 3 of J5. For this, we can use the active_transfer.vqm module as the interface to the active_transfer_library. It accepts a single byte and latches it with a single enable net. Because the active_transfer_library runs at 66 MHz we will need to write some code ensure that the slower C_Enable signal from the Arduino can latch the data into the active_transfer module.



CPLD: Coding up the DesignThe first thing to do is to create a top level file for the project. The top level file will include the input and outputs for the CPLD. These are declared according to the Verilog syntax rules. We won't go through all the rules of Verilog here, but feel free to explore the language more thoroughly at

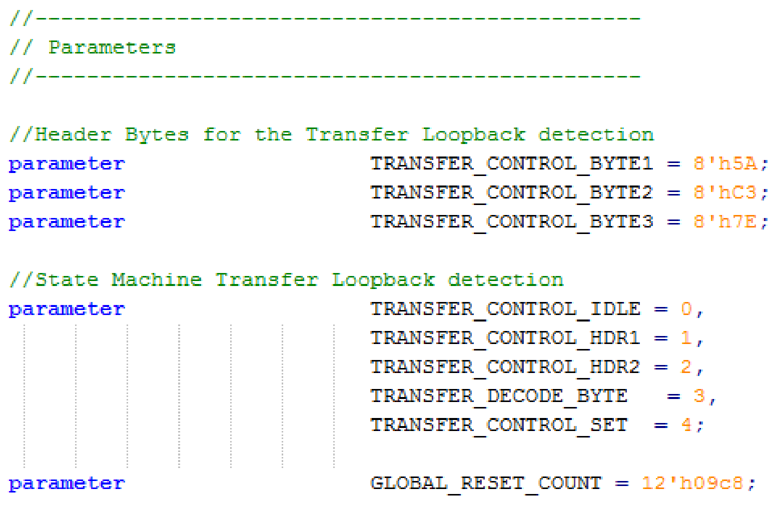
www.asic-world.com/verilog/

We need to add the inputs and outputs for active_transfer_library, user code, leds, and switches. Each port is described as input, output or inout. It is followed by the net type wire or reg. If it is a vector, the array description must be added.



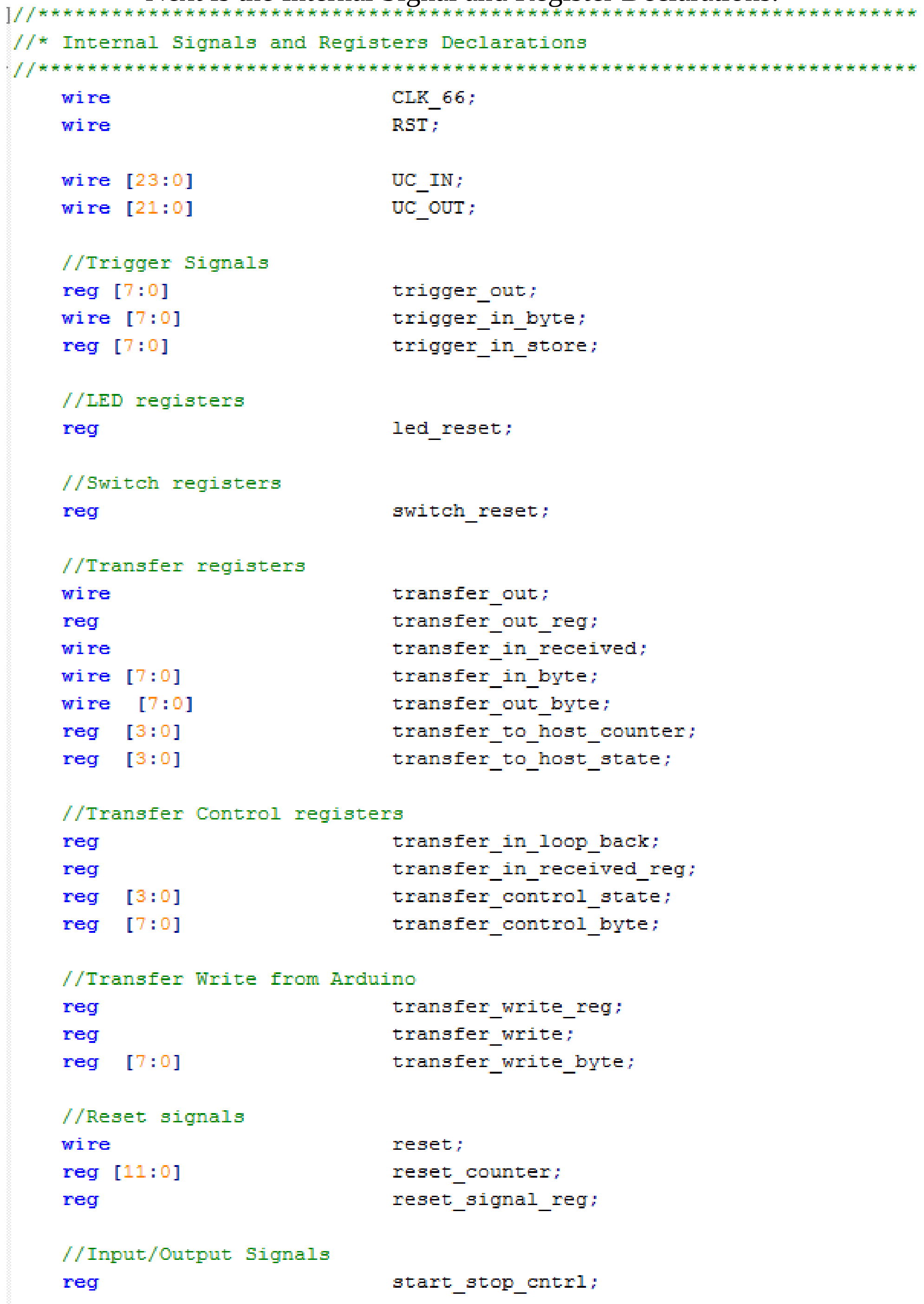
```
module EPT_570_AP_M4_Top (

    input  wire [1:0]          aa,
    input  wire [1:0]          bc_in,
    output wire [2:0]          bc_out,
    inout  wire [7:0]          bd_inout,

    output wire  [7:0]         LB_BYTE_A,        //XIOL -- J3
    input wire   [7:0]         LB_BYTE_B,        //XIOH1 -- J3
    input wire   [7:0]         LB_BYTE_C,        //XIOH2 -- J3
    output reg   [7:0]         LB_BYTE_D,        //COMMS -- J5


    //Transceiver Control Signals
    output wire                TR_DIR_1,
    output wire                TR_OE_1,

    output wire                TR_DIR_2,
    output wire                TR_OE_2,

    output wire                TR_DIR_3,
    output wire                TR_OE_3,

    output reg                 TR_DIR_4,
    output reg                 TR_OE_4,

    input wire                 SW_USER_1,
    input wire                 SW_USER_2,

    output reg [2:0]           LED,
    output wire                LED3
    );
```

Next, the parameter's are defined. These are used as constants in the user code.



```
//-----------------------------------------------
// Parameters
//-----------------------------------------------

//Header Bytes for the Transfer Loopback detection
parameter                      TRANSFER_CONTROL_BYTE1 = 8'h5A;
parameter                      TRANSFER_CONTROL_BYTE2 = 8'hC3;
parameter                      TRANSFER_CONTROL_BYTE3 = 8'h7E;

//State Machine Transfer Loopback detection
parameter                      TRANSFER_CONTROL_IDLE = 0,
                               TRANSFER_CONTROL_HDR1 = 1,
                               TRANSFER_CONTROL_HDR2 = 2,
                               TRANSFER_DECODE_BYTE   = 3,
                               TRANSFER_CONTROL_SET  = 4;

parameter                      GLOBAL_RESET_COUNT = 12'h09c8;
```

Next is the Internal Signal and Register Declarations.



```
//***********************************************************************
//* Internal Signals and Registers Declarations
//***********************************************************************
    wire                        CLK_66;
    wire                        RST;

    wire [23:0]                 UC_IN;
    wire [21:0]                 UC_OUT;

    //Trigger Signals
    reg [7:0]                   trigger_out;
    wire [7:0]                  trigger_in_byte;
    reg [7:0]                   trigger_in_store;

    //LED registers
    reg                         led_reset;

    //Switch registers
    reg                         switch_reset;

    //Transfer registers
    wire                        transfer_out;
    reg                         transfer_out_reg;
    wire                        transfer_in_received;
    wire [7:0]                  transfer_in_byte;
    wire  [7:0]                 transfer_out_byte;
    reg  [3:0]                  transfer_to_host_counter;
    reg  [3:0]                  transfer_to_host_state;

    //Transfer Control registers
    reg                         transfer_in_loop_back;
    reg                         transfer_in_received_reg;
    reg  [3:0]                  transfer_control_state;
    reg  [7:0]                  transfer_control_byte;

    //Transfer Write from Arduino
    reg                         transfer_write_reg;
    reg                         transfer_write;
    reg  [7:0]                  transfer_write_byte;

    //Reset signals
    wire                        reset;
    reg [11:0]                  reset_counter;
    reg                         reset_signal_reg;

    //Input/Output Signals
    reg                         start_stop_cntrl;
```

Next, add the assignments. These assignments will set the direction of the bus transceivers that interface to the Arduino I/O's. The transceivers also include an output enable bit.

```
//****************************************************************************
//*      Signal Assignments
//****************************************************************************
   assign             LB_BYTE_A[7:0] = 8'h0;

   assign             TR_DIR_1  = 1'b1; //1 = A to B; 0 = B to A
   assign             TR_OE_1  = 1'b1;

   assign             TR_DIR_2  = 1'b1; //1 = A to B; 0 = B to A
   assign             TR_OE_2  = 1'b0;

   assign             TR_DIR_3  = 1'b1; //1 = A to B; 0 = B to A
   assign             TR_OE_3  = 1'b0;

   //Clock and Reset
   assign             CLK_66 = aa[1];
   assign             RST = reset;
   assign             reset = reset_signal_reg;

   //Transfer registers
   assign             transfer_out = transfer_out_reg | transfer_write;
   assign             transfer_out_byte = transfer_write_byte;

   //LED3 is used to signify to the  user that the Start
   //switch is enabled
   assign              LED3 =  start_stop_cntrl ? 1'b0 : 1'bz;
```

The reset signal is generated by a counter that starts counting upon power up. When the counter reaches GLOBAL_RESET_COUNT.

```
//*****************************************************
//*      Reset Signal
//*****************************************************
    always @(posedge CLK_IN or negedge aa[0])
    begin
      if(!aa[0])
      begin
            reset_signal_reg <= 1'b0;
            reset_counter <= 0;
      end
      else
      begin
        if( reset_counter < GLOBAL_RESET_COUNT )
        begin
            reset_signal_reg <= 1'b0;
            reset_counter <= reset_counter + 1'b1;
        end
        else
        begin
            reset_signal_reg <= 1'b1;
        end
      end
```

The four LED's are set by the bottom four bits of the active_trigger output register. These trigger outputs can be set by using a function in the Active_Host DLL on the PC. The Data Collector project will use LED3 to indicate the state of the Start/Stop signal.

```
//LED3 is used to signify to the  user that the Start
//switch is enabled
assign                LED3 =  ~start_stop_cntrl;

   //---------------------------------------------
   // LED Set
   //---------------------------------------------

  always @(trigger_in_byte or led_reset or LED or RST)
  begin
    if(!RST)
        LED[2:0] = 3'hz;
    else if(led_reset)
        LED[2:0] = 3'hz;
    else if(trigger_in_byte[3:0])
    begin
        case(trigger_in_byte[3:0])
            3'h1:
                LED[0] = 1'b0;
            3'h2:
                LED[1] = 1'b0;
            3'h4:
                LED[2] = 1'b0;
            default:
                LED[2:0] = LED[2:0];
        endcase
    end

  end
```

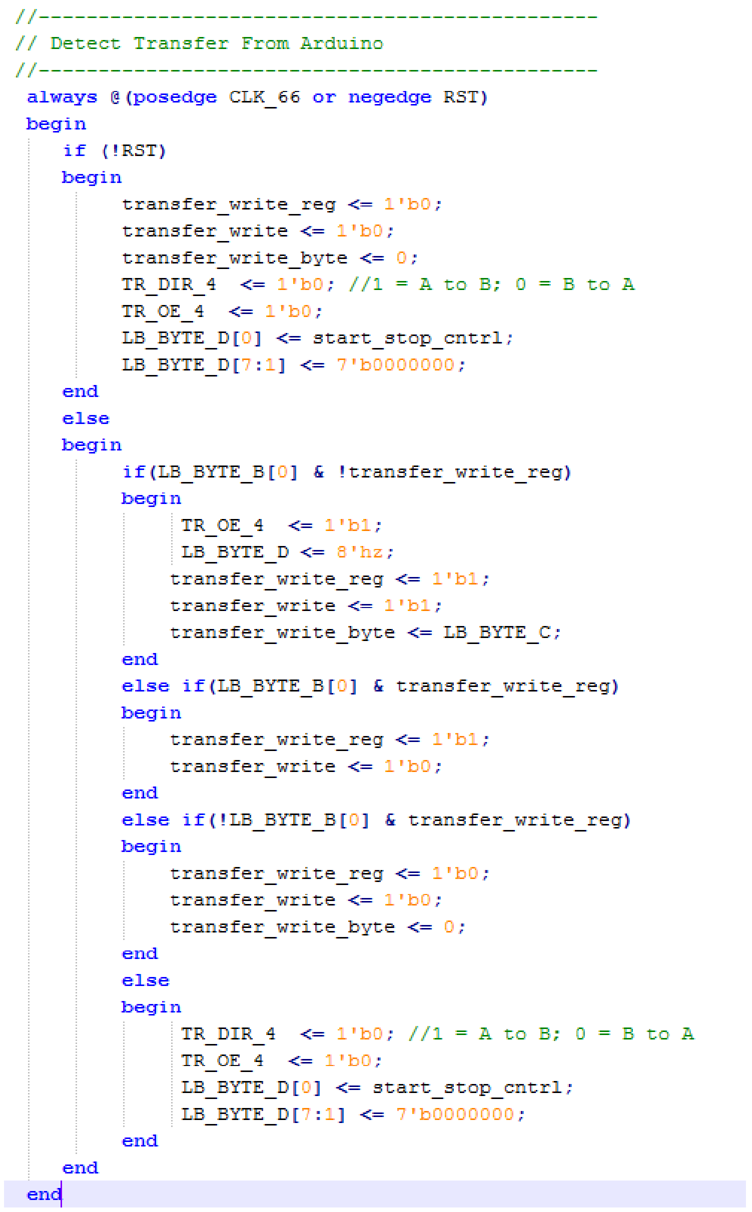
The two user switches are connected to the input trigger register. Pressing a switch will send a trigger to the PC to be decoded by the Active_Host DLL.

```
//---------------------------------------------
// User Switch Trigger
//---------------------------------------------
always @(posedge CLK_IN or negedge RST)
begin
  if(!RST)
  begin
      trigger_out <= 8'h00;
  end
  else
  begin
     if(!SW_USER_1 )
          trigger_out <= 8'h01;
     else if(!SW_USER_2 )
          trigger_out <= 8'h02;
       else if(switch_reset)
          trigger_out <= 8'h04;
       else
          trigger_out <= 8'h00;
  end
end
```

Next, we will add the transfer detection signal from the Arduino. This block will require three registers.

