

EARTH PEOPLE TECHNOLOGY, Inc

UnoProLogic USB-CPLD DEVELOPMENT SYSTEM User Manual

The UnoProLogic 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 UnoProLogic development system provides a convenient, user-friendly work flow by connecting seamlessly with Intel FPGA's Quartus Prime software. The user will develop the code in the Quartus environment on a Windows Personal Computer. The programmable logic code is loaded into the CPLD using only the Quartus Programmer tool and a standard USB-C 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 UnoProLogic development system is a unique combination of hardware and software.

Circuit designs, software and documentation are copyright © 2023, Earth People Technology, Inc

Microsoft and Windows are both registered trademarks of Microsoft Corporation. Intel is a trademark of the Intel Corporation. All other trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.

http://www.earthpeopletechnology.com/



Table of Contents

| 1 | Introdu | ction and General Description | 5 |
|---|---------|--|-----|
| | 1.1 Tes | t Driving the Active Host Test Application | 6 |
| | 1.2 EP | Г-5М57-АР-U2 | 16 |
| | 1.2.1 | Serial USB Communications | 19 |
| | 1.2.2 | Inputs and Outputs | 23 |
| | 1.2.3 | JTAG | 23 |
| | 1.3 Act | ive Host EndTerms | |
| | 1.4 Act | ive Transfer EndTerms | 29 |
| 2 | EPT Dr | ivers | |
| | 2.1 US | B Driver | |
| | 2.2 JTA | AG DLL Insert to Quartus Prime Lite | |
| | 2.2.1 | Installing Quartus | |
| | 2.2.2 | | 35 |
| | 2.2.3 | | |
| | 2.2.4 | Adding the EPT_Blaster to Quartus Prime | |
| | 2.3 Act | ive Host Application DLL | 49 |
| 3 | | Transfer Library | |
| | | Γ Active Transfer System Overview | |
| | | tive Transfer Library | |
| | 3.2.1 | Active Trigger EndTerm | |
| | 3.2.2 | Active Transfer EndTerm | |
| | 3.2.3 | Active Block EndTerm | |
| | | ning Diagram for Active Transfer EndTerms | |
| | 3.3.1 | Active Trigger EndTerm Timing | |
| | 3.3.2 | Active Transfer EndTerm Timing | |
| | 3.3.3 | Active Block EndTerm Timing | |
| 4 | 1 | ing, Synthesizing, and Programming CPLD | |
| | | ting up the Project and Compiling | |
| | 4.1.1 | Selecting Pins and Synthesizing | |
| | 4.1.2 | - 8 8 | |
| 5 | | Host Application | |
| | | gger EndTerm | |
| | | nsfer(Byte) EndTerm | |
| | | ck EndTerm | |
| | | ive Host DLL | |
| | 5.4.1 | Active Host Open Device | |
| | 5.4.2 | Active Host Triggers | |
| | 5.4.3 | Active Host Byte Transfers | 100 |



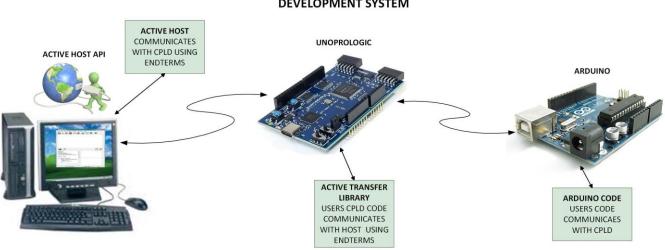
| | 5.4.4 | Active Host Block Transfers | |
|----|----------|--|-----|
| 6 | Assemb | ling, Building, and Executing a .NET Project on the PC | |
| 6 | 5.1 Cre | ating a Project | |
| | 6.1.1 | Setting up the C# Express Environment x64 bit | |
| 6 | 5.2 Ass | embling Files into the Project | |
| | 6.2.1 | Changing Project Name | |
| | 6.2.2 | Add Files to Project | |
| | 6.2.3 | Adding Controls to the Project | |
| | 6.2.4 | Adding the DLL's to the Project | |
| | 6.2.5 | Building the Project | |
| | 6.2.6 | Testing the Project | |
| 7 | Using th | ne Analog to Digital Converter | |
| 7 | 7.1 Reg | sister Descriptions | |
| 7 | 7.2 Cor | version Register | |
| 7 | 7.3 Setu | ıp Register | |
| 7 | 7.4 Ave | eraging Register | |
| 7 | 7.5 Res | et Register | |
| 8 | | oProLyzer Application | |
| 8 | 8.1 Acc | essing the UnoProLyzer Application | |
| 9 | UnoPro | Logic Development Process | |
| ç | 9.1 Des | igning a Simple Data Collection Sampler | |
| | 9.1.1 | The Arduino Microcontroller Board | |
| | 9.1.2 | Create Data Generator | |
| | 9.1.3 | Select I/O's for Fast Throughput on Arduino | |
| | 9.1.4 | Coding the Arduino Data Sampler | 147 |
| | 9.1.5 | Building Arduino Project | |
| | 9.1.6 | Programming the Arduino | |
| | 9.1.7 | CPLD Active Transfer EndTerm Coding and Initiation | |
| | 9.1.8 | CPLD: Define the User Design | |
| | 9.1.9 | CPLD: Compile/Synthesize the Project | |
| | 9.1.10 | CPLD: Program the CPLD | |
| | 9.1.11 | PC: Design the Project | |
| | 9.1.12 | PC: Coding the Project | |
| | 9.1.13 | PC: Compiling the Active Host Application | |
| | 9.1.14 | Adding the DLL's to the Project | |
| | 9.1.15 | Connecting the Project Together | |
| | 9.1.16 | Testing the Project | |
| 10 | APPEN | DIX I | |





1 Introduction and General Description

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



THE EARTH PEOPLE TECHNOLOGY USB-CPLD DEVELOPMENT SYSTEM

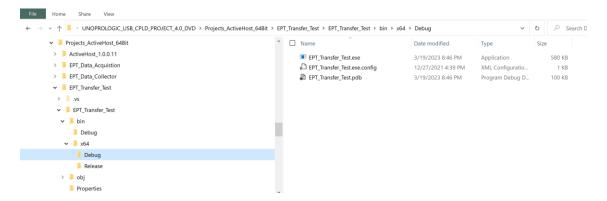
The user's Arduino code is developed to perform particular functions required by the user (such as reading a temperature sensor). The code is downloaded to the microcontroller using the Arduino IDE system provided as part of the microcontroller development system. The UnoProLogic Development System allows 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 UnoProLogic 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



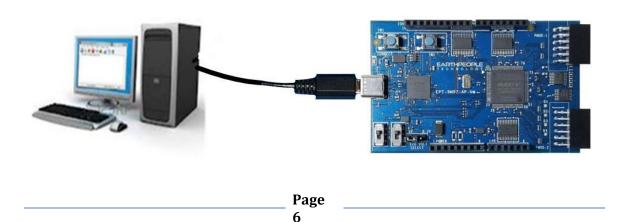
GUI (graphical user interface). Users do not need to interface with the USB Host Driver or any Windows drivers. They need only to include the Active Host dll in their projects. The Active Transfer Libraries must be included in the CPLD project to take advantage of the configurability of the Active Host SDK. All of the drivers, libraries, and project source code are available at <u>www.earthpeopletechnology.com</u>.

1.1 Test Driving the Active Host Test Application

The UnoProLogic 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.

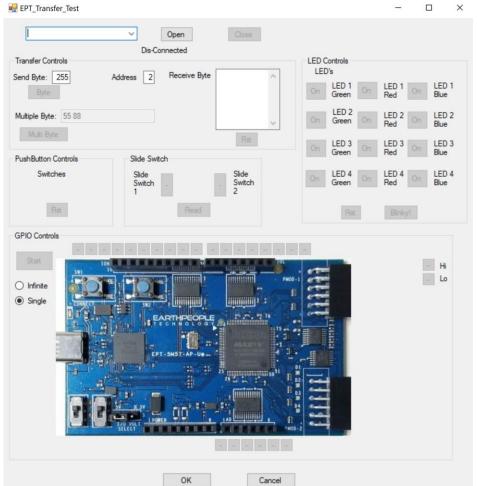


To test drive the application, connect the UnoProLogic to the Windows PC using Type A to USB-C cable. Load the driver for the board. See the section EPT Drivers for instructions on loading the UnoProLogic driver. If the USB driver fails to load, the Windows OS will indicate that no driver was loaded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



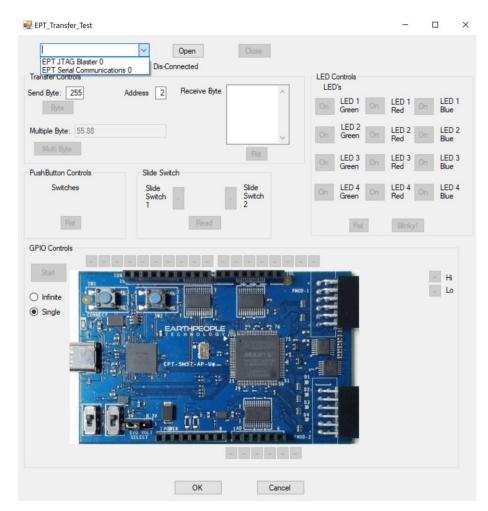


Next, open a Windows Explorer browser. Browse to the Projects_ActiveHost_64Bit\EPT_Transfer_Test\EPT_Transfer_Test\bin\x64\Debug\ folder on the UNO_USB_CPLD_PROJECT_DVD. The application should load with a Windows form.



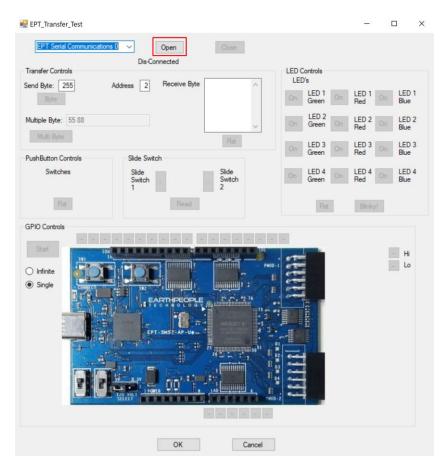
With the application loaded, select the EPT Serial Communications x board from the dropdown combo box.





Next, click on the Open button.





The "Device Connected" label should be seen to indicate the Windows App is now connected to the UnoProLogic.



| EPT_Transfer_Test | | | | | - | 0 | | × |
|--|--|---------------|---|----------------|---|----------------|---|---|
| EPT Serial Communication | ns 0 v Open | Close | | | | | | |
| Transfer Controls Send Byte: 255 Byte Multiple Byte: 55 88 Multi Byte PushButton Controls Switches | Device Connected Address 2 Receive Byte Slide Switch Slide | Rst | LED Controls LED's On LED 1 On Green On LED 2 On LED 3 On LED 3 | On On On | LED 1 Red LED 2 Red LED 3 Red LED 4 | On On On | LED Blue LED Blue LED Blue | 2 |
| Rst | Switch . 1 Read | - Switch 2 | Un Green | Un | Red | | Blue | |
| Start Infinite Single | | | | | | 1 | Hi Lo | |
| | OK | Cance | al | | | | | |

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



| EPT_Transfer_Test | | | | | - | |
|---|---|-----------------------------|--|--|----------------|--|
| EPT Serial Communications | | Close | | | | |
| Transfer Controls iend Byte: 255 Byte Multiple Byte: 55 88 Multi Byte PushButton Controls Switches Rat | Device Connected Address 2 Receive Byte Slide Switch Slide Switch 1 | Rat Slide Switch 2 | On G On G | trols ED 1 On ED 2 On ED 3 On ED 3 On ED 4 On | LED 2 Red C | On LED Blue On LED Blue On LED Blue |
| GPIO Controls | | | LL | Rat | Blinky! | ні п Lo |
| 66 | UNITED OK | Cancel | П 24 П 24 П 24 П 24 РКОВ-2 | | | |
| 391 | | | <u> </u> | | 100 | |



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.

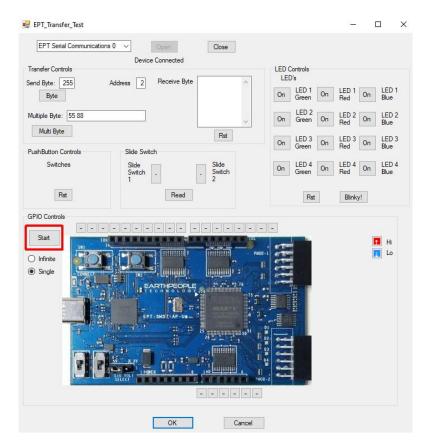




To exercise the Block Transfer EndTerm, click the "Start" button in the GPIO Controls group. The UnoProLogic will sample the state of each Input pin of the CPLD that is connected to a board edge connector. The Transfer Test Window will display the results of each pin, Hi or Lo

| <u>↑</u> | Hi |
|----------|----|
| 1 | Lo |





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



| EPT_Transfer_Test | - 0 |
|---|------------------------------------|
| EPT Serial Communications 0 V Open Close | |
| Device Connected | |
| Transfer Controls | LED Controls |
| Send Byte: 255 Address 2 Receive Byte | LED's |
| Byte | On Green On LED 1 On LED 1 On Blue |
| Multiple Byte: 55 88 | On Green On LED 2 On LED 2 Blue |
| Multi Byte Rst | On LED 3 On LED 3 On LED 3 Blue |
| PushButton Controls Slide Switch | |
| Switches Slide Switch - Slide Switch - Switch 2 | On LED 4 Green On Red On Blue |
| Rst | Rst Blinky! |
| | |
| | |

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





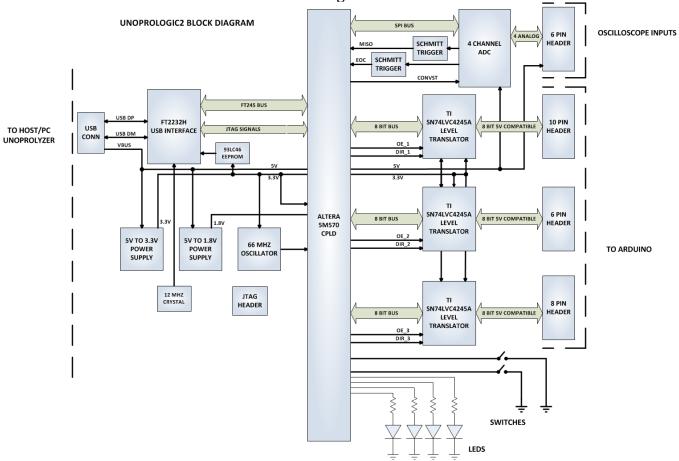
1.2 EPT-5M57-AP-U2

The UnoProLogic board is equipped with an Intel 5M570 CPLD; which is programmed using the Intel Quartus Prime software. The CPLD has 570 Logic Elements which is equivalent to 440 Macrocells. An on board 66 MHz oscillator is used by the EPT Active Transfer Library to provide data transfer rates of up to 0.1 Mega Bytes per second. 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 RGB LED's, two Slide Switches and two Push Buttons that are controllable by the user code. The hardware features are as follows.

- Intel 5M570 CPLD with 440 Macrocells
- FT2232H USB to Serial Interface chip
- 4 Channel ADC 300KSamples/Second
- 66 MHz oscillator for driving USB data transfers and users code
- Three bidirectional voltage translator/bus transceivers
- 24 user Input/Outputs available as three 8 bit ports
- Ports have jumper selectable 3.3V/5 Volt Input/Output

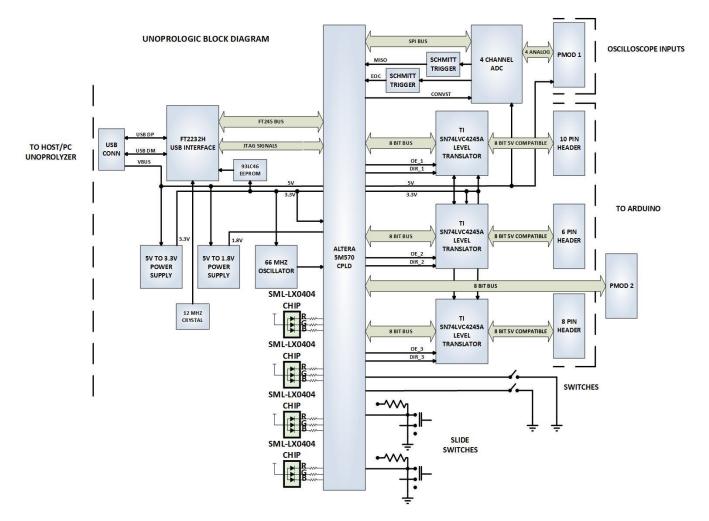


- Four RGB LED's accessible by the user
- Two PCB switches accessible by the user
- Two Slide Switches
- Two PMOD Connectors
- All I/O connectors stack into the Arduino Uno

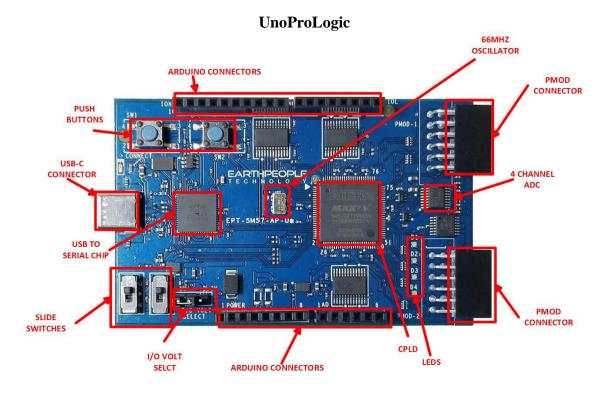


UnoProLogic



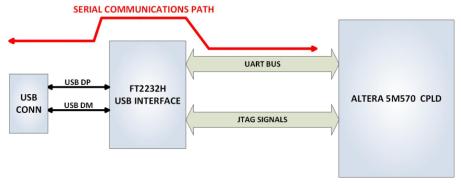






1.2.1 Serial USB Communications

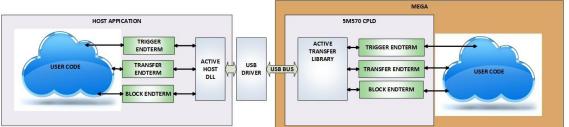
The UnoProLogic CPLD Development system connects an FT2232H Dual High Speed USB (480 Mbits/sec) chip to the CPLD. The CPLD uses a dedicated channel on the FT2232H for serial transfers to the PC. Using the EPT Active Transfer Library, sustained speeds of 0.1 Mbytes/sec can be achieved. The transfers are bi-directional.





The FT2232H chip provides a means of data conversion from USB to serial data and serial 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. Channel B has one dual port 4Kbyte FIFO for transmission from Host PC to the CPLD.

The Serial Communications with the UnoProLogic is handled easily using the Active Host API. 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.

1.2.2 Host PC Connection

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

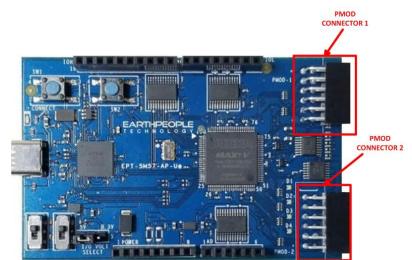


- Power has been applied to the UnoProLogic via USB
- The FT2232HQ chip is working properly
- The Host PC has found the appropriate driver and will communicate with the UnoProLogic



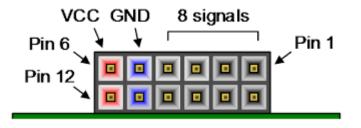
1.2.3 PMOD Connectors

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



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





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

| PMOD | | | | | | | | |
|-------------------------------|------|------|------|------|------|------|------|------|
| I2C Type 1 Type 2 Type 3 Type | | | | | | | | e 4 |
| 101 | 101 | 105 | SS | INT | CTS | INT | CTS | INT |
| 102 | 102 | 106 | MOSI | RST | RTS | RST | ТХ | RST |
| SCL | 103 | 107 | MISO | 107 | RX | 107 | RX | 107 |
| SDA | 104 | 108 | SCK | 108 | ТХ | 108 | RTS | 108 |
| GND | GND | GND | GND | GND | GND | GND | GND | GND |
| 3.3V | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V | 3.3V |

DMOD

The PMOD have the following connections to the MAX V chip:

| PMOD Pin Number | Signal Name | MAX V Pin Number |
|-----------------|-------------|------------------|
| | | |
| 1-1 | PMOD_1_1 | 83 |
| 1-2 | PMOD_1_2 | 81 |
| 1-3 | PMOD_1_3 | 77 |
| 1-4 | PMOD_1_4 | 58 |
| 1-7 | PMOD_1_7 | 76 |
| 1-8 | PMOD_1_8 | 78 |
| 1-9 | PMOD_1_9 | 82 |
| 1-10 | PMOD_1_10 | 84 |
| 2-1 | ADC_AIN0 | NA |
| 2-2 | ADC_AIN1 | NA |
| 2-3 | ADC_AIN2 | NA |
| 2-4 | ADC_AIN3 | NA |



| 2-7 | PMOD_2_7 | 76 |
|------|-----------|----|
| 2-8 | PMOD_2_8 | 78 |
| 2-9 | PMOD_2_9 | 82 |
| 2-10 | PMOD_2_10 | 84 |

1.2.4 Inputs and Outputs

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

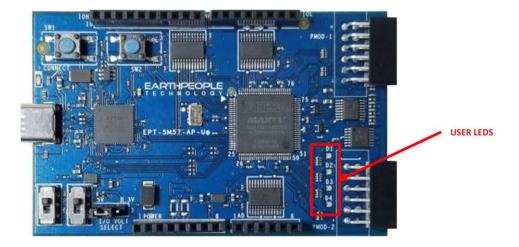


The I/O's are organized as three 8 bit directional ports. Each port must be defined as input or output. This means that all 8 bits of a port will point in the same direction, depending on the direction bit of the transceiver. The direction bit can be changed at any time, so that a port can change from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

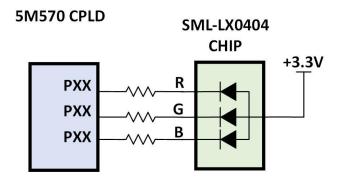
1.2.5 User LEDs

The User LEDs are four RGB LEDs. These LEDs are SML-LX0404 chips and are for use only with +3.3V.





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



Each series resistor uses a 220 Ohm in a resistor array. In order to light up the each LED, the user code must assert a zero on the associated signal for the LED. To turn off the LED, assert High Z on the signal.

The LED RGB signals are organized on the following pins from the MAX V chip:

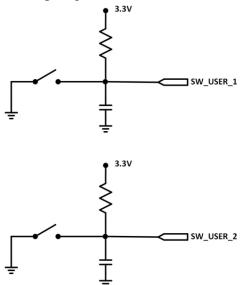
| LED Number | Signal Name | MAX V Pin Number |
|------------|---------------|------------------|
| D1 | LED_GREEN_1_N | 57 |
| D1 | LED_BLUE_1_N | 56 |



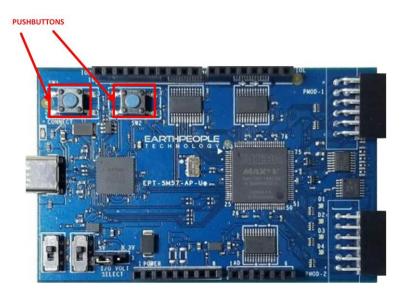
| D1 | LED_RED_1_N | 55 |
|----|---------------|----|
| D2 | LED_GREEN_2_N | 54 |
| D2 | LED _BLUE_2_N | 53 |
| D2 | LED_RED_2_N | 52 |
| D3 | LED_GREEN_3_N | 51 |
| D3 | LED _BLUE_3_N | 50 |
| D3 | LED_RED_3_N | 49 |
| D4 | LED_GREEN_4_N | 48 |
| D4 | LED _BLUE_4_N | 47 |
| D4 | LED_RED_4_N | 43 |

1.2.6 User Push Buttons

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





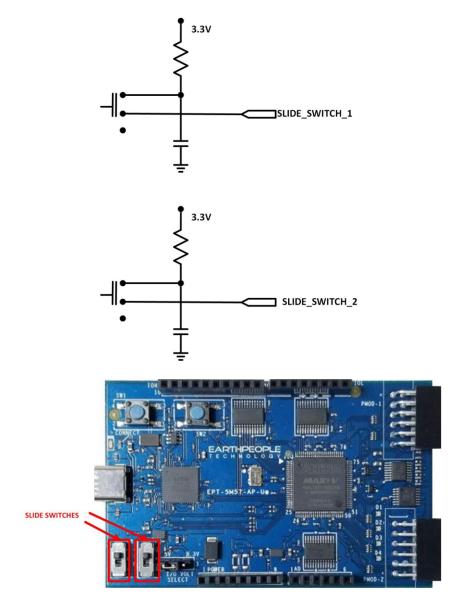


| Component | Net Name | Pin on CPLD | Signal in EPT Project Pinout |
|-----------|-----------|-------------|---------------------------------|
| SW1 | SW_USER_1 | 20 | SW_USER_1 |
| SW2 | SW_USER_2 | 21 | SW_USER_2 |

1.2.7 Slide Switches

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



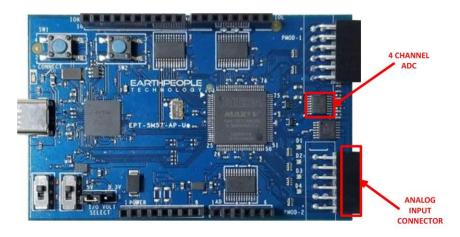


| Component | Net Name | Pin on CPLD | Signal in EPT Project Pinout |
|-----------|----------------|-------------|------------------------------|
| SW3 | SLIDE_SWITCH_1 | 16 | SLIDE_SWITCH_1 |
| SW4 | SLIDE_SWITCH_2 | 17 | SLIDE_SWITCH_2 |



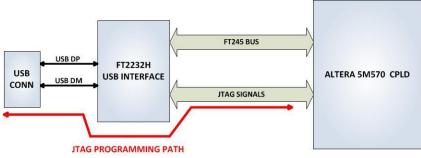
1.2.8 Analog Inputs

The UnoProLogic includes a six pin analog input connector. This connector provides a path from the pins to the input of the four Op-Amp buffers. Each Op-Amp includes a 1MHz low pass filter. Each Op-Amp provides a buffer for the analog signals to the ADC inputs.



