AN70983

Designing a Bulk Transfer Host Application for EZ-USB® FX2LP™/FX3™

Author: Gayathri Vasudevan
Associated Project: Yes
Associated Part Family: CY7C68013A/14A/15A/16A, CYUSB3013,CYUSB3014
Software Version: Microsoft Visual C#, Visual C++, Visual Basics 2010 Express Edition
Related Hardware: CY3684 DVK, CYUSB3KIT - 001
For a complete list of the application notes, click here.

If you have a question, or need help with this application note, contact the author at gaya@cypress.com.

AN70983 describes the development of a USB host application that performs transfers between a PC and Cypress FX2LP and FX3 devices. Example applications are developed in three Microsoft Visual Studio.NET languages: Visual C#, Visual C++, and Visual BASIC using the Cypress USB Suite C# library, CyUSB.dll. This application note also explains the companion firmware used with FX2LP™ and FX3™.

Contents

Introduction

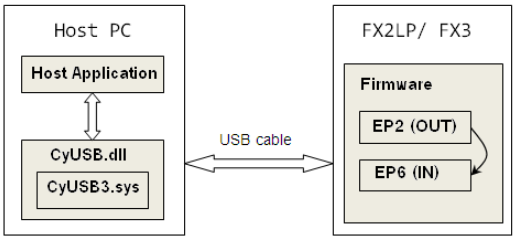
AN70983 demonstrates application development for Cypress products EZ-USB FX2LP and FX3 in three Microsoft Visual Studio.NET languages: Visual C#, Visual C++, and Visual BASIC using Cypress USB Suite C# library, CyUSB.dll.

The host application is built on the Microsoft Visual Studio.NET 2010 platform. The host application writes to BULK IN and BULK OUT endpoints of FX2LP and FX3 using the interfaces provided by CyUsb.dll. CyUsb.dll, in turn, communicates internally with the Cypress USB driver CyUsb3.sys to access these endpoints.

Firmware residing on FX2LP and FX3 transfers the data sent from the host application on the BULK OUT endpoint to the BULK IN endpoint. The PC retrieves the same data using IN transfers. This is called “loopback”.

Functional Overview

Figure 1. Block Diagram



In Figure 1, the 'Host' (PC) and the 'Target' are connected by a USB cable. The host application sends data to the BULK OUT (EP2) endpoint of FX2LP or FX3 by making calls to CyUSB.dll, which in turn communicates internally with the Cypress USB driver CyUsb3.sys. When the BULK data reaches the endpoint EP2, the firmware running on the target device takes this data and copies it back into the BULK IN (EP6) endpoint. When the host issues an IN request to EP6, the BULK data travels back in reverse fashion through CyUsb3.sys to CyUsb.dll and finally reaches the host application. Bulk data sent and received is displayed on the host application, along with the number of transferred bytes.

Preparing the USB Target

Before exploring the Windows code, a suitable USB target must be prepared to execute the loopback function shown in the right side of Figure 1. The companion .zip file to this application note contains two Firmware directories; one for Cypress FX2LP (*Bulkloop_FX2LP*) and the other for Cypress FX3 (*Bulkloop_FX3*). These directories contain the firmware code that you need to download into the respective USB chips to perform the USB device loopback.

Cypress USB Suite Installation

Cypress USB Suite is a set of USB development tools for Visual Studio. Cypress USB Suite includes:

- CyUSB3.sys generic USB kernel mode driver supporting USB 2.0 and USB 3.0 devices
- C#-based Managed class library, CyUSB.dll, to communicate with CyUSB3.sys driver
- C++-based static library, CyAPI.lib, to communicate with CyUSB3.sys driver
- USB3.0 compatible Control Center application to initialize and communicate with EZ-USB devices
- C# example applications with source code: Bulkloop, Control Center, and Streamer
- C++ example application with source code: Bulkloop and Streamer

Follow these steps to download and install the Cypress USB Suite:

1. Download the Cypress USB Suite for Windows, v1.2.2.
2. Unzip the contents of *cy_ssusbsuite_v1.2.2.zip* to the folder *cy_ssusbsuite_v1.2.2*.
3. Copy the extracted contents from the folder *cy_ssusbsuite_v1.2.2\EZ-USB FX3 SDK\1.2* to any convenient location, for example, *C:\Program Files\Cypress\Cypress USBSuite*.

FX2LP

EZ-USB FX2LP operates at a 480-Megabit per second signaling rate while maintaining compatibility with the earlier USB full-speed 12-Megabit per second rate. FX2LP uses high-speed RAM for program storage, so the first step is to load the loopback code into the FX2LP chip. You can do this by using the FX2LP DVK, available at CY3684 DVK.

Cypress provides an application called USB Control Center, available at the Cypress USB Suite. This application has a **Program** tab to load code images into the FX2LP DVK.

To load the loopback code, follow these steps:

1. Configure the DVK board according to this table:

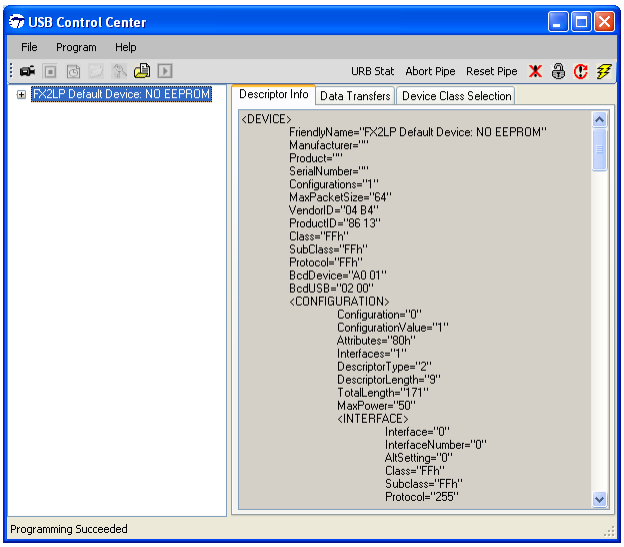
Jumper/Switch	Purpose	Setting
JP1, JP10	3.3V to EZ-USB chip	IN,IN
JP2	Vbus powered	IN
JP3	Activates 4 LEDS	IN(x4)
JP5	3.3V current monitor	IN
JP6, JP7	FX2LP Memory map (int + ext RAM)	OUT, OUT
JP8	Secondary Wakeup	x
SW1	Small/Large EEPROM	x
SW2	EEPROM Enable	OFF (No EEPROM)

SW2 controls how the FX2LP chip appears to a host when first connected to a USB port. In the OFF position the FX2LP enumerates as a default USB device (with VID/PID 0x04B4/0x8613), ready to download user firmware over USB. If SW2 is set to ON when connecting the USB, FX2LP can boot from the onboard EEPROM instead of from the PC. This is not desirable – ensure that SW2 is in OFF position.

2. Start the USB Control Center application.

3. Connect the FX2LP DVK board into a USB port. The board should be recognized as shown in Figure 2.

Figure 2. Control Center

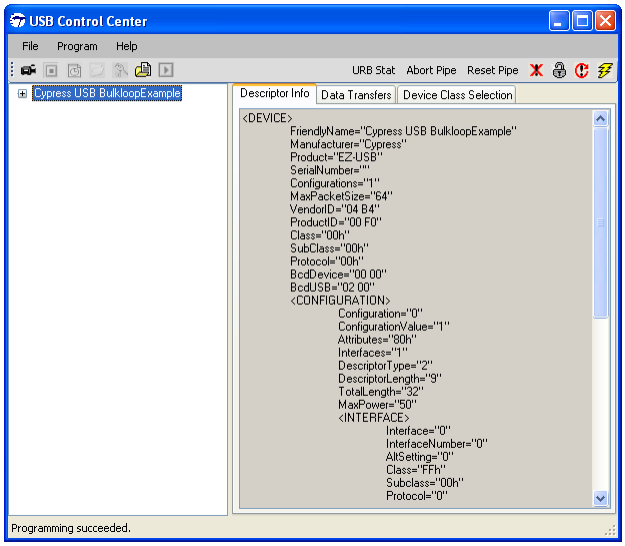


The USB device shown in the left panel of Figure 2 is the default FX2LP device, when the FX2LP board is connected with no EEPROM selected.

If you do not see anything in the left panel, the FX2LP board may be configured incorrectly, or the DVK board driver may not be installed in Windows. Refer to the Appendix in this case.

4. Select **Program > FX2 > RAM** and navigate to the bulkloop.hex file in AN70983_Project\Firmware\ Bulkloop_FX2LP, which comes as attachment with this application note. Double-click on the filename and a "Programming…" message appears at the bottom of the USB Control Center window, followed by a "Programming succeeded" message. LED (D5) on FX2LP DVK should be blinking. If not, make sure the four JP3 jumpers are in place—these jumpers put the LEDS under FX2LP control. The USB Control Center left pane now looks as shown in Figure 3:

Figure 3. Control Center



This is EZ-USB ReNumeration in action. After the BootLoader loaded the Bulkloop example in step 4, the FX2LP device disconnected from USB and automatically re-connected as the "new" device defined by the Bulkloop firmware.