- transfer_write_reg –This is a latch register to hold the state of the C_Enable.
- transfer_write –This register is used to start the active_transfer single byte write to the PC.
- transfer_write_byte –This is an 8 bit register to hold the value of the Data Collection output.

This block will compare the input signal on LB_BYTE_B[0] to a high. The LB_BYTE_B [0] pin is routed to Pin 3 of J5 which is routed to the C_Enable of the Arduino Data Collector. When this bit goes high, the priority encoder goes into statement 1 and sets transfer_write_reg and transfer_write high and latches the value on the LB_BYTE_C[7:0] pins to the transfer_write_byte register. By setting transfer_write_reg high, the priority encoder goes into statement 2 which will set transfer_write register to low and stay in statement 2 of the priority encoder. The back to back high and low on the transfer_write register will cause the active_transfer module to latch the value of transfer_write_byte into the active_transfer_library module and sets up the byte transfer to the PC. When the LB_BYTE_B[0] - C_Enable pin goes low, the encoder will reset transfer_write_reg and transfer_write to low. The encoder goes back to waiting for the LB_BYTE_B[0] - C_Enable to assert high.



```
//-----------------------------------------------
// Detect Transfer From Arduino
//-----------------------------------------------
 always @(posedge CLK_66 or negedge RST)
 begin
    if (!RST)
    begin
         transfer_write_reg <= 1'b0;
         transfer_write <= 1'b0;
         transfer_write_byte <= 0;
         TR_DIR_4  <= 1'b0; //1 = A to B; 0 = B to A
         TR_OE_4  <= 1'b0;
         LB_BYTE_D[0] <= start_stop_cntrl;
         LB_BYTE_D[7:1] <= 7'b0000000;
    end
    else
    begin
         if(LB_BYTE_B[0] & !transfer_write_reg)
         begin
              TR_OE_4  <= 1'b1;
              LB_BYTE_D <= 8'hz;
             transfer_write_reg <= 1'b1;
             transfer_write <= 1'b1;
             transfer_write_byte <= LB_BYTE_C;
         end
         else if(LB_BYTE_B[0] & transfer_write_reg)
         begin
             transfer_write_reg <= 1'b1;
             transfer_write <= 1'b0;
         end
         else if(!LB_BYTE_B[0] & transfer_write_reg)
         begin
             transfer_write_reg <= 1'b0;
             transfer_write <= 1'b0;
             transfer_write_byte <= 0;
         end
         else
         begin
              TR_DIR_4  <= 1'b0; //1 = A to B; 0 = B to A
              TR_OE_4  <= 1'b0;
              LB_BYTE_D[0] <= start_stop_cntrl;
              LB_BYTE_D[7:1] <= 7'b0000000;
         end
    end
 end
```

The 8 bit transceivers are used to connect the +3.3Volt Input/Outputs of the CPLD to the +5Volt Input/Outputs of the Arduino. These 8 bit transceivers incorporate a Direction bit and an Output Enable bit. To guide signals from the CPLD to the Arduino, we turn the Direction bit to "B to A"

TR_DIR_1 <= 1‘b0;

and the Output Enable on.

TR_OE_1 <= 1’b0; (Output Enables are asserted with a zero)

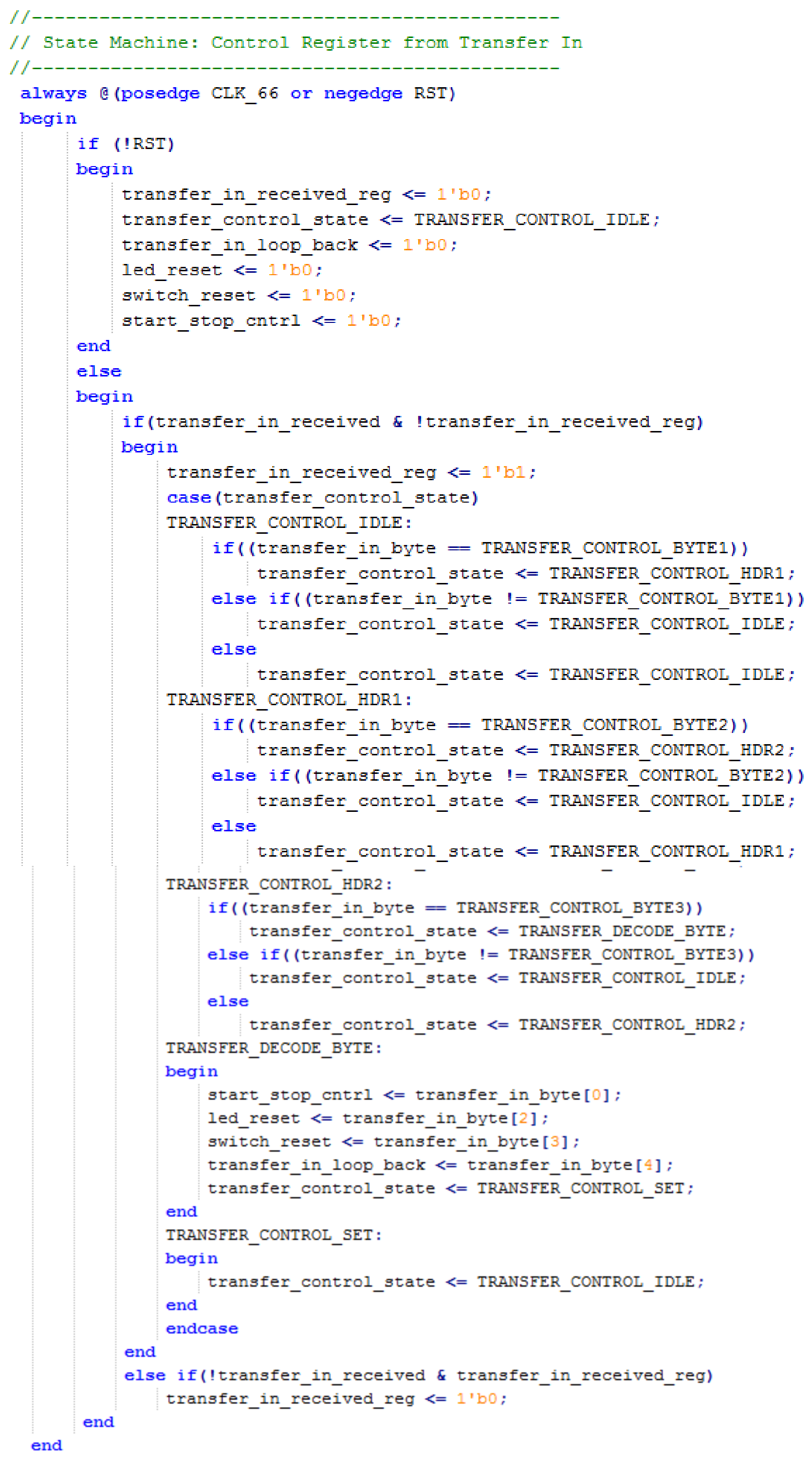
The start_stop_cntrl signal is set by using the TRANSFER_CONTROL state machine in the following section.

Next, we add a TRANSFER_CONTROL state machine to read the Control Register from the Host PC using the active_transfer EndTerm. This state machine will decode the 8 bit control register only after a sequence of three 8 bit bytes in the order of 0x5a, 0xc3, 0x7e. The operation of the state machine is as follows.

- The TRANSFER_CONTROL state machine will stay in the idle state of the parallel encoder until a byte from the active_transfer transfer_to_device register receives a 0x5a.
- This will cause the transfer_control_state to be changed to TRANSFER_CONTROL_HDR1.
- The state machine will stay in the TRANSFER_CONTROL_HDR1 state until the next byte is read from the active_transfer.
- If the byte from transfer_to_device is a 0xc3, the transfer_control_state will be changed to TRANSFER_CONTROL_HDR2.
- If the byte from transfer_to_device is not a 0xc3, the transfer_control_state will go back to idle.
- In the TRANSFER_CONTROL_HDR2 state , the state machine will stay in this state until the next byte from the active_transfer is received.
- If the byte from transfer_to_device is a 0x7e, the transfer_control_state will be changed to TRANSFER_DECODE_BYTE.
- If the byte from transfer_to_device is not a 0x7e, the transfer_control_state will go back to idle.
- In the TRANSFER_DECODE_BYTE state , the state machine will stay in this state until the next byte from the active_transfer.
- The next byte transferred from active_transfer will be decoded as the Control Register.

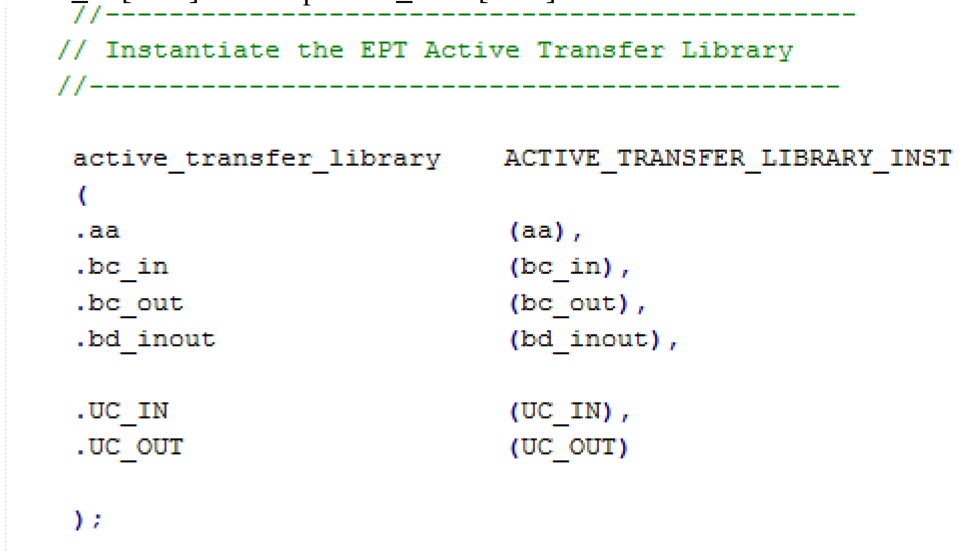
The bits of the Control Register are defined below.