1.2.9 JTAG

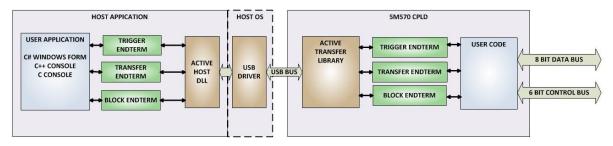
The UnoProLogic uses the second channel of the FT2232H chip as a dedicated CPLD programming port. The CPLD must be programmed via JTAG signals and the FT2232H has built in JTAG signals. The CPLD can be programmed directly from Quartus Prime Lite by using the "jtag_hw_mbftdi_blaster.dll". Just click on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

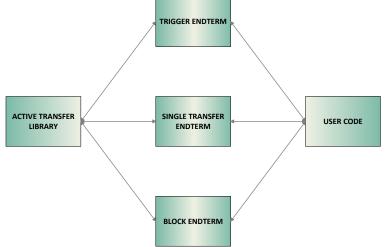
1.4 Active Transfer EndTerms

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

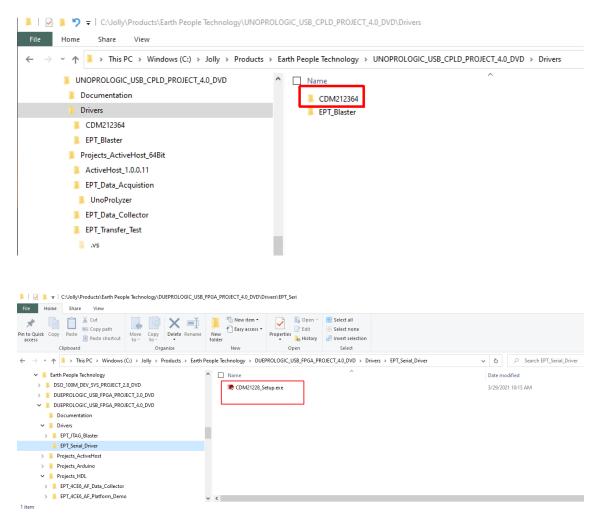
The UnoProLogic Development system requires drivers for any interaction between PC and the board. The communication between the two consists of programming the CPLD and data transfer. In both cases, the USB Driver is required. This will allow Windows to recognize the USB Chip and setup a pathway for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

2.1 USB Driver

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

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





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

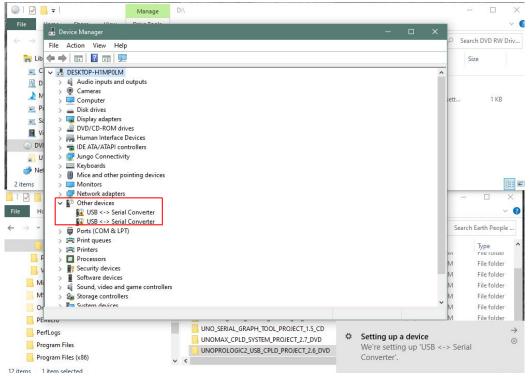
Plug in the UnoProLogic device into an available USB port.





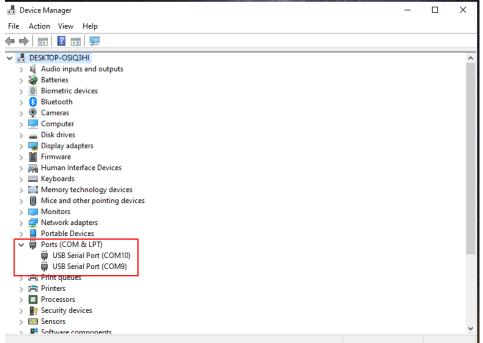
Windows will attempt to locate a driver for the USB device. When it does not find one, it will report a error, "Device driver software was not successfully installed". Ignore this error.

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





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

2.2 JTAG DLL Insert to Quartus Prime Lite

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

2.2.1 Installing Quartus

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



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



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:

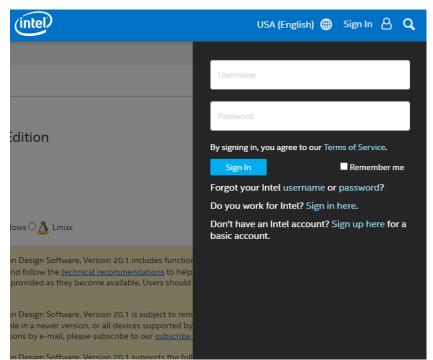
Intel FPGA Quartus Prime Lite

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.

| Products | Solutions | Support | (intel) | USA (English) 🌐 | Sign In | ප | q |
|----------|-----------|--|---|--|---|----------------------------|-------------------------------|
| | Down | load Center fo | or FPGAs | | | | |
| | P | Design Software Embedded Software Archives Licensing rogramming Software Drivers Board System Design Board Layout and Test Lepacy Software | | ntel' Quartus' Prime esign Software | | | |
| | | | The Quartus Prime Lite Edition Design Software, Version keep their software up-to-date and follow the <u>technical reco</u>updates are planned and will be provided as they become avrelease. The Quartus Prime Lite Edition Design Software, Version devices in this release are available in a newer version, or all like to receive customer notifications by e-mail, please subsci | immendations to help improve security. A vallable. Users should promptly install the 20.1 is subject to removal from the web u devices supported by this version are ob | dditional s a latest ver when supp solete. If ye | securi sion u ort fo | ity upon or all ould |

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?"





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.



| | (inte | | | USA (Englis | h) 🌐 M |
|--|--|---|--|---|-------------|
| 10 LP, Cyclone IV, C | clone V, MAX II, MA) | X V, and MAX 10 FPGA. | <u>→ More</u> | | |
| Combined Files | Individual Files | Additional Software | | | |
| Combined Files | Individual Files | Additional Software | | | |
| | nstall instructions | | | | |
| | Software v20.1 Insta | allation FAQ | | | |
| Quick Start Guid | 2 | | | | |
| software compo <i>Included?</i>) to th To achieve a sm | nents. A list of files ne right of the descr aller download and | the Quartus Prime Desi i included in each down ription. The Complete I I installation footprint, y onents in the Individual | load can be viewed i Download includes a You can select device | n the tool tip (<i>What's</i> Il available device fam support in the Multir | nilies. |
| | e Lite Edition Software | ware (Device support i dows.tar | ncluded) What's Includ | led? | 0 |
| ** Nios II ED: requires a | 5 on Windows requires manual installation. | E85C9A1DBAF2F24A7(5 Ubuntu 18.04 LTS on Wind II an Eclipse IDE manually. | | x (WSL), which | |
| on the down | | is a full-featured EDA pr | oduct. Depending of | | nglish) 🧲 |
| Quick Start Guide | | | | | |
| | | the Quartus Prime De | sign Software inclu | des a number of ado | |
| software compon Included?) to the To achieve a smal | right of the descr ler download and | included in each dow ription. The Complete installation footprint, onents in the Individu | nload can be viewe Download include you can select dev | ice support in the M | e families. |
| software compon Included?) to the To achieve a smai Download sectio Quartus Prime Quartus-lit Size: 5.9 GB | right of the descr ler download and n, or select compo Lite Edition Softw 2-20.1.0.711-wind MD5: C0C684441 | ription. The Complete installation footprint, onents in the Individu ware (Device support dows.tar E85C9A1DBAF2F24A ² | The second secon | s all available device ice support in the M onal Software tabs. cluded? | e families. |
| software compon Included?) to the To achieve a smal Download sectio Quartus Prime Quartus-lit Size: 5.9 GB ** Nios II EDS requires a n | right of the descr ler download and n, or select compo Lite Edition Softw 2-20.1.0.711-wind MD5: COC684441 on Windows requires hanual installation. | ription. The Complete installation footprint, onents in the Individu ware (Device support dows.tar | The second secon | s all available device ice support in the M onal Software tabs. cluded? | e families. |



. .

. .

...

| Products Solutions Support | (intel) |
|----------------------------|---|
| | The Combined Files download for the Quartus software components. A list of files included in <i>Included?</i>) to the right of the description. The To achieve a smaller download and installation Download section, or select components in the |
| | Quartus Prime Lite Edition Software (Devic |
| | Quartus-lite-20.1.0.711-windows.tar Size: 5.9 GB MD5: C0C68444E85C9A1DB ** Nios II EDS on Windows requires Ubuntu 18.04 requires a manual installation. ** Nios II EDS requires you to install an Eclipse IDI |
| | Note: The Quartus Prime software is a full-featu download times may be lengthy. |
| | ► System Requirements |
| | Documentation Links |
| | |

The file is 5.9 GB, so this could take a couple of hours depending on your internet connection. When download is complete, store the *.tar file in a directory on your PC.



| 📙 🔄 🗖 🔻 C:\Jolly\Downloads\Altera_Intel_FPGA\Quartus Prime 20.1 | |
|---|--|
| File Home Share View | |
| ← → × ↑ 🔄 > This PC > Local Disk (C:) > Jolly > Downloads > | Altera_Intel_FPGA > Quartus Prime 20.1 |
| Code_FPGA | Name Date modified |
| | |
| | |
| Allegro | |
| Altera_Intel_FPGA | |
| 📙 Quartus Prime 16 | ▲ ▲ |
| Quartus Prime 17.1 | |
| Quartus Prime 18.1 | |
| Quartus Prime 20.1 | |
| AtmosFX | |
| 🔄 Camtasia | |
| FTDI | |
| HyperSerialPort | |
| 💺 🛃 🔜 🗢 C:\Users\nelso\Downloads | |
| File Home Share View | |
| ← → × ↑ 🕂 > This PC > Local Disk (C:) > Users > nelso > Downlo | loads |
| 💻 This PC | ^ Name |
| 🧊 3D Objects | ✓ Today (1) |
| E. Desktop | Quartus-lite-20.1.0.711-windows.tar |
| Documents | Earlier this week (1) |
| 🖶 Downloads | Label-501986027.pdf |
| b Music | ✓ Last week (1) |
| E Pictures | Label-501432798.pdf |
| 🚪 Videos | v (last month (19) |
| Local Disk (C:) | ✓ Last month (18) |

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



| 💐 🔡 💕 🖡 🔻 Quartus-lite | e-20.1.0.711-w | indows - WinZip Evalu | ation Version - | 21 days left | | - 🗆 X |
|--|----------------|--------------------------------|-----------------|---|---|-------------------------------------|
| File Unzip/Share Edit | Backup | Tools Settings | View H | lelp Buy Now | | ~ 3 |
| | BUY | | | our trial runs out! / from our secure online | shop. CNET editors rating | 30 DAY MINISTRAT |
| Files + + Files | > Qu | artus-lite- | 20.1.0. | 711-windo | ws.tar | Actions > Unzip All Files |
| Recent Zip Files Quartus-lite-20.1.0.7tar Quartus Prime 20.1 | | components Type: Folder | | Da | ate modified: 6/6/2020 6:33 PM | Unzip to: ~\Quartus-lite-20.1 |
| Browse & Manage Files | | readme.txt Type: Readme Doc | ument | | ate modified: 6/6/2020 7:09 PM ze: 8.51 KB 🔶 8.51 KB | Convert & Protect Files |
| Frequent Folders | | setup.bat Type: Windows Bat | tch File | | ate modified: 6/6/2020 7:09 PM ze: 1.07 KB → 1.07 KB | When adding files to this Zip file: |
| This PC 648 GB free of 930 GB | | | | | | Reduce Photos Off |
| Network | | | | | | Remove Info Off |
| Shared Files | | | | | | Convert to PDF Off |
| Add Cloud | | | | | | Combine PDFs Off |
| | 🗌 3 i | tem(s) | | | Zip File: 11 item(s), 5.90 (| Save or Share Zip |

The tool will unpack all files.

| WinZip | | | × |
|---------------------------------------|------------|---|---|
| Extracting ModelSimSetup-20.1.0.711-v | vindows ex | e | |
| Excluding models insetup 201101111 | indowsiex. | - | |
| | | | |
| | | | |
| Cancel | | | |

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_lı | ntel_FPGA\Quartus Prime 20.1 | | | | |
|--|-------------------------------------|--------------------------|-------------------------|---|--------------------|
| File Home Share View | | | | | |
| Pin to Quick Copy ath access | Move Copy to * Cot Delete Rename | New item • | Properties ▼ History | Select all Select none Invert selection | |
| Clipboard | Organize | New | Open | Select | |
| ← → × ↑ 📙 > This PC > Local Disk | (C:) > Jolly > Downloads > Alte | era_Intel_FPGA → Quartus | Prime 20.1 | | |
| Quartus Prime 16 | | ^ Name | ^ | Date modified | Туре |
| Quartus Prime 17.1 | | components | | 8/8/2020 12:58 PM | File folder |
| Quartus Prime 18.1 | | Quartus-lite-20 | .1.0.711-windows.tar | 8/8/2020 12:25 PM | WinZip File |
| > 📙 Quartus Prime 20.1 | | readme.txt | | 6/6/2020 7:09 PM | Text Document |
| > AtmosFX | | 💿 setup.bat | | 6/6/2020 7:09 PM | Windows Batch File |
| > Camtasia | | | | | |
| > FTDI | | | | | |
| > HyperSerialPort | | | | | |
| | | | | | |

Click "Next" on the Introduction Window.

| 🕥 Installing Quartus Prime L | ite Edition (Free) 20.1.0.711 — 🗆 🗙 |
|------------------------------|---|
| | Setup - Quartus Prime Lite Edition (Free) 20.1.0.711 |
| inter | 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. |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | < Back Next > Cancel |



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

| 🛜 Installing Quartus Prime Lite Edition (Free) 20.1.0.711 — — | | × |
|--|-----------|----|
| License Agreement | (int | el |
| You can view the full license agreement at the link below or useinstall_lic option from command-line to get the license files before the installation. You must accept the terms of the agreement before continuing with the installation. http://fpqasoftware.intel.com/eula/ | agreement | |
| 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 | | ^ |
| < | > | |
| O you accept this license? I do not accept the agreement I do not accept the agreement | | |
| InstallBuilder | Cance | el |

Click "Next" and accept the defaults.

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



| 📚 Installing Quartus Prime Lite Edition (Free) 20.1.0.711 | - | |
|--|---|---------|
| Select Components | | (intel) |
| 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 11/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) | |
| InstallBuilder — | < Back Next > | Cancel |

Click "Next" to accept the defaults



| S Installing Quartus Prime Lite Edition (Free) 20.1.0.711 | — | o x | |
|--|--------|--------|---|
| Ready to Install | | (intel |) |
| Summary: Installation directory: C:\intelFPGA_lite\20.1 Required disk space: 16760 MB Available disk space: 657864 MB | | | |
| | lext > | Cancel | |

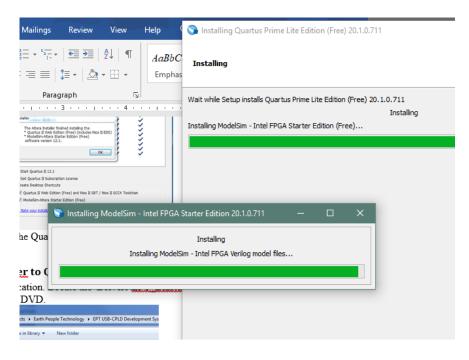
Click "Next" to accept the defaults



| 🕞 Installing Quartus Prime Lite Edition (Free) 20.1.0.711 | | - | |
|--|--------|--------|---------|
| Installing | | | (intel) |
| Wait while Setup installs Quartus Prime Lite Edition (Free) 20.1.0.711 Installing | | | |
| Unpacking files | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| InstallBuilder | | | |
| | < Back | Next > | Cancel |

Wait for the installation to complete.







| 🕤 Installing Quartus Prime L | ite Edition (Free) 20.1.0.711 | - | | × |
|------------------------------|---|------|------|----|
| | Quartus Prime Lite Edition (Free) 20.1.0.711 Installation Complete | | | |
| intel | Setup has finished installing Quartus Prime Lite Edition (Free) 20.1.0.711. Launch USB Blaster II driver installation Create shortcuts on Desktop Launch Quartus Prime Lite Edition Provide your feedback | | | |
| | < Back Fin | iish | Cano | el |

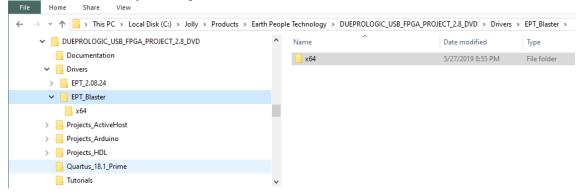
Click "Ok", then click "Finish". The Quartus Prime is now installed and ready to be used.



| 🕞 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 |
| O Buy a Quartus Prime software license |
| O Run the Quartus Prime software |
| O Add an IP license file (for users who have purchased IP) |
| OK Cancel |

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.



Follow these directions:

- 1. Open the C:\:\..\UNOPROLOGIC_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.

| le Home Share View | | | |
|-------------------------------------|---|------------------------|------------------|
| → 👻 🛉 > This PC → Local Disk (C:) → | intelFPGA_lite > 20.1 > quartus > bin64 | | |
| ✓ 20.1 | ^ Name ^ | Date modified | Туре |
| devdata | jam2_api.jam | 6/5/2020 2:55 PM | JAM File |
| > 📙 ip | 📄 jam2_ic.jam | 6/5/2020 2:55 PM | JAM File |
| > 📙 licenses | 🧟 jtag_atlantic.dll | 6/5/2020 3:43 PM | Application exte |
| logs | 🚳 jtag_client.dll | 6/5/2020 3:43 PM | Application exte |
| > modelsim ase | jtag_hw_mbftdi_blaster64.c | dll 10/26/2015 3:01 PM | Application exte |
| > nios2eds | 🚳 jtag_hw_pli-blaster.dll | 6/5/2020 3:45 PM | Application exte |
| | jtag_hw_usb-blaster.dll | 6/5/2020 3:45 PM | Application exte |
| V quartus | 🗟 jtag_pli-blaster_vpi.dll | 6/5/2020 3:45 PM | Application exte |
| > 📙 bin64 | 🔳 jtagconfig.exe | 6/5/2020 3:44 PM | Application |
| > 📙 common | itagserver.exe | 6/5/2020 3:44 PM | Application |
| > drivers | V legality_lab.dll | 6/5/2020 3:39 PM | Application exte |

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.



| Check Email G Google YouTube YouTube Google YouTube YouTube YouTube Societaria Societaria |
|---|
| 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. |
| Click on the "Express 20xx for Windows Desktop" hypertext. $\leftarrow \rightarrow \circlearrowright \ \widehat{\omega} \ \stackrel{\wedge}{\cong} \ ^{\text{https://visualstudio.microsoft.com/vs/express/}$ |
| 🗋 Check Email 💪 Google 💶 YouTube 💆 Yahoo 💡 Google Maps 🛅 EPT 🎦 Banks 🚞 Electronics 🛅 News 🚞 Music 🛅 Pay 🚞 Markets 🛅 Reference |
| Help me choose. I am a |
| Choose from the options below to see what version of Visual Studio is right for you |
| I am a 🗸 |
| |
| |
| |

Still want Visual Studio Express?

Express 2017 for Windows Desktop

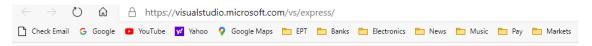
Supports building managed and native desktop applications.*

Express 2015 for Windows Desktop

Supports the creation of desktop applications for Windows.

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





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

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 | Open | |
|---|-------------------------|--------------------------------------|---|
| | Create s | Always open files of this type | sites, web APIs, or real-time online |
| | Expre Provide | Show in folder Copy download link |) pelling, innovative apps for Univers |
| M | vs_WDExpress (2).4 | Cancel | |

| | Х | |
|--|---|--|
| 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. | | |
| <u>C</u> ontinue | | |

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

Click the "Continue" button.





