Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 001-70486

Spec Title: EZ-USB(R) FX2LP(TM) HOST APPLICATION IN VC++ 2008 USING SUITE USB LIBRARY (CYUSB.DLL) - AN70486

Sunset Owner: Gayathri Vasudevan (gaya)

Replaced by: 001-70983

AN70486

EZ-USB® FX2LP™ Host Application in VC++ 2008 Using Suite USB Library (CyUSB.dll)

Author: Gayathri Vasudevan
Associated Project: Yes
Associated Part Family: CY7C68013A
Software Version: Microsoft Visual C# 2008 Express Edition
Related Application Notes: None

Abstract

Cypress's FX2LP is one of the most popular programmable high-speed USB controllers in the industry. Cypress provides SuiteUSB for creating windows applications using Microsoft Visual Studio. This application note demonstrates how to build a host application on Microsoft Visual C++ platform, using the Cypress SuiteUSB C# library, *CyUSB.dll*, to perform USB BULK IN and OUT transfers with FX2LP and the associated project is tested with FX2LP Development kit.

Contents

Overview

AN70486 describes a host application built on the Microsoft Visual C++ 2008 platform that uses *CyUSB.dll* to communicate with Cypress USB driver, *CyUSB.sys*. The host application communicates with the BULK IN and BULK OUT endpoints of FX2LP, using the interfaces provided by the APIs of *CyUSB.dll*. The *CyUSB.dll* in-turn communicates internally with Cypress USB driver (*CyUSB.sys*), for talking to these endpoints. This host application implements the transfer only with devices that pass the particular VID/PID identification. The example device used in this application note is the Bulkloop device. The firmware that is attached along with this application note causes a loop back of data inside the device. Thus this host application, with the attached Bulkloop device, demonstrates the loopback of data.

System Requirements

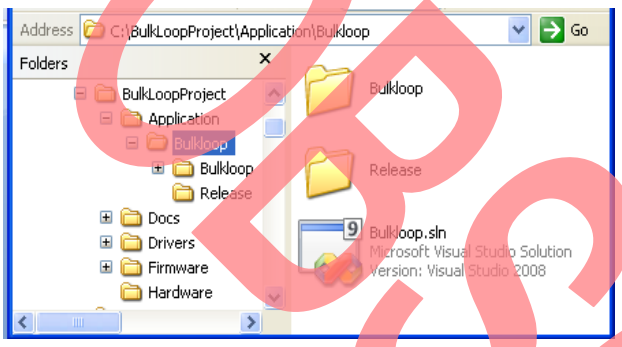
Hardware

A FX2LP DVK (CY3684) board is used as the development and testing platforms for this example. A detailed schematic of the DVK is provided in the 'hardware folder' located in the attachment. More information about the board is available in the 'EZ-USB Advanced Development Board' section of the 'EZ-USB_GettingStarted' document, available at C:\Cypress\USB\doc\General (after DVK install).

Software

1. Control Centre – Available through Suite USB 3.4
2. Keil uVision 2 – The 4 K bytes evaluation version is available with the CY3684 DVK. For the full version, contact Keil.
3. Microsoft Visual C# 2008.

Project Directory Structure



Bulkloop Project

This is the project folder attached along with this application note. This contains the following folders and files.

- Application
- Docs
- Drivers
- Firmware
- Hardware

Application

- **Address:** BulkLoopProject\Application

  The location contains the main project folder (Bulkloop), which consists of the following folders and files.

  (BulkloopProject\Application\Bulkloop).

- Bulkloop
- Release
- Bulkloop.sln

Release

- **Address:** BulkLoopProject\Application\Bulkloop\Release

The location contains the *Bulkloop.exe,* GUI for this host application.

Bulkloop.sln

- **Address:** BulkLoopProject\Application\Bulkloop\Bulkloop.sln

Open the location to view the source code for the host application.

Docs

- **Address:** BulkLoopProject\Docs

The location contains the related datasheets and Technical Reference Manual for EZ-USB FX2LP.

Drivers

- **Address:** BulkLoopProject\Drivers

The previous location contains the *CyUSB.inf* and *CyUSB.sys* file for FX2LP. Use *CyUSB.inf* in the path shown in the following table for the respective OS.

Operating System	Folder Path
Windows XP 32bit	wxp\x86
Windows XP 64bit	wxp\x64
Windows7 32bit	wlh\x86
Windows7 64bit	wlh\x64
Windows Vista 32bit	wlh\x86
Windows Vista 64bit	wlh\x64

Firmware

- **Address:** BulkLoopProject\Firmware

The address contains the Bulkloop firmware that is used in this application note that causes the loopback of data inside the FX2LP device.

Hardware

- **Address:** BulkLoopProject\Hardware

The hardware address contains the schematic of CY3684 development board.