Register	**Bits**	**Description**	**Assertion**
Control	0	Start Stop Cntrl	High
	1	Not Used	
	2	LED Reset	High
	3	Switch Reset	High
	4	Transfer In Loop Back	High
	5	Not Used	
	6	Not Used	
	7	Not Used	
	7	Not Used	



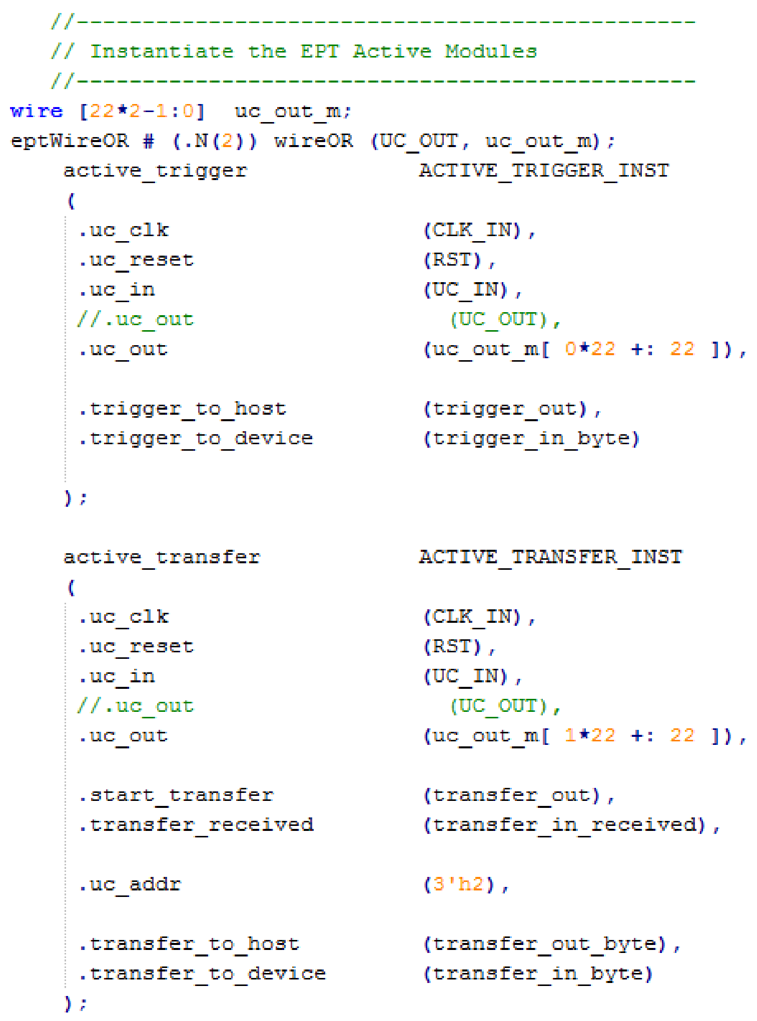
```
//-----------------------------------------------
// State Machine: Control Register from Transfer In
//-----------------------------------------------
 always @(posedge CLK_66 or negedge RST)
 begin
     if (!RST)
     begin
         transfer_in_received_reg <= 1'b0;
         transfer_control_state <= TRANSFER_CONTROL_IDLE;
         transfer_in_loop_back <= 1'b0;
         led_reset <= 1'b0;
         switch_reset <= 1'b0;
         start_stop_cntrl <= 1'b0;
     end
     else
     begin
         if(transfer_in_received & !transfer_in_received_reg)
         begin
             transfer_in_received_reg <= 1'b1;
             case(transfer_control_state)
             TRANSFER_CONTROL_IDLE:
                 if((transfer_in_byte == TRANSFER_CONTROL_BYTE1))
                     transfer_control_state <= TRANSFER_CONTROL_HDR1;
                 else if((transfer_in_byte != TRANSFER_CONTROL_BYTE1))
                     transfer_control_state <= TRANSFER_CONTROL_IDLE;
                 else
                     transfer_control_state <= TRANSFER_CONTROL_IDLE;
             TRANSFER_CONTROL_HDR1:
                 if((transfer_in_byte == TRANSFER_CONTROL_BYTE2))
                     transfer_control_state <= TRANSFER_CONTROL_HDR2;
                 else if((transfer_in_byte != TRANSFER_CONTROL_BYTE2))
                     transfer_control_state <= TRANSFER_CONTROL_IDLE;
                 else
                     transfer_control_state <= TRANSFER_CONTROL_HDR1;
             TRANSFER_CONTROL_HDR2:
                 if((transfer_in_byte == TRANSFER_CONTROL_BYTE3))
                     transfer_control_state <= TRANSFER_DECODE_BYTE;
                 else if((transfer_in_byte != TRANSFER_CONTROL_BYTE3))
                     transfer_control_state <= TRANSFER_CONTROL_IDLE;
                 else
                     transfer_control_state <= TRANSFER_CONTROL_HDR2;
             TRANSFER_DECODE_BYTE:
             begin
                 start_stop_cntrl <= transfer_in_byte[0];
                 led_reset <= transfer_in_byte[2];
                 switch_reset <= transfer_in_byte[3];
                 transfer_in_loop_back <= transfer_in_byte[4];
                 transfer_control_state <= TRANSFER_CONTROL_SET;
             end
             TRANSFER_CONTROL_SET:
             begin
                 transfer_control_state <= TRANSFER_CONTROL_IDLE;
             end
             endcase
         end
         else if(!transfer_in_received & transfer_in_received_reg)
             transfer_in_received_reg <= 1'b0;
     end
 end
```

Next, up is the instantiation for the active_transfer_library. The ports include the input and output pins and the two buses that connect the active modules. These buses are the input UC_IN[23:0] and output UC_OUT[21:0].



```
 //-----------------------------------------------
// Instantiate the EPT Active Transfer Library
//-----------------------------------------------

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

 .UC_IN                      (UC_IN),
 .UC_OUT                     (UC_OUT)

 );
```