The FX2LP DVK is now ready to serve as the USB loopback peripheral for the PC example code in this app note. Note that the "Bulkloop_FX2LP" folder contains the Keil project which compiled to bulkloop.hex, so you may experiment with the source code if you wish.

FX3

EZ-USB FX3 operates at 5 Gigabits per second signaling rate USB ("SuperSpeed") while maintaining compatibility with previous USB generations. Our first step is to load the loopback code into the FX3 chip. This is done using the FX3 DVK (Development Kit), available here CYUSB3KIT-001.

Cypress provides an application called "USB Control Center", available at Cypress USB Suite. This application has a "Program" tab to load code images into the FX3 DVK.

To load the loopback code, follow these steps:

1. Configure the DVK board according to this table.

Jumper/Switch	Purpose	Setting
PMODE Switch	Select USB download	1,2-OFF, 3,4- x
J96,J97	Select USB download	Upper 2 pins (2-3)
J98	Select USB download	Unconnected
J53	Vbus Powered	IN

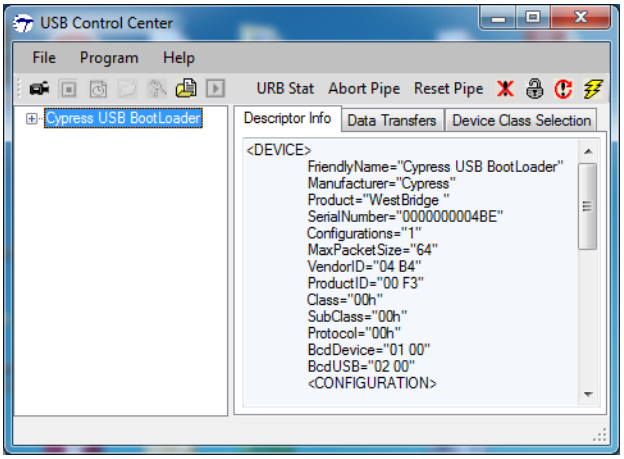
Note: This table specifies the jumper settings for the FX3 Rev3 DVK.

These settings control how the FX3 chip appears to a host when first plugged into a USB port. They allow the FX3 board to enumerate as a loader device, ready to download user firmware.

2. Start the USB Control Center application.

3. Connect the FX3 DVK board into a USB port. The board should be recognized as shown in Figure 4.

Figure 4. Control Center

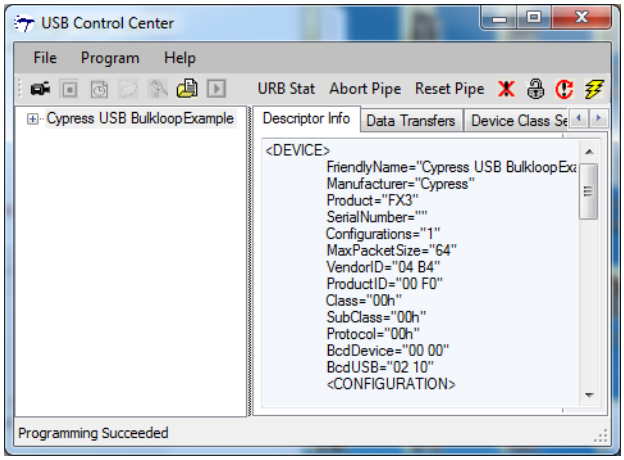


The USB device shown in the left panel is the FX3 board with its boot loader running from internal memory. This means that FX3 is ready to accept a download.

If you don't see anything in the left panel, the FX3 board may be configured incorrectly, or the DVK board dirver may not be installed in Windows. Refer to the Appendix section in this case.

4. Select **Program > FX3 > RAM** and navigate to the file *USBBulkLoopAuto.img* in *AN70983_Project\Firmware\Bulkloop_FX3\USBBulkLoopAuto\Debug*. Double-click the filename and a "Programming…" message appears at the bottom of the USB Control Center window, followed by a "Programming succeeded" message. The USB Control Center left panel now looks like this:

Figure 5. Control Center



This is EZ-USB ReNumeration in action. After the Bootloader loads the Bulkloop example in step 4, the FX3 device is disconnected from the USB and automatically re-connected as the "new" device defined by the bulkloop firmware.

The FX3 DVK is now ready to serve as the USB loopback peripheral for the PC example code in this application note. Note that the *Bulkloop_FX3* folder contains the Eclipse project that is compiled to *bulkloop.img*, so you can experiment with the source code.

To familiarize with the functional blocks in Figure 1, refer to the following documentation:

- CyUSB.dll: The Programmer's Reference C# library (*CyUSB.NET.pdf*) is available with Cypress USB Suite at *C:\Program Files\Cypress\Cypress USBSuite\library\c_sharp* after you copy the Cypress USB Suite contents as instructed in the section Cypress USB Suite Installation Instructions.
- Cyusb3.sys: Cypress CyUsb3.sys Programmers Reference.
- CY3684 (FX2LP DVK): *EZ-USB_GettingStarted.pdf* and *DVK Users Guide.pdf* are available at *C:\Cypress\USB\ doc\General* after you install the DVK from the FX2LP Development Kit.
- FX2LP: *EZ-USB_TRM.pdf* and *fx2lp_datasheet.pdf* are available at *C:\Cypress\USB\doc* after you install the DVK from the FX2LP Development Kit.
- CYUSB3KIT - 001 (FX3 DVK): CYUSB3KIT – 001 User Guide and Quick Start Guide.
- FX3: Getting Started with FX3 - This document serves as a starting point for a new user to get familiar with FX3.
- FX2LP: Getting started with FX2LP - This document serves as a starting point for a new user to get familiar with FX2LP.

Application Overview

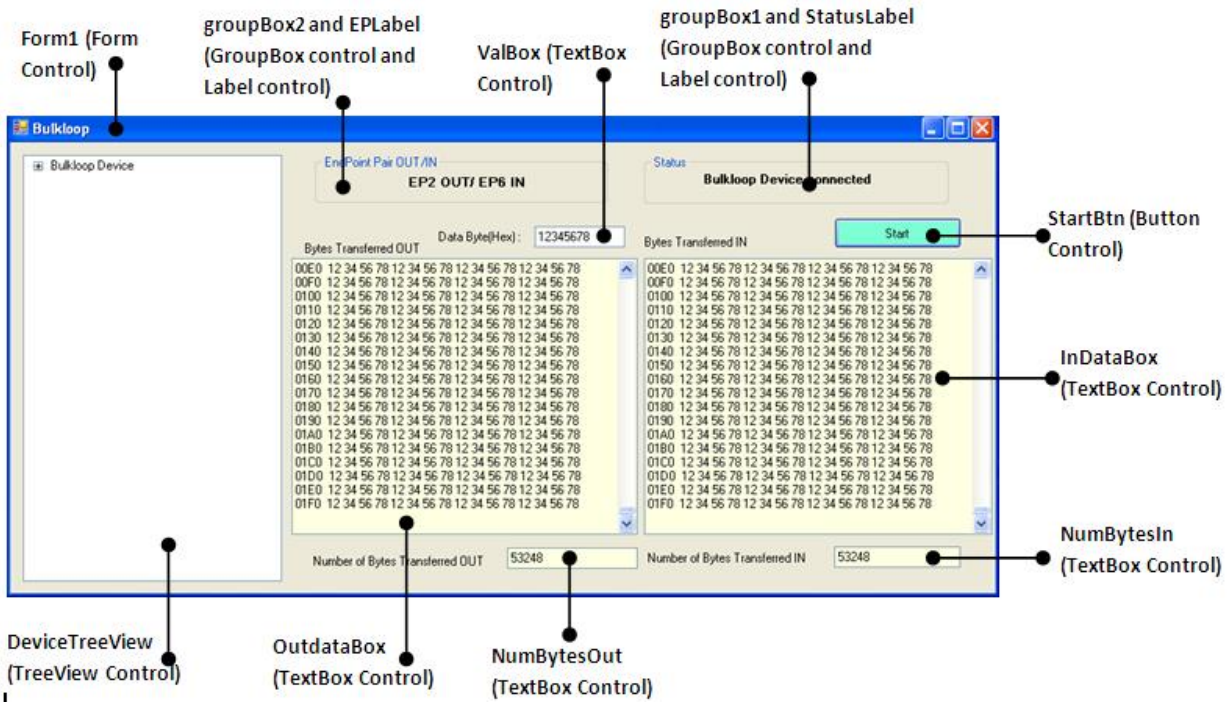
The Visual Studio application files are provided with this application note as *AN70983_Project.zip*. The host application is included in the *AN70983_Project\Application* folder. Refer to the directory structure.