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





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

To use the Active Host Application Software, the Active Host DLL and the ftd2xx DLL must be included in the Microsoft Visual project. The Active Host Application Software will allow the user to create a custom applications on the PC using the EndTerms to perform Triggers and Data Transfer to/from the UnoProLogic. The methods and parameters of the Active Host DLL are explained in the Active Host Application section. Locate the \Projects_ActiveHost_64Bit and \Projects_ActiveHost_32Bit folders on the UnoProLogic Development System CD.

| nize 🔻 New folder | | | | | 800 . | |
|--|---|---------------------------|-----|--------------------|-------------|---|
| 4 📕 Earth People Technology | * | Name | | Date modified | Type | |
| EPT USB-CPLD Development System Arduino_IDE Documentation Divers Projects_ActiveHost_32Bit | | 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_ActiveHost_64Bit | | Projects_Arduino | | 2/12/2013 8:23 AM | File folder | |
| Projects_Arduino Projects_HDI | | Projects_HDL | | 2/12/2013 8:23 AM | File folder | |
| | | \mu Quartus_Programmer | | 2/12/2013 8:23 AM | File folder | |
| Juartus_Programmer | | | 111 | | | |
| File name: RedBoard-v06.zip | | | | | | |
| Save as type: ALZip ZIP File (*.zip) | | | | | | _ |



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

| ganize 👻 Include in library 👻 Share with | - | Burn New folder | | | | 855 | • | |
|--|---|----------------------------|-------------------|-------------|------|-----|---|--|
| 4 퉬 Earth People Technology | * | Name | Date modified | Туре | Size | | | |
| 4 퉬 EPT USB-CPLD Development System | | ActiveHost_1.0.0.8 | 2/12/2013 8:22 AM | File folder | | | | |
| Arduino_IDE | | EPT_570_AP_Data_Collector | 2/12/2013 8:23 AM | File folder | | | | |
| Documentation | | EPT_Transfer_Test | 2/12/2013 8:23 AM | File folder | | | | |
| Drivers Projects ActiveHost 32Bit | | EPTActiveHostConsoleClient | 2/12/2013 8:23 AM | File folder | | | | |
| Projects_ActiveHost_64Bit | | | | | | | | |
| Projects_Arduino | | | | | | | | |
| Projects_HDL | | | | | | | | |
| Quartus_Programmer | - | | | | | | | |

 $\label{eq:locate-basic} Locate the Projects_ActiveHost_64Bit \ActiveHost_1.0.0.8\Bin folder and copy the ActiveHost64.dll and the ftd2xx64.dll.$

| ganize 🔻 🛛 Include in library 👻 🦷 Share with ୟ | Burn New folder | | | 100 - | |
|---|------------------|-------------------|--------------------|--------|--|
| 4 🎉 Earth People Technology | Name | Date modified | Туре | Size | |
| EPT USB-CPLD Development System | ActiveHost64.dll | 2/26/2013 7:20 PM | Application extens | 27 KB | |
| Arduino_IDE Documentation | itd2xx64.dll | 1/18/2013 3:54 PM | Application extens | 252 KB | |
| Drivers Projects_ActiveHost_32Bit | | | | | |
| Projects_ActiveHost_64Bit ActiveHost_1.0.0.8 | | | | | |
| 🍌 Bin | | | | | |
| EPT_570_AP_Data_Collector | | | | | |
| EPT_Transfer_Test | | | | | |
| EPTActiveHostConsoleClient | | | | | |
| Projects Arduino | | | | | |

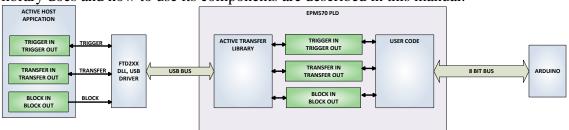
Save the DLL's in the bin\x64\Release folder of the user project under the Microsoft C# Express project. See the Active Host Application section of the UnoProLogic Development System User Manuals for instructions on how to add the dll to the Microsoft C# Express project.



| nize 🕶 🔳 Open with Burn | New f | older | | | 855 | - |
|---|-------|---------------------------------------|--------------------|--------------------|--------|---|
| 4 🎉 Projects_ActiveHost_64Bit | * | Name | Date modified | Туре | Size | |
| ActiveHost_1.0.0.8 EPT_570_AP_Data_Collector | | 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 EPT_Transfer_Test | | 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 | |
| 4 🍑 bin | E | EPT_Transfer_Test.vshost.exe.manifest | 8/31/2009 12:40 AM | MANIFEST File | 1 KB | |
| bebug | | S ftd2xx64.dll | 1/18/2013 3:54 PM | Application extens | 252 KB | |
| 🎍 Release 4 🍱 x64 | | | | | | |
| A 🙀 x04 | | | | | | |
| Release | | | | | | |
| Neledse | | | | | | |
| Properties | | | | | | |

3 Active Transfer Library

The Active Transfer Library is an HDL library designed to transfer data to and from the UnoProLogic 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



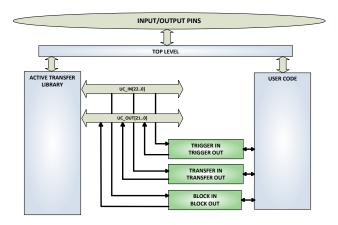
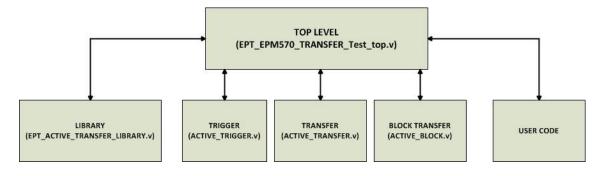


Figure 6 shows how the modules of the EPT Active Transfer Library attach to the overall user project. The EPT Active_Transfer_Library.vqm, Active_Trigger.v, Active_Transfer.v and Active_Block.v modules are instantiated in the top level of the user project. The User_Code.v module is also instantiated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







| 114 - | | | | |
|---------------------------------------|--|---|---|---|
| 1/# 0 | Copyright Earth | People Techr | nology Inc. 2012 | |
| //# | | | | |
| //# | | | | |
| | File Name: EPT_FT: | 2232_Transfe | r_Test_top.v | |
| //# | Service Wieberry | | | |
| | Revision History: DATE | VEDSTON | DETATIS | |
| //# | | | Created | RJJ |
| 1/# | | | | |
| //# | | | | |
| //# | | | | |
| L//### | | ********** | ************ | **************** |
| E'ifde | | | | |
| | include "/src/de: | | | |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | include "/Testber | nch/tb_defin | le.V" | |
| - `endi | lf | | | |
| · · · · · | escale ins/ips | | | |
| Cime | ascale ins/ips | | | |
| | | | | |
| | | | | |
| E//*** | | ********** | ************* | **************** |
| 1/* 1 | Nodule Declaration | | | |
| -1/*** | | | | ***************** |
| | | | | |
| modul | le ept_EPM570_Trans | sfer_Test_to | pp (| |
| 10000 | | | | |
| | | | | |
| | | | | |
| | input wire [1:0] | | | |
| | input wire [1:0] input wire [1:0] | | in, | |
| | | Ъс_ • • | in, | |
| | input wire [1:0] | Ъс • • | in, | |
| | input wire [1:0] // | bc • • • • • | in, | |
| | input wire [1:0] | bc • • • • • | in, rary | |
| | input wire [1:0] // // Instantiate t | bc • • • • | in, rary | |
| | input wire [1:0] // // Instantiate t // active_transfer | bc • • • • | in, rary | |
| | input wire [1:0] // // Instantiate t // active_transfer (| bc • • • • | in, rary EPT_LIB | |
| | input wire [1:0] // // Instantiate t // active_transfer (.aa | bc • • • • | _in, rary EPT_LIB (aa), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in</pre> | bc • • • • | _in, rary EPT_LIB (aa), (bc_in), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out</pre> | bc • • • • | rary EPT_LIB (aa), (bc_in), (bc_out), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in</pre> | bc • • • • | _in, rary EPT_LIB (aa), (bc_in), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout</pre> | bc • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN</pre> | bc • • • • | _in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout</pre> | bc • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), | |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT</pre> | bc • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), | RARY_TOP_INST |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1</pre> | bc • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (data_byte_r | RARY_TOP_INST |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TESI_SIGNAL_1 .STATE_OUT</pre> | bc • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (data_byte_r (ft_245_stat) | <pre>eady), e_machine),</pre> |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS</pre> | bc • • • • • • • • • • • • • • • • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de | <pre>eady), e_machine), code),</pre> |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE</pre> | bc • • • • • • • • • • • • • • • • • • • | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_re | <pre>eady), e_machine), code), gisters_state),</pre> |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE .ENDPOINT_TEST_</pre> | bc • • • • • • • • • • • • • • • • • • • | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_re | <pre>eady), e_machine), code),</pre> |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE</pre> | bc • • • • • • • • • • • • • • • • • • • | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_re | <pre>eady), e_machine), code), gisters_state),</pre> |
| | <pre>input wire [1:0] // // Instantiate t // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE .ENDPOINT_TEST_);</pre> | bc • • • • • • • • • • • • • • • • • • • | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_IN), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_re | <pre>eady), e_machine), code), gisters_state),</pre> |
| | <pre>input wire [1:0] // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE .ENDPOINT_TEST_); //</pre> | bc • • • • • • • • • • • • • • • • • • • | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_wr | <pre>eady), e_machine), code), gisters_state),</pre> |
| | <pre>input wire [1:0] // Instantiate t .explosions.com .ex</pre> | bc • • • • • • • • • • • • • | in, rary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_wr ules | <pre>eady), e_machine), code), gisters_state), ite_to_host)</pre> |
| | <pre>input wire [1:0] // active_transfer (.aa .bc_in .bc_out .bd_inout .UC_IN .UC_OUT .TEST_SIGNAL_1 .STATE_OUT .TEST_BUS .ENDPOINT_STATE .ENDPOINT_TEST_); //</pre> | bc ehe EPT Lib c_library f_OUT BUS | in, Tary EPT_LIB (aa), (bc_in), (bc_out), (bd_inout), (UC_OUT), (UC_OUT), (data_byte_r (ft_245_stat (register_de (endpoint_wr | <pre>eady), e_machine), code), gisters_state), ite_to_host)</pre> |



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.

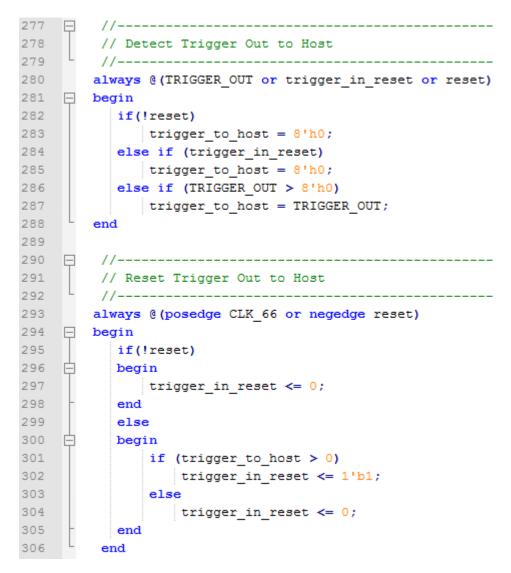
3.2.1 Active Trigger EndTerm

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.

```
743
       wire [22*3-1:0] uc out m;
744
       eptWireOR # (.N(3)) wireOR (UC OUT, uc out m);
745
           active trigger
                                       ACTIVE TRIGGER INST
746 🖃
           (
747
            .uc clk
                                       (CLK 66),
748
            .uc reset
                                       (RST),
749
            .uc in
                                       (UC IN),
            .uc out
750
                                       (uc out m[ 0*22 +: 22 ]),
751
752
            .trigger_to_host
                                       (trigger_to_host),
753
            .trigger to device
                                       (trigger in byte)
754
755
           );
756
```

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.





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.



//-----308 🖵 309 // Detect Trigger In 310 //-----_____ 311 always @(trigger in byte or trigger in reset or reset) 312 📃 begin 313 if(!reset) È 314 begin 315 trigger in detect = 1'b0; 316 end 317 else if (trigger in reset) 318 🚊 begin 319 trigger in detect = 1'b0; 320 end 321 else if (trigger in byte > 8'h0) 322 😑 begin 323 trigger in detect = 1'b1; 324 end 325 end 326 //-----327 328 // Store the value of Trigger In //-----329 330 always @ (posedge CLK 66 or negedge reset) 331 begin 332 if(!reset) 333 白 begin 334 trigger in store <= 8'h0f;</pre> 335 trigger in reg <= 1'b0;</pre> 336 trigger in reset <= 1'b0;</pre> 337 end 338 else if (trigger in detect & !trigger in reg) 339 📋 begin 340 if(trigger in byte != 0) 341 trigger in store[7:0] <= trigger in byte[7:0];</pre> trigger_in_reg <= 1'b1;</pre> 342 343 end 344 else if (trigger in reg) 345 😑 begin 346 trigger in reg <= 1'b0;</pre> 347 trigger in reset <= 1'b1;</pre> 348 end 349 else if (!trigger in detect) 350 Ē begin 351 trigger in reg <= 1'b0;</pre> 352 trigger in reset <= 1'b0;</pre> 353 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.

```
757
                                      ACTIVE TRANSFER INST
           active_transfer
758
    (
759
           .uc clk
                                      (CLK 66),
760
            .uc reset
                                      (reset).
761
            .uc in
                                      (UC IN),
762
            .uc out
                                       (uc_out_m[ 1*22 +: 22 ]),
763
            .start_transfer
764
                                      (transfer out reg),
765
            .transfer received
                                       (transfer in received),
766
767
            .uc addr
                                       (3'h2),
768
            .transfer_to_host
.transfer_to_device
769
                                      (transfer out byte),
770
                                       (transfer_in_byte)
771
           ):
772
```

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.



| 181 | | // |
|-----|----|---|
| 182 | Ч | // Transfer byte to Device |
| 183 | L | // |
| 184 | | always @(TRANSFER OUT EN or reset) |
| 185 | | begin |
| 186 | Ч | if(!reset) |
| 187 | L. | begin |
| 188 | Ч | transfer out detect = 1'b0; |
| 189 | | end |
| 190 | | else |
| 191 | L | begin |
| 192 | Ч | if(transfer to device reset) |
| 193 | | transfer out detect = 1'b0; |
| 194 | | else if(TRANSFER OUT EN) |
| 195 | H | begin |
| 196 | Т | transfer out byte = TRANSFER OUT BYTE; |
| 197 | | transfer out detect = 1'b1; |
| 198 | | end |
| 199 | _ | end |
| 200 | L | end |
| 201 | | |
| 202 | | // |
| 203 | Τ | // Reset transfer to device reset |
| 204 | L | // |
| 205 | | always @(posedge CLK 66 or negedge reset) |
| 206 | | begin |
| 207 | Т | if (!reset) |
| 208 | Ē. | begin |
| 209 | | <pre>transfer to device reset <= 1'b0;</pre> |
| 210 | - | end |
| 211 | | else |
| 212 | ¢ | begin |
| 213 | | <pre>if(transfer_out_detect)</pre> |
| 214 | | <pre>transfer_to_device_reset <= 1'b1;</pre> |
| 215 | | else |
| 216 | | <pre>transfer_to_device_reset <= 1'b0;</pre> |
| 217 | - | end |
| 218 | L | end |
| 010 | | |



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.

| 220 | | // |
|-----|---|---|
| 221 | T | // Transfer to Host |
| 222 | L | // |
| 223 | | always @(posedge CLK 66 or negedge reset) |
| 224 | | begin |
| 225 | T | if (!reset) |
| 226 | | begin |
| 227 | T | <pre>transfer out <= 1'b0;</pre> |
| 228 | | transfer out reg <= 1'b0; |
| 229 | | <pre>transfer out byte <= 8'h0;</pre> |
| 230 | - | end |
| 231 | | else |
| 232 | þ | begin |
| 233 | | <pre>if(start_transfer_byte & !transfer_out)</pre> |
| 234 | 白 | begin |
| 235 | | <pre>transfer_out_byte <= TRANSFER_HOST_BYTE;</pre> |
| 236 | | <pre>transfer_out_reg <= 1'b1;</pre> |
| 237 | | <pre>transfer_out <= 1'b1;</pre> |
| 238 | F | end |
| 239 | | <pre>else if(start_transfer_byte & transfer_out)</pre> |
| 240 | 白 | begin |
| 241 | | <pre>transfer_out_reg <= 1'b0;</pre> |
| 242 | | <pre>transfer_out <= 1'b1;</pre> |
| 243 | - | end |
| 244 | | <pre>else if(!start_transfer_byte & transfer_out)</pre> |
| 245 | 白 | begin |
| 246 | | <pre>transfer_out_reg <= 1'b0;</pre> |
| 247 | | <pre>transfer_out <= 1'b0;</pre> |
| 248 | F | end |
| 249 | F | end |
| 250 | L | 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.

```
811
           active block
                                       BLOCK TRANSFER INST
812
     (
                                       (CLK_66),
813
            .uc clk
814
            .uc reset
                                       (RST),
815
            .uc in
                                       (UC IN),
816
            .uc out
                                       (uc out m[ 2*22 +: 22 ]),
817
                                      (block_out_reg),
818
            .start transfer
819
            .transfer received
                                       (block in rcv),
820
821
            .transfer ready
                                       (block byte ready),
822
823
            .uc addr
                                       (3'h4),
824
            .uc length
                                       (BLOCK COUNT 8),
825
826
            .transfer to host
                                      (block out byte),
827
            .transfer to device
                                       (block in data),
828
829
           .STATE OUT
                                       (block state out),
830
           .TEST BUS
                                       (block out test bus)
831
832
           );
833
```

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 transferred to theUSB chip. This process continues until the number of bytes desicribed by the uc_length have been transferred into the USB chip.



| 542 | | // | | | | | |
|-----|--------|---|--|--|--|--|--|
| 543 | T | | | | | | |
| | | <pre>// Registers to start Block Transfer Out //</pre> | | | | | |
| 544 | | // | | | | | |
| 545 | _ | always @(posedge CLK_66 or negedge RST) | | | | | |
| 546 | F | begin | | | | | |
| 547 | | if(!RST) | | | | | |
| 548 | Ē | begin | | | | | |
| 549 | | <pre>block_out_reg <= 1'b0;</pre> | | | | | |
| 550 | | <pre>start_block_transfer_reg <= 1'b0;</pre> | | | | | |
| 551 | - | end | | | | | |
| 552 | | else | | | | | |
| 553 | ¢ | begin | | | | | |
| 554 | | <pre>if(start_block_transfer & !start_block_transfer_reg)</pre> | | | | | |
| 555 | | <pre>start_block_transfer_reg <= 1'b1;</pre> | | | | | |
| 556 | | <pre>else if(start_block_transfer_reg & !block_out_reg)</pre> | | | | | |
| 557 | ¢ | begin | | | | | |
| 558 | | <pre>block out reg <= 1'b1;</pre> | | | | | |
| 559 | - | end | | | | | |
| 560 | | else if(block out counter >= BLOCK COUNT 8) | | | | | |
| 561 | L L | begin | | | | | |
| 562 | Т | <pre>block out reg <= 1'b0;</pre> | | | | | |
| 563 | | <pre>start block transfer reg <= 1'b0;</pre> | | | | | |
| 564 | - | end | | | | | |
| 565 | - | end | | | | | |
| 566 | L | end | | | | | |
| 567 | | | | | | | |
| 568 | | // | | | | | |
| 569 | T | // Data for Block Transfer Out | | | | | |
| 570 | L | // Data for block framster out | | | | | |
| 571 | | always @ (posedge CLK 66 or negedge RST) | | | | | |
| 572 | | begin | | | | | |
| 573 | Т | if(!RST) | | | | | |
| 574 | | begin | | | | | |
| 575 | T. | block out counter <= 0; | | | | | |
| 576 | | end | | | | | |
| 577 | | else | | | | | |
| 578 | H | begin | | | | | |
| 579 | T | if(block_byte_ready) | | | | | |
| 580 | | | | | | | |
| 581 | T | <pre>begin block out counter <= block out counter + 1'd1;</pre> | | | | | |
| 581 | | | | | | | |
| 583 | | end | | | | | |
| 584 | Ц | <pre>else if(block_out_counter >= BLOCK_COUNT_8) begin</pre> | | | | | |
| | T | begin | | | | | |
| 585 | | <pre>block_out_counter <= 0; oud</pre> | | | | | |
| 586 | Γ | end | | | | | |
| 587 | Γ | end | | | | | |
| 588 | - | end | | | | | |
| | | _ | | | | | |



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

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

3.3 Timing Diagram for Active Transfer EndTerms

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

3.3.1 Active Trigger EndTerm Timing

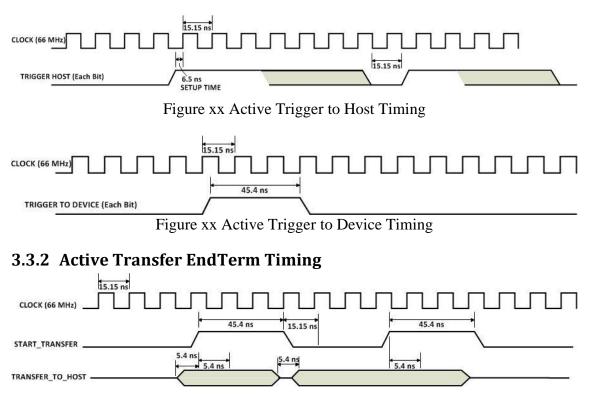


Figure xx Active Transfer To Host Timing



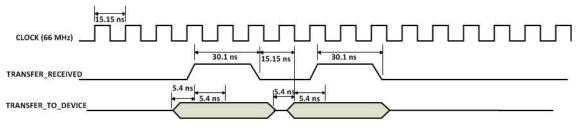


Figure xx Active Transfer To Device Timing

3.3.3 Active Block EndTerm Timing

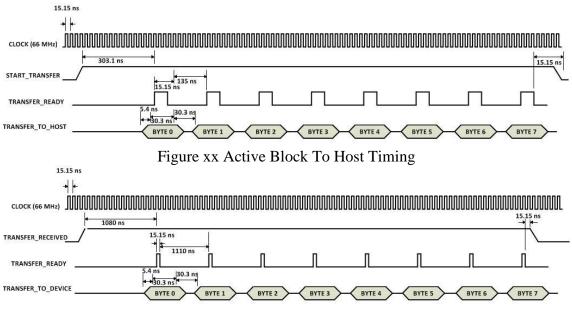
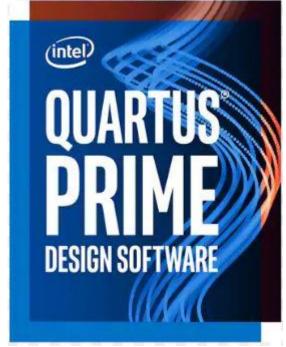


Figure xx Active Block To Device Timing



4 Compiling, Synthesizing, and Programming CPLD



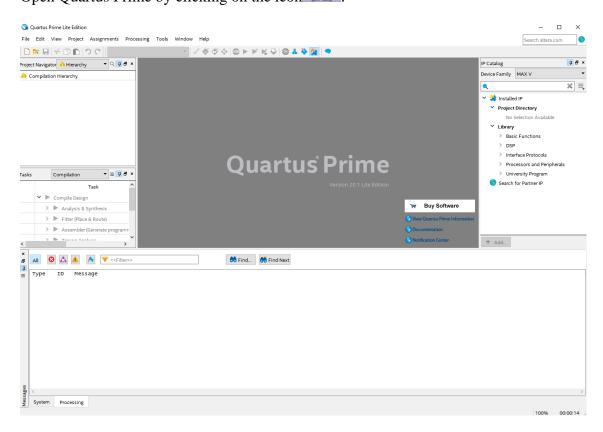
The CPLD on the UnoProLogic-U2 can be programmed with the Active Transfer Library and custom HDL code created by the user. Programming the CPLD requires the use of the Quartus Prime software and a standard USB cable. There are no extra parts to buy, just plug in the USB cable. Once the user HDL code 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".

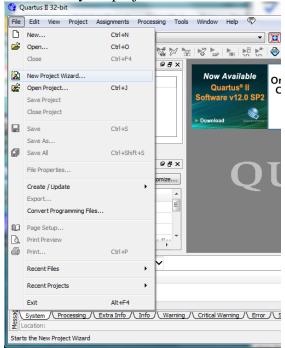


| Organize - 🤤 Open Include in library - Share with | | | | 100 - | |
|---|--|-------------------|-------------|-------|--|
| | * Name | Date modified | Туре | Size | |
| 🕌 bin64 | EPT_Transfer_Test | 3/13/2013 8:52 PM | File folder | | |
| 🕌 common | 🎍 fir_filter | 3/3/2013 1:38 AM | File folder | | |
| 📕 cusp | incr_comp_makefile | 3/3/2013 1:38 AM | File folder | | |
| bildrivers | whol_verilog_tutorial | 3/3/2013 1:38 AM | File folder | | |
| eda | | | | | |
| extibs32 | | | | | |
| libraries | | | | | |
| 📕 Imf | | | | | |
| \mu qdesigns | | | | | |
| EPT_Transfer_Test | | | | | |
| 🍑 fir_filter | | | | | |
| incr_comp_makefile | | | | | |
| whdl_verilog_tutorial | | | | | |
| sopc_builder | | | | | |
| altera_inspector.log.zip | - | | | | |
| EPT_Transfer_Test Date modified: 3/13/2013 8:52 PM File folder | | | | | |
| us Prime by clicking or | Quaritys II 12/15p21 Webb Helition | | | | |





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



At the Top-Level Entity page, browse to the

C:\intelFPGA_lite\xxx.x\quartus\qdesignscreate directory to store your project. Type in a name for your project "EPT_570_AP_U2_Top".



| Directory, Name, Top-Level Entity [page 1 o | f 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 | l must exactly m | atch the entity na | ame in the design fil | e. |
| EPT_570_AP_U2_Top | | | | | |
| Use Existing Project Settings | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

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

\Projects_HDL\EPT_Transfer_Test \src folder of the UnoProLogic 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_U2_Top.v



| | | Device. |
|--|---|---------|
| General | Files | |
| 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. | |
| Libraries | Select the design mes you want to include in the project, end, and all to add all design mes in the project directory to the project. | |
| Operating Settings and Conditions Voltage Temperature | File name: | Add |
| Compilation Process Settings | Туре | Add All |
| Early Timing Estimate | /EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/write_control_logic.v Verilog HDL File | |
| Incremental Compilation | /EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/read_control_logic.v Verilog HDL File | Remove |
| Physical Synthesis Optimizations | /EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/mem_array.v Verilog HDL File | Up |
| EDA Tool Settings | /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 | Up |
| Design Entry/Synthesis Simulation | PEPT USB-CPLD Development System CD/Projects_HDL/EPT_Iransfer_lest/src/EPT_5/0_AP_02_1op.v Verilog HDL Hie /EPT USB-CPLD Development System CD/Projects HDL/EPT_Transfer_Test/src/active_trigger.vgm Verilog Quartus M | Down |
| Formal Verification | (EP1058-CPLD Development System CD/Projects HDL/EPT Transfer Test/src/active transfer library.vam Verlidg Quartus M | |
| Board-Level | (EPT USB-CPLD Development System CD/Projects HDL/EPT Transfer_Test/src/active_transfer.vqm Verlog Quartus M | |
| Analysis & Synthesis Settings | /EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/active_block.vqm Verilog Quartus M | |
| Fitter Settings TimeQuest Timing Analyzer Assembler Design Assistant SignaTap II Logic Analyzer Logic Analyzer Interface PowerPlay Power Analyzer Settings SSN Analyzer | | |
| | | |

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



| Family: MAX II Package: Any Devices: All Pin count: Any Target device Speed grade: Any O Auto device selected by the Fitter Name filter: Image: Comparison of the comparison of t | ▼ ▼ ▼ |
|---|--------------------------|
| Devices: All Pin count: Any Target device Auto device selected by the Fitter Name filter: | |
| Target device Speed grade: Any O Auto device selected by the Fitter Name filter: | |
| Auto device selected by the Fitter Name filter: | • |
| Auto device selected by the Fitter Name filter: | |
| | |
| Specific device selected in 'Available devices' list Show advanced devices | |
| | HardCopy compatible only |
| 🔘 Other: n/a | |
| ······ | blocks |
| EPM570T100C3 3.3V 570 1 | |
| EPM570T100C4 3.3V 570 1 | |
| EPM570T100C5 3.3V 570 1 | |
| EDME 707 10015 0 0V | |
| | |
| EPM570T144A5 3.3V 570 1 | |
| EPM570T144A5 3.3V 570 1 EPM570T144C3 3.3V 570 1 | |
| EPM570T144A5 3.3V 570 1 | |

Select Next, leave defaults for the EDA Tool Settings.



| Tool Name Timing Symbol Signal Integrity Boundary Scan | Format(s) | Run t Run t Run t V | l Automatically his tool automatically jate-level simulation a | |
|--|--|----------------------------------|--|-------------------|
| <none> <none> Timing Symbol Signal Integrity</none></none> | <none><none><none><none></none></none></none></none> | Run g | | |
| <none> Timing Symbol Signal Integrity</none> | <none><none><none></none></none></none> | • • | ate-level simulation a | utomatically afte |
| Timing Symbol Signal Integrity | <none> <none> <none></none></none></none> | • | | |
| Symbol Signal Integrity | <none></none> | • | | |
| Signal Integrity | <none></none> | - | | |
| | | | | |
| Boundary Scan | <none></none> | | | |
| | | - | | |
| | | | | |
| | | 11 | | |

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



| 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>)</none></none> |
| Simulation: | <none> (<none>)</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_U2_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.