Finally, we instantiate the Active EndTerms. For the Data Collection project, we only need active_transfer and active_trigger EndTerms. The uc_out port for both modules must be shared. Since they both drive this bus, a bus wide wired-or circuit is used so that they don't drive each other. The active_transfer EndTerm has a port for the address (uc_addr). This allows the PC to address up to 8 different modules. Just add a three bit address to this port and the PC must add this same address to communicate with this module.



```
    //------------------------------------------------
    // Instantiate the EPT Active Modules
    //------------------------------------------------
wire [22*2-1:0]  uc_out_m;
eptWireOR # (.N(2)) wireOR (UC_OUT, uc_out_m);
    active_trigger              ACTIVE_TRIGGER_INST
    (
     .uc_clk                    (CLK_IN),
     .uc_reset                  (RST),
     .uc_in                     (UC_IN),
     //.uc_out                    (UC_OUT),
     .uc_out                    (uc_out_m[ 0*22 +: 22 ]),

     .trigger_to_host           (trigger_out),
     .trigger_to_device         (trigger_in_byte)

    );

    active_transfer             ACTIVE_TRANSFER_INST
    (
     .uc_clk                    (CLK_IN),
     .uc_reset                  (RST),
     .uc_in                     (UC_IN),
     //.uc_out                    (UC_OUT),
     .uc_out                    (uc_out_m[ 1*22 +: 22 ]),

     .start_transfer            (transfer_out),
     .transfer_received         (transfer_in_received),

     .uc_addr                   (3'h2),

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

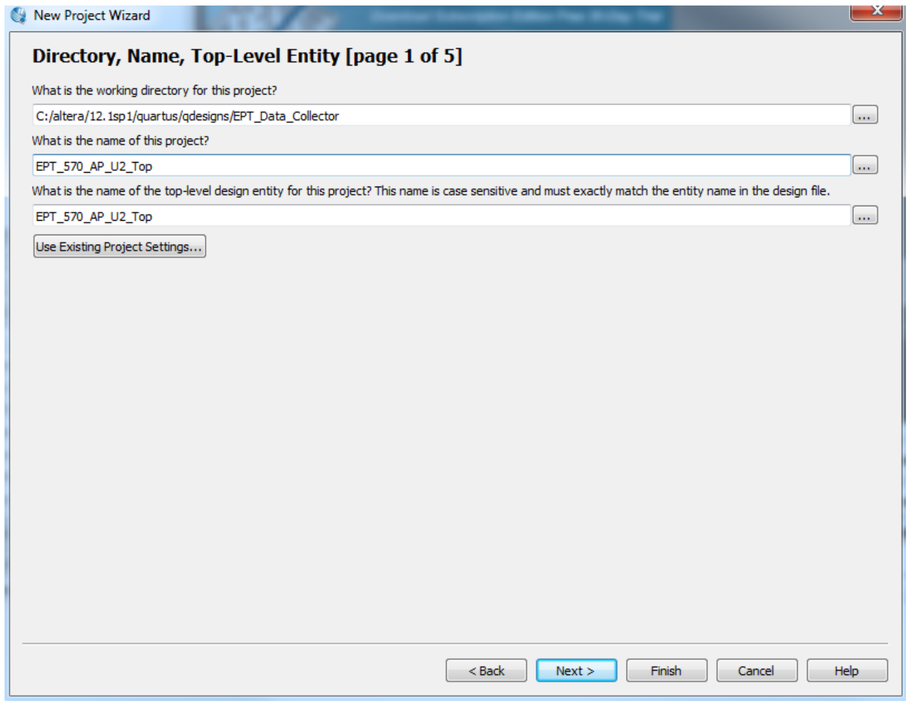
Next, we are ready to compile and synthesize.

6.3.9 CPLD: Compile/Synthesize the Project

The Quartus Prime application will compile/ synthesize the user code, active_transfer_library, and the active EndTerms. The result of this step is a file containing the CPLD code with "*.pof". First, we need to create a project in the Quartus Prime environment. Follow the directions in the section: "Compiling, Synthesizing, and Programming CPLD".
Bring up Quartus Prime, then use Windows Explorer to browse to c:/altera/xxx/quartus/qdesigns create a new directory called: "EPT_Data_Collector".

Open Quartus Prime by clicking on the icon .

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



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_Data_Collector

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.

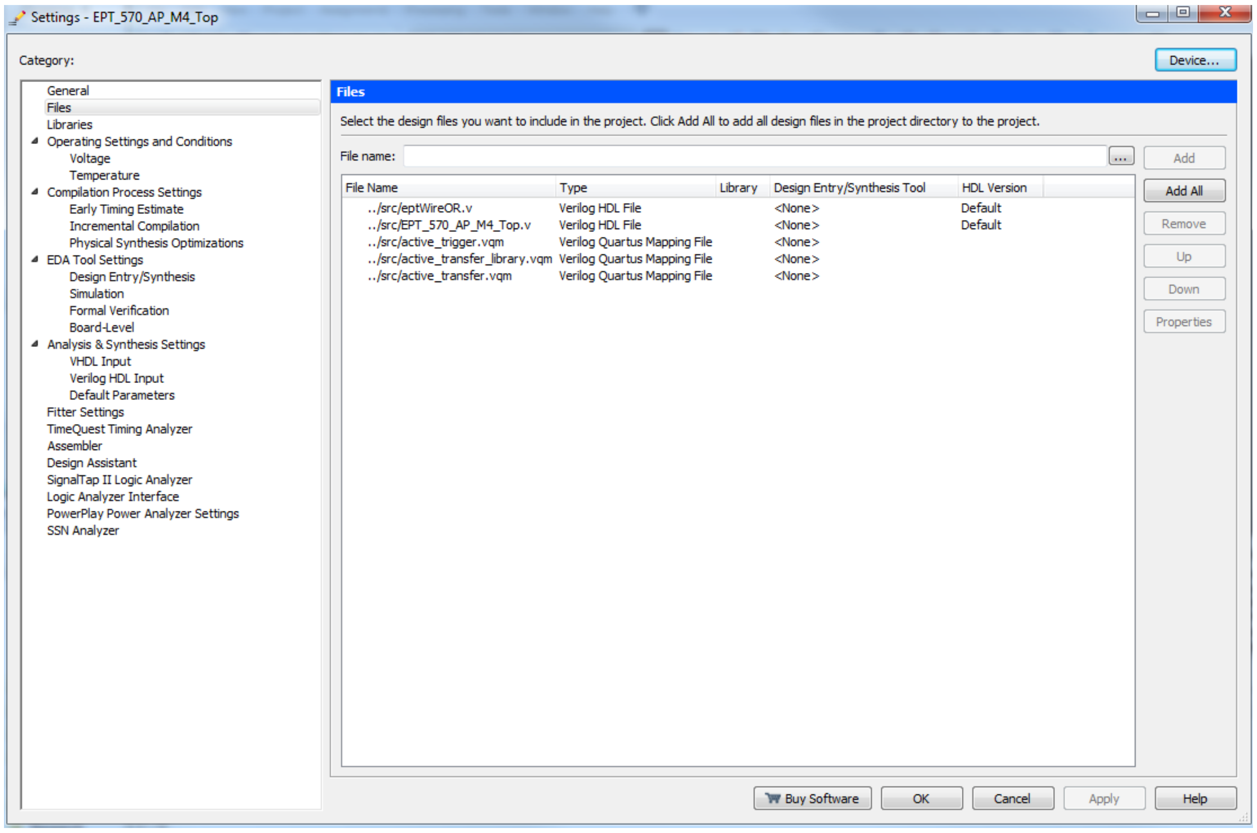
EPT_570_AP_U2_Top

At the Top-Level Entity page, browse to the c:\altera\xxx\quartus\qdesigns\EPT_Data_Collector directory to store your project. Type in a name for your project “EPT_570_AP_M4_Top”.

Follow the steps up to Add Files. At the Add Files box, click on the Browse button and navigate to the project Data Collector install folder in the dialog box. Browse to the \Projects_HDL\EPT_570_AP_Data_Collector \EPT_570_AP_M4_Top folder of the EPT USB-CPLD Development System CD. Copy the files from the \src directory.

- Active_transfer.vqm
- Active_trigger.vqm
- Active_transfer_library.vqm
- eptWireOr.v
- EPT_570_AP_M4_Top.v

Add the 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	Type	Library	Design Entry/Synthesis Tool	HDL Version
../src/eptWireOR.v	Verilog HDL File		<None>	Default
../src/EPT_570_AP_M4_Top.v	Verilog HDL File		<None>	Default
../src/active_trigger.vqm	Verilog Quartus Mapping File		<None>	
../src/active_transfer_library.vqm	Verilog Quartus Mapping File		<None>	
../src/active_transfer.vqm	Verilog Quartus Mapping File		<None>	

Continue following the instructions by adding a device and finishing the project instantiation. Then, bring up the Pin Planner.

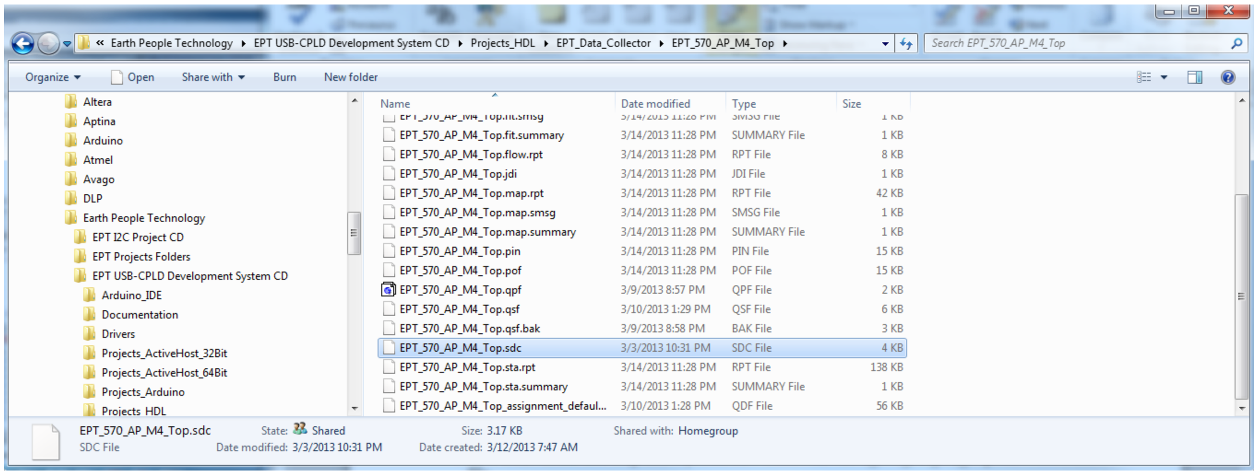
- Under Assignments, Select Import Assignments.

- At the Import Assignment dialog box, browse to the \Projects_HDL\EPT_Data_Collector \EPT_570_AP_M4_Top folder of the EPT USB-CPLD Development System CD. Select the Quartus Specification file, “EPT_570_AP_M4_Top.qsf” file.

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

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

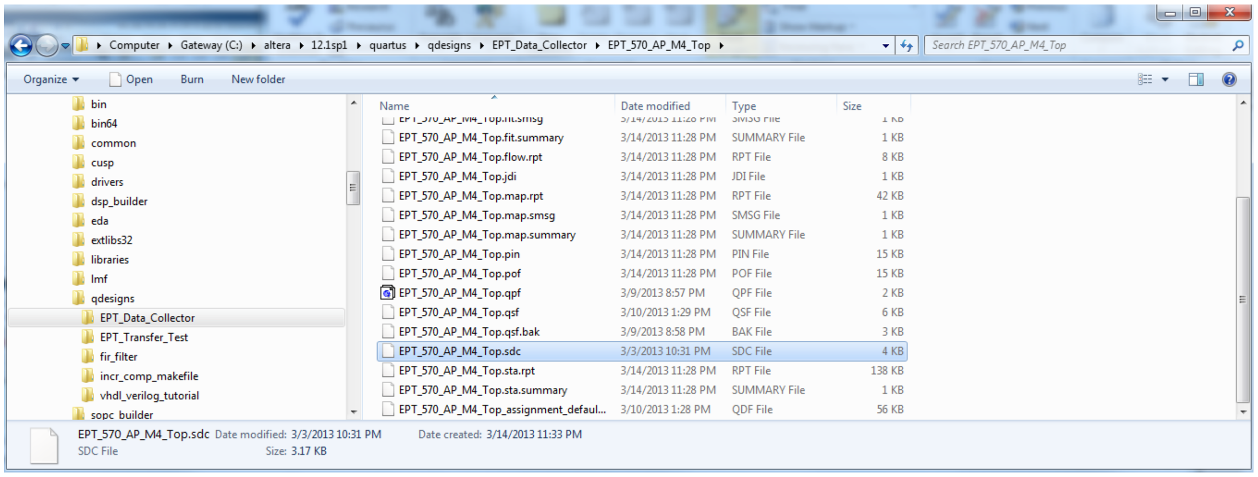
Browse to the \Projects_HDL\EPT_Data_Collector \Altera_EPM570_M4 folder of the EPT USB-CPLD Development System CD. Select the "EPT_570_AP_M4_Top.sdc" file.



Name	Date modified	Type	Size
EPT_570_AP_M4_Top.fit.smsg	3/14/2013 11:28 PM	SMSG File	1 KB
EPT_570_AP_M4_Top.fit.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top.flow.rpt	3/14/2013 11:28 PM	RPT File	8 KB
EPT_570_AP_M4_Top.jdi	3/14/2013 11:28 PM	JDI File	1 KB
EPT_570_AP_M4_Top.map.rpt	3/14/2013 11:28 PM	RPT File	42 KB
EPT_570_AP_M4_Top.map.smsg	3/14/2013 11:28 PM	SMSG File	1 KB
EPT_570_AP_M4_Top.map.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top.pin	3/14/2013 11:28 PM	PIN File	15 KB
EPT_570_AP_M4_Top.pof	3/14/2013 11:28 PM	POF File	15 KB
EPT_570_AP_M4_Top.qpf	3/9/2013 8:57 PM	QPF File	2 KB
EPT_570_AP_M4_Top.qsf	3/10/2013 1:29 PM	QSF File	6 KB
EPT_570_AP_M4_Top.qsf.bak	3/9/2013 8:58 PM	BAK File	3 KB
EPT_570_AP_M4_Top.sdc	3/3/2013 10:31 PM	SDC File	4 KB
EPT_570_AP_M4_Top.sta.rpt	3/14/2013 11:28 PM	RPT File	138 KB
EPT_570_AP_M4_Top.sta.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top_assignment_defaul...	3/10/2013 1:28 PM	QDF File	56 KB

EPT_570_AP_M4_Top.sdc State: Shared Size: 3.17 KB Shared with: Homegroup
SDC File Date modified: 3/3/2013 10:31 PM Date created: 3/12/2013 7:47 AM

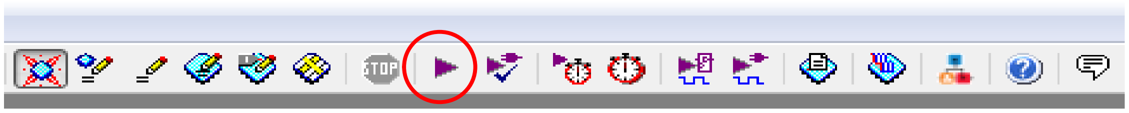
Copy the file and browse to c:\altera\xxx\quartus\qdesigns\EPT_Data_Collector directory. Paste the file.



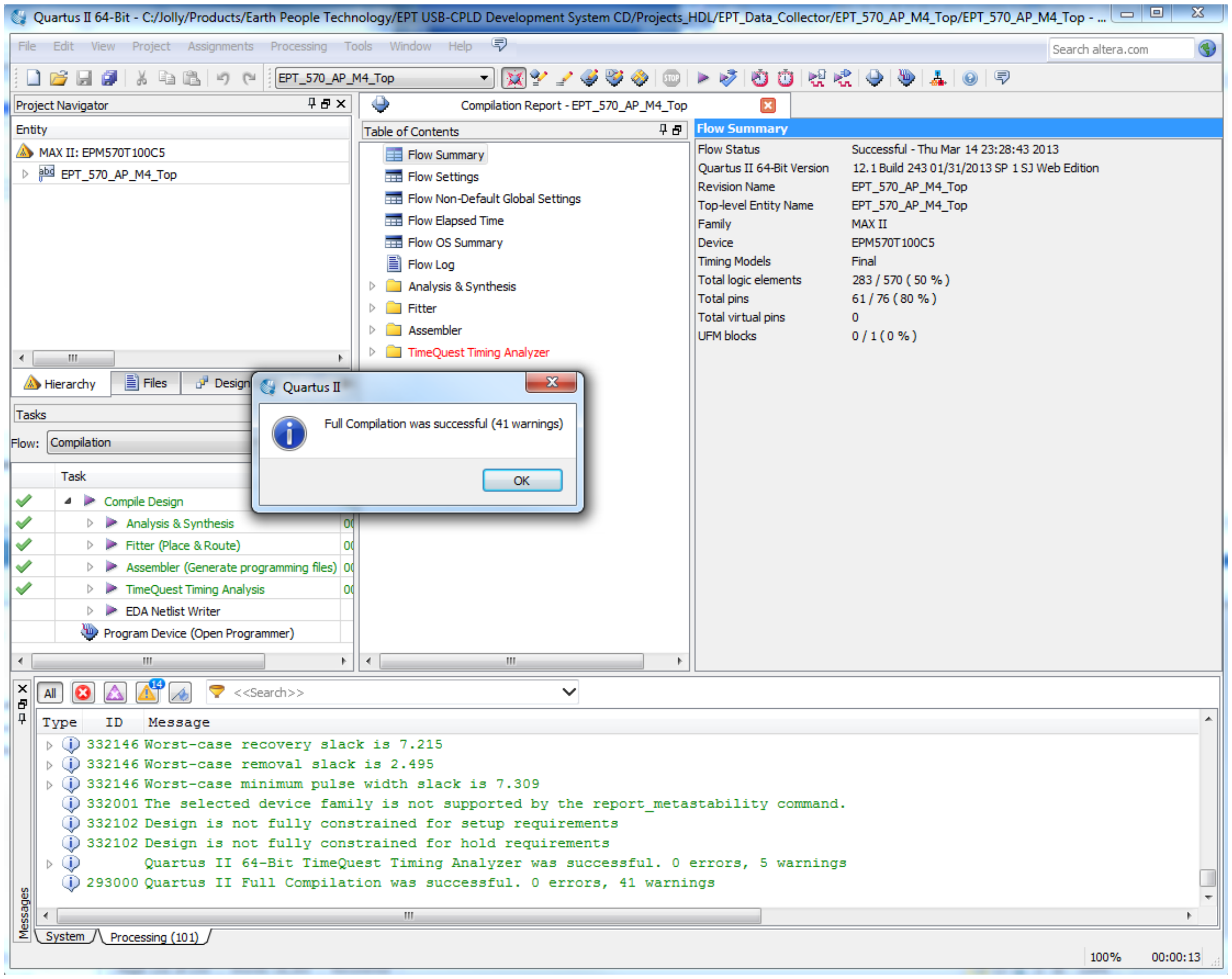
Name	Date modified	Type	Size
EPT_570_AP_M4_Top.fit.smsg	3/14/2013 11:28 PM	SMSG File	1 KB
EPT_570_AP_M4_Top.fit.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top.flow.rpt	3/14/2013 11:28 PM	RPT File	8 KB
EPT_570_AP_M4_Top.jdi	3/14/2013 11:28 PM	JDI File	1 KB
EPT_570_AP_M4_Top.map.rpt	3/14/2013 11:28 PM	RPT File	42 KB
EPT_570_AP_M4_Top.map.smsg	3/14/2013 11:28 PM	SMSG File	1 KB
EPT_570_AP_M4_Top.map.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top.pin	3/14/2013 11:28 PM	PIN File	15 KB
EPT_570_AP_M4_Top.pof	3/14/2013 11:28 PM	POF File	15 KB
EPT_570_AP_M4_Top.qpf	3/9/2013 8:57 PM	QPF File	2 KB
EPT_570_AP_M4_Top.qsf	3/10/2013 1:29 PM	QSF File	6 KB
EPT_570_AP_M4_Top.qsf.bak	3/9/2013 8:58 PM	BAK File	3 KB