To launch the application, navigate to the language you are using in the Application folder, and click *BulkLoop_xxx.exe* in the application folder. Information about the application's user interface with application operation details and source code details are explained in the following sections.

Features

The user interface of the host application is shown in Figure 6.

Figure 6. User Interface



This application user interface is designed to show:

- **Status:** Reports whether the target FX2LP device is currently connected or disconnected.
- **Endpoint Pair OUT/IN:** Endpoint pair on the target device, currently in use as OUT and IN endpoints for transferring data to and getting data from the target. This is informational only (not user-editable), reflecting the settings in the source code.
- **Data Bytes (Hex):** A user-selectable data pattern (maximum allowed length of 8 nibbles or 4 bytes), which will be transferred in loopback fashion during bulk-loop operation.
- **Bytes Transferred OUT:** The log of bytes transferred from the host PC to the target. Each line begins with the starting address of bytes in the line.
- **Bytes Transferred IN:** Same as OUT, except that the bytes transferred are IN bytes from the target to the host PC.
- **Number of Bytes transferred OUT:** Number of bytes transferred from the PC to the target.
- **Number of Bytes transferred IN:** Number of bytes transferred from the target to the PC.
- **Start/Stop:** Bulk-loop operation start/stop button.
- **DeviceTreeView**: Displays all the connected devices bound to Cyusb3.sys.

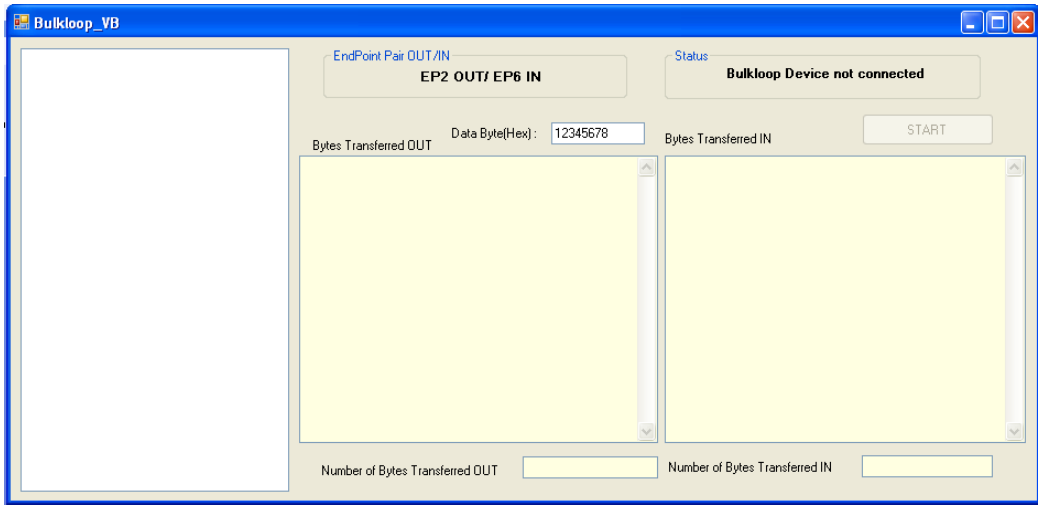
Operating Instructions

Note: The applications provided with this application note have been tested using FX2LP and FX3 DVK in Windows XP (32-bit) and Windows 7 (32-bit and 64-bit) systems.

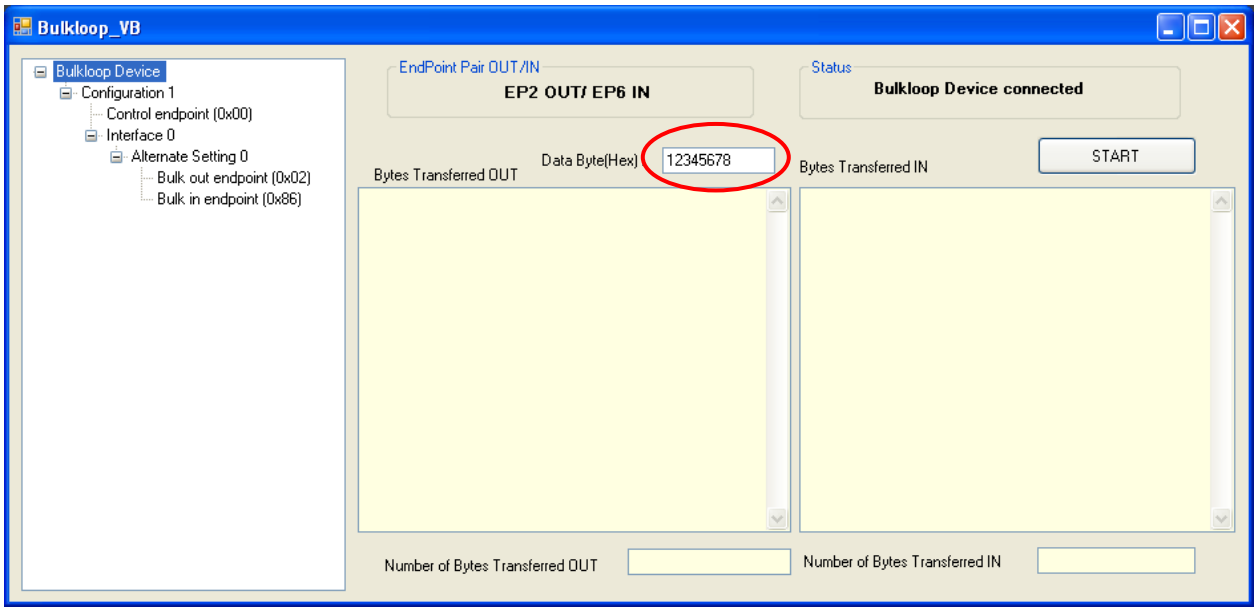
1. Open the Host Application placed under the folder Application. To launch the application, click **BulkLoop_xxx.exe**, placed under the application folder.

   Note: .NET framework 4.0 is required for running these Visual Studio projects.

2. If the bulkloop device is not connected, the GUI appears as shown in the following figure:

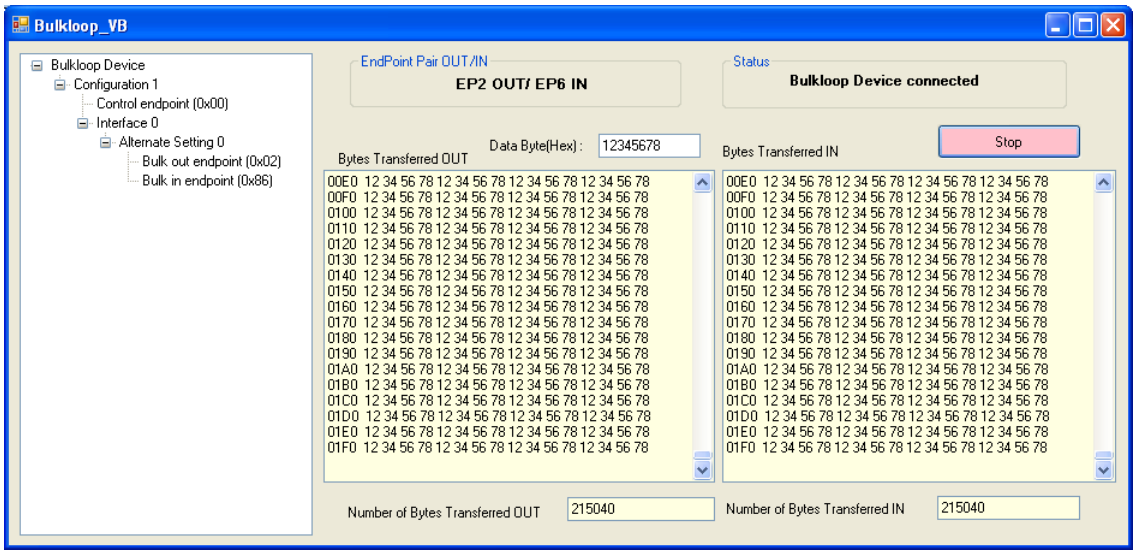


   If the bulkloop device (FX2LP/ FX3) is connected, then the GUI appears as shown in the following figure:

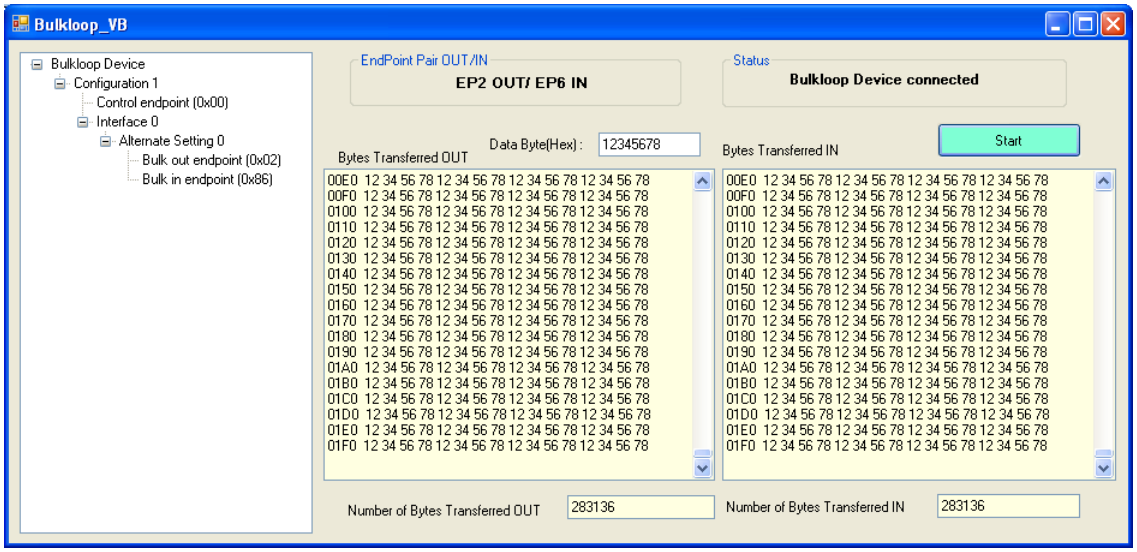


3. Enter the data byte (in Hex) to be transferred in the Textbox control (this is circled in the previous figure).

4. Click the **Start** button. Both BULK IN and OUT transfers start.



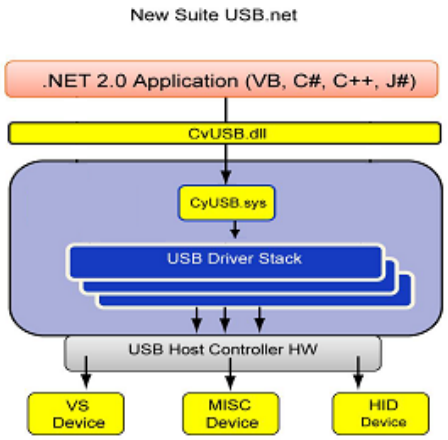
5. Click the Stop button to stop the transfers. Bytes transferred so far (OUT and IN) and numbers of bytes transferred so far (OUT and IN) are displayed in the corresponding textboxes.



Adding Reference to CyUSB.dll

CyUSB.dll is a managed Microsoft .NET class library. It provides a high-level, powerful programming interface to USB devices. Because *CyUSB.dll* is a managed .NET library, you can access its classes and methods from any of the Microsoft Visual Studio.NET managed languages such as Visual Basic.NET, C#, Visual J#, and managed C++.

Figure 7. Application Development with CyUSB.dll



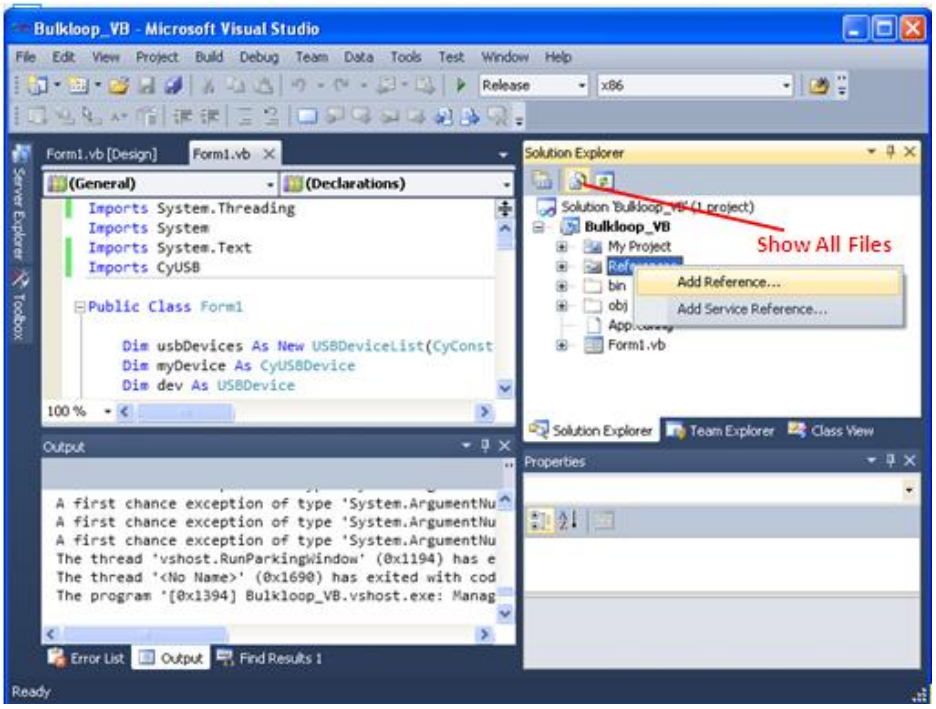
To use the library, you need to add a reference to CyUSB.dll. After this library is added, any source file that accesses the CyUSB namespace requires a line to include the namespace in the appropriate syntax. For more information about CyUSB.dll, refer to the Programmer's Reference C# library (*CyUSB.NET.pdf*) that is present with the Cypress USB Suite at *C:\Program Files\Cypress\Cypress USBSuite\library\c_sharp* after you copy the Cypress USB Suite contents as instructed in section Cypress USB Suite Installation Instructions.

Adding Reference to CyUSB.dll in Visual Basics Application:

To view the source, open *BulkLoop_VB.sln* under *Application\VB Application\Source\Bulkloop_VB*.

- Expand the project *Bulkloop_VB* in solution explorer.
- Click **Show all files** as shown in Figure 8. Right-click **Reference** and **Add Reference**, under the Solution Explorer window as shown in Figure 8. Select the **Browse** tab and browse to the installation directory of Cypress USB Suite and double-click **CyUSB.dll**.

Figure 8. Adding Reference to CyUSB.dll in VB



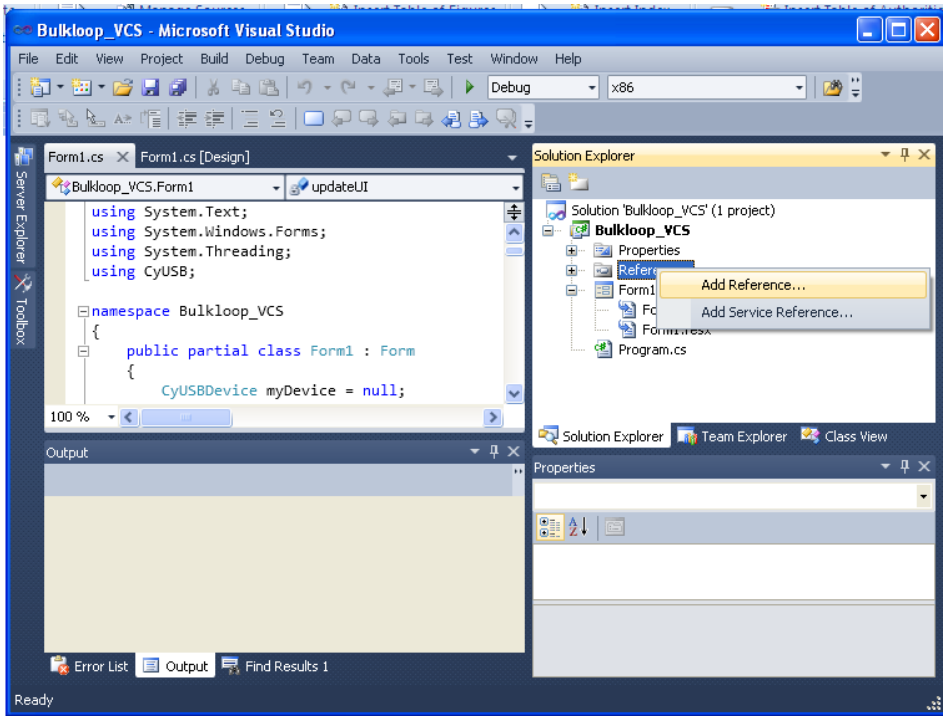
- Click **View > Code** and add the statement "`Imports CyUSB`" before "`Public Class Form1`" as shown in the code window (Form1.vb) of Figure 8. .

Adding Reference to CyUSB.dll in Visual C# Application:

To view the source, open *BulkLoop_VCS.sln* under *Application\VCS Application\Source\Bulkloop_VCS*.

- Expand the project *Bulkloop_VCS* in the solution explorer.
- Right-click **Reference** and **Add Reference**, under the Solution Explorer window as shown in Figure 9. Select the **Browse** tab and browse to the installation directory of Cypress USB Suite and double-click **CyUSB.dll**.

Figure 9. Adding Reference to CyUSB.dll in VC#



- Click **View > Code** and add the statement "`using CyUSB`" before "`namespace Bulkloop_VCS`" as shown in the code window (Form1.cs) of Figure 9.