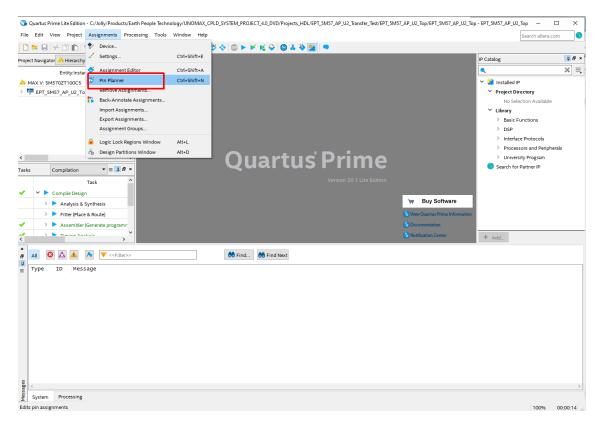
| | ology/UNOMAX_CPLD_SYSTEM_PROJECT_4.0_DVD/Projects_HDL/EPT_SM57_AP_U2_Transfer_Test | |
|---|--|----------------------------------|
| File Edit View Project Assignments Processing Tools | | Search altera.com |
| 🗋 🚾 🔲 🖌 🖺 💼 🔤 🍢 Device | ŏ ♦ ∞ ► ¥ K ♀ ⊜ & ≯ 🙀 📍 | |
| Project Navigator 🔥 Hierarchy 🧹 Settings | Ctrl+Shift+E | IP Catalog 📮 🗗 🗙 |
| Entity:Instar 🗳 Assignment Editor | Ctrl+Shift+A | |
| A MAX V: 5M570ZT100C5 | Ctrl+Shift+N | Y 💐 Installed IP |
| EPT_5M57_AP_U2_To Remove Assignments | | Y Project Directory |
| Back-Annotate Assignments | | No Selection Available |
| Import Assignments Export Assignments | | ✓ Library |
| Export Assignments Assignment Groups | | > Basic Functions > DSP |
| | | > Interface Protocols |
| Logic Lock Regions Window | Alt+L | > Processors and Peripherals |
| C Design Partitions Window | Quartus Prime | > University Program |
| Tasks Compilation - = 🗜 🗗 × | Qualtus Fillin | Search for Partner IP |
| Task | | tion |
| ✓ ✓ ► Compile Design | | |
| Analysis & Synthesis | | w Buy Software |
| Fitter (Place & Route) | | S View Quartus Prime Information |
| Assembler (Generate programm | | © Documentation |
| | | Notification Center + Add |
| < >> | | T Add |
| × All 3 ▲ ▲ ▼ < <filter>></filter> | 60 Find 60 Find Next | |
| Type ID Message | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| System Processing | | > |
| System Processing | | |
| Imports assignments from other sources | | 100% 00:00:14 |

At the Import Assignment dialog box, Browse to the \Projects_HDL\EPT_Transfer_Test \ EPT_UnoProLogic_TOP folder of the UnoProLogic Development System DVD. Select the "EPT_570_AP_U2_Top.qsf" file.

| V Import Assignments | X |
|--|--------------------------|
| Specify the source and categories of assignments to import. | |
| File name: echnology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/Altera_EPM570_U2/EPT_570_ | AP_U2_Top.qsf Categories |
| Copy existing assignments into EPT_570_AP_Transfer_Test.qsf.bak before importing | Advanced |
| | OK Cancel Help |
| | |

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





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



| ile | Edit | View Processing | Tools Window | Help 🐬 | | | Sear | ch altera.com | 0 |
|--------|---|--|---|--|---|--|----------|--|---|
| j [| Report | | ₽ ₽ × | | | | 11 | | |
| | Rep | ort not available | | | | ●▲♥●●♥●▲●●○○○ | 0000000 | 1 | |
| 3 | | | | | : • | | Ē | 24 | |
| 3 | | | | | | | | 73 | |
| 2 | | | | | 1 | Top View | | 78 | |
| 2 | | | | | :8 | Wire Bond | 2 | | |
| 2 | | | | | :Ă | | ě | er | |
| | | | | | ĽŽ | | V | - | |
| Ç | | | | | | | | а а а а | |
| | | | | | | | - | 42 81 | |
| ' L | Tasks | | 무 🗗 🗙 | | : . | | X | - | |
| 1 | | Show I/O Ba | nks 🔺 | | | MAX II | 6 | | |
| | | Show VREF (| Groups | | | | | | |
| | | Show Edges | | | 28 | EPM570T100C5 | | 22 24 | |
| E. | | Show Edges | OS Dine | | :8 | | Ě | | |
| 1 | | • | | | -8 | ∆⊽oooo⊽o∆●●●ɑ | Ö AVAGOO | a1 | |
| 8 | | 🕨 x4 Mode | | | 24 27 38 25 88 | | | | |
| | ∢ | | | | | | | | _ |
| - 113 | | ed: * 🛛 👻 | » Edit: 💢 🖌 | | | | F | Filter: Pins: all | |
| | 7 7 | NodeName | Direction | Location | I/O Bank | I/O Standard | Reserved | Current Strength | |
| | ۲ | LB_LOWER[0] | Unknown | PIN_52 | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| | > | LB_LOWER[1] | Unknown | PIN_53 | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| ۱I | | LB_LOWER[2] | Unknown | PIN_54 | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| | (2) | LB_LOWER[3] | Unknown | PIN_55 | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| - 11 | | | | | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| - - | > | LB_LOWER[4] | Unknown | PIN_56 | - | | | | |
| • | 2 | LB_LOWER[5] | Unknown | PIN_57 | 2 | 3.3-V LVdefault) | | 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] | Unknown Unknown | PIN_57 PIN_58 | 2 | 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] | Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 | 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] | Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 | 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] | Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 | 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] | Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 | 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[3] | Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_92 | 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 | 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_UOPER[0] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[3] LB_UPPER[4] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_92 PIN_91 | 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_UOPER[0] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[3] LB_UPPER[4] LB_UPPER[5] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_95 PIN_92 PIN_91 PIN_89 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_UPVER[7] LB_UPVER[0] LB_UPVER[1] LB_UPVER[2] LB_UPVER[3] LB_UPVER[4] LB_UPVER[5] LB_UPVER[6] LB_UPVER[7] LED[0] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_92 PIN_92 PIN_91 PIN_87 PIN_86 PIN_74 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| 2 | | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[2] LB_UPPER[3] LB_UPPER[5] LB_UPPER[5] LB_UPPER[5] LB_UPPER[7] LB_UPPER[7] LB_UPPER[7] LB_UPPER[7] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_95 PIN_95 PIN_92 PIN_92 PIN_87 PIN_86 PIN_74 PIN_75 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3 v LVdefault) 3.3 v LVdefault) | | 16mA (default) 16mA (default) | |
| 2 | $\begin{array}{c} \diamond \diamond$ | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[3] LB_UPPER[3] LB_UPPER[4] LB_UPPER[6] LB_UPPER[6] LB_UPPER[6] LB_UPPER[7] LED[0] LED[1] LED[2] | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_95 PIN_92 PIN_91 PIN_89 PIN_87 PIN_86 PIN_75 PIN_98 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| 2 | $\begin{array}{c} \diamond \diamond$ | LB_LOWER(5) LB_LOWER(6) LB_LOWER(6) LB_UPPER(0) LB_UPPER(1) LB_UPPER(2) LB_UPPER(2) LB_UPPER(3) LB_UPPER(5) LB_UPER(5 | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_95 PIN_92 PIN_91 PIN_89 PIN_87 PIN_86 PIN_74 PIN_75 PIN_75 PIN_98 PIN_99 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| 2 | $\begin{array}{c} \diamond \diamond$ | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[2] LB_UPPER[4] LB_UPPER[5] LB_UPPER[5] LB_UPPER[5] LB_UPPER[7] LED[0] LED[1] LED[2] LED[3] SW_USER_1 | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_57 PIN_61 PIN_97 PIN_95 PIN_95 PIN_92 PIN_91 PIN_89 PIN_87 PIN_86 PIN_87 PIN_86 PIN_75 PIN_98 PIN_75 PIN_98 PIN_99 PIN_15 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| | | LB_LOWER(5) LB_LOWER(6) LB_LOWER(7) LB_UPPER(0) LB_UPPER(1) LB_UPPER(2) LB_UPPER(2) LB_UPPER(4) LB_UPPER(5) LB_UPPER(6) LB_UPPER(6) LB_UPPER(7) LED(0) LED(1) LED(2) LE | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_58 PIN_61 PIN_97 PIN_96 PIN_95 PIN_95 PIN_92 PIN_89 PIN_87 PIN_87 PIN_74 PIN_75 PIN_98 PIN_98 PIN_99 PIN_155 PIN_100 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (default) 16mA (default) | |
| | S | LB_LOWER[5] LB_LOWER[6] LB_LOWER[7] LB_UPPER[0] LB_UPPER[1] LB_UPPER[2] LB_UPPER[2] LB_UPPER[4] LB_UPPER[5] LB_UPPER[5] LB_UPPER[5] LB_UPPER[7] LED[0] LED[1] LED[2] LED[3] SW_USER_1 | Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown Unknown | PIN_57 PIN_57 PIN_61 PIN_97 PIN_95 PIN_95 PIN_92 PIN_91 PIN_89 PIN_87 PIN_86 PIN_87 PIN_86 PIN_75 PIN_98 PIN_75 PIN_98 PIN_99 PIN_15 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | 3.3-V LVdefault) 3.3-V LVdefault) | | 16mA (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

 $https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/VHD\ L/Timequest.pdf$

Browse to the $Projects_HDL EPT_Transfer_Test \\ EPT_UnoProLogic_TOP folder of the UnoProLogic Development System DVD. Select the "EPT_570_AP_U2_Top.sdc" file.$



| rganize 🕶 🚺 Open Share with 🕶 Burn New f | older | | | | |
|---|---------------------------|--------------------|-------------|------|--|
| Documentation · | Name | Date modified | Туре | Size | |
| Ju Drivers | 🔒 db | 3/3/2013 1:26 AM | File folder | | |
| Projects_ActiveHost_32Bit | incremental_db | 1/30/2013 8:42 PM | File folder | | |
| Projects, ActiveHost, 64Bit Projects, Activino Projects, J-DL EPT_570, AP_Data, Collector EPT_Transfer_Test | 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 | |
| Altera_EPM570_U2 | EPT_570_AP_U2_Top.qws | 3/3/2013 1:26 AM | QWS File | 2 KB | |
| Altera_EPMD/0_02 | EPT_570_AP_U2_Top.sdc | 1/29/2013 10:30 PM | SDC File | 4 KB | |
| Quartus, Programmer Switch FDI Linear Tech NP | | | | | |
| Telegesis | | | | | |

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

| ganize Open Burn New fold FPT Transfer Test | er | Name | Date modified | - | Size |
|--|----|---------------------------|--------------------|---------------------|------|
|) fir_filter | | Marrie | 3/13/2013 9:42 PM | Type File folder | 3126 |
| incr_comp_makefile | | incremental db | 3/13/2013 9:38 PM | File folder | |
| whdl_verilog_tutorial | | output_files | 3/13/2013 9:42 PM | File folder | |
| 🕌 sopc_builder | | simulation | 3/13/2013 9:42 PM | File folder | |
| altera_inspector.log.zip | | EPT_570_AP_U2_Top.jdi | 3/13/2013 9:42 PM | JDI File | 1 KB |
| 91 | | EPT_570_AP_U2_Top.qpf | 3/13/2013 9:10 PM | QPF File | 2 KB |
| Brother | E | 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 |
| Documents and Settings | | EPT_570_AP_U2_Top.sdc | 1/29/2013 10:30 PM | SDC File | 4 KB |
| Jolly | | | | | |
| Business Opportunities | | | | | |
| Capitol_College | | | | | |
| Code_FPGA | | | | | |
| Documents | | | | | |
| \mu Download | | | | | |
| Drivers | - | | | | |

Select the Start Compilation button.



| ile Edit View Project Assignments Processing Tools Win | | | Search altera.com |
|--|-------------------------------|--------------------------------|---------------------------|
| 🗋 🔚 🕂 🗋 🛍 🏷 🦿 EPT_5M57_AP_U2_Top | - ∠ ♦ ♦ ∞ ∞ ► × € € ⊜ ≛ ≫ 🔛 📍 | | |
| oject Navigator 🍌 Hierarchy 🛛 🔻 🦳 🗗 🗙 | | IP Catalog | 1 - 7 |
| Entity:Instance | | | × = |
| MAX V: 5M570ZT100C5 | | 🗡 🚢 Instal | led IP |
| EPT_SM57_AP_U2_Top | | | ct Directory |
| | | | o Selection Available |
| | | ✓ Librar | ry asic Functions |
| | | | |
| | | > Int | terface Protocols |
| | | > Pr | rocessors and Peripherals |
| > | Quartus Prime | | niversity Program |
| sks Compilation 🔻 🗏 🖪 🗶 | Quartasi init | Search | h for Partner IP |
| Task ^ | | | |
| V Compile Design | | | |
| > > Analysis & Synthesis | | 👾 Buy Software | |
| Fitter (Place & Route) | 6 | View Quartus Prime Information | |
| Assembler (Generate programm) | 6 | Documentation | |
| Timing Applicie | 6 | Notification Center + Add | |
| · · · · · · · · · · · · · · · · · · · | | | |
| All 😣 🛕 \land 💙 < <filter>></filter> | 😽 Find 😽 Find Next | | |
| Type ID Message | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| System Processing | | | |

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 | | | 23 |
|---|---|---|--|-------------------|-----------|
| File Edit View Project Assignments Processing | Tools Window Help 💎 | | | Search altera.com | • |
| 📄 💕 🚽 🥔 👗 🖓 👘 🔃 🕫 (* 1975-570_AP) | U2_Top 🔹 😵 🖓 🐨 | N 10 10 10 10 10 10 10 10 10 10 10 10 10 | : 🎍 🐌 👗 💿 💎 | | |
| Project Navigator 4 🗗 🗙 | Compilation Report - EPT_570_AP_U2_Top | | | | |
| Entity | Table of Contents | Flow Summary | | | |
| MAX II: EPM570T100C5 ID EPT_570, AP_U2, Top matrix active_transfer:ACTIVE_TRANSFER_INST matrixe_transfer:ACTIVE_TRANSFER_L active_transfer_IoPary:ACTIVE_TRANSFER_L active_transfer_IoPary:ACTIVE_TRANSFER_L active_transfer_IOPary:ACTIVE_TRANSFER_L active_transfer_IOPary:ACTIVE_TRANSFER_L | Flow Summary Flow Settings Flow Non-Default Global Settings Flow Non-Default Global Settings Flow OS Summary Flow OS Summary | Flow Status Quartus II 64-Bit Version Revision Name Top-level Entity Name Family Device | Flow Failed - Wed Mar 13 21:26 12.1 Build 243 01/31/2013 SP 1 EPT_570_AP_U2_Top EPT_570_AP_U2_Top MAX II EPM570T100C5 | | |
| sync_ffeebil.OCK_IN_FIFO develockeb.OCK_TRANSFER_INST develockeb.OCK_TRANSFER_INST | ■ Flow Log ▶ ▲ Analysis & Synthesis | Timing Models | Final | | |
| Task Image: Completion Flow: Completion Task Image: Completion Image: Completion Image: Completion | e mpilation was NOT successful (3 errors, 3 warnings) | | | | |
| × 🖪 😋 🛆 🚺 🧟 💎 < <search>></search> | ~ | | | | |
| Type ID Message II28 Elaborating entity "epth" II28 Elaborating entity "acti II28 Elaborating entity "acti II28 Elaborating entity "acti II206 Node instance "BLOCK_IN II4001 Generated suppressed mess Quartus II 64-Bit Analys | <pre>ireOR" for hierarchy "eptWireOR:wireOf ve_trigger" for hierarchy "active_tri ve_transfer" for hierarchy "active_trive ve_hick" for hierarchy "active_toix" FIFO instantiates undefined entity " isages file ('altera/12 lapl/quarturg/ is 6 Synthesis was unsuccessful. 1 er ion was unsuccessful. 3 errors, 3 war m</pre> | <pre>gger:ACTIVE_TRIGGER ansfer:ACTIVE_TRANS EBLOCK_TRANSFER_INS sync_fifo" gdesigns/EFT_Transf cor, 3 warnings</pre> | FER_INST" T" | EPT_570_AP_U2_T | op.m E |
| System (1) Processing (32) | | | | 9% 00 | :00:03 |

Click Ok, the select the "Error" tab to see the error.



| 🚱 Quartus II 64-Bit - C:/altera/12.1sp1/quartus/qdesigns | /EPT_Transfer_Test/EPT_570_AP_U2_Top - EPT_570_AP_U2 | Тор | 1 m | | |
|--|---|---|--|------------------|----------|
| File Edit View Project Assignments Processing | g Tools Window Help 💎 | | | Search altera.co | m 🚯 |
| 1 📄 💕 🗐 🥔 🕺 🖧 🛍 🖄 🕫 🕬 (* 1975-570_AP) | U2_Top 🔹 🔀 💓 🗸 🗸 💞 🧇 💿 | ► Ø Ø Ø ₩ ₹ | 2 4 4 5 | | |
| Project Navigator # # × | Compilation Report - EPT_570_AP_U2_Top | | | | |
| Entity | Table of Contents \$ | _ | | | |
| AMA E PM/S7110/CS // E PT 570_4P12_150 // E PT 570_4P12_150 // E PT 570_4P12_150 // E PM/S7110/CS // E PM/S710/CS // E PM/S710/CS // E PM/S710/CS // E PM/S710/C | Flow Summary Flow Actings Flow Actings Flow Actings Flow Isonofault Global Settings Flow Lagsed Time Flow Cos Flow Log Analysis & Synthesis | How Status Quantus 164-Bit Version Revision Name Top-level Anthy Name Family Device Timing Models | Piov Failed - Wed Mar 12.22 12.1 Build 243 01/21/2013 24 PT _570 JeU (20 pp EPT_570 JeU (20 pp HAT 30 JeU (20 pp) HAT 30 JEU (20 pp | | |
| | × | | | | |
| × ▲ 😰 🔉 🔊 🕢 🔻 < <search>></search> | ~ | | | | |
| Type ID Message 12006 Node instance "BLOCK_IN Quartus II 64-Bit Amaly 293001 Quartus II Full Compilet | FIFO" instantiates undefined entity is & Synthesis was unsuccessful. 1 eri ion was unsuccessful. 3 errors, 3 warn | for, 5 warnings | | | |
| System / Processing (3) / | m | | | | ŀ |
| (a) (a) (a) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c | | | | 9% | 00:00:03 |

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.

| tegory: | | | Device |
|--|--|--|------------|
| General | Files | | |
| 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 director | roject. | |
| Libraries | | | |
| Operating Settings and Conditions Voltage | File name: olly/Products/Earth People Technology/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Te | | Add |
| Temperature Compilation Process Settings | | Туре | Add All |
| Early Timing Estimate Incremental Compilation | blogy/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/write_control_logic.v blogy/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/read_control_logic.v | Verilog HDL File Verilog HDL File | Remove |
| Physical Synthesis Optimizations EDA Tool Settings | plogy/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/mem_array.v plogy/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/eptWireDRv | Verilog HDL File Verilog HDL File | Up |
| Design Entry/Synthesis | plogy/EPT USB-CPLD Development System CD/Projects_HDL/EPT_Transfer_Test/src/EPT_570_AP_U2_Top.v plogy/EPT USB-CPLD Development System CD/Projects HDL/EPT Transfer Test/src/active trigger.ygm | Verilog HDL File Verilog Quartus Mapp | Down |
| Design Entry/Synthesis Simulation Formal Verification Board-Level Analysis & Synthesis Settings VHDL Input Default Parameters Fitter Settings TimeQuest Timing Analyzer Assembler Design Assistant SignaTipp II.logic Analyzer Logic Analyzer Interface PowerPlay Power Analyzer Settings SSN Analyzer | blogy/EPT US8-CPLD Development System CD/Projects_HDU/EPT_Transfer_Text/src/active_transfer_libraryvqm blogy/EPT US8-CPLD Development System CD/Projects_HDU/EPT_Transfer_Text/src/active_block.vqm blogy/EPT US8-CPLD Development System CD/Projects_HDU/EPT_Transfer_Text/src/active_block.vqm | Verilog Quartus Mapp Verilog Quartus Mapp Verilog Quartus Mapp | Properties |
| | < III III III III III III III IIII III | | |



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 | Тор | | |
|---|--|--|--|-------------------|
| File Edit View Project Assignments Processing | Tools Window Help 🐬 | | | Search altera.com |
| 🗋 💕 😹 🥔 🐰 🖏 🖏 🤟 (*) EPT_570_AP_U | J2_Top 🔹 🔀 💓 🔹 | No | : 🍚 🦥 👗 🔞 🐬 | |
| Project Navigator 🕂 🗗 🗙 | Compilation Report - EPT_570_AP_U2_Top | × | | |
| Entity | Table of Contents 4 🗗 | Flow Summary | | |
| MAX II: EPM570T100C5 | Flow Summary | Flow Status | Successful - Wed Mar 13 22:0 | |
| < | Flow Settings Flow Non-Default Global Settings Flow Non-Default Global Settings Flow Logs Flow Log Flow Log Flow Log Content Content | Quartus II 64-Bit Version Revision Name Top-level Entity Name Family Device Timing Models Total logic elements Total pins Total pins Total pins UFM blocks | 12.1 Build 243 01/31/2013 SP: EPT_570_AP_U2_Top EPT_570_AP_U2_Top MAX II EPM5700100C5 Final 557 / 570 (98 %) 51 / 76 (67 %) 0 0 / 1 (0 %) | 1 SJ Web Edition |
| Hierarchy Files P Design Units P | Full Compilation was successful (47 warnings) | UFM blocks | 0/1(0%) | |
| Task | | | | |
| Compile Design 00 | | | | |
| 🖉 🕨 🕨 Analysis & Synthesis 🛛 00 | | | | |
| Fitter (Place & Route) 00 | | | | |
| Assembler (Generate programmin 00 | | | | |
| TimeQuest Timing Analysis 00 | | | | |
| EDA Netlist Writer 00 | | | | |
| Program Device (Open Programmer) | | | | |
| | < + | | | |
| All 🙆 🖾 💒 🚲 👎 < <search>></search> | \sim | | | |
| Type ID Message | | | | |
| 332102 Design is not fully cons | | | | |
| | est Timing Analyzer was successful. 0 | errors, 5 warnings | | |
| Running Quartus II 64-Bi | + FDA Notligt Writer | | | |
| | ead settings files=offwrite setting | s files=off EPT 57 | D AP U2 Top -c EPT 57 | 0 AP U2 Top |
| | AP_U2_Top.vho" and "EPT_570_AP_U2_Top | | | |
| Quartus II 64-Bit EDA Ne | tlist Writer was successful. 0 errors, | 0 warnings | | |
| a (1) 293000 Quartus II Full Compilat | ion was successful. 0 errors, 47 warni | ngs | | |
| y 293000 Quartus 11 Full Compilat | | | | |
| ö | | | | |

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.