EPT_570_AP_M4_Top.sdc	3/3/2013 10:31 PM	SDC File	4 KB
EPT_570_AP_M4_Top.sta.rpt	3/14/2013 11:28 PM	RPT File	138 KB
EPT_570_AP_M4_Top.sta.summary	3/14/2013 11:28 PM	SUMMARY File	1 KB
EPT_570_AP_M4_Top_assignment_defaul...	3/10/2013 1:28 PM	QDF File	56 KB

EPT_570_AP_M4_Top.sdc Date modified: 3/3/2013 10:31 PM Date created: 3/14/2013 11:33 PM
SDC File Size: 3.17 KB

and select the Start Compilation button.



This will cause the compile and synthesization process. After successful completion, the screen should look like the following:



Quartus II 64-Bit - C:/Jolly/Products/Earth People Technology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Data_Collector/EPT_570_AP_M4_Top/EPT_570_AP_M4_Top - ...

Compilation Report - EPT_570_AP_M4_Top

Flow Summary

Flow Status	Successful - Thu Mar 14 23:28:43 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	EPT_570_AP_M4_Top
Top-level Entity Name	EPT_570_AP_M4_Top
Family	MAX II
Device	EPM570T100C5
Timing Models	Final
Total logic elements	283 / 570 (50 %)
Total pins	61 / 76 (80 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

Quartus II

Full Compilation was successful (41 warnings)

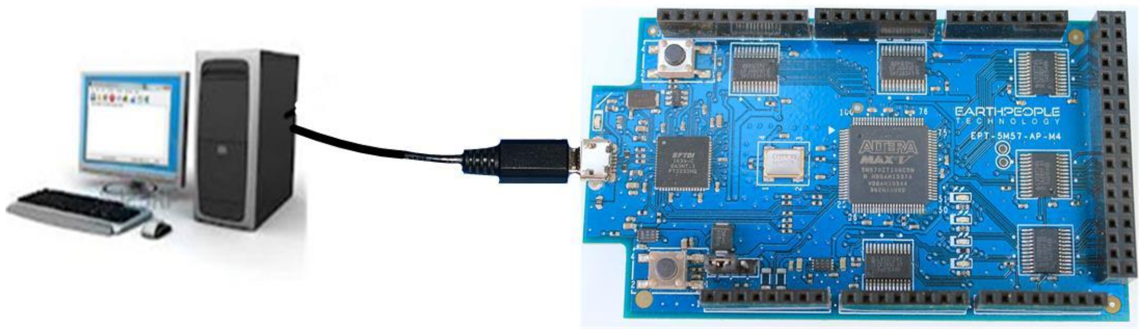
```
Type  ID     Message
      332146 Worst-case recovery slack is 7.215
      332146 Worst-case removal slack is 2.495
      332146 Worst-case minimum pulse width slack is 7.309
      332001 The selected device family is not supported by the report_metastability command.
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
             Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 5 warnings
      293000 Quartus II Full Compilation was successful. 0 errors, 41 warnings
```

If the synthesis fails, you will see the failure message in the message window. Note that in addition to fatal errors, the compile process can produce “warnings” which do not necessarily prevent execution of the code but which should be corrected eventually.

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

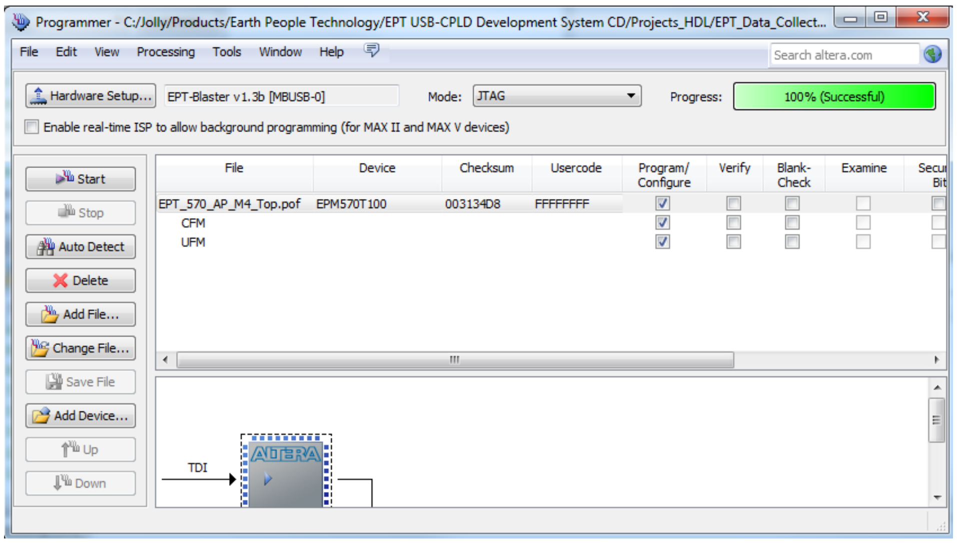
6.3.10 CPLD: Program the CPLD

The final step is programming the “*.pof” file into the CPLD. Follow the section: “Programming the CPLD”.



- Connect the MegaProLogic to the PC,
- Open up Quartus Prime,
- Open the programmer tool
- In the upper left corner of the Programmer Tool, there is a button labeled “Hardware Setup”. Verify that EPT-Blaster v1.0b” has been selected. If not, go to the section JTAG DLL Insert to Quartus Prime and follow the directions.
- Check the box under Program/Configure
- Click the Start button.

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


Programmer - C:/Jolly/Products/Earth People Technology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Data_Collect...
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup...
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
TDI

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

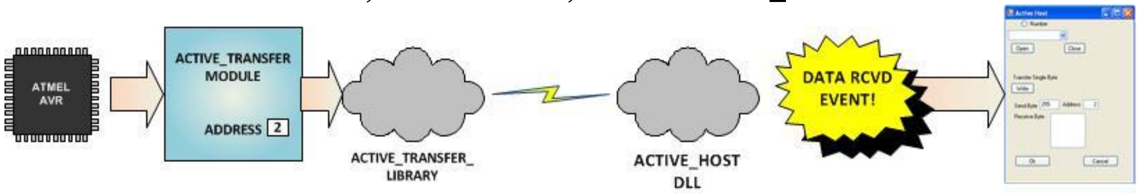
Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Secu Bit
EPT_570_AP_M4_Top.pof	EPM570T100	003134D8	FFFFFFFF	✓				
CFM				✓				
UFM				✓				

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

6.3.11 PC: Design the Project

The final piece of the Data Collection Sampler is the PC application. This application will fetch the data from the CPLD of the MEGAPROLOGIC and display it on the screen. It includes user code, windows form, and the 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, from the user code on the PC, makes the data transfer transparent. The data just appears in memory and the user code will direct the data to a textbox on the Windows Form.

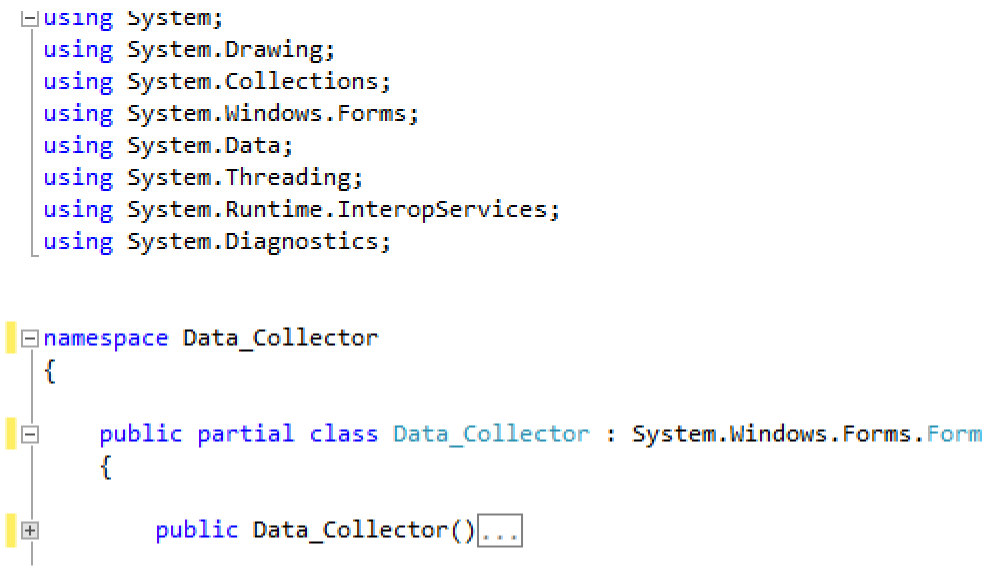
The Data Collector project will perform the following functions.

- Find MEGAPROLOGIC Device.
- Open MEGAPROLOGIC Device.
- Start the Arduino data collection process.
- Wait for data from MEGAPROLOGIC.
- Display data from MEGAPROLOGIC in textbox.

6.3.12 PC: <u>Coding the Project</u>

The user code is based on the .NET Framework and written in C#. The language is great for beginners as it is a subset of the C++ language. It has the look and feel of the familiar C language but adds the ease of use of classes, inheritance and method overloading. C# is an event based language which changes the method of writing code for this project. See the section "Assembling, Building, and Executing a .NET Project on the PC" for a better description of event based language programming.

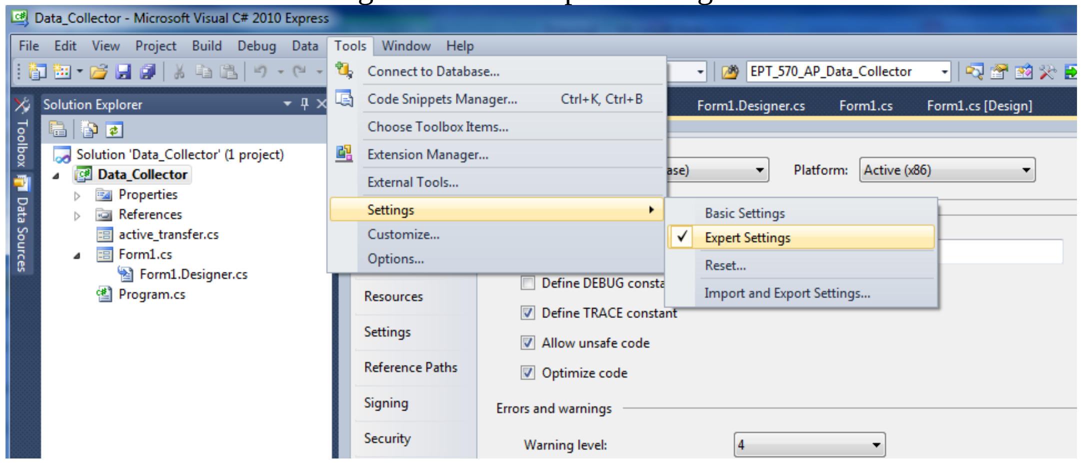
To start the project, follow the section "Assembling, Building, and Executing a .NET Project on the PC". Use the wizard to create project called "Data_Collector". When the wizard completes, the C# Express main window will look like the following.



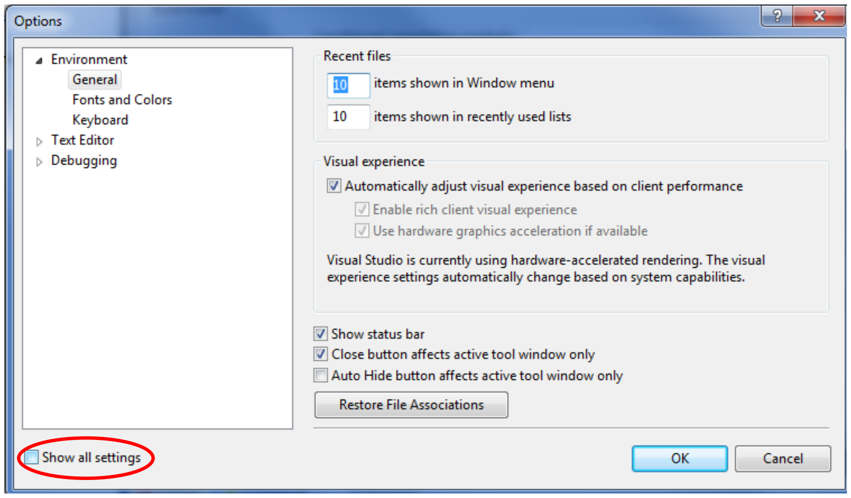
```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace Data_Collector
{
    public partial class Data_Collector : System.Windows.Forms.Form
    {
        public Data_Collector()...
```

These statements setup the namespace and the class for the project. There are several other files that are created by the wizard such as Form1.Designer.cs, Program.cs, Form1.resx. We don’t need to go into these support files, we will just focus on the Form1.cs as this is where all the user code goes.

The project environment must be set up correctly in order to produce an application that runs correctly on the target platform. Visual C# Express defaults new projects to 32 bits. If you OS is a 64 bit platform, use the following directions to set up a 64 bit project. 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.


Options

- Environment
  - General
  - Fonts and Colors
  - Keyboard
- Text Editor
- Debugging

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

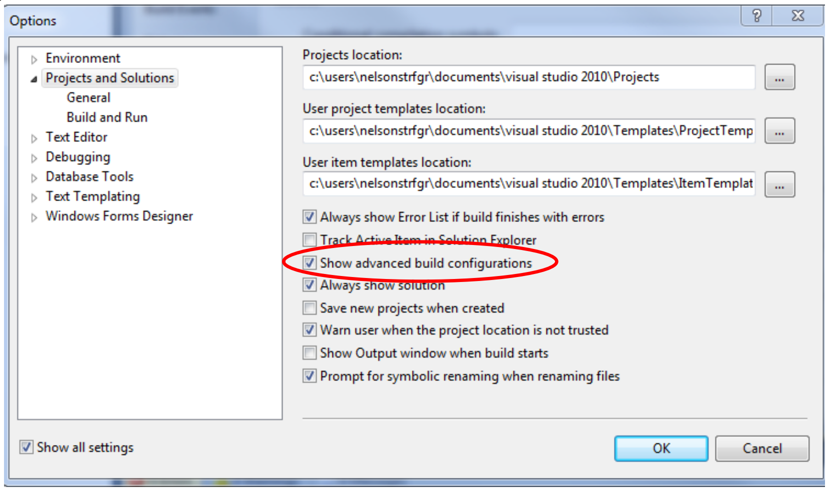
☐ Auto Hide button affects active tool window only

Restore File Associations

☐ Show all settings

OK Cancel

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


Options

- Environment
- Projects and Solutions
  - General
  - Build and Run
- Text Editor
- Debugging
- Database Tools
- Text Templating
- Windows Forms Designer

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

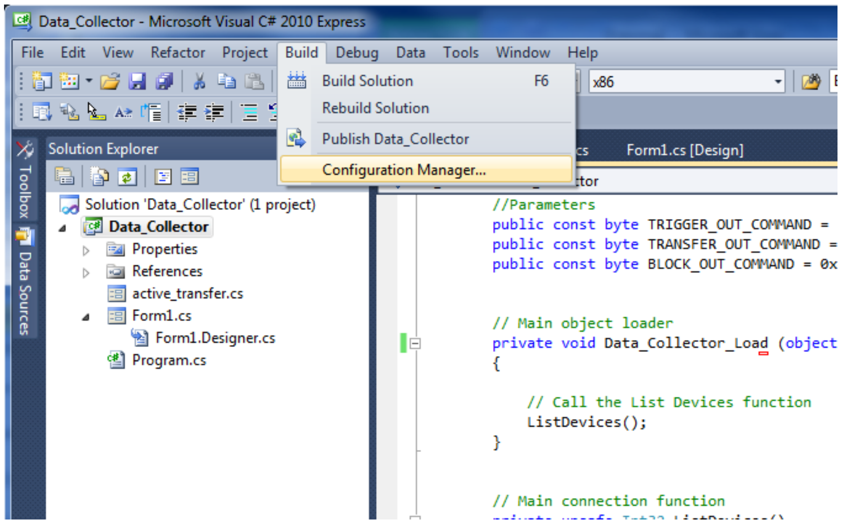
☐ Show Output window when build starts

☑ Prompt for symbolic renaming when renaming files

☑ Show all settings

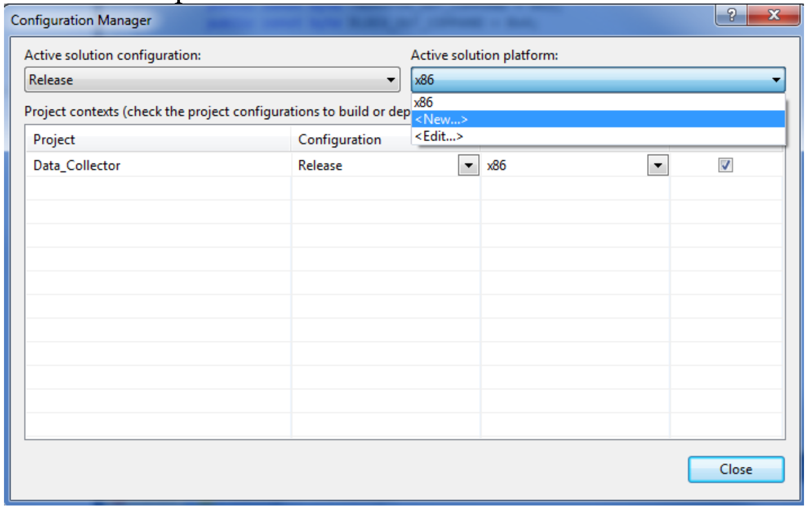
OK Cancel

Go to Build->Configuration Manager.