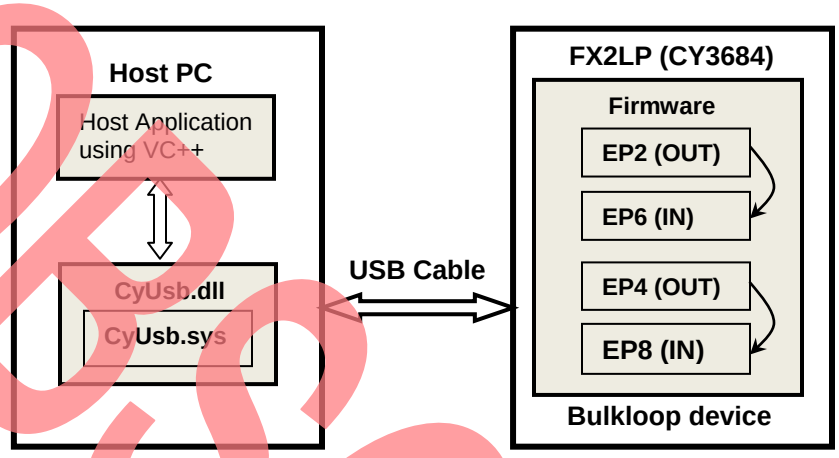
Prerequisites

The prerequisites for testing this application are,

1. *CY3684 EZ-USB_GettingStarted.pdf* and *DVK Users Guide.pdf* available at *C:\Cypress\USB\doc\General* (after DVK install) – contains information on how to use the CY3684 kit for the first time
2. Introduction to CyUSB.dll Based Application Development Using C# – Helps in getting started with developing host applications using VC#.

3. Getting Started with FX2LP™ – This document serves as an introduction for new users to get familiar with FX2LP.

Block Diagram



The 'Host' (Host PC) and the 'target' (FX2LP based CY3684 board) are connected through USB cable. The host application sends data to BULK OUT (EP2 or EP4) endpoints of FX2LP™ through *CyUSB.dll*, which in-turn communicates internally with Cypress USB driver, *CyUSB.sys*. When the bulk data reaches the endpoint EP2 (or EP4), the bulkloop firmware running on Fx2LP, takes this data and copies it back to EP6 IN (or EP8 IN) endpoint. The bulk data travels back in reverse through *CyUSB.sys* to *CyUSB.dll,* and finally reaches the host application. The host application displays the bulk data transferred.

For familiarizing with different components of block diagram, see the following documentation after installing Suite USB.

***CyUSB.dll*:** *Cypress Suite USB 3.4.x\CyUSB.NET\ CyUSB.NET.pdf*

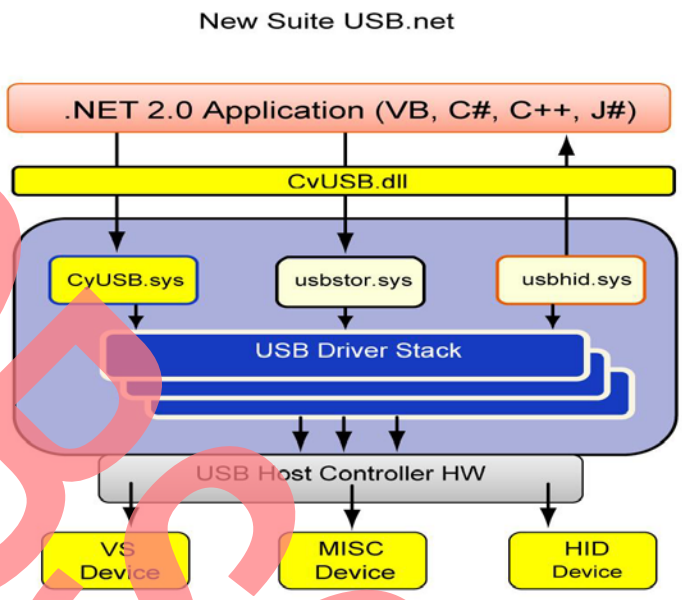
***CyUSB.sys*:** *Cypress Suite USB 3.4.x\Driver\CyUSB.pdf,* after installing the DVK from FX2LP Development kit.

CY3684: *EZ-USB_GettingStarted.pdf* and *DVK Users Guide.pdf* available at *C:\Cypress\USB\doc\General*.

FX2LP: *EZ-USB_TRM.pdf* and *fx2lp_datasheet.pdf* available at *C:\Cypress\USB\doc.*

Introduction to *CyUSB.dll*

Earlier, the application writing process involved making direct calls to drivers. The process of writing an application was cumbersome; the application had to first get a device handle and then call device I/O controls or read/write files. This made access, especially to mass storage class devices, very difficult. The new generation of application development tools from Cypress has a simple, more powerful API. SuiteUSB 3.4 supports the *CyUSB.sys*, *usbstor.sys*, and *usbhid.sys* device drivers increasing the spectrum of devices that can be accessed with this tool. The following figure represents the development of application with the new *SuiteUSB.net*.