Adding reference to CyUSB.dll in Visual C++ Application:

To view the source, open the *BulkLoop_VCPP.sln* under *Application\VC++ Application\Source\Bulkloop_VCPP*.

- Expand the project *Bulkloop_VCPP* in the solution explorer.
- Go to **Project > Properties**. In the dialog box, select **Common Properties > Add New Reference** as shown in Figure 10 and Figure 11. In the window that opens, select the **Browse** tab and point to the CyUSB.dll in the installation directory of Cypress USB Suite.

Click **View Code** and add the statement `#using "C:\Program Files\Cypress\Cypress USBSuite\library\c_sharp\lib\CyUSB.dll"` after `#pragma once` as shown in Figure 12. The path to be included here depends on the install directory path of the Cypress USB Suite.

Add the statement "`using namespace CyUSB`" after the various "using namespace" statements as shown in the following code (also see Figure 12).

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

Figure 10. Adding Reference to CyUSB.dll in VC++

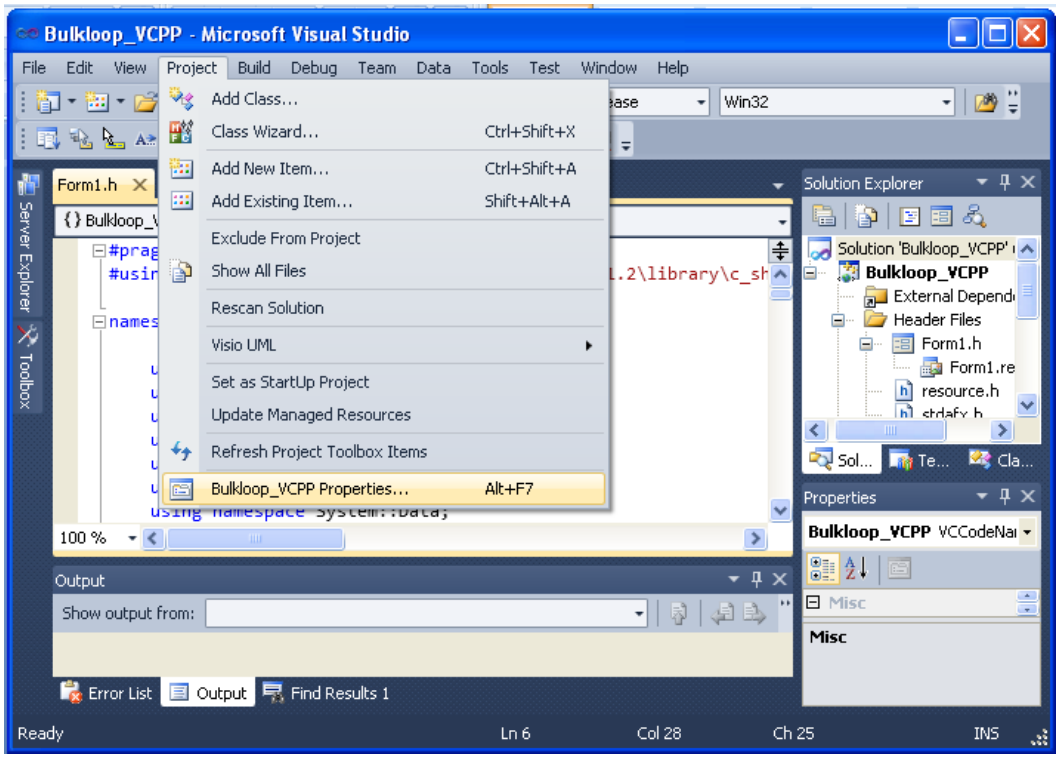


Figure 11. Adding Reference to CyUSB.dll in VC++

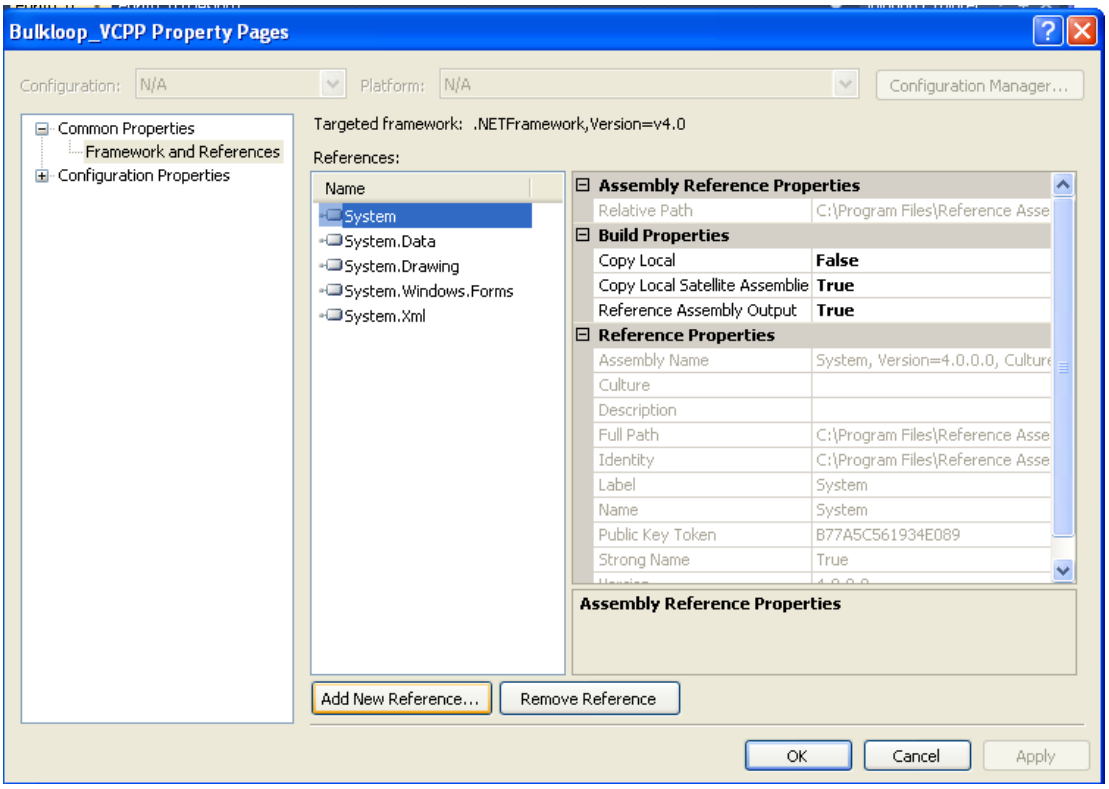
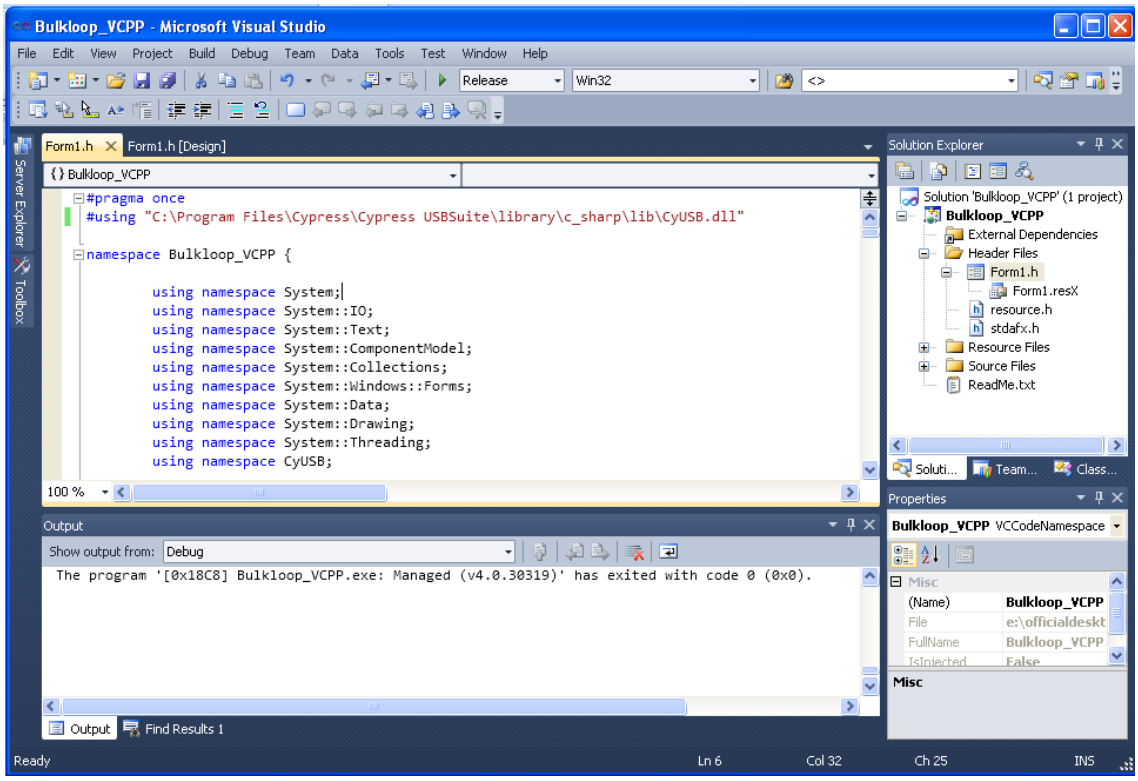


Figure 12. Adding Reference to CyUSB.dll in VC++



Source Code Overview

This section explains the main functionalities of the host application code. For more information about the various CyUSB.dll APIs and classes, refer to the Programmer's Reference C# library (CyUSB.NET.pdf) available with the Cypress USB Suite.