```
//Parameters
public const byte TRIGGER_OUT_COMMAND =
public const byte TRANSFER_OUT_COMMAND =
public const byte BLOCK_OUT_COMMAND = 0x

// Main object loader
private void Data_Collector_Load (object
{

    // Call the List Devices function
    ListDevices();
}

// Main connection function
```

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



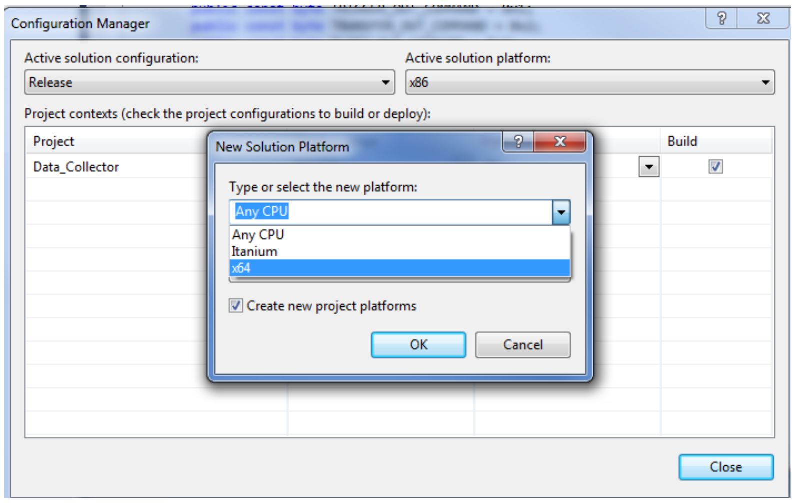
Active solution configuration: Release

Active solution platform: x86

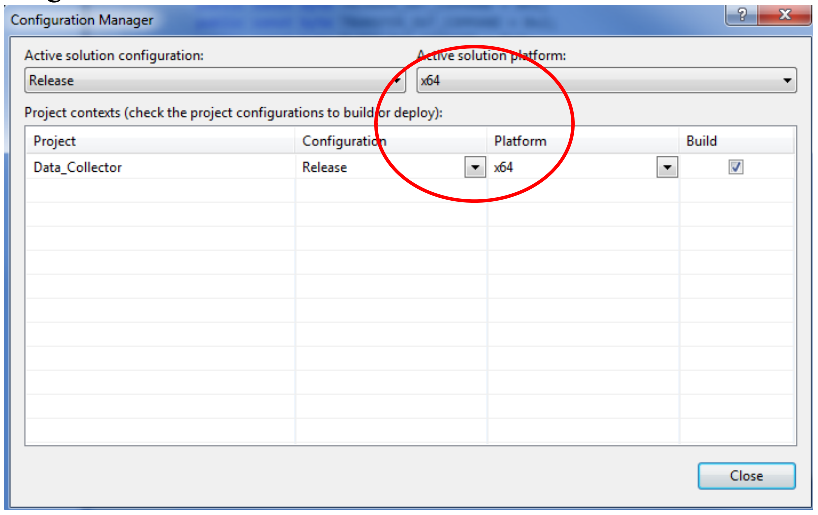
Project contexts (check the project configurations to build or dep

Project	Configuration		
Data_Collector	Release	x86	✓

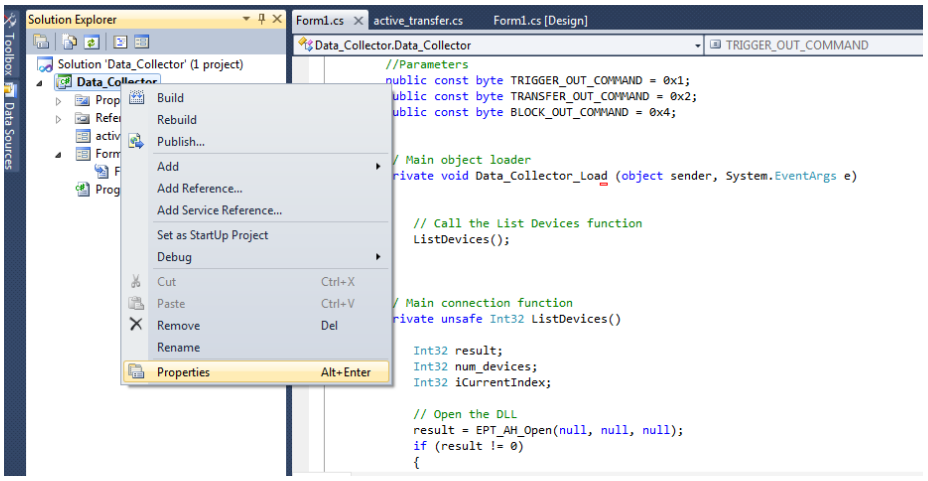
In the New Solution Platform window, click on the drop down box under “Type or select the new platform:”. Select “x64”.


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

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

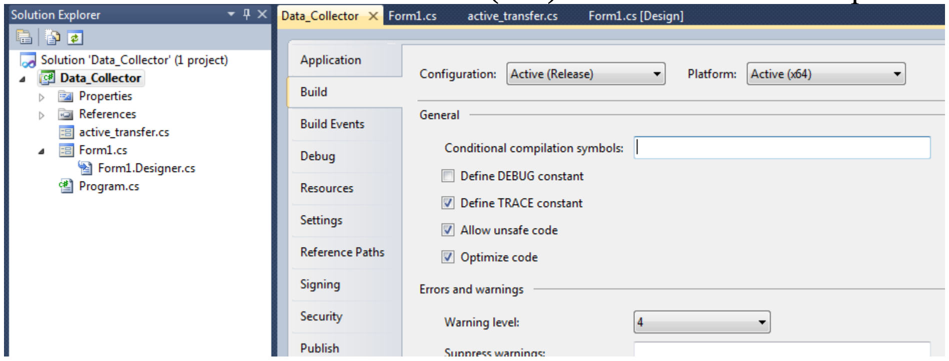

Configuration Manager
Active solution configuration:
Release
Active solution platform:
x64
Project contexts (check the project configurations to build or deploy):
Project
Configuration
Platform
Build
Data_Collector
Release
x64
Close

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.



```
        //Parameters
        public const byte TRIGGER_OUT_COMMAND = 0x1;
         ublic const byte TRANSFER_OUT_COMMAND = 0x2;
         ublic const byte BLOCK_OUT_COMMAND = 0x4;

         / Main object loader
         rivate void Data_Collector_Load (object sender, System.EventArgs e)

            // Call the List Devices function
            ListDevices();

         / Main connection function
         rivate unsafe Int32 ListDevices()

            Int32 result;
            Int32 num_devices;
            Int32 iCurrentIndex;

            // Open the DLL
            result = EPT_AH_Open(null, null, null);
            if (result != 0)
            {
```

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



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

Conditional compilation symbols:

☐ Define DEBUG constant

☑ Define TRACE constant

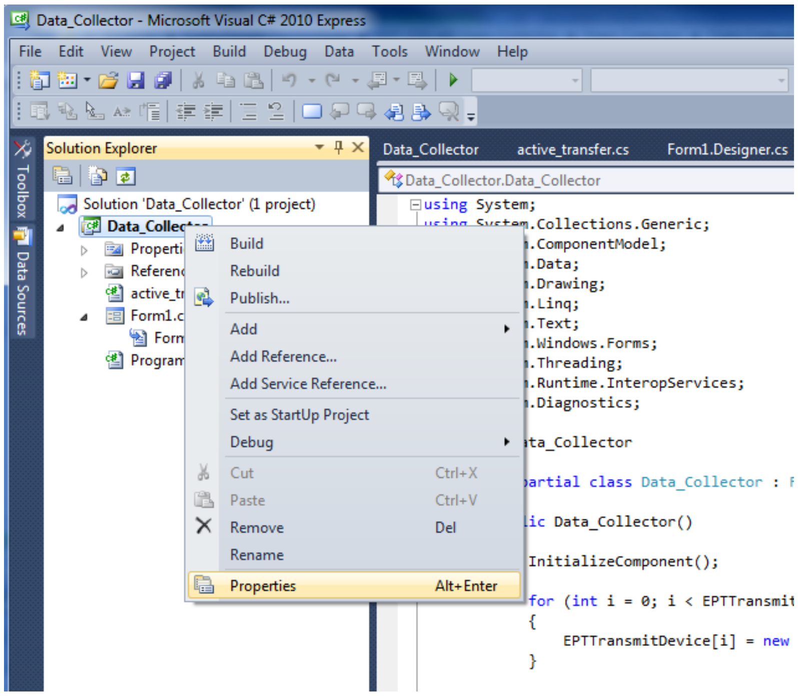
☑ Allow unsafe code

☑ Optimize code

Warning level: 4

Suppress warnings:

Next, unsafe code needs to be allowed so that C# can be passed pointer values from the Active Host. Right click on the “Data Collector” project in the Solution Explorer. Select Properties.



```
using System;
using System.Collections.Generic;
.ComponentModel;
.Data;
.Drawing;
.Linq;
.Text;
.Windows.Forms;
.Threading;
.Runtime.InteropServices;
.Diagnostics;

ta_Collector

artial class Data_Collector : F

ic Data_Collector()

 InitializeComponent();

 for (int i = 0; i < EPTTransmit
 {
     EPTTransmitDevice[i] = new
 }
```

Now we are ready to start coding.

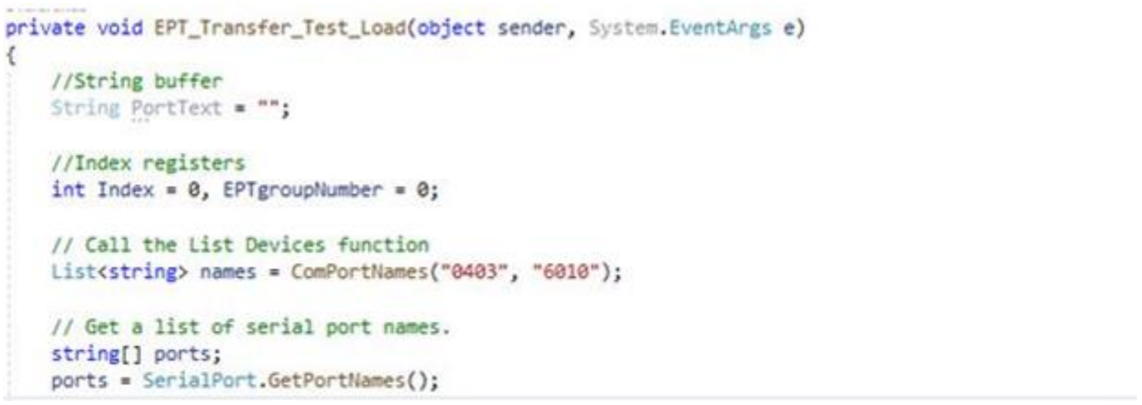
Next, we add two classes for our device. One class stores the information useful for our device for Transmit to the EndTerms such as, address of module, length of transfer etc.

```
//Create an array of the Transfer Class for device
Transfer[] EPTTransmitDevice = new Transfer[8];
```

The next class is used to store parameters for receiving data from the device.