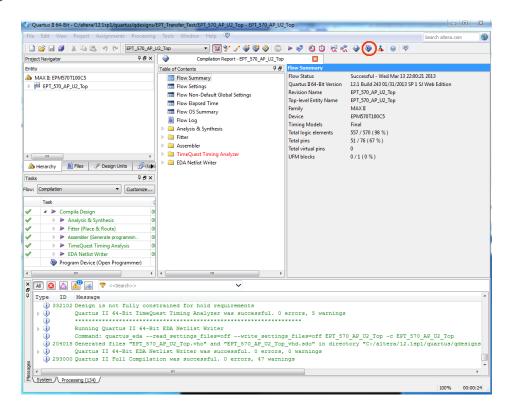
4.1.2 Programming the CPLD

Programming the CPLD is quick and easy. All that is required is a standard USB-C cable and the EPT_Blaster Driver DLL. Connect the UnoProLogic 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.



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



| w background progra | amming (for MAX II and | | | | | | | |
|---------------------|------------------------|----------|----------|-----------------------|-----------|-----------------|-----------------|-----------------|
| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | Securi Bit |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | Configure | Configure | Configure Check | Configure Check |

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

| Hardware Setup | | | | | | |
|--|----------|--------------|-----------------|------------|-----------------|----|
| Hardware Settings JTAC | Settings | | | | | |
| Select a programming hard hardware setup applies only | | | | levices. T | his programming | |
| Currently selected hardware | EPT-JT | AG-Blaster v | 1.0 (64) [MBUSB | 8-0] | | • |
| Hardware frequency: | | | | | | Hz |
| Available hardware items - | | Server | Port | | Add Hardware | |
| EPT-JTAG-Blaster v1.0 (6 | 4) | Local | MBUSB-0 | | Remove Hardware | e |
| | | _ | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | Clos | |
| | | | | | Clos | e |



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 UnoProLogic device.

| | | | | | | | iearch altera | | |
|-------------------------|-------------------------|---------------------------|--------------|----------|-----------------------|--------|-----------------|---------|--|
| | EPT-Blaster v1.3b [MBUS | - | | • | Progress: | | | | |
| Enable real-time ISP to | allow background progra | amming (for MAX II and MA | X V devices) | | | | | | |
| 🔊 Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | |
| Julie Stop | | | | | | | | | |
| Auto Detect | | | | | | | | | |
| | | | | | | | | | |
| 💢 Delete | | | | | | | | | |
| | | | | | | | | | |
| 🆖 Add File | | | | | | | | | |
| Change File | | | | | | | _ | | |
| G Change File | • | | m | | | | | | |
| Save File | (| | ш | | | | | | |
| Change File | ۹ [| | m | | | | | | |
| Change File | 4 | | m | | | | | | |

Select the 5M570 under "Device".

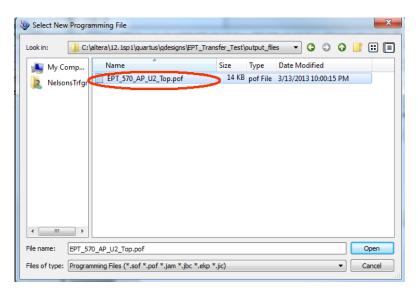
| Hardware Setup | EPT-JTAG-Blaster v1.0 (64) [MBUSB-0] | | Mo | ode: JTAG | | • | Progress | : | | |
|--|--|------------|----------|-----------|-----------------------|--------|-----------------|---------|-----------------|------|
| Enable real-time | ISP to allow background programming when a | vailable | | | | | | | | |
| ▶ ¹ ¹ ¹ Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | Security Bit | Eras |
| ^{∎™} Stop | C:/Jolly/Products/Earth People Technolog | 5M570ZT100 | 002E7DAA | 002E7D78 | | | | | | |
| Auto Detect | СЕМ | | | | | | | | | |
| × Delete | UFM | | | | | | | | | |
| Add File | | | | | | | | | | |
| Change File | < | | | | | | | | | |
| Save File | | | | | | | | | | |
| Add Device | | | | | | | | | | |
| ி ^{ங்} Up | TDI | | | | | | | | | |
| | | | | | | | | | | |
| ¹ [™] Down | | | | | | | | | | |
| | 5M570ZT100 TDO | | | | | | | | | |
| | | | | | | | | | | |



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

| Programmer - C:/ | Select New Progra | mming File | | | |
|----------------------|------------------------|-----------------------------------|---------------------|-----------------------|--------------|
| ile Edit View | Look in: | altera\12.1sp1\quartus\qdesign | s\EPT_Transfer_Test | - 0 0 0 | * • • |
| ine cuit view | My Comp | Name | Size Type | Date Modified | |
| 🔔 Hardware Setup. | NelsonsTrfgr | 퉬 db | Filder | 3/13/2013 10:00:22 PM | |
| Enable real-time I | Meisons mgr | incremental_db | | 3/13/2013 9:38:49 PM | |
| | • | output_files | | 3/13/2013 10:00:21 PM | |
| ⊌ [™] Start | | simulation | Filder | 3/13/2013 9:42:12 PM | r |
| Stop | | | | | |
| Auto Detect | | | | | |
| 🗙 Delete | | | | | |
| 🍐 Add File | | | | | |
| 🔓 Change File) | | | | | |
| Save File | 4 | | | | |
| 🚰 Add Device | File name: | | | | Open |
| the Dp | Files of type: Program | mming Files (*.sof *.pof *.jam *. | jbc *.ekp *.jic) | • | Cancel |
| J [™] Down | | | | | |
| | | | | | |
| | | | | | |

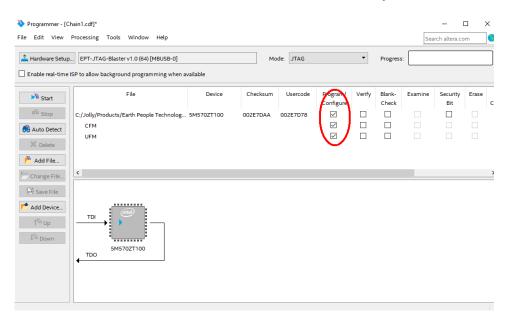
Click on the EPT_5M57_AP_U2_Top.pof file to select it.



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.



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



| Programmer - [Ch | nain1.cdf]* Processing Tools Window Help | | | | | | | Sea | | |
|--|--|------------|----------|----------|-----------------------|--------|-----------------|---------|-----------------|-------|
| Hardware Setup. | . [EPT-JTAG-Blaster v1.0 (64) [MBUSB-0] SP to allow background programming when | available | Mo | de: JTAG | | Y | Progress | | 6% | |
| ^{▶₩} Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | Security Bit | Erase |
| 📲 Stop | C:/Jolly/Products/Earth People Technolog. | 5M570ZT100 | 002E7DAA | 002E7D78 | | | | | | |
| Auto Detect | CFM | | | | \checkmark | | | | | |
| X Delete | UFM | | | | \checkmark | | | | | |
| Add File | < | | | | | | | | | |
| Change File Save File Add Device 1th Up Up Ub Down | TDI 5M5702T100 | | | | | | | | | |

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



| dware Setup E | PT-JTAG-Blaster v1.0 (64) [MBUSB-0] | | Mo | de: JTAG | | • | Progress | s: 10 | 0% (Succes | sful) |
|----------------------|---------------------------------------|------------|----------|----------|-----------------------|--------|-----------------|---------|-----------------|-------|
| ble real-time ISP to | o allow background programming when a | vailable | | | | | | | | |
| ⁰ Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | Security Bit | Era |
| Stop C:/ | Jolly/Products/Earth People Technolog | 5M570ZT100 | 002E7DAA | 002E7D78 | | | | | | |
| to Detect | CFM | | | | \checkmark | | | | | |
| Delete | UFM | | | | \checkmark | | | | | |
| | | | | | | | | | | |
| dd File | | | | | | | | | | |
| ange File < | | | | | | | | | | |
| Save File | | | | | | | | | | |
| | | | | | | | | | | |
| d Device | TDI | | | | | | | | | |
| [%] Up — | | | | | | | | | | |
| Down | | | | | | | | | | |
| | 5M570ZT100 | | | | | | | | | |
| + | TDO | | | | | | | | | |
| | | | | | | | | | | |

At this point, the UnoProLogic 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.

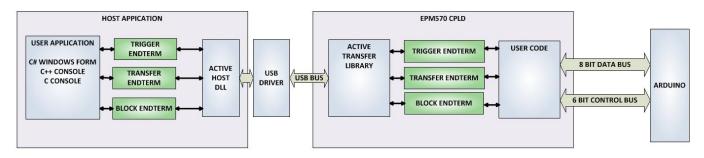
5 Active Host Application

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



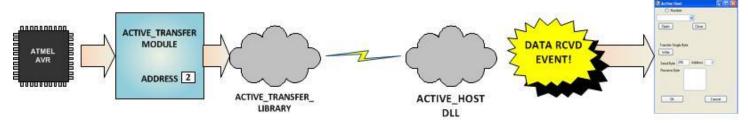
circular buffer in memory. This allows the user code to fetch the transfers with out losing bytes.

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)
       1
           //Compare port name with the found VID/PID
           //combinations. Add them to Matching port list
           //and comboDevList
           if (names.Contains(port))
            5
               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.

| 🖳 EPT_Transfer_Test | | |
|--|---|------|
| l | ~ | Open |
| EPT JTAG Blaster 0 EPT Serial Communications 0 Transfer Controls | | |

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)
        ſ
            serialPort AH.Open();
            btnOpenDevice.Enabled = false;
            btnCloseDevice.Enabled = true;
            //textBox1.ReadOnly = false;
            return true:
        3
    }
    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

```
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)
    ł
        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)
                 ſ
                     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)
 ſ
     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 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)
ł
    int BlockAddress = (int)data;
    byte[] cBuf = new Byte[device[BlockAddress].Length];
    if ((device[BlockAddress].Repititions > 0) &
        !device[BlockAddress].TransferPending & !BlockTransferStop)
    ſ
        device[BlockAddress].TransferPending = true;
        Buffer.BlockCopy(block 8 in payload, 0, cBuf, 0,
            device[BlockAddress].Length);
        fixed (byte* pBuf = cBuf)
        ł
          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--;
    }
3
```

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)
    Ł
        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 UnoProLogic. A standalone project can be range from a simple program to display and send data from the user to/from the Arduino Uno. 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.

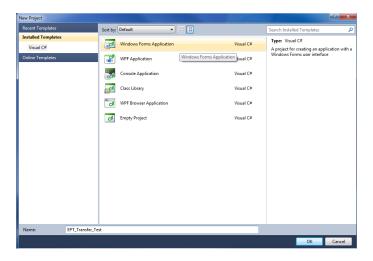


6.1 Creating a Project

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

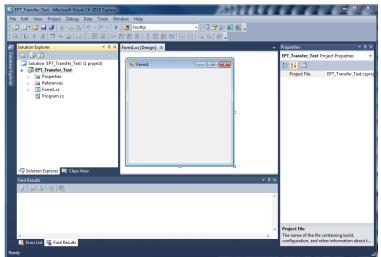
| 🖪 N | licrosoft Visual C# 2010 Express | | | - 0 × |
|------|----------------------------------|--------------|-----------------------|--------------|
| File | Edit View Debug Tools Wi | ndow Help | | |
| 67 | New Project | Ctrl+Shift+N | 🖄 tooltip 🔹 🚽 🖓 🎓 🖄 : | - |
| đ | Open Project | Ctrl+Shift+O | | |
| 2 | Open File | Ctrl+O | | |
| | Close | | | |
| đ | Close Solution | | | |
| | Save Selected Items | Ctrl+S | | |
| | Save Selected Items As | | | |
| 9 | Save All | Ctrl+Shift+S | | |
| | Export Template | | | |
| | Page Setup | | | |
| 8 | Print | Ctrl+P | | |
| | Recent Files | • | | |
| | Recent Projects and Solutions | • | | |
| | Exit | Alt+F4 | | - ₽ × |
| | \$ \$ \$ \$ \$ \$ | | | |
| | | | | * |
| | | | | |
| | | | | |
| | | | | |
| | 4 | | | * |
| | 🙀 Error List 🖳 Find Results | | | |
| Read | ······ | | | |
| | | | | |

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



The project creation is complete.





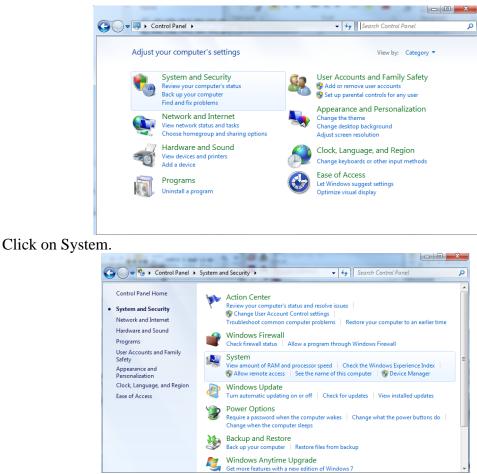
Save the project, go to File->Save as, browse to a folder to create EPT_Transfer_Test folder. The default location is c:\Users\<Users Name>\documents\visual studio 2010\Projects.

| Project Location | | | | | | × |
|--|---|------|--------------------|------------------------------|------|----|
| Goo ✓ Kearth People Technology ▶ EPT_Transfer_Test ▶ EPT_Transfer_Test | | | ▼ 49 | ✓ ✓ Search EPT_Transfer_Test | | P |
| Organize 🔻 New folder | | | | | | 0 |
| ActiveHost_1.0.0.6 | * | Name | | Date modified | Туре | |
| ActiveHost_1.0.0.7 | | | | | | |
| Boards | | | No items match you | ır search. | | |
| EPT_570_AP_Active_Transfer | | | | | | |
| EPT_570_AP_Data_Collector | | | | | | |
| EPT_FT201X_Interface | | | | | | |
| EPT_FT2232_Interface | - | | | | | |
| EPT_FT2232H_JTAG | - | | | | | |
| EPT_FT2232H_SVF_Programmer | | | | | | |
| EPT_FT2232H_USBTransfer | | | | | | |
| 4 퉲 EPT_Transfer_Test | | | | | | |
| EPT_Transfer_Test | | | | | | |
| HSP_FT2232H_FPGA | | | | | | |
| ISP_FT2232H_SPI | - | • | | | | • |
| | | | | | | |
| Folder: EPT_Transfer_Test | | | | | | |
| | | | (| Select Folder | Canc | el |

6.1.1 Setting up the C# Express Environment x64 bit

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





Check under System type:



| | | A busiced and | |
|--------------------------------------|--|---|---------------------------|
| 🚱 🔍 💌 🕨 Control Panel 🕨 | System and Security + System | - 47 Search C | Control Panel |
| Control Panel Home | View basic information | n about your computer | • • |
| 🛞 Device Manager | Windows edition | | |
| 😵 Remote settings | Windows 7 Home Premiu | m | |
| 😵 System protection | Copyright © 2009 Micros | oft Corporation. All rights reserved. | |
| Advanced system settings | 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 0 | Gateway. |
| | Installed memory (RAM) | 6.00 GB.(5.71 GB usable) | |
| | System type: | 64-bit Operating System | |
| | Pen and rouch. | No Perror Touch input is available for this Display | |
| | Gateway support | | |
| | Website: | Online support | |
| | Computer name, domain, an | d workgroup settings | |
| | Computer name: | Gateway-DkTp-A8 | 🛞 Change settings |
| See also | Full computer name: | Gateway-DkTp-A8 | |
| Action Center | Computer description: | | |
| Windows Update | Workgroup: | WORKGROUP | |
| Performance Information and Tools | Windows activation | | |
| | Windows is activated | | with first and the second |

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 Expre | ress | |
|--|--------------------------------------|-------------------------------------|
| File Edit View Project Build Debug Data | Tools Window Help | |
| । 🛅 🛅 • 📂 🖬 🕼 🕼 🖓 • 🗠 - 🗠 • | 😘 Connect to Database | - Martin PT_570_AP_Data_Collector - |
| 神 同を引 空の四 母凱協様 | Code Snippets Manager Ctrl+K, Ctrl+B | - |
| Solution Explorer 🗸 🕂 🗙 | Choose Toolbox Items | |
| | 🖳 Extension Manager | |
| Image: Solution 'EPT_Transfer_Test' (1 project) | External Tools | |
| | Settings • | Basic Settings |
| | Customize | ✓ Expert Settings |
| Form1.cs | Options | Reset |
| Program.cs | | Import and Export Settings |
| | | |



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

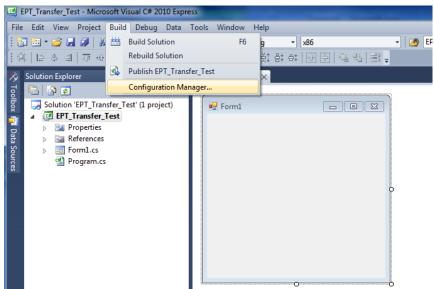
| ▲ Environment General Fonts and Colors Keyboard | Recent files items shown in Window menu 10 items shown in recently used lists |
|--|---|
| Debugging | Visual experience |
| | Automatically adjust visual experience based on client performance Enable rich client visual experience Use hardware graphics 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.

| Environment | Projects location: |
|---|---|
| Projects and Solutions | c:\users\nelsonstrfgr\documents\visual studio 2010\Projects |
| Build and Run | User project templates location: |
| Text Editor | c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ProjectTemp |
| Debugging | User item templates location: |
| Database Tools Tool Tool State | c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ItemTemplat |
| Text Templating Windows Forms Designer | 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.





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

| Active solution configuration: | | Active solut | ion platform: | | |
|-----------------------------------|---------------------------------|--------------------|---------------|---|----------|
| Debug | • | x86 | | | |
| roject contexts (check the projec | t configurations to build or de | x86 <new></new> | | | |
| Project | Configuration | <edit></edit> | | | |
| EPT_Transfer_Test | Debug | - | x86 | • | V |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

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



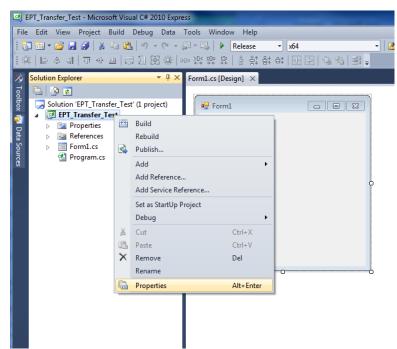
| Active solution configurat Debug | • | | |
|---|---|-------|-------|
| Project contexts (check the Project EPT_Transfer_Test | e project configurations to build or co New Solution Platform Type or select the new platf Any CPU Any CPU Itanium x64 I Create new project platform | iorm: | Build |

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

| Configuration Manager | | \frown | ? × |
|-----------------------------------|------------------------------------|---------------------------|-------|
| Active solution configuration: | | Active solution platform: | |
| Release | | x64 | |
| Project contexts (check the proje | ct configurations to build or depl | loy): | |
| Project | Configuration | Platform | Build |
| EPT_Transfer_Test | Release | ▼ x64 | ▼ |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | Close |

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





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

| -31 | eri_transfer_rest - inicrosoft visual C+ 2010 Expre | | |
|----------------------------|--|--|---|
| | Edit View Project Build Debug Data | | |
| 10 | 🕽 🖼 • 💕 🛃 🥔 👗 🖏 🛍 👘 • 🤍 • 🤇 | 📮 • 🖳 🕨 Release | 🔹 x64 🔹 🧭 EPT_570_AP_Data_Collector 📼 🔩 😒 🎇 🔮 |
| 🦗 🕫 Toolbox 🛺 Data Sources | Solution Explorer Solution Explorer Solution Explorer Solution EXProperties BERTIANSETENT DEFINITIONSETENT DEFINITIONS DEFINI | Pelesee EPT_Transfer_Test* X Application Build Build Events Debug Resources Settings Reference Paths Signing Security Publish | |
| | | | |

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 UnoProLogic Development System CD installed on your PC. Browse to the EPT_Transfer_Test folder where the Project files reside, copy the*.cs files, and install them in the top level folder of your EPT_Transfer_Test project.

| IFT USB-CPLD Development System CD If Arduino JDE Drivers Drivers Projects, ActiveHost, 32Bit Projects, ActiveHost, 42Bit Projects, 4 | Date modified 3/2/2013 11:29 PM 3/2/2013 11:29 PM 3/2/2013 11:29 PM 3/2/2013 11:28 PM 3/2/2013 11:28 PM 2/7/2013 11:11 PM 2/7/2013 11:36 PM | Type File folder File folder File folder Visual C# Source f Visual C# Project f Visual Studio Proj Visual C# Source f | Size 7 K 5 K 1 K 19 K |
|---|--|--|-----------------------------------|
| | 2/4/2013 9:05 PM 2/7/2013 11:34 PM | Visual C# Source f .NET Managed Re Visual C# Source f | 36 K 6 K 1 K |

6.2.1 Changing Project Name

NOTE

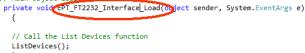
If you named your project something other than EPT_Transfer_Test, you will have to make changes to the *.cs files above. This is because Visual C# Express links the project files and program files together. These chages can be made by modifying the following:

- 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.

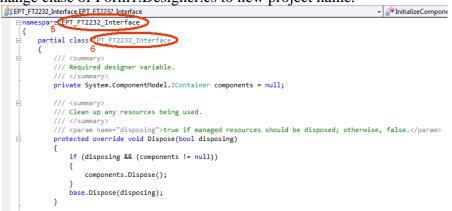


| <pre>%EPT_FT2232_Interface.EPT_FT2232_Interface</pre> |
|---|
| <pre>using System.Drawing; using System.Collections; using System.Windows.Forms; using System.Data;</pre> |
| using System.Collections; using System.Windows.Forms; using System.Data; |
| using System.Windows.Forms; using System.Data; |
| using System.Data; |
| |
| using System.Threading: |
| |
| using System.Runtime.InteropServices; |
| using System.Diagnostics; |
| 1 |
| |
| □ namespace EPT_FT2232_Interface |
| |
| 2 |
| public partial class cP1_FT2232_Interface |
| |
| |
| public _PT_FT2232_Interface() |
| |
| <pre>InitializeComponent();</pre> |
| |
| <pre>for (int i = 0; i < device.Length; ++i)</pre> |
| { |
| <pre>device[i] = new Transfer();</pre> |
| } |
| } |

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



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

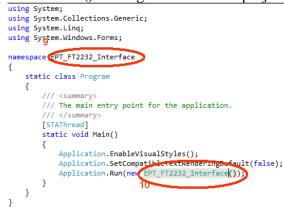


- 7. Change the this.Name and this.Text in Form1Designer.cs to new project name.
- 8. Change this.Load in Form1Designer.cs to include new project name.





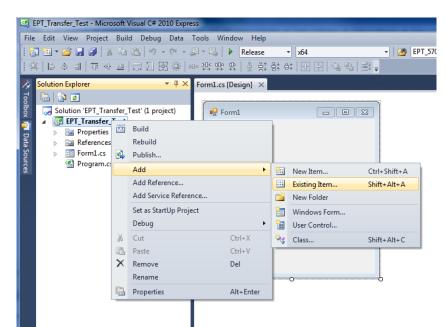
- 9. Change namespace in Program.cs to new project name
- 10. Change Application.Run() in Program .cs to new projectname.