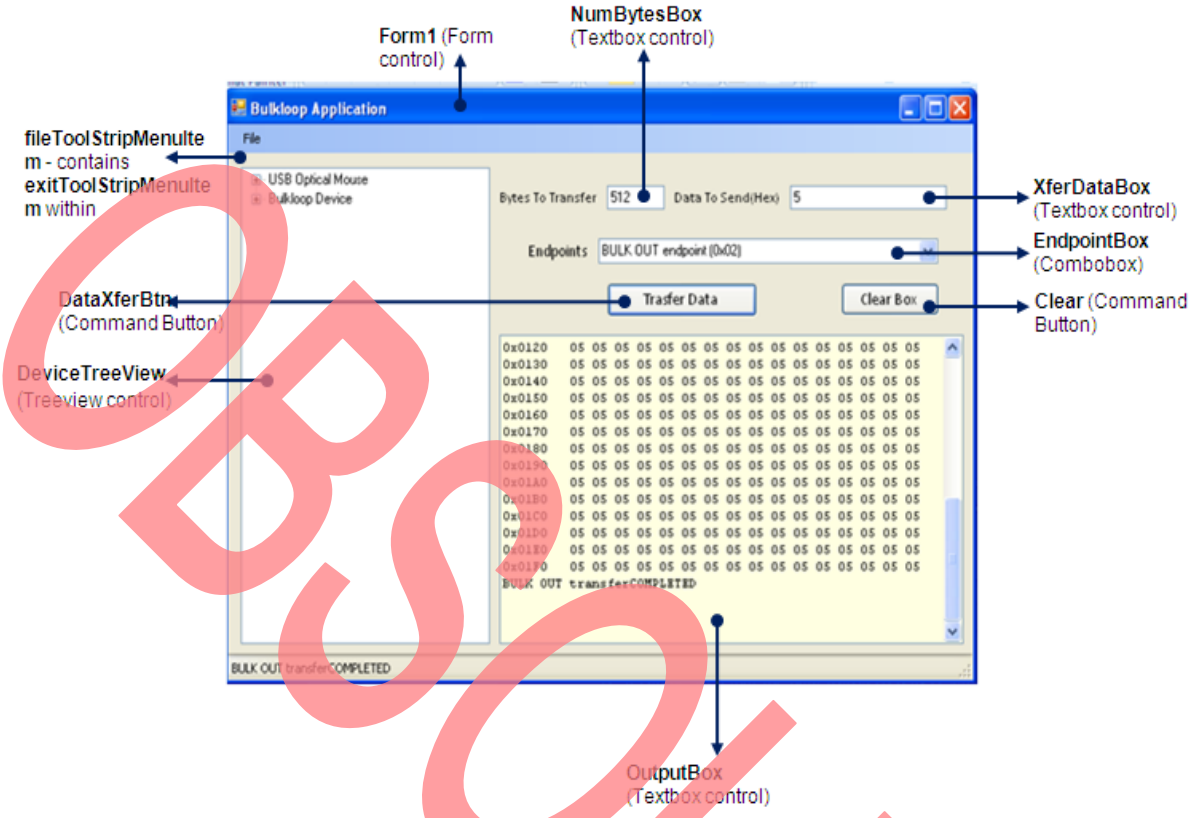


CyUSB.dll is a managed Microsoft *.NET* class library, and *CyUSB.dll* classes and methods can be accessed from any of the Microsoft Visual Studio .NET managed languages, such as Visual Basic.NET, C#, Visual J#, and managed C++. *CyUSB.dll* provides a high-level, powerful programming interface to USB devices. To use the library, you need to add a reference to *CyUSB.dll*. Then, any source file that accesses the *CyUSB* namespace will need to include a line with the namespace in the appropriate syntax. For more information on *CyUSB.dll* see the Cypress CyUSB .NET DLL Programmer's Reference.

Operation

Host Application

The user interface of the host application (*Bulkloop.exe*) is available in the attached source directory *\BulkLoopProject\Application\Bulkloop\Release* folder.



This application user interface is designed to show:

All the USB Devices Connected: Treeview control is used only to display all the connected USB devices. This is used to demonstrate that all the devices (supported by *CyUSB.sys*, *usbstor.sys*, *usbhid.sys*) are supported by *CyUSB.dll*.

Bytes to Transfer: This is user selectable byte that represents the number of bytes to be transferred during the BULK transfer.

Data to Send (Hex): This is user selectable byte, which is transferred in loopback fashion, during bulk-loop operation.

Endpoints: Endpoints available in Bulkloop device (device with the VID/PID = 0x04B4/ 0x1004) are listed under the combobox.

Bytes Transferred: A textbox (OutputBox) is used to display the actual data bytes transferred and the result of the transfer.

Transfer Data: This command button (DataXferBtn) is used for initiating the bulk transfers.

Clear: This command button is used for clearing the OutputBox that displays the actual data transferred.

Exit: MenustripItem (Exit) for exiting the application.

Operation – Broad View

The firmware, which is provided in the attachment along with this application note, when downloaded into the FX2LP device causes it to enumerate with VID/PID of 0x04B4/0x1004 as a bulkloop device. This particular device (let us call it the Bulkloop device) has four Endpoints, configured as follows:

- EP2 – BULK OUT, double buffered, size 512
- EP6 – BULK IN, double buffered, size 512
- EP4 – BULK OUT, double buffered, size 512
- EP8 – BULK IN, double buffered, size 512

The firmware basically performs a loop back of data. Whenever there is any data in the EP2 OUT, it is immediately transferred to EP6 IN; and whenever there is any data in EP4 OUT, it is transferred to EP8 IN.

The host application that is explained in this application note is used to send the data needed for the loopback to the OUT endpoint of the device, and retrieve the data from the IN endpoint of the Bulkloop device. This host application can be used to send data to OUT endpoint, and receive data from the IN endpoint of any device having the particular VID/PID. But in this application note, this is demonstrated using bulkloop firmware. Since the