```
//Create a Receive object of the Transfer Class.
Transfer EPTReceiveData = new Transfer();
```

The first function called when the Windows Form loads up is the Data_Collector_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!");
```

The ListDevices() function calls the

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

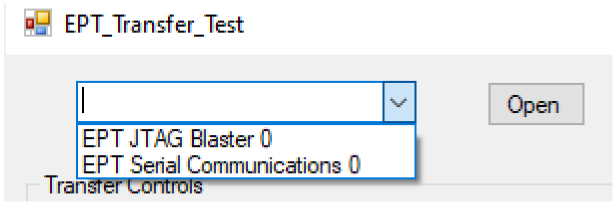
to determine the Serial devices attached to the PC. Next,

```
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);
```

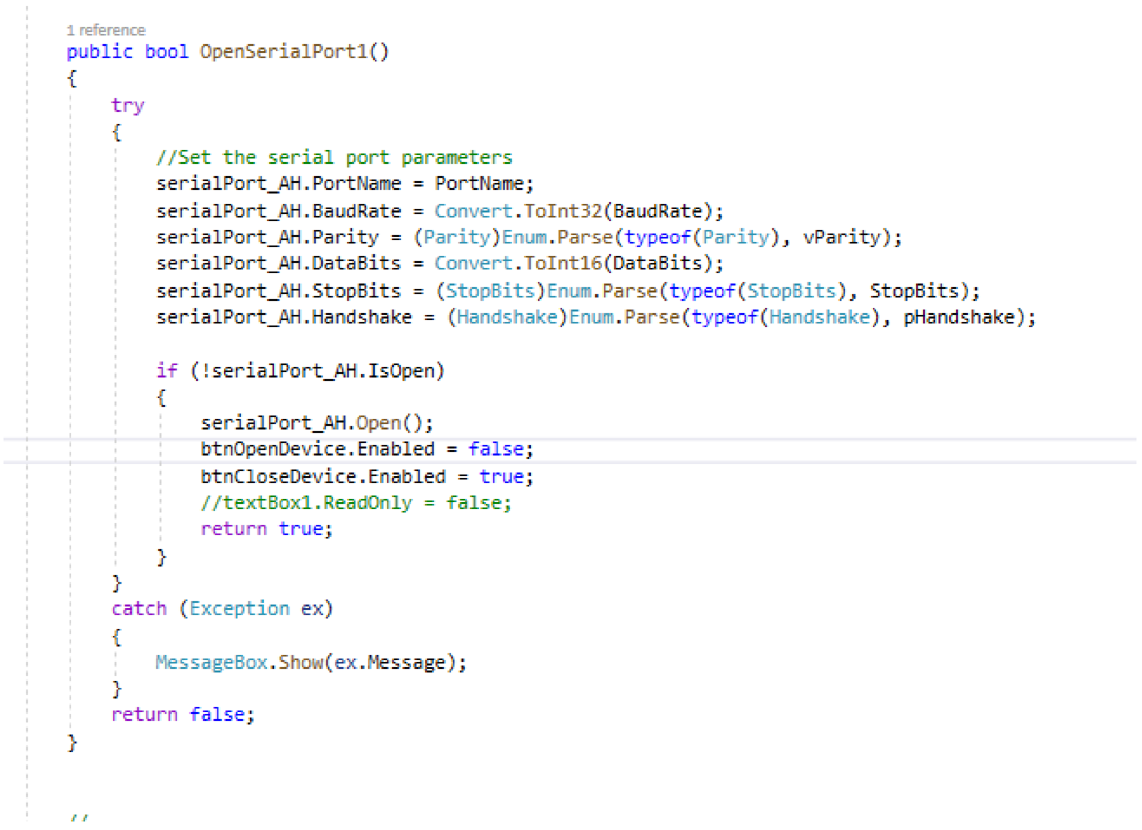
The user will select the device from the drop down combo box. This can be seen when the Windows Form is opened and the cmbDevList combo box is populated with all the devices. The selected device will be stored as an index number in the variable device_index.



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

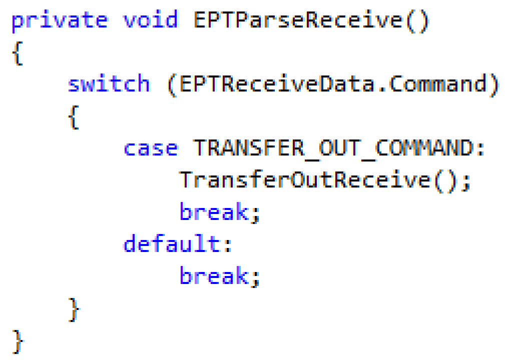
```
OpenSerialPort1()
```

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;
}


//
```

When EPTReadFunction() callback is called and passed parameters from the Active Host dll, it populates the EPTReceiveData object. It then calls EPTParseReceive() function. This function uses a case statement to call the TransferOutReceive() function.



```
private void EPTParseReceive()
{
    switch (EPTReceiveData.Command)
    {
        case TRANSFER_OUT_COMMAND:
            TransferOutReceive();
            break;
        default:
            break;
    }
}
```

TransferOut Receive() creates a string from the EPTReceiveData.Payload parameter. Then sends the string to the textbox, tbDataBytes.

```
public void TransferOutReceive()
{
    string WriteRcvChar = "";
    WriteRcvChar = String.Format("{0}", (int)EPTReceiveData.Payload);
    tbDataBytes.AppendText(WriteRcvChar + " ");
}
```

Controls such as buttons are added to the Form1.cs[Design] window which allow turning on and off signals. These include

- btnWriteByte
- btnTransferReset
- btnOk
- btnClose
- btnResetBlock

Refer to section 6.1.4 Adding Controls to the Project for details about using the ToolBox to place controls on a design. The btnWriteByte click event calls the EPT_AH_SendTransferControlByte(). This function is used to turn on/off bits in the Control Register in the CPLD code. The btnWriteByte will set the start_stop_cntrl signal in the CPLD to one. This signal starts the Arduino Data Collector sending its random word to the CPLD.

```
private void btnWriteByte_Click(object sender, EventArgs e)
{
    int address_to_device;
    address_to_device = Convert.ToInt32(tbAddress.Text);
    EPT_AH_SendTransferControlByte((char)2, (char)1);
}
```

The btnTransferReset sets the start_stop_cntrl bit in the Control Register to zero. This action will cause the Arduino Data Collector to stop sending the random word to the CPLD.

```
private void btnTransferReset_Click(object sender, EventArgs e)
{
    int address_to_device;
    address_to_device = Convert.ToInt32(tbAddress.Text);
    EPT_AH_SendTransferControlByte((char)address_to_device, (char)0);
}
```

The btnResetBlock button will clear the tbDataBytes textblock. The Clear() method is inherited from the textbox class.

```
private void btnResetBlock_Click(object sender, EventArgs e)
{
    tbDataBytes.Clear();
}
```

The btnOk and btnClose buttons are used to end the application. It calls the function EPT_AH_CloseDeviceByIndex() to remove the device from the Active Host dll. The buttons btnOpen and btnClose have their Enabled parameter set to true and false respectively. The Enabled parameter controls whether the button is allowed to launch an event or not. If it is not enabled, the button is grayed out. At the end of each click event, the Application.Exit() method is called. This exits the form.

```
private void btnOk_Click(object sender, EventArgs e)
{
    EPT_AH_CloseDeviceByIndex(device_index);
    btnOpenDevice.Enabled = true;
    btnCloseDevice.Enabled = false;

    lblDeviceConnected.Text = "";
    Application.Exit();
}

private void btnCancel_Click(object sender, EventArgs e)
{
    EPT_AH_CloseDeviceByIndex(device_index);
    btnOpenDevice.Enabled = true;
    btnCloseDevice.Enabled = false;

    lblDeviceConnected.Text = "";
    Application.Exit();
}
```

This is all that is needed for the Data Collector project. The Arduino will generate a random 8 bit word. It then transmits that word to the CPLD using the A0 (WRITE_ENABLE) signal. The CPLD transmits the 8 bit word to the PC using the ACTIVE TRANSFER module of the Active_Transfer Library. The dll reads the 8 bit word into local memory. It then calls the Callback function, EPTReadFunction. The 8 bit is finally displayed to screen using the MessageBox.Show().

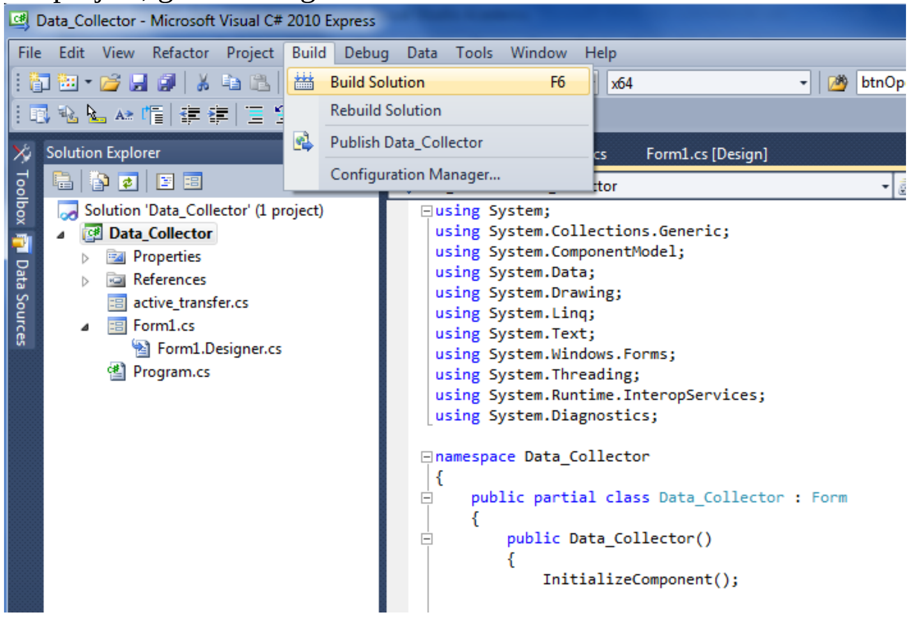
6.3.13 PC: Compiling the Active Host Application

Building the Data_Collector project will compile the code in the project and produce an executable file. It will link all of the functions declared in the opening of the Data_Collector Class with the Active Host dll. The project will also automatically link the FTD2XX.dll to the object code. Follow section: Assembling, Building, and Executing a .NET Project on the PC. Browse to the \Projects_ActiveHost_xxBit\EPT_Data_Collector \Data_Collector\ folder of the EPT USB-CPLD Development System CD. Copy the following files into the project.

- Active_transfer_xxx.cs
- Data_Collector.csproj

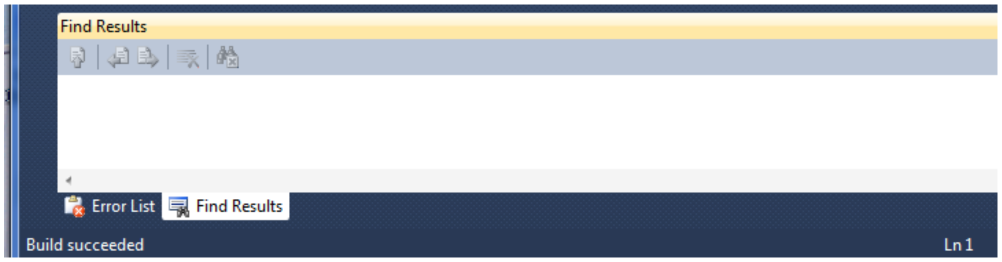
- Data_Collector.csproj.user
- Form1.cs
- Form1.Designer.cs
- Program.cs

To build the project, go to Debug->Build Solution.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace Data_Collector
{
    public partial class Data_Collector : Form
    {
        public Data_Collector()
        {
            InitializeComponent();
```

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”.



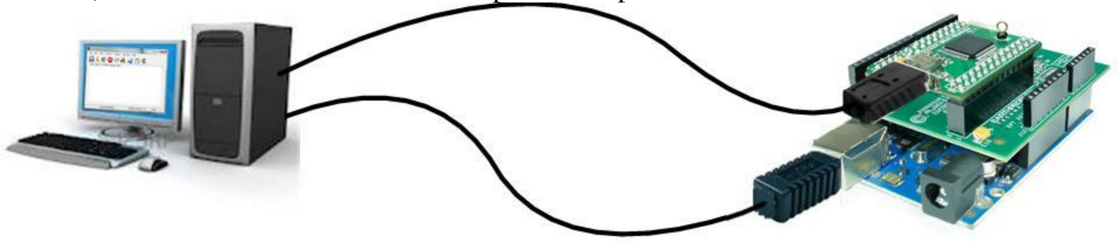
Build succeeded

If the build fails, you will have to examine each error in the “Error List” and fix it accordingly. If you cannot fix the error using troubleshooting methods, post a topic in the Earth People Technology Forum. All topics will be answered by a member of the technical staff as soon as possible.