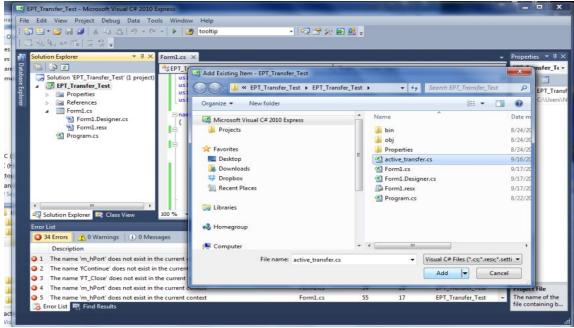
6.2.2 Add Files to Project

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



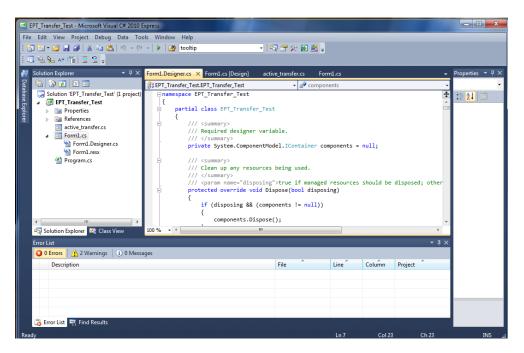


Browse to the EPT_Transfer_Test project folder and select the active_transfer_xx.cs file. Click Add.





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.



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 \gg button to launch the Toolbox.



| 😂 EPT_Transfer_Test - Microsoft Visual C# 2010 Express | |
|---|---|
| File Edit View Project Debug Data Tools Window Help | |
| 🗄 🛅 🖼 🕶 🛃 🥔 👃 🛍 🛍 🥙 - 🔍 - 🕨 🧭 tooltip 💿 - 🖓 🕾 💥 😓 | |
| [許 他今年 司今日 司迎國恭 言將將將國家禁弊軍国国司的國⇒ | |
| | Toolbox |
| Solution 'EPT_Transfer_Test' (1 project) Image: Solution 'EPT_Transfer_Test' (1 project) Image: Solution 'EPT_Transfer_Test Image: Solution 'EPT_Transfer_Test Image: Solution 'EPT_Transfer_Test' Image: Solution 'EPT_Test' Image: Solution 'EPT_Test' Image: Solution 'EPT_Test' | All Windows Forms All Common Controls |
| Solution (EPT_Transfer_Test' (1 project) 🖳 Active Host | Pointer |
| A G Properties | ab Button |
| References | CheckBox |
| active_transfer.cs | CheckedListBox |
| Form1.cs Form1.Designer.cs | 📑 ComboBox |
| S Form1.resx | DateTimePicker |
| Transfer Single Byte Transfer Multiple Bytes | A Label A LinkLabel |
| Write LoopBack Multi Byte | A LinkLabel |
| Send Byte 255 Address 2 | 232 ListView |
| Receive Byte | #- MaskedTextBox |
| | MonthCalendar |
| | |
| 🕐 timerUSB | NumericUpDown |
| | PictureBox ProgressBar |
| 💫 Solution Explorer 💐 Class View | Progressbar PadioRutton T |
| Error List | - ₽ × |
| 😮 0 Errors 🔢 1 0 Messages | |
| Description File Line | Column Project |
| | |
| | |
| 🚴 Error List 🖳 Find Results | |
| Ready | <u></u> |

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



| 📴 E | PT_Transfer_Test - Microsoft Visual C# 2010 Exp | ress | | | |
|------------------|---|---------------------------------|--|------|-----------------------------------|
| File | Edit View Project Debug Data Forma | at Tools Window Help | | | |
| | 1 🛅 • 💕 🛃 🥔 🐰 🗈 🛍 🍏 • (* - | 🕨 🖄 tooltip | - 🔍 🖀 🏷 🛃 🖳 🖕 | | |
| ÷ ‡ | 비 [[[후 희] ㅠ ~ 프] 글 때 [[[] [[[] [[[] | 마 많 많 많 음 탉 탉 타 균 [| | | |
| | Solution Explorer 🛛 👻 F | Form1.cs* Form1.Designer.cs* Fo | orm1.cs [Design]* × active_transfer.cs | - | Toolbox → 및 × |
| Database Explore | | | | * | > All Windows Forms |
| abası | Solution 'EPT_Transfer_Test' (1 project) | 🖳 Active Host | | | Common Controls Pointer |
| Ē | EPT_Transfer_Test Properties | | Number | | Pointer ab Button ■ |
| plore | Properties References | • | | = | CheckBox |
| | active_transfer.cs | Buttor L. Close | Description | | CheckedListBox |
| | ▲ Form1.cs | Buttor | Description | | ComboBox |
| | Form1.Designer.cs | | | | DateTimePicker |
| | Program.cs | Transfer Single Byte | Transfer Multiple Bytes | | A Label |
| | | Write LoopBack | Multi Byte | | A LinkLabel |
| | | | | | ■ ListBox |
| | | Send Byte 255 Address 2 | | | 222 ListView |
| | | Receive Byte | | | MaskedTextBox |
| | | | | - | MonthCalendar |
| | | | | | NotifyIcon |
| | | 😳 timerUSB | | | PictureBox |
| | < > | | | | ProgressBar |
| | 🔄 Solution Explorer 😵 Class View | | | | PadioPutton |
| | Error List | | | | ~ ₽ × |
| | 🔕 0 Errors 🛛 🕂 2 Warnings 🗍 🕦 0 Message | 25 | | | |
| | Description | | File | Line | Column Project |
| | | | | | |
| | | | | | |
| | | | | | |
| | 📸 Error List 🗏 Find Results | | | | |
| Read | dv | | | | |

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.



| 명 EPT_Transfer_Test - Microsoft Visual C# 2010 Express | | |
|--|----------------|--|
| File Edit View Project Debug Data Format Tools Window Help | | |
| : 🛅 🖼 = 📸 🛃 🐇 🐁 🖄 🥙 = 🔍 = 🕨 🦉 tooltip 💿 = 🖓 🛣 🔆 🛃 🖳 🖕 | | |
| [前] [[수리] 그 에 데 엄간 않 않 ! ~ 않 않 않 않 않 다 요 집 / 관 정 ! 않 - | | |
| Solution Explorer 🔹 🕂 × Form1.cs* Form1.Designer.cs* Form1.cs [Design]* 🗙 active_transfer.cs | - Toolbo | x – a × |
| Solution 'EPT_Transfer_Test' (1 project) Properties References active Host Form1.cs Form1.cs Form1.resx Program.cs Program.cs Image: Program.cs Program.cs Program.cs References Proml.resx Program.cs Program.cs Program.cs | | Mindows Forms Pointer Pointer Button CheckBox CheckedListBox ComboBox DateTimePicker Label LinkLabel ListBox ListView MaskedTextBox MonthCalendar |
| ✓ IIII → Rolution Explorer Reg Class View | | NotifyIcon NumericUpDown PictureBox ProgressBar PadiaButton |
| Error List | | - 4 × |
| 3 0 Errors 2 Warnings 0 Messages | | |
| Description File | Line Colur | mn Project |
| Ready | 15,71 <u>-</u> | 2 × 4i |

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 UnoProLogic Development System CD installed on your PC. Browse to the Projects_ActiveHost folder. 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.

| Irganize 👻 🔟 Open with Share with 💌 Burn | N | ew folder | | | |
|--|---|---------------------------------------|--------------------|--------------------|--------|
| 🌗 Arduino_IDE | * | Name | Date modified | Туре | Size |
| Documentation | | ActiveHost64.dll | 3/2/2013 9:44 PM | Application extens | 27 K |
| Drivers | | EPT_Transfer_Test.exe | 3/2/2013 11:28 PM | Application | 28 KI |
| Projects_ActiveHost_32Bit | | EPT_Transfer_Test.pdb | 3/2/2013 11:28 PM | Program Debug D | 56 KE |
| Projects_ActiveHost_64Bit | | EPT_Transfer_Test.vshost.exe | 3/2/2013 11:21 PM | Application | 12 KI |
| ActiveHost_1.0.0.8 | | EPT_Transfer_Test.vshost.exe.manifest | 8/31/2009 12:40 AM | MANIFEST File | 1 KE |
| EPT_Data_Collector EPT_Transfer_Test EPT_Transfer_Test EPT_Transfer_Test Din | | 🚳 ftd2xx64.dll | 1/18/2013 3:54 PM | Application extens | 252 KE |
| | | | | | |
| | | | | | |
| Debug | | | | | |
| Release | | | | | |
| iii kacuse iii x64 iii Debug | | | | | |
| | | | | | |
| 3 Release | | | | | |
| 🌗 obj | | | | | |
| Properties | - | | | | |

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.



| _ | EPT_Tran | nsfer_Test - Micro | osoft \ | Visual (| C# 2010 Exp | ress | | | X |
|------------------|-------------------|--------------------------------|----------|-----------|---------------------------------|----------|------------|---|-------|
| File | Edit | View Project | Deb | oug D | ata Tools | Window | Help | | |
| | - 🔝 ר | 🗃 🖬 🗿 🛛 🖌 | Þ | Start | Debugging | | F5 | - 💀 🕾 🎋 🖬 🔮 - | |
| | 3 👞 🖗 | ⊾ 🗚 👘 📃 | # | Build | d Solution | | F6 | | |
| 1 | | n Explorer | %⊒ (⊒ | | | | F11 F10 | | ų× |
|)ataba: | | 🚱 🛃 🗵 📰 olution 'EPT_Trar | 11 | | gle Breakpoin | nt | | EPT_Transfer_Test | _ |
| Database Explore | ▲ [2] ⊳ | EPT_Transfer_ Properties | | Wind | | | • | <pre>ic bool BlockTransferInfinite = false; ic bool BlockTransferReps = false;</pre> | |
| orer | Þ | References | | | r All DataTips rt DataTips | | | <pre>ic bool BlockTransferStop = false; ate void btnOpen Click(object sender, System.EventArgs e)</pre> | |
| | 1 | Form1.cs Form1.i Form1.r | | · · · | ort DataTips . ons and Setti | | | Open_Selected_Device(); | |
| | ۲ (کې Sol | Program.cs | | lass Viev | • • 10 | 00 % - 4 | ł | <pre>vate void btnClose_Click(object sender, System.EventArgs e) if (m_hPort != 0) { fontinue = 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; "" * </pre> | · ₽ × |
| | Error Li | st | | | | | | ▼ | Ψ× |
| | 00 | Errors 🛛 🔥 2 Wa | irning | js (i |) 0 Message | IS | | | |
| | | Description | | | | | | File Line Column Project | |
| | | | | | | | | | |
| | | | | | | | | | |
| | 📷 Err | ror List 🖳 Find I | Result | ts | | | | | |
| Rea | dy | | | | | | | Ln 58 Col 35 Ch 35 IN | IS 🔡 |

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".

| | Find Results | |
|------|-----------------------------|------|
| | ◎ @ ■ 素 酏 | |
| | | |
| | | |
| | | |
| | 😫 Error List 🔄 Find Results | |
| | | |
| Buil | d succeeded | Ln 1 |

6.2.6 Testing the Project

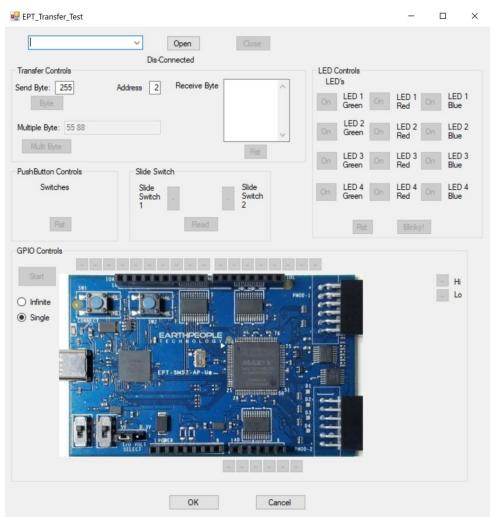
Once the project has been successfully built, it produces an *.exe file. The file will be saved in the Release or Debug folders.



| rganize 👻 💼 Open Share with 👻 Burn New | / fold | | | | |
|--|--------|--|--|--|--|
| Arduino_DE Documentation Drivers Projects_ActiveHost_32Bit ActiveHost_10.08 EPT_Data_Collector EPT_Transfer_Test EPT_Transfer_Test Debug Release M54 EPEses | | Name ActiveHost64.dll EFT_Transfer_Test.pdb EFT_Transfer_Test.vshost.exe EFT_Transfer_Test.vshost.exe.manifest 6 ftd2xx64.dll | Date modified 3/2/2013 9:44 PM 3/2/2013 11:28 PM 3/2/2013 11:28 PM 8/31/2009 12:40 AM 1/18/2013 3:54 PM | Type Application extens Application Program Debug D Application MANIFEST File Application extens | Size 27 KB 28 KB 56 KB 12 KB 1 KB 252 KB |
| obj Properties | | | | | |

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





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



| Byte Image: State st | 🖷 EPT_Transfer_Test | - 🗆 X |
|--|--|--|
| Send Byte: 255 Address 2 Point Multiple Byte: 55 88 Multiple Byte: 55 88 Multiple Byte: Switches Slide Switch Slide Switch <t< td=""><td>EPT JTAG Blaster 0 EPT Serial Communications 0</td><td>LED Controls</td></t<> | EPT JTAG Blaster 0 EPT Serial Communications 0 | LED Controls |
| Bite Multiple Byte: 55 88 Multiple Byte: 55 88 Red Red </td <td></td> <td>LED's</td> | | LED's |
| Multiple Byte: 35 25 Multiple Byte: 35 25 Multiple Byte: 95 25 | Byte | On Green On Red On Blue |
| Multi Byte PushButton Controls Switches Switches Side Switch 1 Ret Blue Side Switch 1 Ret Blue Switch 1 Ret Blue Switch 1 Ret Blue Switch 1 Ret Blue Blue Blue Ret Blue | | |
| PushButon Controls Switches Slide Switch Sl | Multi Byte Rst | LED 3 LED 3 LED 3 |
| Switch 1 2 Witch 2 Ref Bud GPIO Controls Start • Infinite • Single | PushButton Controls Slide Switch | |
| Ret Red Binkyt GPIO Controls Image: I | Switch Switch | On LED 4 On LED 4 On LED 4 Green On Red On Blue |
| GPIO Controls Infinite Single Finite Finit | | and another |
| Start C Infinite Single FARTHPROOLS FA | | Hist Blinky! |
| | Satt Infinite Single Infinite FT : SH37: AP Usin | |
| OK Cancel | | |

Click on one of the LED buttons in the middle of the window. The corresponding LED on the UnoProLogic-U2 board should light up.

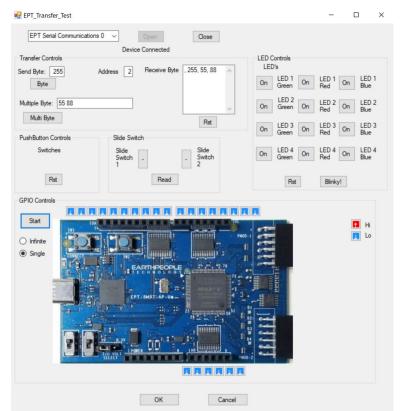
To exercise the Single Byte Transfer EndTerm, click the "Byte" button in the Transfer Controls group. Type in several numbers separated by a space and less 256 into the Multiple Byte textbox. Then hit the Multi Byte button. The numbers appear in the Receive Byte textbox.





To exercise the Block Transfer EndTerm, click the "Start" button in the GPIO Controls group. The UnoProLogic will sample the state of each Input pin from around the board and display the result of each pin on the window.

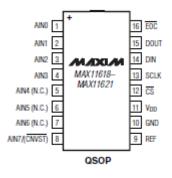




Press the PCB switches on the UnoProLogic to view the Switch Controls in action.

7 Using the Analog to Digital Converter

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



Page 129



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

7.1 Register Descriptions

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



| REGISTER NAME | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
|---------------|-------|--------|--------|--------|---------|---------|--------|--------|
| Conversion | 1 | CHSEL3 | CHSEL2 | CHSEL1 | CHSEL0 | SCAN1 | SCAN0 | Х |
| Setup | 0 | 1 | CKSEL1 | CKSEL0 | REFSEL1 | REFSELO | Х | Х |
| Averaging | 0 | 0 | 1 | AVGON | NAVG1 | NAVG0 | NSCAN1 | NSCANO |
| Reset | 0 | 0 | 0 | 1 | RESET | Х | Х | Х |

Table 1. Input Data Byte (MSB First)

X = Don't care.

7.2 Conversion Register

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

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

Table 2. Conversion Register*

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



| CHSEL1 | CHSEL0 | SELECTED CHANNEL (N) |
|--------|--------|-------------------------|
| 0 | 0 | AINO |
| 0 | 1 | AIN1 |
| 1 | 0 | AIN2 |
| 1 | 1 | AIN3 |

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

7.3 Setup Register

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



Table 3. Setup Register*

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

*See below for bit details.

| CKSEL1 | CKSEL0 | CONVERSION CLOCK | ACQUISITION/SAMPLING | CNVST CONFIGURATION |
|--------|--------|-----------------------|--------------------------------|---------------------|
| 0 | 0 | Internal | Internally timed | CNVST |
| 0 | 1 | Internal | Externally timed through CNVST | CNVST |
| 1 | 0 | Internal | Internally timed | AIN15/AIN11/AIN7** |
| 1 | 1 | External (4.8MHz max) | Externally timed through SCLK | AIN15/AIN11/AIN7** |

**For the MAX11618/MAX11619, CNVST has its own dedicated pin.

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

7.4 Averaging Register

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



| Table 4. Averaging Register* | |
|------------------------------|--|
|------------------------------|--|

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

*See below for bit details.

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

X = Don't care.

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

7.5 Reset Register

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



| | | 5 | | |
|----------|---------|--|--|--|
| BIT NAME | BIT | FUNCTION | | |
| _ | 7 (MSB) | Set to 0 to select reset register. | | |
| _ | 6 | Set to 0 to select reset register. | | |
| _ | 5 | et to 0 to select reset register. | | |
| _ | 4 | Set to 1 to select reset register. | | |
| RESET | 3 | Set to zero to reset all registers. Set to 1 to clear the FIFO only. | | |
| Х | 2 | Don't care. | | |
| Х | 1 | Don't care. | | |
| Х | 0 (LSB) | Don't care. | | |

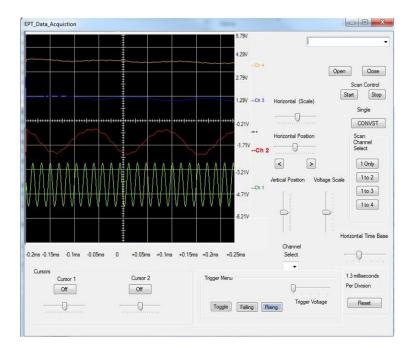
Table 5. Reset Register

8 The UnoProLyzer Application

The source DVD for the UnoProLogic2 comes with the UnoProLyzer application project. This project allows the user to display 1 to 4 channels of analog input in a graphing application. The project utilizes the PC to perform all data storage and graphing. The PC sends commands to and receives the data from the UnoProLogic2 and stores each channel data in its own separate buffer in memory. The UnoProLyzer collects all samples from each channel by streaming across up to four dedicated communication "pipes".

The UnoProLogic2 commands the ADC to start a conversion on the channels selected by the user. It then waits for the ADC to complete the conversion on all channels. It transfers the data for each channel across its own dedicated communication pipe. Then starts the process over again. The UnoProLyzer application will accept each data word and decode the pipe number it came across. It stores each word into a separate buffer for each channel. The UnoProLyzer then performs post processing on each data word. It performs trigger detection, smoothing, sorting, scaling and searching. It then displays the data set in 500 data point segments.





8.1 Accessing the UnoProLyzer Application

Locate the EPT_Data_Acquistion folder in the Drivers folder of the UnoProLogic Development System DVD using Windows Explorer.

| ganize ▼ Include in library ▼ Share with ▼ Burn No | ew folder | | | | |
|--|-----------|----------------------------|--------------------|---------------------|-------|
| 🕌 CTS Crystal | * | Name | Date modified | Туре | Size |
| Diodes Incorporated | | 퉬 bin | 5/12/2015 11:34 PM | File folder | |
| | | 퉬 obj | 5/12/2015 11:34 PM | File folder | |
| Earth People Technology | | Properties | 5/12/2015 11:34 PM | File folder | |
| EPT I2C Project CD | | active_transfer_x64.cs | 3/6/2015 9:14 AM | Visual C# Source f | 10 KB |
| EPT Projects Folders UNOPROLOGIC2_USB_CPLD_PROJECT_1P4_DVD | | EPT_Data_Acquistion.csproj | 3/18/2015 10:07 PM | Visual C# Project f | 5 KB |
| Documentation | | Eorm1.cs | 5/12/2015 11:22 PM | Visual C# Source f | 47 KB |
| Documentation | | Form1.Designer.cs | 3/26/2015 11:26 PM | Visual C# Source f | 42 KB |
| Projects_ActiveHost_32Bit | E | 🛃 Form1.resx | 3/26/2015 11:26 PM | .NET Managed Re | 6 KB |
| Projects_ActiveHost_52Bit | | 🐴 Program.cs | 3/2/2015 10:59 PM | Visual C# Source f | 1 KB |
| ActiveHost 1.0.0.10 | | ScaleFactorMenu.cs | 3/12/2015 12:19 AM | Visual C# Source f | 14 KB |
| EPT_Data_Acquisition | | SignalSetup.cs | 3/28/2015 12:17 AM | Visual C# Source f | 30 KB |
| EPT_Data_Acquisition | | | | | |
| EPT_Data_Collector | | | | | |
| EPT_Transfer_Test | | | | | |
| EPTActiveHostConsoleClient | | | | | |
| Projects_Arduino | - | | | | |

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



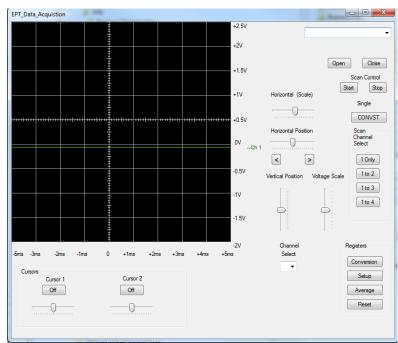
| - 🕆 🖡 - Jolly > Products > Earth People Technology > UNOPRO | LOGIC_USB_CPLD_PROJECT_4.5_DVD > Projects_ActiveHost_64Bit > | EPT_Data_Acquistion > Uno | ProLyzer > EPT_Data_Acquis | tion > bin > x64 > |
|---|--|---------------------------|----------------------------|--------------------|
| ✓ | ↑ □ Name | Date modified | Туре | bize |
| Documentation | EPT_Data_Acquistion.vshost.exe | 3/3/2015 9:08 AM | Application | 12 KB |
| > 📜 Drivers | UnoProLyzer.exe | 1/6/2022 6:11 PM | Application | 54 KB |
| ✓ | UnoProLyzer.pdb | 1/6/2022 6:11 PM | Program Debug D | 88 KB |
| > 🖡 ActiveHost_1.0.0.11 | | | | |
| ✓ | | | | |
| ✓ | | | | |
| EPT_Data_Acquistion | | | | |
| 🗸 📙 bin | | | | |
| Debug | | | | |
| ✓ <mark></mark> ×64 | | | | |
| Debug | | | | |
| Release | | | | |
| > <mark> </mark> obj | • | | | |

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

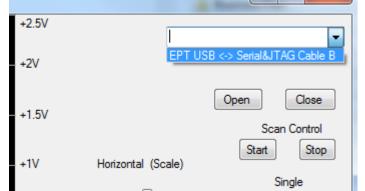


Double click on the EPT_Data_Acquisition.exe file and the UnoProLyzer Application will open.





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



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





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

| a family | |
|---------------------------------|---|
| EPT USB <> Se | rial&JTAG Cable 🔻 |
| Open Horizontal (Scale) | Close Scan Control Start Stop Single CONVST Scan |
| Ch 1 | Channel Select |
| Vertical Position Voltage Scale | 1 to 2 1 to 3 1 to 4 |
| Channel Select | Registers Conversion Setup Average Reset |

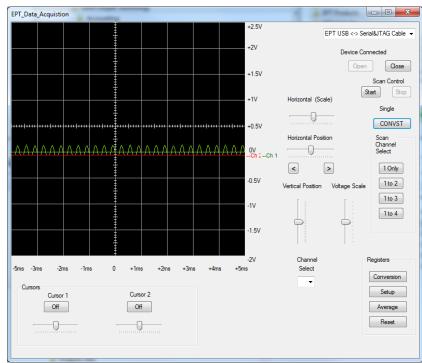
Next, click on the Start button.