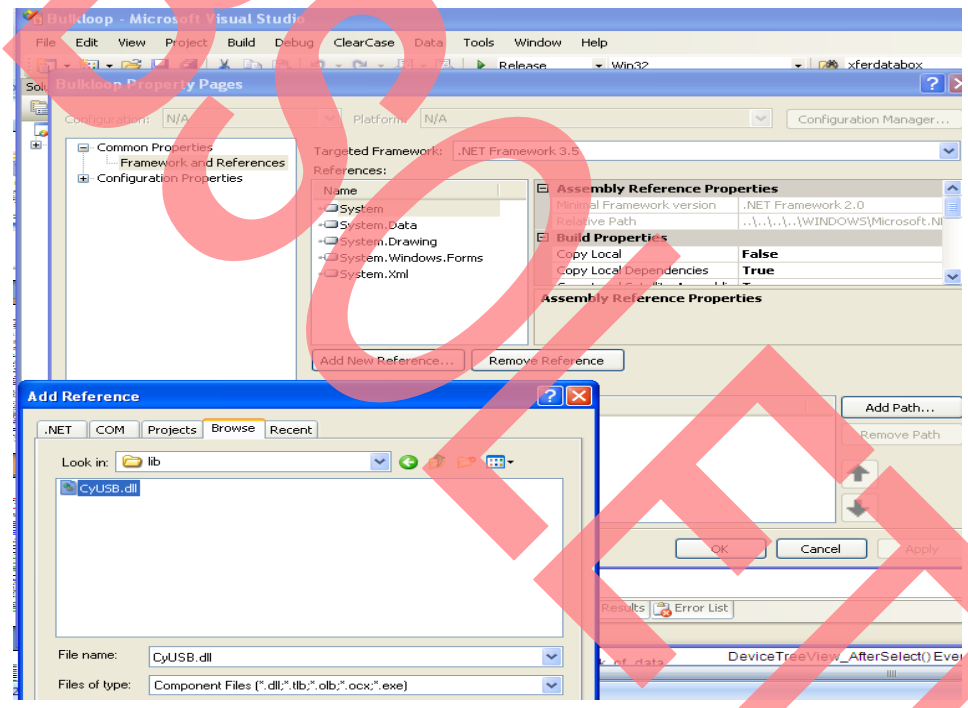
Bulkloop firmware immediately transfers the data in the EP2 OUT (EP4 OUT) to EP6 IN (EP8 IN), the data received from the IN endpoint is the same as that was sent to the corresponding OUT endpoint.

In the GUI developed in this host application, a Treeview control is used to display all the USB devices attached. Only if the Bulkloop device is attached, and is selected from the Treeview, the remaining controls in the form are enabled. In this example, Bulkloop device is taken to have the VID/PID values of 0x04B4/0x1004. If the Bulkloop device is selected from the Treeview, then the Endpoints available are listed in the combobox. Select one endpoint from the list and input the number of bytes to be transferred and the actual data byte (hex) into the corresponding textbox. Then click on the DataXferBtn to initiate the transfer. The actual bytes transferred and the result of the transfer is displayed in the textbox (OutputBox).

Adding Reference to *CyUSB.dll* in a VC++ project

If a new VC++ project is started, to use *CyUSB.dll*, we have to add reference to it as follows.

Go to **Project > Properties**. In the dialog box, select **Common Properties > Add New Reference**. In the window that opens select **Browse** tab and point to the ***CyUSB.dll*** (*C:\Cypress\Cypress Suite USB 3.4.4\CyUSB.NET\lib\CyUSB.dll*).



Source Code Outlook

For viewing the source, open the '*BulkLoop.sln*' file, placed under 'Application\Bulkloop' folder of the attachment. When the bulkloop projects opens in Microsoft visual studio, open the solution explorer using View → solution explorer. In solution explorer window, under BulkLoop project, Form1.h is placed. Select Form1.h and view its sources using View → Code.

When the host application is running, the execution sequence is as follows:

- The global declarations part and the form constructor get executed first, as soon as the GUI is opened.
- When you connect the USB device, the connection between the host and the device gets established, and the event_handler function usbDevices_DeviceAttached() gets called. This function in turn calls a method for updating the Treeview control with the current list of USB devices attached.
- When any device is selected from the Treeview, DeviceTreeView_AfterSelect()EventHandler executes and from that, only if bulkloop device is selected, the remaining controls on the form are enabled, else they are disabled.

- When the target device is disconnected, the event_handler usbDevices_DeviceRemoved() gets called. This function also calls a method for updating the Treeview control with the current list of USB devices attached. If the removed device is bulkloop device, then the remaining controls on the form are disabled.
- After bulkloop device is connected and selected from the Treeview control, when you press the 'Transfer Data' button, the DataXferBtn_Click() function gets called. This function takes the user written data byte value from the 'Data to Send (Hex)' field and the 'Bytes to Transfer' field on user interface and then depending on the endpoint selected from the combobox, it calls the Xferdata method.
- This XferData() function is core to the USB communication and it is used for doing synchronous data communication on endpoints. For more information on these functions, check for comments in the code section that explains the functionality. Following code snippets explains essential part of the code.

Code Snippets

The following code snippets are directly taken from the form design sources.

Pre-compiler Directives: You can see the line #pragma once on top. You have to include the following path below that line:

```
#using "C:\Cypress\Cypress Suite USB 3.4.4\CyUSB.NET\lib\CyUSB.dll"
```

The compiler searches for references along the path specified in the #using statement. Also you can see the directives starting with word "using" inside the namespace. All the directives except the last one are automatically added when the application is created.

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace CyUSB;
```

- `using` CyUSB: This is to inform the application that *CyUSB.dll*, which is added as the reference library to this project, is being used. The CyUSB namespace provides classes and interfaces for cypress USB communication library (*CyUSB.dll*).

Object Initialization: Inside the Form1 class (`public ref class Form1 : public System::Windows::Forms::Form`) instances of the various classes required are created.