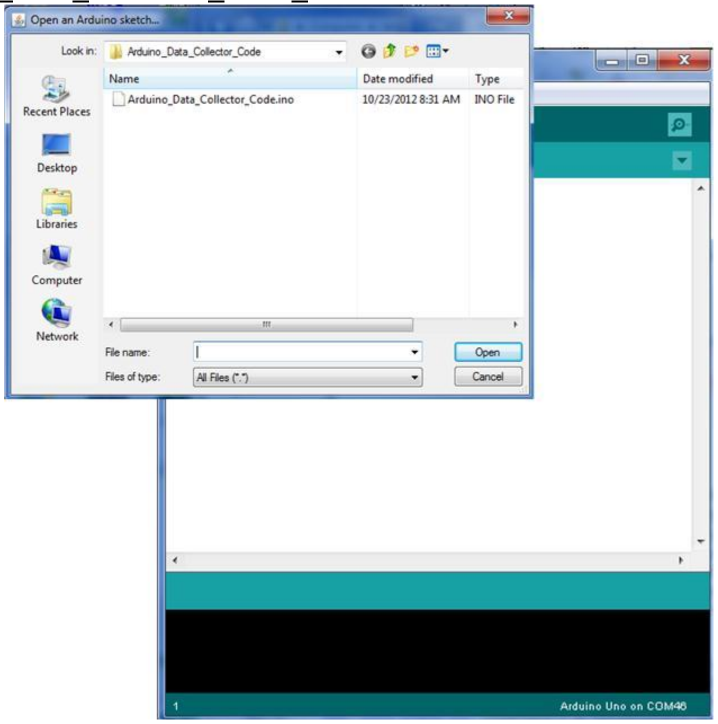
At this point, the environment has produced an executable file and is ready for testing. Next, we will connect everything together and see it collect data and display it.

6.3.14 Connecting the Project Together

Now we will connect the Arduino, EPT 570-AP-M4, and the PC to make a Data Collector. First, connect a USB cable from a USB port on the PC to the Arduino. Second, connect a USB cable from a open USB port on the PC to the EPT 570-AP-M4.



Next, open the Arduino IDE and select File->Open and select your sketch created earlier, Arduino_Data_Collector_Code_M4.ino.

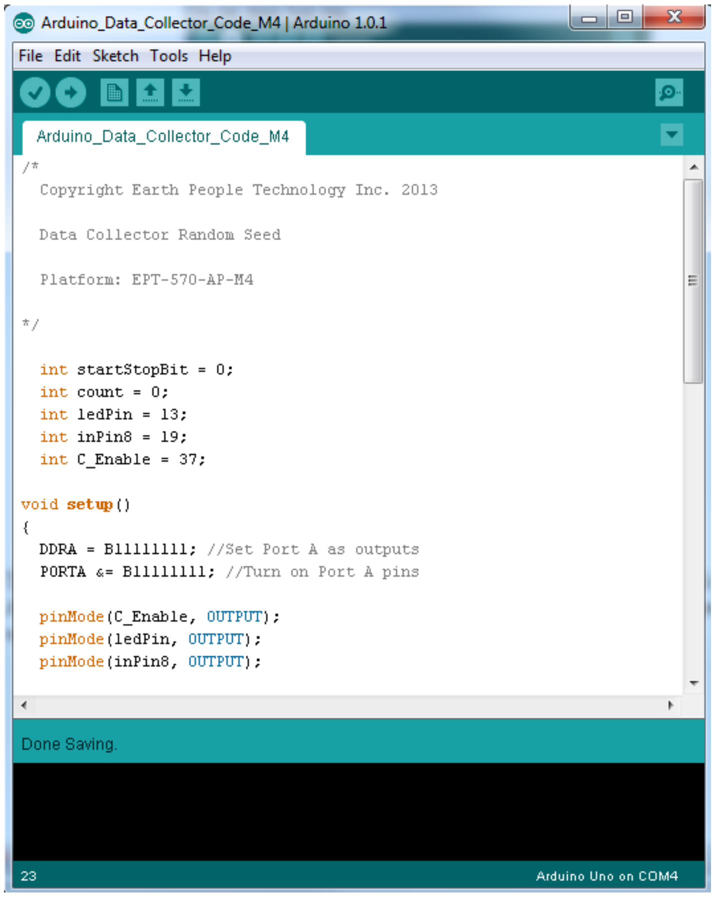

Open an Arduino sketch...
Look in: Arduino_Data_Collector_Code
Recent Places
Desktop
Libraries
Computer
Network
Open
Cancel

Name	Date modified	Type
Arduino_Data_Collector_Code.ino	10/23/2012 8:31 AM	INO File

File name:
Files of type: All Files (*.*)

1 Arduino Uno on COM48

Select the file and click Open. The sketch will now populate the Arduino IDE window. Compile and Download the sketch into the Arduino microcontroller using the Upload button.

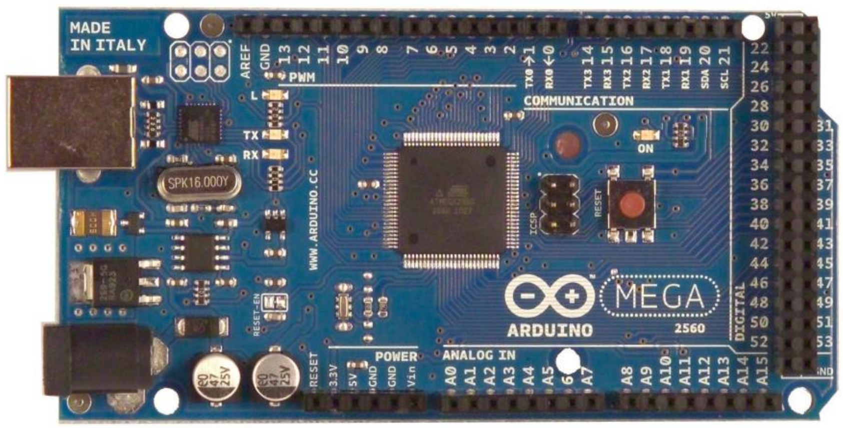

Arduino_Data_Collector_Code_M4 | Arduino 1.0.1
File Edit Sketch Tools Help
Arduino_Data_Collector_Code_M4

```
/*
  Copyright Earth People Technology Inc. 2013

  Data Collector Random Seed

  Platform: EPT-570-AP-M4

*/

  int startStopBit = 0;
  int count = 0;
  int ledPin = 13;
  int inPin8 = 19;
  int C_Enable = 37;

void setup()
{
  DDRA = B11111111; //Set Port A as outputs
  PORTA &= B11111111; //Turn on Port A pins

  pinMode(C_Enable, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(inPin8, OUTPUT);
```

Done Saving.

23 Arduino Uno on COM4

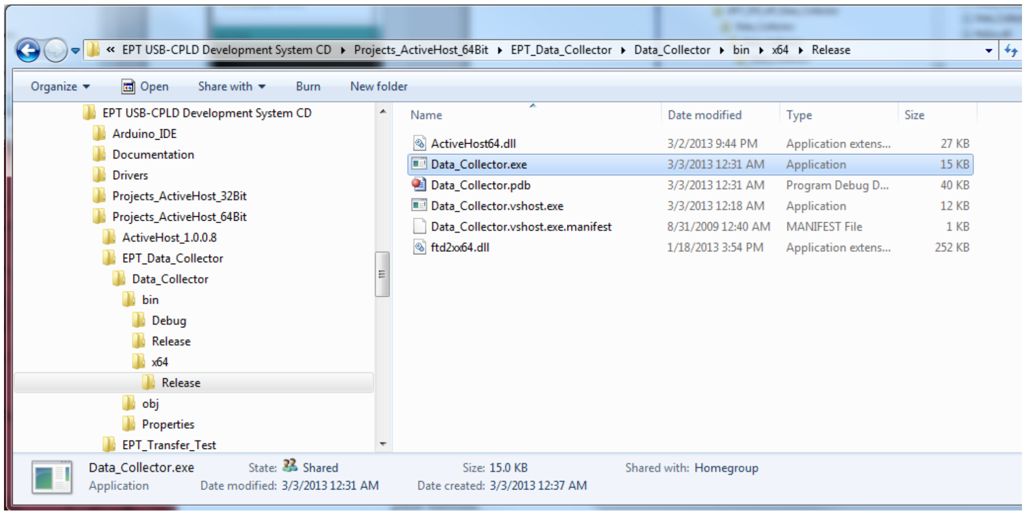
The Arduino IDE will compile the project, then transmit the machine level code into the ATMega328 SRAM to start the program. When this is complete, the Yellow L LED will blink about once per second.



If this LED is blinking at the rate of once per second, the Arduino and the Data Collector project are ready for the EPT 570-AP-M4 code.

The CPLD should already be programmed with its Data Collector Project. If it isn't, follow the instructions in section 3.1.10.

Open the EPT Data Collector on the PC by browsing to the Data Collector project folder. Locate the executable in the \bin\x64\Release folder.

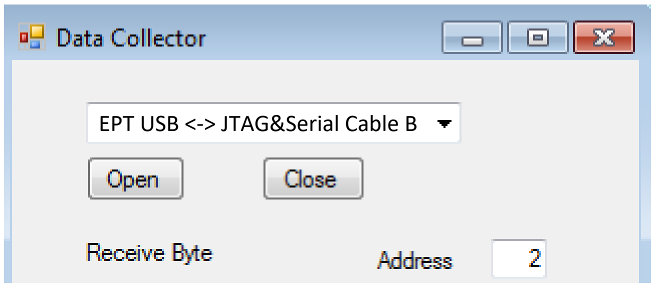

« EPT USB-CPLD Development System CD ▸ Projects_ActiveHost_64Bit ▸ EPT_Data_Collector ▸ Data_Collector ▸ bin ▸ x64 ▸ Release
Organize
Open
Share with
Burn
New folder

- EPT USB-CPLD Development System CD
  - Arduino_IDE
  - Documentation
  - Drivers
  - Projects_ActiveHost_32Bit
  - Projects_ActiveHost_64Bit
    - ActiveHost_1.0.0.8
    - EPT_Data_Collector
      - Data_Collector
        - bin
          - Debug
          - Release
          - x64
            - Release
        - obj
        - Properties
    - EPT_Transfer_Test

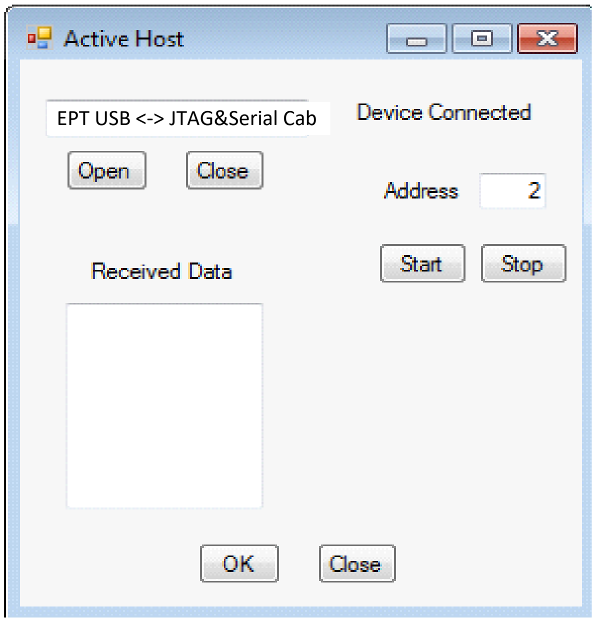
Name	Date modified	Type	Size
ActiveHost64.dll	3/2/2013 9:44 PM	Application extens...	27 KB
Data_Collector.exe	3/3/2013 12:31 AM	Application	15 KB
Data_Collector.pdb	3/3/2013 12:31 AM	Program Debug D...	40 KB
Data_Collector.vshost.exe	3/3/2013 12:18 AM	Application	12 KB
Data_Collector.vshost.exe.manifest	8/31/2009 12:40 AM	MANIFEST File	1 KB
ftd2xx64.dll	1/18/2013 3:54 PM	Application extens...	252 KB

Data_Collector.exe State: Shared Size: 15.0 KB Shared with: Homegroup
Application Date modified: 3/3/2013 12:31 AM Date created: 3/3/2013 12:37 AM

Initiate the application by double clicking the application icon in the \Release folder of the project. The application will open and automatically load the Active Host dll. The application will locate the EPT 570-AP-M4 device. Next, the combo box at the top will be populated with the name of the device.

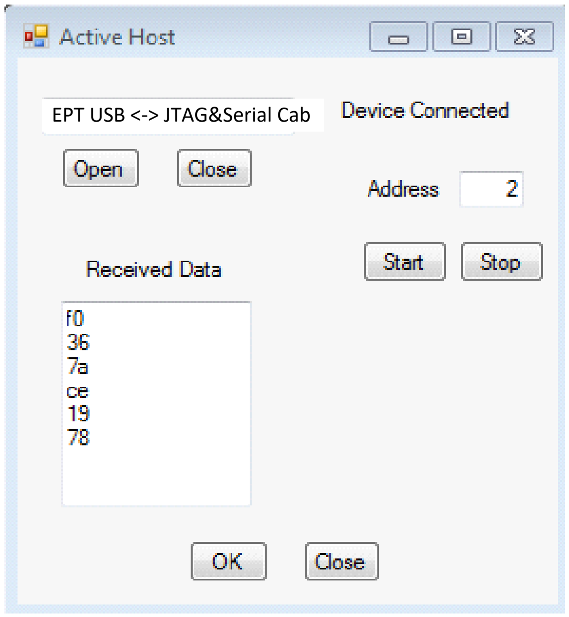


Select the EPT 570-AP device and click the Open button. If the Active Host application connects to the device, a label will indicate "Device Connected". Next, select the address of the Active Transfer module in the CPLD. In our case it is "2".



6.3.15 Testing the Project

To test our Data Collector project, just click on the Start button. As soon as the device connects, the data from the Arduino will appear in the received data textBox.



```
f0
36
7a
ce
19
78
```

And that’s all there is to the Data Collector Project. It’s up to the user to use this project as a base to create much larger projects. You can easily make a volt meter using this project by turning off the Random number generator in the Arduino and reading the

Analog Pins. Also, reformat the textBox display that it shows in decimal instead of the Hexadecimal display.

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