The important steps in the code are given in the following table:

Action/ Function	Description	CyUSB.dll API
Object initialization	• Create and initialize instances of the classes USBDeviceList, CyUSBDevice, CyBulkEndPoint and other required variables. • "usbDevices" represents a dynamic list of USB devices that are accessible via the class library, which populates itself with USBDevice objects representing all the USB devices bound to Cyusb3.sys, served by the device selector mask, "DEVICES_CYUSB".	• USBDeviceList represents a dynamic list of USB devices that are accessible via the class library. When an instance of USBDeviceList is created, it populates itself with USBDevice objects representing all the USB devices served by the indicated device selector mask. These USBDevice objects have all been properly initialized and are ready for use. • The CyUSBDevice class represents a USB device attached to the Cyusb3.sys device driver. • CyBulkEndPoint is a subclass of the CyUSBEndPoint abstract class.
Device attach and removal detection	• When the user connects the target hardware and the connection between the host and the target is established, the event_handler function usbDevices_DeviceAttached() is called. • When the target device is disconnected, the event_handler function usbDevices_DeviceRemoved() is called. • Both of these event handlers in turn call function RefreshDevice().	
RefreshDevice()	• Instantiates "myDevice" with the device of interest, based on the VID and PID of the device. The bulkloop firmware used in this example uses the VID/PID: 0x04B4/ 0x00F0. • If there are multiple devices with this VID/PID, "myDevice" will be instantiated with the handle to the first connected device. • The status label and the Start/ stop button is enabled or disabled accordingly. • If the device of interest if found, then the "BulkOutEndpt" and "BulkInEndpt" are instantiated with endpoints found in the device, EP2 and EP6 respectively, since the bulkloop firmware in this example uses the same. • Clears and updates the treeview with the latest list of devices bound to Cyusb3.sys.	• USBDeviceList[int VID, int PID]: This index operator provides access to elements of the USBDeviceList based on the VendorID and ProductID properties of the USBDevice objects in the list. • EndpointOf(byte addr): Returns the CyUSBEndPoint object whose Address property is equal to addr. Returns null if no endpoint with Address = addr is found.
Data Transfer Operation	• StartBtn_Click() eventhandler is invoked on clicking "Start" button, • This function first takes the user's value from the 'Data Bytes(Hex)' field of the application user interface, changes the text of the button to "Stop" and starts the thread "tXfers" where the data communicate on is actually carried out. • The other parent thread from now on just take cares of application user interface events and other OS events. • If the thread is already running, this eventhandler aborts the thread and changes the text of the button back to "Start"	

Action/ Function	Description	CyUSB.dll API
TransfersThread()	• Calls SetOutputData() to setup the data for OUT transfers. • After the thread is started, data transfers in the out direction and IN direction are started using the XferData() API.	XferData(ref byte[] buf, ref int 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 (that is, blocking) I/O operations and does not return until the transaction completes or the endpoint's TimeOut has elapsed. Returns true if the transaction successfully completes before TimeOut has elapsed. The number of bytes actually transferred is passed back in *len*.
SetOutputData()	• It takes the user-given data in the 'Data Bytes(Hex)' field of the application user interface and fills the 'outData[]' buffer with these bytes, which are transferred out in the OUT transfers.	
StatusUpdate()	• This is the call back function for updating the UI and is called from TransfersThread. This updates the textboxes in the UI with the Bytes transferred and number of Bytes transferred.	

Asynchronous Communication

In this application, synchronous communication is used to communicate data between the host and the target. In this method, the data transfer function 'XferData' does not return until the full data is transferred. In other words, the application gets locked until the given data transfer returns. After that, the next data transfer operation starts. This method is suitable for most applications. But with applications that need to use the whole data bandwidth available for transfers, the more advanced asynchronous communication method is used. However, only very advanced users should use this transfer method. The 'Streamer' example that comes with Cypress USBSuite shows how to use the Asynchronous communication method.

Firmware

Firmware configures the endpoints and the interface to carry out the bulk data loopback operation. Two endpoints are configured to handle bulk transfer: one OUT endpoint and one IN endpoint. Data sent from the host is stored in an OUT endpoint buffer. This data is transferred to the IN endpoint and then sent back to the host on request.

FX2LP Bulkloop Firmware

The following code snippets give an explanation of important code functionalities of the FX2LP bulkloop firmware:

	Description	Code Snippets	Function Name	File name

Descriptor Information	• VID: 0x04B4
	• PID: 0x00F0
	• Device Class: Vendor
	• Number of configurations: 1
	• Number of interfaces: 1
	• Number of endpoints: 2
	• Endpoints supported:
	  ▫ EP2 BULK OUT, 512 byte maxpacket size
	  ▫ EP6 BULK IN, 512 bytes maxpacketsize
				dscr.a51

	Description	Code Snippets	Function Name	File name
Initialization	• Different component initialization of the FX2LP-based target is done in the TD_Init() function. This function is called from main() for initialization.		TD_Init()	bulkloop.c
	• Sets the CPU clock to 48 MHz	`CPUCS =((CPUCS & ~bmCLKSPD) | bmCLKSPD1);`		
	• Configures the interface of the FX2LP. • The FX2LP is configured in the slave FIFO mode • Interface clock is chosen to be internal 48MHz clock.	`IFCONFIG |= 0x40;`		

	The endpoints are configured as follows:
	• Endpoint 2 – OUT, Bulk, double buffered, 512 bytes
	• Endpoint 6 – IN, Bulk, double buffered, 512 bytes
		`EP2CFG = 0xA2;`
		`SYNCDELAY;`
		`EP6CFG = 0xE2;`
		`SYNCDELAY;`		

	• The OUT endpoints, after configuring, need to be armed to accept packets from the host. Because the endpoints are double buffered, two packets must be skipped to arm the endpoint. Arming is essentially freeing up the buffers and making them available to the host to receive packets.
	• By writing a 1 to bit 7 of the byte count register, the packet is skipped.
		`EP2BCL = 0x80;`
		`SYNCDELAY;`
		`EP2BCL = 0x80;`
		`SYNCDELAY;`		

	• Enables the AUTO pointer used for data transfer in the TD_Poll function.	`AUTOPTRSETUP |= 0x01;`		

	Description	Code Snippets	Function Name	File name
Data Loopback	• The bulkloop implementation is carried out in the TD_Poll() function. This function is called repeatedly when the device is idle		TD_Poll()	bulkloop.c

	• Endpoint 2 is first checked to see if it has data. This is done by reading the endpoint 2 empty bit in the endpoint status register (EP2468STAT).
	• If endpoint 2 has data (sent from the host), the capability of endpoint 6 to receive the data is checked. This is done by reading the endpoint 6 'Full' flag indicated by a Full bit in the endpoint status register.
	• If endpoint 6 is not full, then the data is transferred.
		`if(!(EP2468STAT & bmEP2EMPTY))`
		`{`
		`if(!(EP2468STAT & bmEP6FULL))`
		`{`
		`}`
		`}`		

	• For transferring the data, the data pointers are initialized to the corresponding buffers. The first auto pointer is initialized to the first byte of the endpoint 2 FIFO buffer.	`APTR1H = MSB( &EP2FIFOBUF );` `APTR1L = LSB( &EP2FIFOBUF );`		
	• The second auto pointer is initialized to the first byte of the endpoint 6 FIFO buffer.	`AUTOPTRH2 = MSB( &EP6FIFOBUF );` `AUTOPTRL2 = LSB( &EP6FIFOBUF );`		
	• The number of bytes to be transferred is read from the byte count registers of Endpoint 2. The registers EP2BCL, EP2BCH contain the number of bytes written into the FIFO buffer by the host. These two registers give the byte count of the data transferred to the FIFO in and OUT transaction as long as the data is not committed to the peripheral side.	`count = (EP2BCH << 8) + EP2BCL;`		

	• As auto pointers have been enabled, the pointers increment automatically, and the statement 'EXTAUTODAT2 = EXTAUTODAT1' transfers data from endpoint 2 to endpoint 6. Each time this statement is executed, the auto pointer is incremented. This statement is executed over and over to transfer each byte from endpoint 2 to 6.	`for( i = 0x0000; i < count; i++ )`
		`{`
		`EXTAUTODAT2 = EXTAUTODAT1;`
		`}`		