```
USBDeviceList^ usbDevices;
// create dynamic list of USB devices to
represent all USB devices.
CyFX2Device^ dev;
// Create USB(Cypress FX2) device for our
application called dev
CyFX2Device^ bulkloopdev;
// Create USB(Cypress FX2) device for
application called bulkloopdev
CyFX2Device^ fx2_selected;
// Create USB(Cypress FX2) device for
application called fx2_selected
CyUSBEndPoint^ ept;
// Create object of the abstract class,
CyUSBEndPoint
int nodecount;
System::String^ datastr;
```

In the previous snippet, the USBDeviceList, CyFX2Device, CyUSBDevice, CyUSBEndPoint classes are from CyUSB namespace. For details of these classes, see *CyUSB.NET* documentation inside *Cypress Suite USB 3.4.x\CyUSB.NET\CyUSB.NET.pdf* of cypress Suite-USB installation, on your PC.

The first initialization is for dynamic list of USB devices called usbDevices. This list is further used in selecting and instantiating 'dev', 'bulkloopdev', and 'fx2_selected'. It is also used in adding the device attachment and detachment event handlers. (The next three initializations creates USB devices for the application. 'bulkloopdev' represents the Bulkloop device attached. In this application note, the device with VID = 0x04B4, PID= 0x1004). 'fx2_selected' represents the device selected by you from the Treeview. 'dev' is used for populating the Treeview control with all the USB devices attached. The next initialization is for the endpoint, ept, that is used in the project.

Form Constructor

```
//Create a list of CYUSB devices
usbDevices = gcnew
USBDeviceList(CyConst::DEVICES_CYUSB |
CyConst::DEVICES_MSC |
CyConst::DEVICES_HID);
// All devices served by
CyUSB.sys,usbstor.sys,usbhid.sys is added
to
// the list usbDevices

//Adding event handlers for device
attachment and device removal
usbDevices->DeviceAttached += gcnew
System::EventHandler(this,&Form1::usbDevice
s_DeviceAttached);
// Eventhandler assigned to DeviceAttached
to handle the event when a
// device is attached
usbDevices->DeviceRemoved += gcnew
System::EventHandler(this,&Form1::usbDevice
```

```
s_DeviceRemoved);
// Eventhandler assigned to DeviceRemoved
to handle the event when a
// device is removed
```

- `usbDevices = gcnew USBDeviceList(CyConst::DEVICES_CYUSB | CyConst::DEVICES_MSC | CyConst::DEVICES_HID);` Creates an instance of USBDeviceList and populates itself with USBDevice objects representing all the USB devices that are served by the device selector. The device selector is indicated by passing the appropriate parameter to the class method. Device selector passed here is ORed combination of DEVICES_CYUSB, DEVICES_HID, DEVICES_MSC. CyConst is a static class that contains these constants and several other constants. See *CyUSB.net* documentation for more details.

- `usbDevices->DeviceAttached,usbDevices->DeviceRemoved;` When a USB device is connected or disconnected from the bus, the connection or the removal event can be detected and some action can be taken. Detection of the event is automatically setup by the USBDeviceList object. Handling of the event requires that an EventHandler object be assigned to the DeviceAttached / DeviceRemoved event handler. Thus 'usbDevices_DeviceAttached' EventHandler is assigned to the DeviceAttached and 'usbDevices_DeviceRemoved' EventHandler is assigned to the DeviceRemoved. These EventHandlers are explained later.

```
for(int i=0 ; i < usbDevices->Count ;
i++)
{
dev = (static_cast<CyFX2Device^>(usbDevi
ces[i]));
//Assign all devices in
usbDevices(dynamic USBDeviceList)one
by// one to dev
DeviceTreeView->Nodes->Add(dev->Tree);
//adding the devices in the usbDevices
to the treeview control
}
bulkloopdev = (static_cast<CyFX2Device^>
(usbDevices[0x04B4,0x1004]));
```

- `usbDevices->Count`: The Count property reflects the number of USBDevice objects in the USBDeviceList. Thus inside the 'for' loop, all devices in 'usbDevices'(dynamic USBDeviceList) are assigned one by one to 'dev', and every device is then added to the Treeview control using the Tree property of the CyUSBDevice class.

- `bulkloopdev = static_cast<CyFX2Device^>(usbDevices[0x04B4,0x1004]));` This checks the USBDeviceList for a device with the given VID/PID (here, 0x04B4/0x1004) and assigns that to 'bulkloopdev' of the type CyFX2Device. If bulkloopdev returns true, then Bulkloop device is attached, else it is not attached. Thus the 'ToolStripStatusLabel' is updated accordingly.

Device Attach and Removal Action

- When a device is attached, the 'usbDevices_DeviceAttached' Eventhandler executes and calls the RefreshDeviceTree() method.

- Updates the Treeview control with the current list of USB devices attached. The code is similar to the one in form constructor for populating the Treeview control.

- When a device is removed, the usbDevices_DeviceRemoved EventHandler executes and calls RefreshDeviceTree method. It also checks whether the removed device is bulkloop device. If so, the remaining controls on the form are disabled, and the combobox cleared.

DeviceTreeView_AfterSelect() EventHandler: This is the EventHandler that executes as soon as the Treeview control is selected.