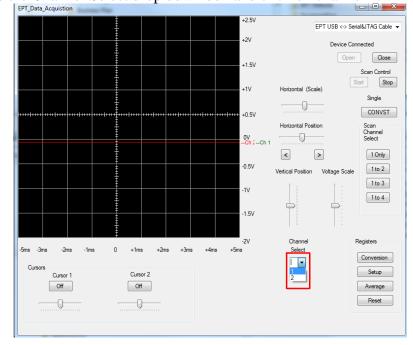
| / | EPT | USB <-> Seria | al&JTAG Cable 👻 |
|-----------|--------------------|---------------|--|
| / Ch 1 | Horizontal (Scale) | L | Close Scan Control att Stop Single CONVST Scan Channel Select 1 Only 1 to 2 1 to 3 1 to 4 |
| | Channel Select | F | Conversion Setup Average Reset |

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



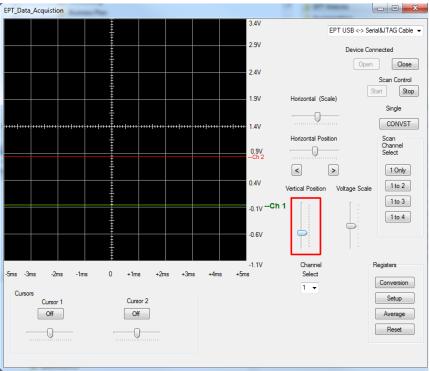


Next, locate the Channel Select drop down box and click on channel 1.





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



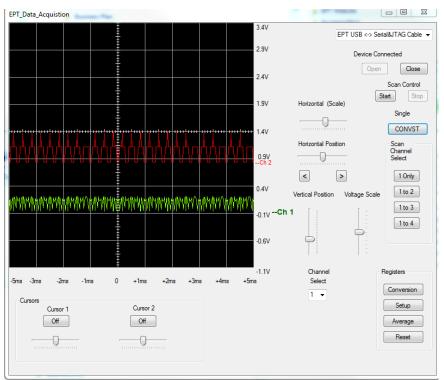
The selected channel will show up as a large icon. Its position indicates the zero position of the data. The magnitude information along the y-axis is only for the selected channel.

Then connect a signal to the channel 1 input on the UnoProLogic2.





If you don't have a 0-5 Volt signal to connect to the UnoProLogic2, you can use your finger and touch it to the bottom of the Analog Input Connector. The ambient electricity from your body has just enough current to give the Analog inputs a deflection from zero.



Now the UnoProLogic2 and UnoProLyzer are ready to measure an 0-5VDC signals.

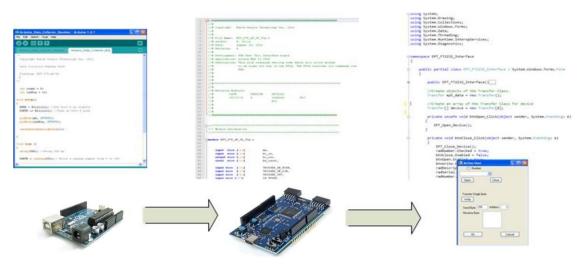


9 UnoProLogic Development Process

There is no standard for developing embedded electronics. The best method is the one that works for the user. These methods can range from a top down approach where the design is written down first and all code is written, then compile, execute and test. Or a bottom up approach can be pursued where a small piece of the project is assembled and verified (i.e. I2C communication to a sensor). Then the next piece is assembled and verified (i.e. collect sensor data in a storage buffer) and connected to the first. And so on, until the whole design is complete. Or, you could use any combination of these two extremes.

9.1 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, UnoProLogic-U2 CPLD, and a Windows 7 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.



9.1.1 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.

9.1.2 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 UnoProLogic 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.

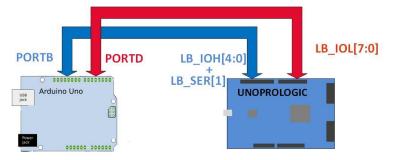
The randomSeed() function must be called during the setup() function. It takes as input parameter the output of the Analog Pin 1. The output of this Pin 1 will have a small amount of random noise on it. Because of this noise, the randomSeed() function will produce a different seed every time the sketch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

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

9.1.3 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 UnoProLogic. 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.

PORTD maps to Arduino digital pins 0 to 7

DDRD - The Port D Data Direction Register - read/write PORTD - The Port D Data Register - read/write PIND - The Port D 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.

```
int ledPin = 13;
void setup()
{
    DDRD = B1111111; //Set Port D as outputs
    PORTD &= B1111111; //Turn on Port D pins
    pinMode(A0, OUTPUT);
}
```



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

9.1.4 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 UnoProLogic. This output pin (from the UnoProLogic) 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 UnoProLogic on J10 PIN 1. On the Arduino, this is PIN 8 of the Digital pins. Each iteration of the loop() function, the startStopBit variable stores the state of DigitalPin8. 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 UnoProLogic. 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:

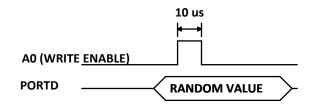


```
/*
  Copyright Earth People Technology Inc. 2012
 Data Collector Random Seed
 Platform: EPT-570-AP-U2
*/
 int startStopBit = 0;
 int count = 0;
 int ledPin = 13;
 int inPin8 = 8;
void setup()
{
 DDRD = B11111111; //Set Port D as outputs
 PORTD &= B11111111; //Turn on Port D pins
 pinMode(A0, OUTPUT);
 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
     PORTD = random(255);
     //Set the Write Enable Pin High
       digitalWrite(A0, HIGH);
```



```
//Set the LED Pin High
digitalWrite(ledPin, HIGH);
delay(500); //Delay 500 ms
//Set the LED Pin Low
digitalWrite(ledPin, LOW);
//Set teh Write Enable Pin Low
digitalWrite(A0, LOW);
}
```

Notice that PORTD 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 A0 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 A0, this produces a write enable pulse of 10 microseconds.



The RANDOM VALUE will be stable before the A0(WRITE ENABLE) asserts thus guaranteeing a successful transfer of data from Arduino to CPLD.

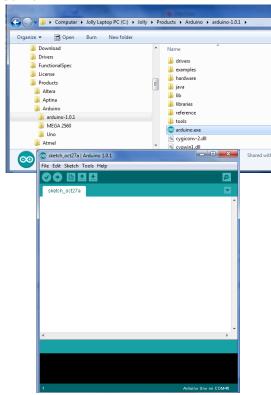
9.1.5 Building Arduino Project

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 UnoProLogic Development System CD. Copy Arduino_Data_Collector_Code_U2.ino .



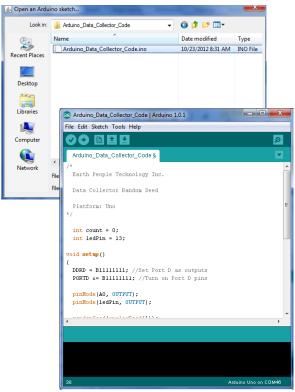
To compile your code,

• Open up the Arduino IDE



• Load your code into the Sketch.



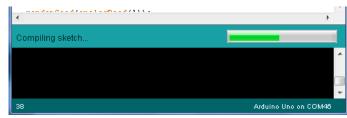


• Click the Verify button



• The sketch will compile





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

| | F. |
|---|------------------|
| Done compiling. | |
| | ^ |
| Binary sketch size: 1,786 bytes (of a 32,256 byte max | (imum) |
| 38 Ardu | ino Uno on COM46 |

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

9.1.6 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

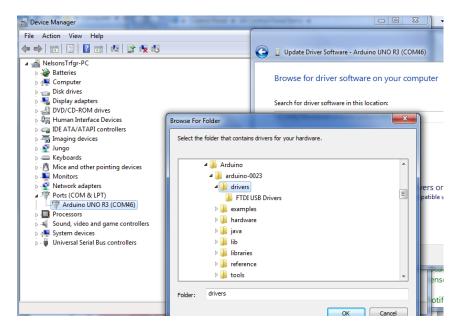
• Connect the USB cable from PC to Arduino



- 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 UNO (COMxx)"
- Right click on the "Arduino UNO (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 Uno's driver file, named "ArduinoUNO.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.



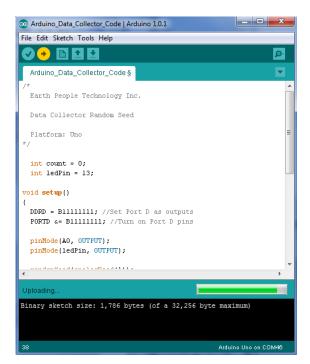
• Once the driver is loaded, we can set the COM Port. Click on Tools and select Serial Port, then click the available port.



| 💿 Arduino_Data_Coll | ector_Code Arduino | 1.0.1 | | X |
|--|---------------------------------------|------------------|-------------------|----------|
| File Edit Sketch To | ols Help | | | |
| | Auto Format | Ctrl+T | | 9 |
| Arduino Data | Archive Sketch | | | |
| /* | Fix Encoding & Rela | | | |
| Earth People | Serial Monitor | Ctrl+Shift+M | | - Â |
| | Board | | | |
| Data Collec | Serial Port | ۱. | ✓ COM46 | |
| Platform: U | Programmer | + | | E |
| */ | Burn Bootloader | | | |
| <pre>int count = 0; int ledPin = 1</pre> | 3: | | 1 | |
| PORTD &= B1111 | 11; //Set Port D 1111; //Turn on P | - | | |
| pinMode(AO, OU pinMode(ledPin | | | | |
| • | 1 | | | + |
| Done compiling. | | | | |
| | | | | * |
| Binary sketch si: | e: 1,786 bytes (| of a 32,256 byte | maximum) | - |
| 38 | | | Arduino Uno on C(| DM46 |

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





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.



9.1.7 CPLD Active Transfer EndTerm Coding and Initiation

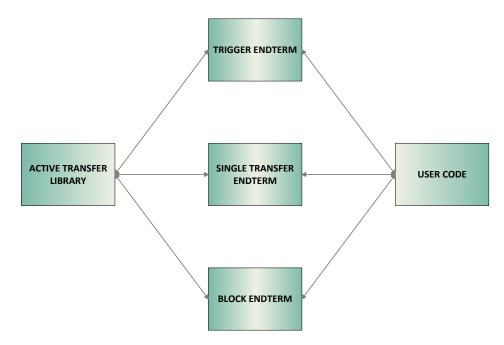
The UnoProLogic will accept the data collected by the Arduino and transfer it to the PC. It is designed to plug directly into the Arduino Uno and there is no need for external wires to be added. The Active Transfer EndTerms are used to connect the Active Transfer Library to the user code. This makes it easy to transfer data to and from the PC via the USB. The user needs to create a state machine to control the transfer between



the incoming data and the Active Transfer EndTerms. We will now go through exercise of creating the CPLD code for the Data Collector Sampler.

9.1.8 <u>CPLD:</u> Define the User Design.

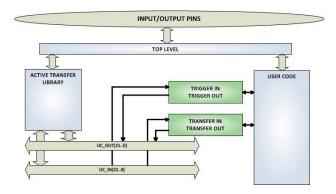
In this step we will define the user's code and include EndTerms and the EPT Active Transfer Library. The Active Transfer Library contains a set of files with a ".vqm" name extension which select particular operations to perform (e.g., byte transfer, block transfer, trigger). The active_transfer_library.vqm file must be included in the top level file of the project. The EndTerms 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 Project CD.



We will build our CPLD project using Quartus Prime software from Intel. The primary file defining the user's CPLD project is named "EPT_570_AP_U2_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 J8 connector and a write enable on Pin 1 of J9. 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 A0 (write 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_U2_Top (

| input wire | [1:0] | aa, | | |
|-------------|-------|------------------|--------|-----|
| input wire | [1:0] | bc_in, | | |
| output wire | [2:0] | bc_out, | | |
| inout wire | [7:0] | bd_inout, | | |
| | | | | |
| input wire | [1:0] | TRIGGER_IN_HIGH, | //XIOH | J10 |
| input wire | [5:0] | TRIGGER_IN_LOW, | //AD | J9 |
| output reg | [7:0] | LB_LOWER, | //XIOH | J10 |
| input wire | 17.01 | LB UPPER, | //XTOT | .78 |
| imput with | [7:0] | LD_OFFER, | //AIOD | |

| <pre>//Transceiver Control output reg output reg</pre> | Signals TR_DIR_1, TR_OE_1, |
|--|----------------------------------|
| output wire | TR_DIR_2, |
| output wire | TR_OE_2, |
| output wire | TR_DIR_3, |
| output wire | TR_OE_3, |
| input wire | SW_USER_1, |
| input wire | SW_USER_2, |
| <pre>output reg [2:0] output wire);</pre> | LED, LED3 |



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

| // // Parameters // | |
|---|---|
| //Header Bytes for the Tra | nsfer Loopback detection |
| parameter | TRANSFER_CONTROL_BYTE1 = 8'h5A; |
| parameter | TRANSFER_CONTROL_BYTE2 = 8'hC3; |
| parameter | <pre>TRANSFER_CONTROL_BYTE3 = 8'h7E;</pre> |
| //State Machine Transfer L parameter | oopback detection 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;
                         UC OUT;
   wire [21:0]
   //Trigger Signals
   reg [7:0]
                         trigger out;
   wire [7:0]
                         trigger in byte;
                         trigger in store;
   reg [7:0]
   //LED registers
   reg
                         led reset;
   //Switch registers
   reg
                         switch reset;
   //Transfer registers
                         transfer out;
   wire
                         transfer_out_reg;
   reg
                        transfer_in_received;
   wire
                        transfer in byte;
   wire [7:0]
   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;
                          transfer control state;
   reg [3:0]
                          transfer control byte;
   reg [7:0]
   //Transfer Write from Arduino
   reg
                         transfer write reg;
   reg
                         transfer write;
                         transfer write byte;
   reg [7:0]
   //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
              TR DIR 2 = 1'b1; //1 = A to B; 0 = B to A
              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;
  assign
  //Clock and Reset
  assign
             CLK 66 = aa[1];
             RST = reset;
  assign
  assign
              reset = reset signal reg;
  //Transfer registers
  assign
              transfer out = transfer out reg | transfer write;
              transfer out byte = transfer write byte;
  assign
  //LED3 is used to signify to the user that the Start
  //switch is enabled
  assign
               LED3 = ~start stop cntrl;
```

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



```
11*
      Reset Signal
always @(posedge CLK IN or negedge aa[0])
   begin
    if(!aa[0])
    begin
         reset signal reg <= 1'b0;</pre>
         reset counter <= 0;</pre>
    end
    else
    begin
      if ( reset_counter < GLOBAL_RESET_COUNT )
      begin
         reset_signal_reg <= 1'b0;</pre>
         reset counter <= reset counter + 1'b1;
      end
      else
      begin
         reset signal reg <= 1'b1;</pre>
      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;</pre>
  end
  else
 begin
    if(!SW USER 1 )
         trigger out <= 8'h01;</pre>
    else if(!SW USER 2 )
         trigger out <= 8'h02;</pre>
     else if (switch reset)
         trigger out <= 8'h04;</pre>
     else
         trigger out <= 8'h00;</pre>
  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 A0(Write 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 TRIGGER_IN_LOW[1] to a high. The TRIGGER_IN_LOW[1] pin is routed to Pin 1 of J9 which is routed to the A0(Write 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_UPPER[7:0] pins to the transfer_write_byte register. By setting 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_module



and sets up the byte transfer to the PC. When the TRIGGER_IN_LOW[1] -A0(Write Enable) pin goes low, the encoder will reset transfer_write_reg and transfer_write to low. The encoder goes back to waiting for the TRIGGER_IN_LOW[1] -A0(Write Enable) to assert high.

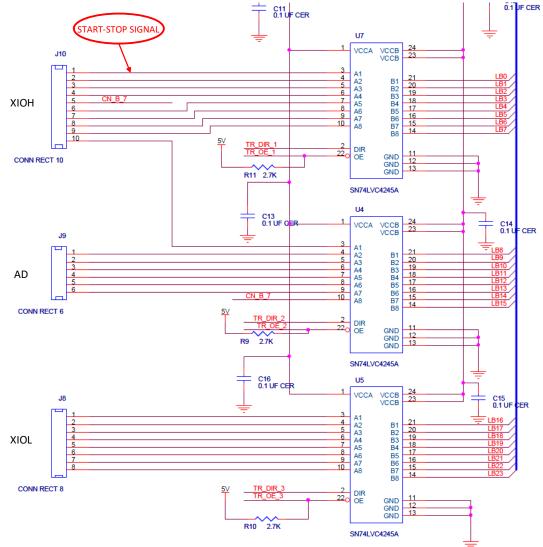
```
//--
// Detect Transfer From Arduino
//-----
always @ (posedge CLK IN or negedge RST)
begin
    if (!RST)
    begin
         transfer write reg <= 1'b0;</pre>
         transfer write <= 1'b0;</pre>
         transfer write byte <= 0;</pre>
    end
    else
    begin
         if (TRIGGER IN LOW[1] & !transfer write reg)
         begin
              transfer write reg <= 1'b1;</pre>
              transfer write <= 1'b1;</pre>
              transfer write byte <= LB UPPER;
         end
         else if (TRIGGER IN LOW[1] & transfer write reg)
         begin
              transfer write reg <= 1'b1;</pre>
              transfer write <= 1'b0;</pre>
         end
         else if(!TRIGGER IN LOW[1] & transfer write reg)
         begin
              transfer write reg <= 1'b0;</pre>
              transfer write <= 1'b0;</pre>
              transfer write byte <= 0;</pre>
         end
    end
```

end

This block of code takes care of reading the random word from the Arduino using the A0(Write Enable) Pin. However, because the Arduino is expecting a Start/Stop bit on Digital Pin8, the CPLD code has to provide this bit. This presents a problem, the UnoProLogic has 3 eight bit bi-directional ports. Which means each port is has a

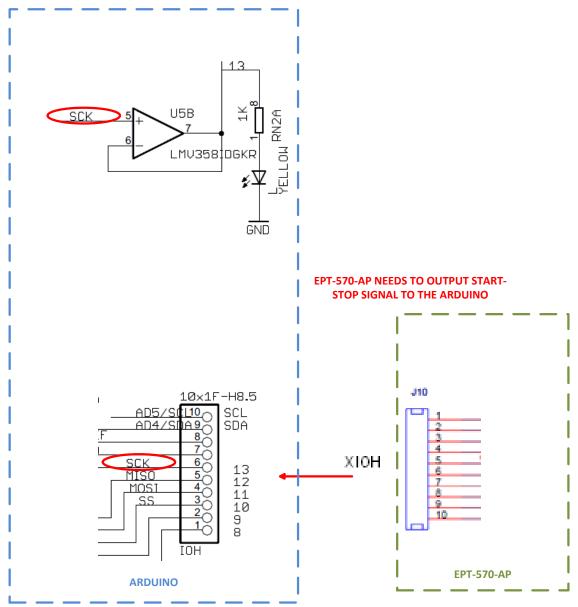


direction which is either input or ouput at a given time. However, the ports can be switched between input and output at any time. Two of the three ports must be used as inputs into the CPLD for the random word and the A0(Write Enable) Pin. So, the third port can be used as the output port.



This, however, causes another problem! The Arduino XIOH connector needs to output the Amber LED state. So, if one pin on the connector needs to be an output, the UnoProLogic port on J10 (XIOH) cannot be an output! This would interfere with the turning on and turning off of the LED.





So, we can fix this problem by noting that the 8 bit bi-directional ports on the UnoProLogic have Output Enables that allow the CPLD to "float" the signals of the port at any time. By floating the port, we can multiplex the signals of the port. When we need to drive the signals from the UnoProLogic port to the Arduino, we turn on the Output Enables of the port. And when we need to let the Arduino drive its signals, we turn off the Output Enables of the port.



In the reset section of the synchronous block, 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. So, if the start_stop_cntrl signal is set, the Output Enable is turned on and the signal will appear on DigitalPin8 on the Arduino XIOH connector. As the Data Collector code cycles through its loop() function, it will cause the if statement to branch into its conditional statement. The Data Collector code will assert the A0(Write Enable) Pin in its conditional statement. The A0(Write Enable) Pin will cause the CPLD code to enter into its first conditional statement. This first statement turns off the Output Enables of the Port J10. With the Port turned off, the Arduino can set the LED on when it executes its code. When the A0(Write Enable) Pin is de-asserted, the Output Enable of Port J10 is turned back on and the whole process can start over.

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 IN or negedge RST)
begin
     if (!RST)
     begin
          transfer_in_received_reg <= 1'b0;</pre>
          transfer control state <= TRANSFER LOOPBACK IDLE;
          transfer in loop back <= 1'b0;</pre>
          led reset <= 1'b0;</pre>
          switch_reset <= 1'b0;</pre>
      end
      else
     begin
          if(transfer in received & !transfer in received reg)
          begin
              transfer in received reg <= 1'b1;</pre>
              case(transfer control state)
              TRANSFER CONTROL IDLE:
                  if((transfer in byte == TRANSFER CONTROL BYTE1))
                      transfer_control_state <= TRANSFER CONTROL HDR1;</pre>
                  else if((transfer in byte != TRANSFER CONTROL BYTE1))
                      transfer control state <= TRANSFER CONTROL IDLE;</pre>
                  else
                     transfer control state <= TRANSFER CONTROL IDLE;
              TRANSFER CONTROL HDR1:
                  if((transfer in byte == TRANSFER CONTROL BYTE2))
                      transfer control state <= TRANSFER CONTROL HDR2;</pre>
                  else if((transfer in byte != TRANSFER CONTROL BYTE2))
```



```
transfer_control_state <= TRANSFER_CONTROL_IDLE;</pre>
                  else
                      transfer control state <= TRANSFER CONTROL HDR1;
              TRANSFER CONTROL HDR2:
                  if((transfer in byte == TRANSFER CONTROL BYTE3))
                      transfer control state <= TRANSFER DECODE BYTE;</pre>
                  else if((transfer in byte != TRANSFER CONTROL BYTE3))
                      transfer control state <= TRANSFER CONTROL IDLE;
                  else
                      transfer_control_state <= TRANSFER CONTROL HDR2;</pre>
              TRANSFER DECODE BYTE:
              begin
                  transfer in loop back <= transfer in byte[0];</pre>
                  led reset <= transfer in byte[2];</pre>
                  switch reset <= transfer in byte[3];</pre>
                  transfer loopback state <= TRANSFER LOOPBACK 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;</pre>
     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 ]),
                          (transfer_out) ,
    .start_transfer
    .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.

9.1.9 <u>CPLD:</u> 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 | (X |
|--|-----|
| 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 | |
| Use Existing Project Settings | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| < Back Next > Finish Cancel Hi | elp |
| | |

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_U2_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_Data_Collector \EPT_570_AP_U2_Top folder of the UnoProLogic Development System CD. Copy the files from the \src directory.

- Active_transfer.vqm
- Active_trigger.vqm
- Active_transfer_library.vqm
- eptWireOr.v
- ETP_570_AP_U2_Top.v

Add the files:

| General | Files | | | | | |
|---|------------|---|--|---------------|---|-------------|
| Files Libraries Operating Settings and Conditions Voltage | | esign files you want to include in the | project. Click Add All to add all de | sign files in | the project direct | tory to the |
| Temperature | File name: | | | | | Add |
| Compilation Process Settings Early Timing Estimate | File Name | | Туре | Library | Design Entry/ | Add All |
| Incremental Compilation Physical Synthesis Optimizations | /src/E | ptWireOR.v PT_570_AP_U2_Top.v active_trigger.vqm | Verilog HDL File Verilog HDL File Verilog Overhein Manning F | :l- | <none> <none> <none></none></none></none> | Remove |
| EDA Tool Settings Design Entry/Synthesis Simulation | /src/a | active_transfer_library.vqm active_transfer_library.vqm active_transfer.vqm | Verilog Quartus Mapping File Verilog Quartus Mapping File Verilog Quartus Mapping File | | <none> <none></none></none> | Up |
| Formal Verification Board-Level | | | | | Shorez | Down |
| Analysis & Synthesis Settings VHDL Input Verligg HDL Input Default Parameters Fitter Settings TimeQuest Timing Analyzer Assembler Design Assistant SignalTap II Logic Analyzer Logic Analyzer Interface PowerPlay Power Analyzer Settings SSN Analyzer | 4 | 11 | | | | |

Continue following the instructions by adding a device and finishing the project instantiation. Then, add the Pins.

• Under Assignments, Select Import Assignments.