	• After the data is transferred, endpoint 2 has to be re-armed to accept a new packet from the host. Endpoint 6 has to be "committed", that is, make the FIFO buffers available to the host for reading data from the Endpoint 6	`EP6BCH = EP2BCH;`
		`SYNCDELAY;`
		`EP6BCL = EP2BCL;`
		`SYNCDELAY;`
		`EP2BCL = 0x80;`		

FX3 Bulkloop Firmware

The following code snippets give an explanation of important code functionalities of the FX3 bulkloop firmware:

	Description	FX3 APIs	Function Name	File name
Descriptor Information	• VID: 0x04B4 • PID: 0x00F0 • Device Class: Vendor • Number of configurations: 1 • Number of interfaces: 1 • Number of endpoints: 2 • Endpoints supported: ▫ EP2 BULK OUT, maxpacket size of 1024 byte for SS, 512 byte for HS ▫ EP6 BULK IN, maxpacket size of 1024 bytes for SS, 512 byte for HS			cyfxbulklpdscr.c

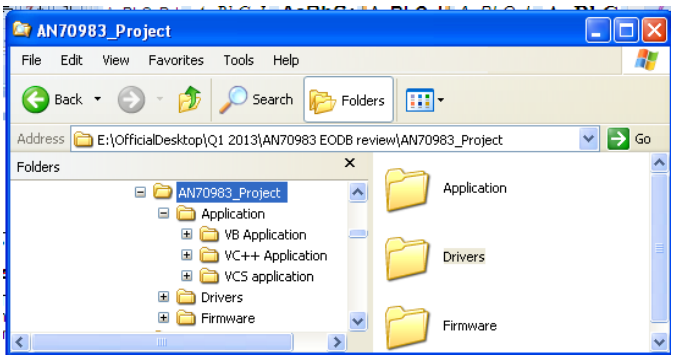
Initialization	• FX3 device is initialized
	• Sets the functional mode of the 61 IO pins available in FX3
	• The RTOS kernel is invoked next from the main(). This is a non-returning call and sets up the Threadx kernel. This function needs to be invoked as the last function from the main () function.
		`CyU3PDeviceInit()`
		`CCyU3PDeviceConfigureIOMatrix()`
		`CyU3PKernelEntry()`
			main()	cyfxbulklpauto.c

	• FX3 application definition function. This is invoked from the system module after the device and all the modules are initialized. The application threads and other required OS primitives can be created here. At-least one thread must be created. This function should not be explicitly invoked. • BulkLpAppThread_Entry is the application thread created.	`CyU3PThreadCreate()`	CyFxApplicationDefine()	
	• CyFxBulkLpApplnDebugInit() initializes the debug module • CyFxBulkLpApplnInit() initializes the bulkloop application		BulkLpAppThread_Entry()	
	• Start the USB device mode driver in the USB 3.0 platform and also to create the DMA channels required for the control endpoint. • Register a USB setup request handler with the USB driver. • Registers a USB event callback function with the USB driver. CyFxBulkLpApplnUSBEventCB is registered in this application. • Register a USB 3.0 LPM request handler callback. • Register USB descriptors with the USB driver. The driver is capable of remembering one descriptor each of the various supported types as well as up to 8 different string descriptors. The driver only stores the descriptor pointers that are passed in to this function, and does not make copies of the descriptors. The caller therefore should not free up these descriptor buffers while the USB driver is active. • Enable/disable the USB connection. This function is used to enable or disable the USB PHYs on the USB 3.0 platform and to control the connection to the USB host in that manner.	• `CyU3PUsbStart()` • `CyU3PUsbRegisterSetupCallback()` • `CyU3PUsbRegisterEventCallback()` • `CyU3PUsbRegisterLPMRequestCallback()` • `CyU3PUsbSetDesc()` • `CyU3PConnectState()`	CyFxBulkLpApplnInit()	

	Description	FX3 APIs	Function Name	File name
	• CyFxBulkLpApplnStart () is called upon Set Configuration event, which in turn configures USB endpoint's properties.	• `CyU3PSetEpConfig()`	CyFxBulkLp ApplnStart ()	
Data Loopback	• Creates a DMA channel. A DMA channel requires a producer socket, a consumer socket, a set of buffers and a callback function. In this example, an auto channel is created between EP2 as producer socket and EP6 as consumer socket. Whenever, EP2 receives any data it will be automatically committed to EP6. • Set a transfer on the DMA channel. The function starts a transaction the selected DMA channel.	• `CyU3PDmaChannelCreate()` • `CyU3PDmaChannelSetXfer()`	CyFxBulkLp ApplnStart ()	

Project Directory Structure

The following figure shows the directory structure of the project folder attached with this application note:



AN70983_Project

This project folder contains the following folders:

- Application
- Drivers
- Firmware

Application

This folder contains the following folders that contain the host application in Visual Studio.NET language as indicated by the folder name:

- VB Application
- VC++ Application
- VCS Application

Each of these folders contains the following sub-folders:

Application: Contains application executable (Bulkloop_xxx.exe) along with other supporting files.

Source: Contains bulkloop application sources. You can open the sources by opening BulkLoop_xxx.sln in Microsoft Visual Studio 2010.

Drivers

This folder contains the drivers required to connect FX2LP/FX3 to hosts that have different operating systems. Each of the folders contains a CyUSB.inf and Cyusb3.sys file for FX2LP/FX3. The following table shows the folder path for different operating systems:

Operating System	Folder Path
Windows XP 32-bit	wxp\x86
Windows7 32-bit	win7\x86
Windows7 64-bit	win7\x64
Windows Vista 32-bit	wlh\x86
Windows Vista 64-bit	wlh\x64

Firmware

This folder contains the following folders that contain the firmware that runs on the target device.

Bulkloop_FX2LP:

This folder contains 'BulkLoop' firmware for FX2LP and provides loopback. You can see the firmware sources by opening *bulkloop.Uv2* using Keil uVision 2.

Bulkloop_FX3:

This folder contains 'BulkLoop' firmware for FX3 and provides loopback. You can open the firmware sources using Eclipse IDE.

System Requirements

Hardware

FX2LP

An FX2LP DVK (CY3684) board is used as the development and testing platform for this example for FX2LP. A detailed schematic of the DVK is provided in the hardware folder after FX2LP DVK install. More information about the board is available in the 'EZ-USB Advanced Development Board' section of the *EZ-USB_Getting Started* document available with the FX2LP DVK.

FX3

An FX3 DVK (CYUSB3KIT - 001) board is used as the development and testing platform for this example for FX3. A detailed schematic of the DVK is provided in the hardware folder (after you install the FX3 DVK).

Software

- Microsoft .NET framework 4.0
- ControlCenter - Available by installing the Cypress USB Suite.
- Keil uVision 2 – The 4K-byte evaluation version is available with the CY3684 DVK. For the full version, contact Keil.
- Eclipse IDE – Available by installing FX3 SDK.
- Microsoft Visual Studio 2010.

Additional Resources

- Cypress USBSuite Application Development- Quick Start Guide, available as part of Cypress USB Suite, helps in getting started with developing host applications in Visual C# using CyUSB.dll; and Visual C++ using CyAPI.lib.
- Programmer's Reference C# library (CyUSB.NET.pdf) is available with Cypress USB Suite at *C:\Program Files\Cypress\Cypress USBSuite\library\c_sharp* after copying the Cypress USB Suite contents as instructed in the section Cypress USB Suite Installation Instructions.
- More application examples are provided with the Cypress USB Suite such as:
  - Streamer: Can be used to test the data transfer.
  - Control Center: User interface to communicate with USB devices served by CyUSB3.sys.
- EZ-USB® FX2LP™/ FX3™ Developing Bulk-Loop Example on Linux describes how you can use libusb to develop a USB host application on a Linux-based operating system for Cypress EZ-USB FX2LP/ FX3 products.
- EZ-USB® FX2LP™ - Developing USB Application on MAC OS X using LIBUSB describes how libusb-1.0 can be used to develop USB host applications (Cocoa Application) on MAC OS X 10.6/10.7 for Cypress EZ-USB FX2LP products.