```
TreeNode^ selnode =
DeviceTreeView->SelectedNode;
while(selnode->Parent)
selnode = selnode->Parent; //finds the
selected parent node
fx2_selected =
static_cast<CyFX2Device^>(selnode->Tag);
//finds the seleced device.
```

- This code snippet inside the EventHandler finds the device which is selected from the Treeview control, and assigns that to 'fx2_selected'.

- Following that, if the device selected is same as the bulkloop device, then the remaining controls in the form are enabled and calls the Setup_EndPointBox() method, else the remaining controls are disabled and EndPointBox (combobox displaying the available endpoints) is cleared.

Setup_EndPointBox() method

- This function sets up the combobox (EndPointBox) with the endpoints available in the bulkloop device.

- For more information on EndPointCount, EndPoints, bIn, Address properties, see *CyUSB.NET* documentation.

xferDataBox_KeyUp() method

- This function sets up the NumBytesBox(Bytes To transfer field) with number of bytes to be transferred. If

the length of data bytes entered by you in the XferDataBox(Data to send(hex)) field is greater than 1 Byte, then this function computes the number of bytes entered by you, and NumBytesBox is updated accordingly and disabled. The number of bytes to be transferred (Bytes To Transfer field) can be updated by you only if you enter a data length of 1 Byte. (Data to Send field). If you enter a data length greater than 1 Byte, then the xferDataBox_KeyUp() function updates the 'Bytes To Transfer' field and disables.

EndpointBox_SelectedIndexChanged() method

- This function enables the NumBytesBox, whenever an endpoint is selected from the combobox, and restores the default values of 512 into 'Byte to Transfer' field, and 5 to 'Data to Send(hex)' field.

xferDataBox_Leave() method

- This function ensures that the XferDataBox(Data to send(Hex)) field is never empty. Accidently if you leave the field empty, the function assigns it the default value of 5, and number of bytes field is assigned the default value of 512.

DataXferBtn_Click() method

Event Handler for the Click event of the DataXferBtn

- From the Endpoint, which is selected from the 'EndPointBox' (combobox), the address of the selected endpoint is found and stored in 'addr'.
- In the 'if-else' ladder, the data transfer is done according to the endpoint selected.
- `if(addr == "0x02")`: If the selected endpoint is EP2 OUT, then first the buffer, buf, is filled by the data to be transferred to EP2 OUT (from the user entered XferDataBox). For more information on EndPointOf(addr), BulkOutEndPt, XferData see *CyUSB.net* documentation.

```
bulkloopdev->BulkOutEndPt =
safe_cast<CyUSB::CyBulkEndPoint^> (
bulkloopdev->EndPointOf(0x02) );
//Assign EP2 as BulkOutEndPt
BulkResult =
bulkloopdev→BulkOutEndPt→XferData(buf,
len);
//calls the xferData function for bulk
transfer(OUT) in the
//cyusb.dll
ept =
static_cast<CyUSBEndPoint^>(bulkloopdev
→BulkOutEndPt);
//Assign currently selected endPoint
(here EP2) to 'ept'
```

- Here BulkOutEndPt is assigned as EP2, which is returned by `bulkloopdev→EndPointOf(0x02)`.
- `BulkResult = bulkloopdev→BulkOutEndPt→XferData(buf, len)`: The XferData method sends or receives len bytes of data from / into buf. This is the primary I/O method of the library for transferring data. It performs synchronous (i.e., blocking) I/O operations and does not return until transaction completes or the endpoint's TimeOut has elapsed. For all non-control endpoints, the direction of the transfer is implied by the endpoint itself. Returns true if the transaction successfully completes before TimeOut has elapsed. The number of bytes actually transferred is passed back in len. So here, for EP2, this method sends len bytes of data from the buf, and the result of transfer is stored in BulkResult.
- `ept = static_cast<CyUSBEndPoint^>(bulkloopdev →BulkOutEndPt)`: Assigns currently selected endPoint (here EP2) to 'ept'.
- The code is similar for the following endpoints selected: EP4 and EP2.