- At the Import Assignment dialog box, browse to the \Projects_HDL\EPT_Data_Collector \EPT_570_AP_U2_Top folder of the UnoProLogic Development System CD. Select the Quartus Specification file, "EPT_570_AP_U2_Top.qsf".
- 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 \EPT_UnoProLogic_TOP folder of the UnoProLogic Development System CD. Select the "EPT_570_AP_U2_Top.sdc" file.

| ganize 🛪 📄 Open Share with 🛪 Burn I | New folde | | | | |
|-------------------------------------|-----------|--------------------------------------|-----------------------------------|----------------------|--------|
| | vew folde | · | | | |
| Earth People Technology | ^ | Name | Date modified 3/3/2013 1:12 AM | Type SUMMARY File | Size |
| EPT I2C Project CD | | EPT_570_AP_U2_Top.flow.rpt | 3/3/2013 1:12 AM | RPT File | 7 KB |
| EPT Projects Folders | | EPT_570_AP_U2_Top.jdi | 3/3/2013 1:12 AM | IDI File | 1 KB |
| EPT USB-CPLD Development System CD | | EPT_570_AP_U2_Top.map.rpt | 3/3/2013 1:12 AM | RPT File | 42 KB |
| 🌽 Arduino_IDE | | EPT_570_AP_U2_Top.map.smsg | 3/3/2013 1:12 AM | SMSG File | 1 KB |
| Documentation | | EPT_570_AP_U2_Top.map.summary | 3/3/2013 1:12 AM | SUMMARY File | 1 KB |
| Projects ActiveHost 32Bit | | EPT_570_AP_U2_Top.pin | 3/3/2013 1:12 AM | PIN File | 15 KB |
| Projects_ActiveHost_32Bit | E | EPT_570_AP_U2_Top.pof | 3/3/2013 1:12 AM | POF File | 15 KB |
| Projects_ActiveHost_04bit | | Tep: | 10/22/2012 12:04 | OPF File | 2 KB |
| Projects HDL | | EPT_570_AP_U2_Top.qsf | 1/28/2013 12:07 AM | OSF File | 6 KB |
| EPT Data Collector | | EPT_570_AP_U2_Top.qsf.bak | 10/22/2012 12:05 | BAK File | 3 KB |
| EPT_570_AP_M4_Top | | EPT_570_AP_U2_Top.qws | 3/3/2013 1:31 AM | QWS File | 2 KB |
| ET 1570 AP U2 Top | | EPT_570_AP_U2_Top.sdc | 1/24/2013 10:01 PM | SDC File | 4 KB |
| src | | EPT_570_AP_U2_Top.sta.rpt | 3/3/2013 1:12 AM | RPT File | 139 KB |
| EPT Transfer Test | | EPT_570_AP_U2_Top.sta.summary | 3/3/2013 1:12 AM | SUMMARY File | 1 KB |
| Quartus Programmer | - | EPT_570_AP_U2_Top_assignment_default | 1/24/2013 9:18 PM | QDF File | 56 KB |

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



| ganize 🕶 📄 Open 🛛 Burn New fold | ler | | | | | |
|---|-----|---------------------------|--------------------|-------------|------|--|
| 🎉 qdesigns | * | Name | Date modified | Туре | Size | |
| EPT_Data_Collector | | 🕌 db | 3/14/2013 12:07 AM | File folder | | |
| 🎉 EPT_Transfer_Test | | EPT_570_AP_U2_Top.qpf | 3/14/2013 12:06 AM | OPF File | 2 KB | |
| 🍌 fir_filter | | EPT_570_AP_U2_Top.qsf | 3/14/2013 12:07 AM | OSF File | 4 KB | |
| incr_comp_makefile | | EPT_570_AP_U2_Top.qsf.bak | 3/14/2013 12:08 AM | BAK File | 4 KB | |
| whole who | | EPT_570_AP_U2_Top.sdc | 1/24/2013 10:01 PM | SDC File | 4 KB | |
| end{tabular} end{ | E | | | | | |
| Jolly Business Opportunities Capitol_College Code_FPGA | | | | | | |

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:/altera/12.1sp1/quartus/qdesigns/E | PT_Data_Collector/EPT_570_AP_U2_Top - EPT_570_AP_U | _Top | | | |
|--|--|---|--|------------------|---------|
| File Edit View Project Assignments Processing | Tools Window Help 🕏 | | | Search altera.co | m 🚯 |
| 1 📑 💕 🗐 🥔 👗 😘 🔃 🕫 🕬 🕬 | 2_Top 🔹 😨 🔮 🖉 🔹 | > ⊘ ⊗ ⊙ છ ⊗ | . 🕘 🕲 👗 📵 🖻 | | |
| Project Navigator 🕴 🗗 🗙 | Compilation Report - EPT_570_AP_U2_Top | | | | |
| Entity | Table of Contents | Flow Summary | | | |
| MAX E EPAS/07LD0C5 ME EPT_5/0, AP, U2, Top Metrotic transfer: ACTIVE_TRANSFER_INST Metrotic transfer: ACTIVE_TRANSFER_L detive_transfer_lbmar:AACTIVE_TRANSFER_L detive_transfer_lbmar:AACTIVE_TRIGGER_INST Metrotic transfer_lbmar:AACTIVE_TRIGGER_INST Metrotic transfer_lbmar:AACTIVE_TRIGGER_INST Metrotic transfer_lbmar:AACTIVE_TRIGGER_INST | Flow Sammary Flow Stings Flow Stings Flow Shon-Default Global Settings Flow Elapsed Time Flow OS summary Flow Cog Flow Log Flow Flow Log | Flow Status Quartus II 64-Bit Version Revision Name Top-level Entity Name Family Device Timing Models | Successful - Thu Mar14 00:11: 12.1 Build 243 01/31/2013 SP 1 EPT_570_AP_U2_Top EPT_570_AP_U2_Top MAX II EPM570T100C5 Final | | |
| Herardry ≧ Files J ² Design Units ∰kits Tradis ⊕ Ø | Quartus II Full Compilation was successful (49 warnings) CK | Total logic elements Total pins Total virtual pins UFM blocks | Pinai 283 / 570 (50 %) 51 / 76 (67 %) 0 0 / 1 (0 %) | | |
| Task (✓ ▶ Analysis & Synthesis 0 ✓ ▶ Fatter (Place & Roste) 0 ✓ ▶ Fatter (Place & Roste) 0 ✓ ▶ Assembler (Generate programmin 0 ✓ ▶ Assembler (Generate programmin 0 ✓ ▶ TimeQuest Timing Analysis 0 ✓ ▶ DA Netlist Writer 0 ✓ ▶ Program Device (Open Programmer) | د ا | | | | |
| × AI 🙆 🛆 🖉 🚲 マ < <search>></search> | ~ | | | | |
| P Type ID Message 0 392102 Design is not fully const 0 Quarcus II 64-Bit TimeQue 0 Running Quarcus II 64-Bit TimeQue 0 Running Quarcus II 64-Bit Que 0 Command: quartus_edares 0 204018 Generated files "FEF 570 0 Quarcus II 64-Bit CDA Net | est Timing Analyzer was successful. 0 | ****** s_files=off EPT_57 _vhd.sdo" in direc 0 warnings | D_AP_U2_Top -c EPT_57 | | designs |

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.

9.1.10 <u>CPLD:</u> Program the CPLD

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





- Connect the UnoProLogic 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 v1.6" 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.

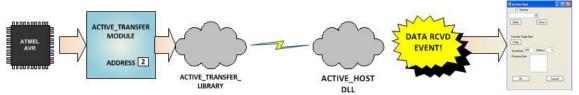
| | | indow Help 🐬 | | | | | Search alte | ra.com | |
|--------------------|----------------------------|-------------------------|------------------|----------|------------------------------|--------|-----------------|------------|---|
| Hardware Setup. | EPT-Blaster v1.3b [MB | JSB-0] | Mode: JTAG | | Progress | s: | 100% (Su | iccessful) | |
| nable real-time IS | P to allow background prog | ramming (for MAX II and | d MAX V devices) | | | | | | |
| Start Start | File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank- Check | Examine | S |
| Stop | output_files/EPT_570_ | A EPM570T100 | 00313A26 | FFFFFFF | V | | | | |
| | CFM | | | | \checkmark | | | | |
| etect | UFM | | | | \checkmark | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | 4 | | | | | | | | |
| ו | | | | | | | | | - |
| | | | | | | | | | - |
| •• | | | | | | | | | : |
| | | | | | | | | | |
| | TDI | | | | | | | | |
| | | | | | | | | | |



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

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

9.1.12 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.



| ⊢ <mark>using System;</mark> |
|--|
| using System.Drawing; |
| using System.Collections; |
| using System.Windows.Forms; |
| using System.Data; |
| using System.Threading; |
| <pre>using System.Runtime.InteropServices;</pre> |
| using System.Diagnostics; |
| <pre>Image Data_Collector { </pre> |
| <pre>public partial class Data_Collector : System.Windows.Forms.Form {</pre> |
| <pre>public Data_Collector()</pre> |

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

| Data_Collector - Microsoft Visual C# 2010 Express | | and the second second second | | |
|--|---------------------|------------------------------|--------------|--|
| File Edit View Project Build Debug Data | Tools Window Help | | | |
| - M - C 🔊 🖬 🌾 📓 🖓 - C - | 🔍 Connect to Databa | se | | 🝷 🛛 🙆 EPT_570_AP_Data_Collector 💿 🝷 🛃 🎇 🎇 |
| 🏷 Solution Explorer 🛛 🝷 🕂 🗙 | Code Snippets Ma | nager Ctrl+K, Ctrl+B | | Form1.Designer.cs Form1.cs Form1.cs [Design] |
| 7 🛅 🗿 🕢 | Choose Toolbox It | ems | | |
| Image: Solution 'Data_Collector' (1 project) | Extension Manage | | L, | |
| Data_Collector Image: Second | External Tools | | ase) | Platform: Active (x86) |
| References | Settings | • | | Basic Settings |
| တို 📰 active_transfer.cs | Customize | | \checkmark | Expert Settings |
| Argenetics Argenetics Argenetics Argenetics Argenetics Argenetics Argenetics Argenetics Argenetics | Options | | | Reset |
| Form1.Designer.cs | Resources | Define DEBUG consta | | Import and Export Settings |
| | | Define TRACE consta | nt | |
| | Settings | Allow unsafe code | | |
| | Reference Paths | Optimize code | | |
| | Signing | Errors and warnings | | |
| | Security | Warning level: | | 4 |



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

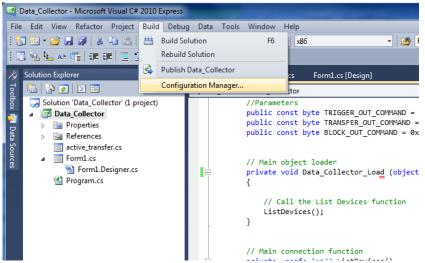
| ▲ Environment General Fonts and Colors Keyboard | Recent files items shown in Window menu 10 items shown in recently used lists |
|--|---|
| Debugging | Visual experience |
| | Automatically adjust visual experience based on client performance |
| | ✓ Enable rich client visual experience |
| | Use 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.

| Environment | Projects location: |
|---|---|
| Projects and Solutions | c:\users\nelsonstrfgr\documents\visual studio 2010\Projects |
| Build and Run | User project templates location: |
| Text Editor | c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ProjectTemp |
| Debugging | User item templates location: |
| Database Tools Tool Tool State | c:\users\nelsonstrfgr\documents\visual studio 2010\Templates\ItemTemplat |
| Text Templating Windows Forms Designer | 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.





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

| onfiguration Manager Active solution configuration: | No. of Concession, Name | Astive colution platforms | ? × | | | |
|--|----------------------------------|---------------------------|-------|--|--|--|
| Release | | Active solution platform: | | | | |
| | | x86 x86 | • | | | |
| Project contexts (check the project | ct configurations to build or de | *P <new></new> | | | | |
| Project | Configuration | <edit></edit> | | | | |
| Data_Collector | Release | ▼ x86 | ▼ | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | Close | | | |

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



| ctive solution configuration: | Active solution platform: | |
|--------------------------------|---|--|
| Release | ▼ x86 | |
| oject contexts (check the proj | ject configurations to build or deploy): | |
| Project | New Solution Platform Build | |
| Data_Collector | Type or select the new platform: Any CPU Any CPU Itanium Itanium < | |

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

| Configuration Manager | | | ? × |
|----------------------------------|-------------------------------------|---------------------------|-------|
| Active solution configuration: | | Active solution platform: | |
| Release | | x64 | • |
| Project contexts (check the proj | ect configurations to build or depl | loy): | |
| Project | Configuration | Platform | Build |
| Data_Collector | Release | ▼ x64 | ▼ |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | Close |
| | | | 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.



| Solution Explorer | → ‡ × | Form1.cs × act | tive_transfer.cs Form1.cs [Design] | |
|--|----------------------------------|--------------------------------------|---|---|
| 🗧 🕒 💽 🗷 | 8 | Applied The Collector | r.Data_Collector | TRIGGER_OUT_COMMAND |
| Solution Data_ Solution Data_ Prop Prop Prog | Build Rebuild | / | <pre>//Parameters ublic 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 send // Call the List Devices function ListDevices();</pre> | |
| 8 7 10 10 10 10 10 10 10 10 10 10 10 10 10 | Cut Paste Remove Rename | Ctrl+X Ctrl+V Del Alt+Enter | <pre>/ Main connection function rivate unsafe Int32 ListDevices() Int32 result; Int32 num_devices; Int32 iCurrentIndex;</pre> | |
| | | | <pre>// Open the DLL result = EPT_AH_Open(null, null, null); if (result != 0) {</pre> | |

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

| | - | |
|--|-----------------|--|
| | | |
| Solution 'Data_Collector' (1 project) | Application | Configuration: Active (Release) Platform: Active (x64) |
| Data_Collector Properties | Build | |
| References active_transfer.cs | Build Events | General |
| ▲ I Form1.cs Form1.Designer.cs | Debug | Conditional compilation symbols: |
| Program.cs | Resources | Define DEBUG constant |
| | Settings | Define TRACE constant Allow unsafe code |
| | Reference Paths | ☑ Optimize code |
| | Signing | Errors and warnings |
| | Security | Warning level: |
| | Publish | |

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.



| | 🗉 • 📂 🖃 🧋 | χI | | | *) |
|--------------|--|----------|---|-----------------------|--|
| | lution Explorer | | ▼ ₽ × | Data_Collector | active_transfer.cs Form1.Designer.c |
| Toolbox | l 🗗 🧟 | | | Applied The Collector | .Data_Collector |
| 2 | Solution 'Data_ Data_Collect Properti | | Build | ⊟using Sys | <pre>tem; +=m.Collections.Generic; .ComponentModel; .Data;</pre> |
| Data Sources | ▶ a Reference 2 active_tr 2 active_tr 2 Form1.cc 2 Form 2 Program | . | Rebuild Publish Add Add Reference Add Service Reference | .e | <pre>1.Drawing; 1.Linq; 1.Text; 1.Windows.Forms; 1.Threading; 1.Runtime.InteropServices;</pre> |
| | | ¥ | Set as StartUp Project Debug Cut | t Ctrl+X | <pre>.Diagnostics; ta_Collector</pre> |
| | | × | Paste Remove | Ctrl+V Del | <pre>partial class Data_Collector : ic Data_Collector()</pre> |
| | | | Rename | | <pre>InitializeComponent();</pre> |
| | | | Properties | Alt+Enter | <pre>for (int i = 0; i < EPTTransm {</pre> |

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();
```

UnoProLogic

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().



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

The ListDevices() function calls the EPT_AH_Open() function to load up the ActiveHost Dll. Next, it calls EPT_AH_QueryDevices() which searches through the registry files to determine the number of EPT devices attached to the PC. Next, EPT_AH_GetDeviceName() is called inside a for loop to return the ASCII name of each device attached to the PC. It will automatically populate the combo box, cmbDevList with all the EPT devices it finds.

```
// List Devices function
private unsafe Int32 ListDevices ()
   {
  Int32 result;
  Int32 num devices;
  Int32 iCurrentIndex;
  // Open the DLL
  result = EPT_AH_Open(null, null, null);
  if (result != 0)
     MessageBox.Show("Could not attach to the ActiveHost library");
      return 0;
      }
  // Query connected devices
  num_devices = EPT_AH_QueryDevices();
  //Prepare the Combo box for population
  iCurrentIndex = cmbDevList.SelectedIndex;
  cmbDevList.Items.Clear();
  // Go through all available devices
  for (device_index = 0; device_index < num_devices; device_index++)</pre>
      {
          String str;
          str = Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetDeviceName(device_index));
          cmbDevList.Items.Add(str);
      }
  return 0;
}
```



The user will select the device from the drop down combo box. This value can be sent to the OpenDevice() function using the button Click of the Open button.

| <pre>exit(0); }</pre> | device_index)); |
|---|-----------------|
| USB High Speed Transfer B Open Close | |

The device_index variable is used to store the index of the device selected from the combo box. This variable is passed into the EPT_AH_OpenDeviceByIndex(). This process is started by the user clicking on the "Open" button. If the function is successful, the device name is displayed in the label, labelDeviceCnt. Next, the device is made the active device and the call back function is registered using the RegisterCallBack() function. Finally, the Open button is grayed out and the Close button is made active.



```
// Open the device
public unsafe Int32 OpenDevice()
{
    device_index = (int)cmbDevList.SelectedIndex;
    if (EPT_AH_OpenDeviceByIndex(device_index) == 0)
    {
        String message = "Could not open device " +
           Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetDeviceName(device_index)) + ", " +
           Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetDeviceSerial(device_index));
        MessageBox.Show(message);
        return 0;
    }
    else
    {
        labelDeviceCnt.Text = "Connected to device " +
            Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetDeviceName(device_index)) + ", " +
            Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetDeviceSerial(device_index));
    }
    // Make the opened device the active device
    if (EPT_AH_SelectActiveDeviceByIndex(device_index) == 0)
    {
        String message = "Error selecting device: %s " +
           Marshal.PtrToStringAnsi((IntPtr)EPT_AH_GetLastError());
        MessageBox.Show(message);
        return 0;
    }
    // Register the read callback function
    RegisterCallBack();
    btnOpenDevice.Enabled = false;
    btnCloseDevice.Enabled = true;
    return 0;
}
```

Next, the callback function is populated. This function will be called from the Active Host dll. When the EPT Device has transferred data to the PC, the callback function will do something with the data and command.



```
// Actual callback function which will read messages coming from the EPT device
unsafe void EPTReadFunction(Int32 device_id, Int32 device_channel, byte command, byte payload,
ł
   byte* message = data;
   // Select current device
   EPT_AH_SelectActiveDeviceByIndex(device_id);
   //Add command and device_channel to the receive object
   EPTReceiveData.Command = ((command & COMMAND_DECODE) >> 3);
   EPTReceiveData.Address = device_channel;
   //Check if the command is Block Receive. If so,
    //use Marshalling to copy the buffer into the receive
   //object
   if (EPTReceiveData.Command == BLOCK OUT COMMAND)
   {
        EPTReceiveData.Length = data_size;
       EPTReceiveData.cBlockBuf = new Byte[data_size];
       Marshal.Copy(new IntPtr(message), EPTReceiveData.cBlockBuf, 0, data size);
   }
   else
   {
       EPTReceiveData.Payload = payload;
    3
   EPTParseReceive();
}
```

Because the callback function communicates directly with the dll and must pass pointers from the dll to the C#, marshaling must be used. Marshaling is an advanced topic and will not be covered in this manual.

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 1.6.4 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 btn0k_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)
ł
    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().

9.1.13 PC: Compiling the Active Host Application

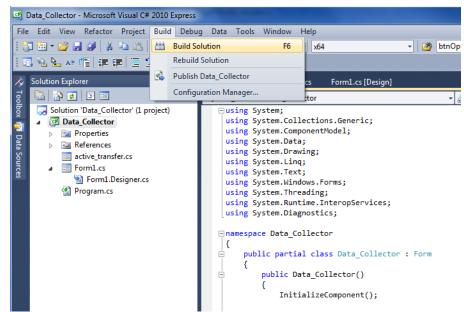
Building the Data_Collector project will compile the code in the project and produce an executable file. It will link all of the functions declared in the opening of the Data_Collector Class with the Active Host dll. The project will also automatically link the FTD2XX.dll to the object code. Follow section: Assembling, Building, and Executing a .NET Project on the PC. Browse to the

\Projects_ActiveHost_xxBit\EPT_Data_Collector \Data_Collector\ folder of the UnoProLogic 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.





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".



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.

9.1.14 Adding the DLL's to the Project

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

- ActiveHostXX.dll
- ftd2xxXX.dll

and install them in the binx64 folder of your EPT_Data_Collector project.



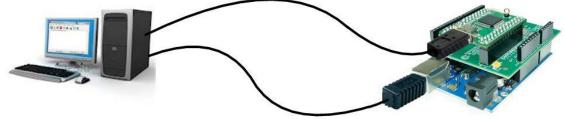
| Irganize 👻 🔟 Open with Share with 👻 Bur | n Ne | ew folder | | | |
|---|------|---------------------------------------|--------------------|--------------------|--------|
| 🌗 Arduino_IDE | * | Name | Date modified | Туре | Size |
| Documentation | | ActiveHost64.dll | 3/2/2013 9:44 PM | Application extens | 27 KE |
| Drivers | | EPT_Transfer_Test.exe | 3/2/2013 11:28 PM | Application | 28 KE |
| Projects_ActiveHost_32Bit | | EPT_Transfer_Test.pdb | 3/2/2013 11:28 PM | Program Debug D | 56 KE |
| Projects_ActiveHost_64Bit | | EPT_Transfer_Test.vshost.exe | 3/2/2013 11:21 PM | Application | 12 KE |
| ActiveHost_1.0.0.8 | | EPT_Transfer_Test.vshost.exe.manifest | 8/31/2009 12:40 AM | MANIFEST File | 1 KE |
| EPT_Data_Collector | | S ftd2xx64.dll | 1/18/2013 3:54 PM | Application extens | 252 KE |
| EPT_Transfer_Test | | | | | |
| EPT_Transfer_Test | | | | | |
| Debug | | | | | |
| i Felose i Aclese i x64 i Debug i Relesse i Relesse i obj | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Save the project.

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.

9.1.15 Connecting the Project Together

Now we will connect the Arduino, UnoProLogic, 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 UnoProLogic.



Next, open the Arduino IDE and select File->Open and select your sketch created earlier, Arduino_Data_Collector_Code_U2.ino.



| Look in: | Arduino Dat | ta_Collector_Code | - 0 🕽 🖻 🛄 - | - | |
|----------------------------------|---|--------------------------|--------------------|---------------------|--------|
| 0 | Name | | Date modified | Туре | |
| Recent Places | Arduino_Data_Collector_Code.ino | | 10/23/2012 8:31 AM | | ۹ ۱ |
| Libraries Computer Network | ✓ File name: Files of type: | " I Al Files (*.*) | • |) Open Cancel | |
| | | | | 112 | |
| | | | | | |
| | X | | | | , |
| | z | | | | ţ |

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.





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.

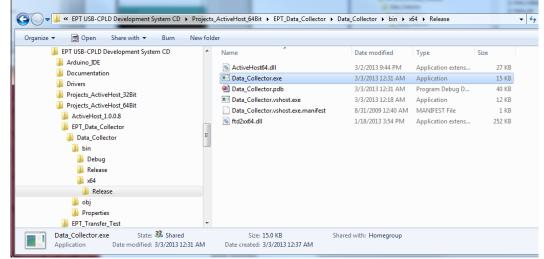


If this LED is blinking at the rate of once per second, the Arduino and the Data Collector project are ready for the EPT 570-AP-U2 code.



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 $\frac{1}{2}$



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-U2 device. Next, the combo box at the top will be populated with the name of the device.

| 🖳 Da | ta Collector | | _ | |
|------|-----------------|--------------|---------|---|
| | | | | |
| | EPT USB <-> JTA | G&Serial Cab | le B 🔻 | |
| | Open | Close | | |
| | Receive Byte | | Address | 2 |

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".



| 🖳 Active Host | |
|-----------------------------|------------------|
| EPT USB <-> JTAG&Serial Cab | Device Connected |
| Open Close | Address 2 |
| Received Data | Start Stop |
| | |
| | |
| | |
| ОК | Close |

9.1.16 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.

| 🖳 Active Host | |
|----------------------------------|------------------|
| EPT USB <-> JTAG&Serial Cab | Device Connected |
| Open Close | Address 2 |
| Received Data | Start Stop |
| f0 36 7a ce 19 78 | |
| ОК | llose |



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.



10 APPENDIX I

EPT

FIFO

FTDI

| | List of Abbreviations and Acronyms |
|------------|------------------------------------|
| Earth Pe | eople Technology |
| First In - | - First Out |
| Future T | echnology 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 Intel 5M570 CPLD