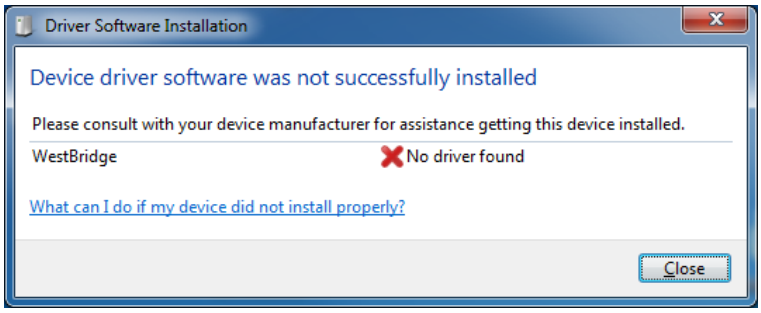
Glossary

Term	Meaning
Microsoft Visual Studio	Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework,.NET Compact Framework and Microsoft Silverlight.
Microsoft Visual C#	The Microsoft version of the C# language designed for managed environments.
Microsoft Visual C++	Commercial, integrated development environment (IDE) product from Microsoft for the C, C++, and C++/CLI programming languages.
Microsoft Visual Basic	Visual Basic .NET (VB.NET) is an object-oriented computer programming language that can be viewed as an evolution of the classic Visual Basic (VB), implemented on the .NET Framework. Visual Basic is a third-generation event-driven programming language and integrated development environment(IDE) from Microsoft for its COM programming model
Cy3684 DVK	Cypress proprietary development kit for the EZ-USB FX2LP family. For more information, see the Cypress webpage.
CYUSB3KIT - 001	Cypress proprietary development kit for the EZ-USB FX3. For more information, see the Cypress webpage
EZ-USB	Generic name for all Cypress proprietary high-speed USB products such as FX2LP.
FX2LP	Cypress proprietary high-speed USB controllers. For more information, see the Cypress webpage.
FX3	Cypress's EZ-USB FX3 is the next-generation USB 3.0 peripheral controller, providing integrated and flexible features. For more information, see the Cypress webpage.
Cypress USB Suite	Cypress proprietary USB development tools for Visual Studio. For more information, see the Cypress webpage.
CyUSB.dll	Cypress proprietary library for software development using .NET technologies. For more information, see the Programmer's Reference C# library (CyUSB.NET.pdf) that is available with Cypress USB Suite.
Cyusb3.sys	Cypress proprietary library for USB communication. For more information, see the Cypress CyUsb3.sys Programmers Reference.
USB Control Center	Cypress proprietary application for general-purpose USB functionality such as device identification and image downloads. The application is available through Cypress USB Suite.
Keil uVision	The 4K-byte evaluation version is available with the CY3684 DVK. For the full version, contact Keil.
Eclipse IDE	As part of the development tools, the FX3 SDK provides the Eclipse IDE for C/C++ developers. It consists of Eclipse base platform and the CPP feature. This firmware development environment helps you to develop, build, and debug firmware applications for FX3.

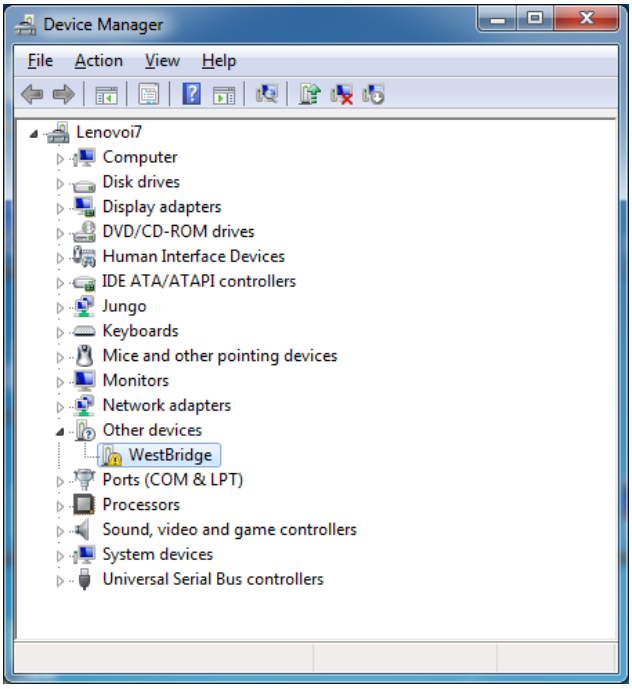
Appendix

Windows Install of FX2LP/ FX3 DVK Driver

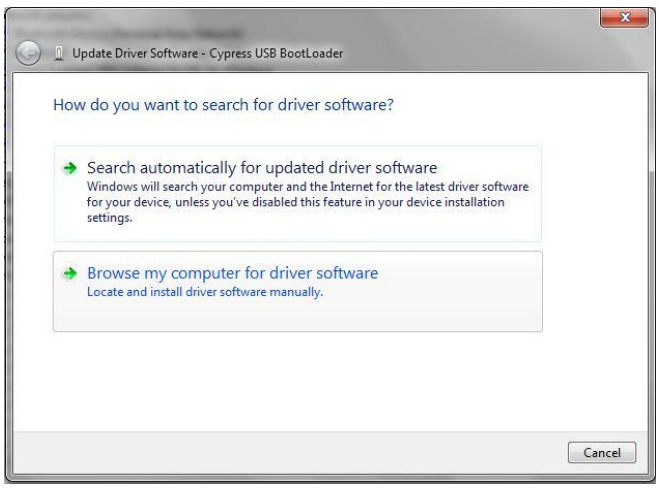
If you have not already installed the FX2LP/ FX3 DVK kit on a Windows computer, the first time you connect the DVK to the computer you will see the following message:



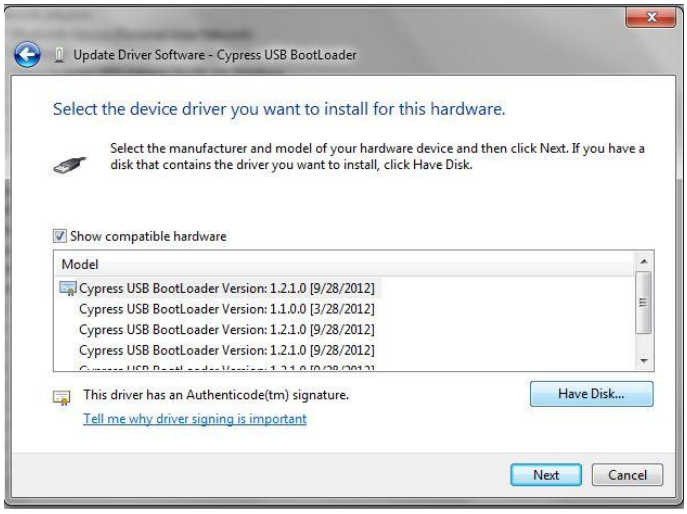
Close the message box and navigate to Windows Device Manager. To do this, click the Windows **Start** button, right-click **Computer** in the right-hand column, and select **Properties** to bring up System Information. Then, click **Device Manager** at the top of the left column.



Right-click the device with the yellow exclamation and select **update driver.**



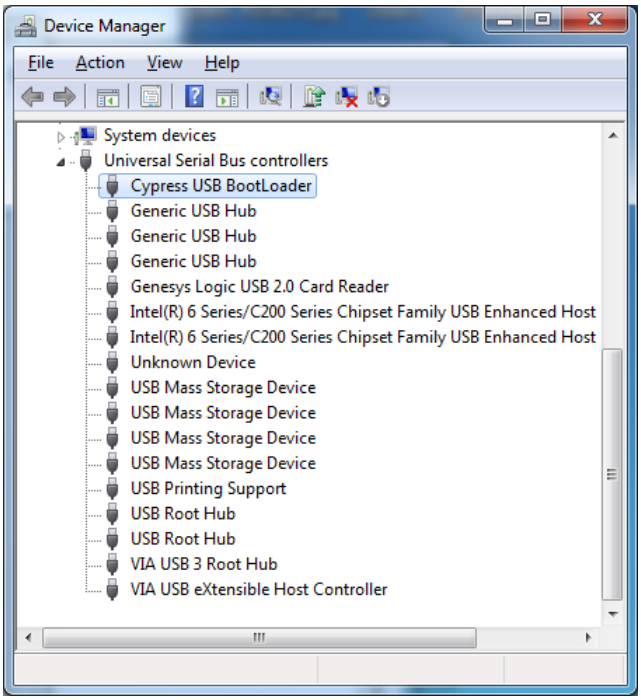
Select **Browse my computer for driver software**.



Click the **Have Disk** button and browse to the location *AN70983_Project\Drivers* to choose the CyUSB.inf and click **OK**. Use inf from the folder (*AN70983_Project\Drivers*) as shown in the following table for the respective operating system:

Operating System	Folder Path
Windows XP 32-bit	wxp\x86
Windows7 32-bit	win7\x86
Windows7 64-bit	win7\x64
Windows Vista 32-bit	wlh\x86
Windows Vista 64-bit	wlh\x64

Click '**Next**' and continue to successfully install the driver. After you have successfully installed the driver, the Device Manager window should remove the entry with the yellow exclamation and show as recognized devices, successfully bound to the driver.



Document History

Document Title: AN70983 – Designing a Bulk Transfer Host Application for EZ-USB® FX2LP™/FX3™

Document Number: 001-70983

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3315759	SSJO/ GAYA	07/15/2011	New Application Note
*A	3521406	GAYA	02/09/2012	Converted code example to application note.
*B	4023136	GAYA	06/07/2013	Updated document. Added VB and VC++ examples using CyUSB.dll, that can communicate with FX2LP and FX3.

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
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 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

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

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