```
else if(addr == "0x86")
//IF EP6 selected
{
bulkloopdev→BulkInEndPt =
safe_cast<CyUSB::CyBulkEndPoint^>  (
bulkloopdev→EndPointOf(0x86) );
//Assign EP6 as BulkInEndPt
BulkResult = bulkloopdev→BulkInEndPt-
>XferData(buf,len);
//calls the XferData function for bulk
transfer(IN) in the
//cyusb.dll
ept =
static_cast<CyUSBEndPoint^>(bulkloopdev
→BulkInEndPt);
//Assign currently selected endPoint
(here EP6) to 'ept'
}
```

- If the selected endpoint is EP6 IN,
  - Here BulkInEndPt is assigned as EP6, which is returned by `bulkloopdev->EndPointOf(0x86)`
  - `BulkResult = bulkloopdev→BulkInEndPt→XferData(buf,len)`: Calls the XferData function for bulk transfer(IN) from EP6 IN of the device. Since the endpoint here is 'BulkInEndPt' it receives len bytes of data into buf.
  - `ept = static_cast<CyUSBEndPoint^>(bulkloop`

`dev→BulkInEndPt):` Assigns currently selected endPoint (here EP6) to 'ept'

- The code is similar for the following endpoints selected: EP6 and EP8.
- The rest of the section in the EventHandler deals with forming the strings: resultstr (to display the Result of the transfer) and datastr (to display the bytes transferred)

Clear_Click() Method: This is the EventHandler for the Clear command button for the Click event. This clears the OutBox (textbox control)

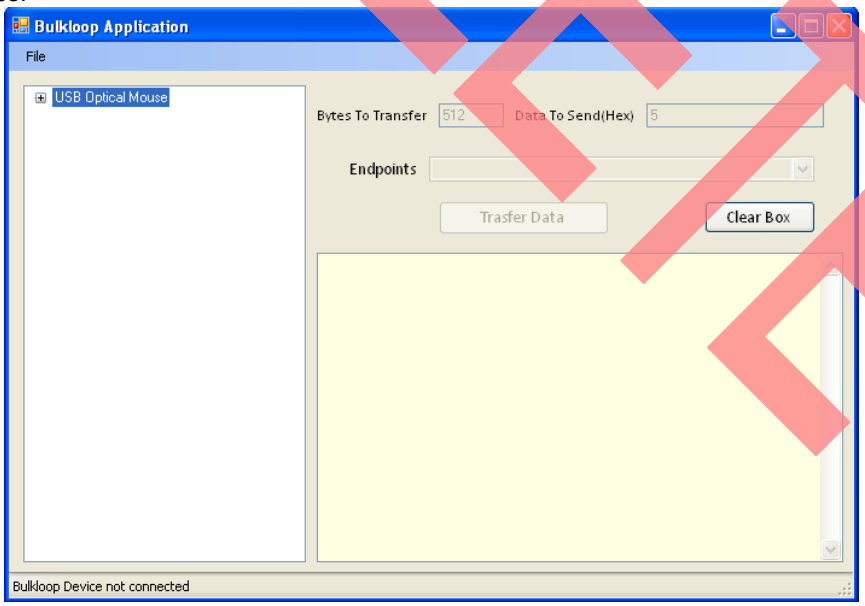
exitToolStripMenuItem_Click() Method: This is the EventHandler for the exitToolStripMenuItem for the Cilck event. This exits from the application.

Hardware Connections

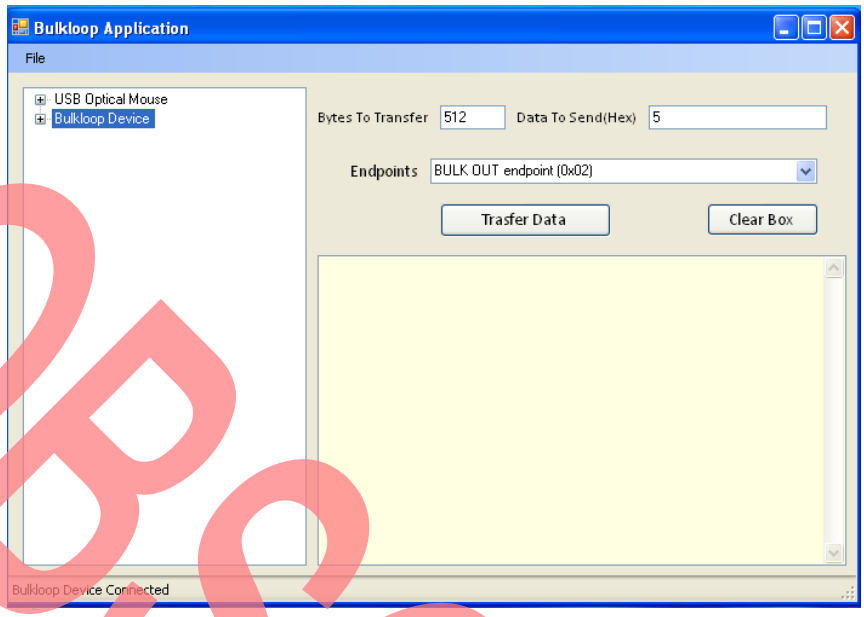
For this example, all the jumpers on CY3684 should be left to their default states. You can refer to the default jumper settings of the DVK in the Table 1 of EZ-USB_GettingStarted.pdf. This document is available in 'USB \doc\General' folder of the DVK install. The Schematic of CY3684 development board can be found inside the folder Hardware in the associated project.

How to Test the Application

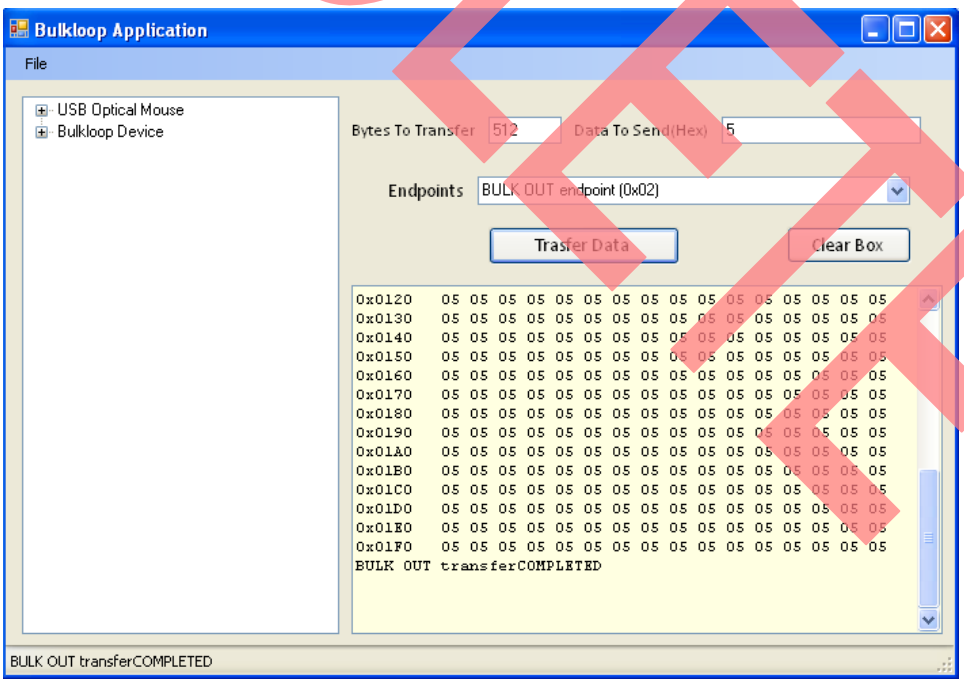
1. Install SuiteUSB 3.4. This installs **Control Centre** utility.
2. In CY3684 FX2LP DVK, select "EEPROM ENABLE" switch to 'No EEPROM' position. Connect the (FX2LP DVK) board to the PC. It enumerates with the default internal descriptor. Use the *CyUSB.inf* file in the driver's folder to bind with the device. For more information on binding the driver, see MatchingDriverToUSBDevice section in *CyUSB.chm* inside folder Drivers or the link Drivers for FX1/FX2LP to bind with the device.
3. Change the 'EEPROM ENABLE' switch position to "EEPROM" position, and the EEPROM SELECT switch to the LARGE EEPROM position on the device.
4. Open the Control Center application present at 'Start → programs → Cypress → Cypress Suite USB 3.4.4 → Control Center'. Download the *Bulkloop.iic* from *Firmware\Bulkloop* folder to the large EEPROM present on the FX2LP DVK using the Control Centre utility. Reset the FX2LP, and the device then enumerates running the bulkloop firmware.
5. If Windows pops up asking you to bind the driver, repeat the steps explained in Step 2.
6. Now open the Host Application GUI by opening *Bulkloop.exe* file from *Application\Bulkloop\Release folder*. Bulkloop device can be plugged in even after opening the GUI (follow the previous 5 steps for that as well). If the bulkloop device is not connected, GUI appears as shown in the following figure.



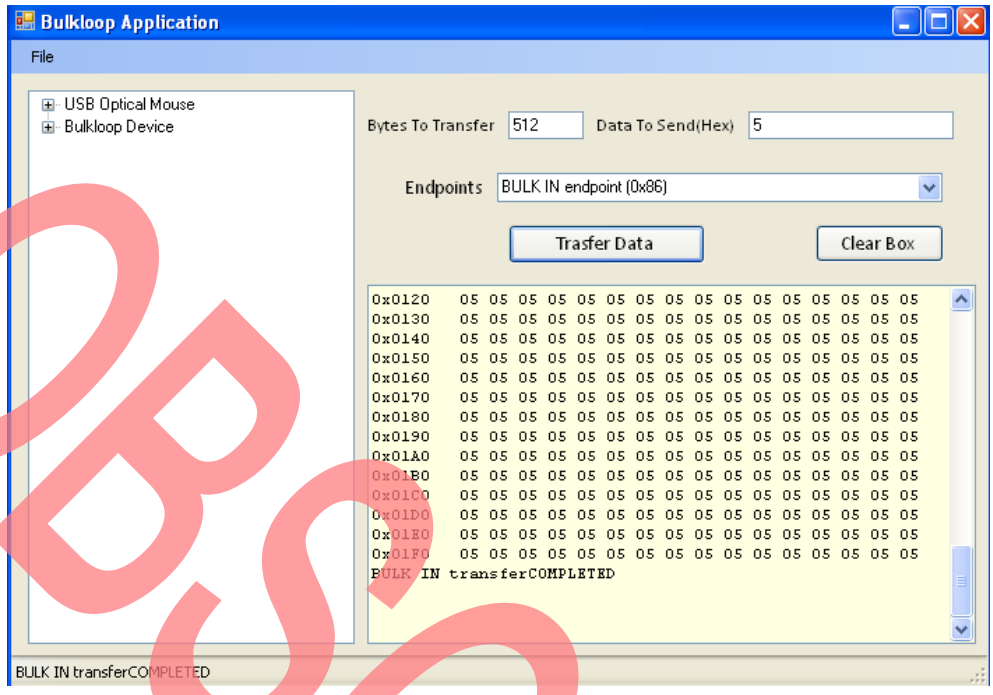
7. If the bulkloop device is connected and selected, then GUI appears as shown in the following figure.



8. If the bulkloop device is connected, select the bulkloop device from the Treeview control and also select the OUT endpoint, say EP2 OUT, from the combobox.
9. Enter the data byte (in Hex) to be transferred and the number of bytes to be transferred, in corresponding Textbox controls.
10. Click the 'Transfer Data' command button for transferring data OUT. The data bytes transferred and the result is displayed in the OutputBox (Textbox control).



11. Now select the corresponding IN endpoint, EP6 IN from the combobox and click 'Transfer Data' button for transferring the data in.



12. It can be seen that the data transferred IN is same as that is transferred OUT, due to the bulkloop firmware running in the device.
13. Similarly, it can be tested using the EP4 OUT and EP8 IN as well.

Additional Resources

1. Cypress CyUSB .NET DLL Programmer's Reference: For more information on CyUSB.dll
2. Introduction to CyUSB.dll Based Application Development Using C# - Helps in getting started with developing host applications using VC#
3. Introduction to CyAPI.lib Based Application Development Using VC++ - Helps in getting started with developing host applications using VC++
4. Getting Started with FX2LP™

About the Author

Name:	Gayathri Vasudevan
Title:	Applications Engineer
Contact:	gaya@cypress.com

Document History

Document Title: EZ-USB® FX2LP™ Host Application in VC++ 2008 Using Suite USB Library (CyUSB.dll) – AN70486

Document Number: 001-70486

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3319379	GAYA	07/18/2011	New code example.
*A	3518978	GAYA	02/07/2012	Converted code example to application note
*B	4428148	GAYA	07/02/2014	Obsolete document. Completing Sunset Review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com