AN70983

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

Author: Gayathri Vasudevan

Associated Part Family: CY7C6801XA, CYUSB3014
Related Resources: For a complete list, click here.
To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/AN70983

Cypress provides software development tools for the host PC in the form of a Visual Studio .NET library. This library simplifies USB coding at the Windows level. This application note introduces the .NET class library and shows how to create a Windows example to send and retrieve data using a “bulkloop” firmware example running on an FX2LP or FX3 Development Kit (DVK).

Contents

1 Introduction

This application note demonstrates how to use the Cypress library for Microsoft .NET languages to implement host PC applications to communicate with Cypress’s FX2LP and FX3 devices. Using this library, a Visual C#, Visual Basic, or Visual C++ program can communicate with an FX2LP or FX3-based device at a high level of abstraction. For example, the USBDeviceList and CyUSBDevice classes (explained in the Source Code Overview section) expose devices and endpoints as simple class members. Communicating with USB devices using this high-level model is a great improvement over multiple low-level Win32 API calls. For example, you can program a complete USB device tree display with only 15 lines of C# code.

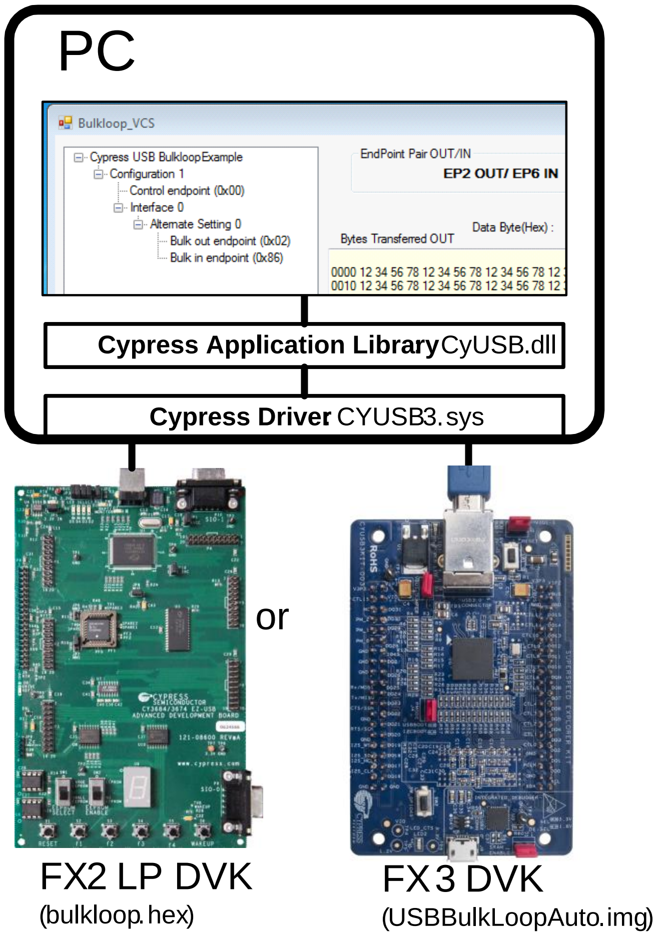
A Visual Studio solution along with this application note provides example projects in Visual C#, Visual Basic, and Visual C++ languages. The host projects are built and tested using Microsoft Visual Studio Express 2012. The host application writes to the BULK IN and BULK OUT endpoints of FX2LP or FX3. The library classes communicate with the provided Cypress USB driver CyUsb3.sys to access these devices and endpoints.

To exercise the Windows Visual Studio code, an example firmware called “bulkloop” is provided for FX2LP and FX3 evaluation boards along with this application note. This firmware accepts host PC data and sends the same data back to the PC on command.

This application note gives an overview of the Windows and device sides of the USB connection. It explains the software layout for the example code associated with this application note and provides details to prepare the FX2LP and FX3 evaluation boards as test targets. The application note also demonstrates how to test drive the pre-compiled Windows code. This allows you to run the Windows application without Visual Studio.

2 Functional Overview

Figure 1. System Block Diagram



Note FX3 DVK shown in Figure 1 is only representative. CYUSB3KIT-001/CYUSB3KIT-003 can be used.

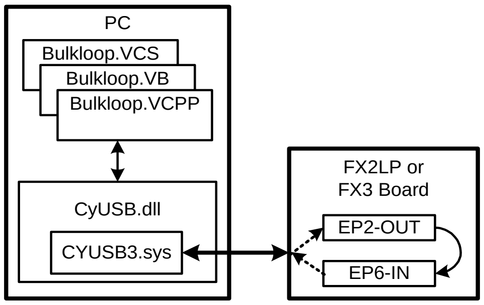
In Figure 1, the host PC connects to one of two Cypress target boards with a USB cable. The FX2LP or FX3 evaluation board is loaded with firmware that accepts BULK data on Endpoint 2-OUT and copies it for PC retrieval to the Endpoint 6-IN buffer. The firmware file name is shown at the bottom of each board in Figure 1.

Note USB direction is host-centric; therefore, OUT is PC to device and IN is device to PC.

The host application sends data to EP2-OUT by making library calls to CyUSB.dll, which in turn communicates with the Cypress USB driver CYUSB3.sys. When the host issues an IN request to EP6, the BULK data travels back in reverse through CyUsb3.sys and the CyUsb.dll library call, and finally reaches the host application for display. Bulk data sent and received is displayed in a PC text box, along with the number of transferred bytes. This data flow is depicted in Figure 2.

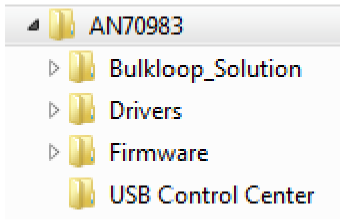
The Microsoft Visual Studio solution contains projects for Visual C# (shown in Figure 1 as Bulkloop_VCS), Visual Basic, and Visual C++. They produce identical results.

Figure 2. Bulkloop Data Flow



3 Application Note Folder Layout

Figure 3. Application Note Folder Layout



Cypress provides suites of software tools to develop firmware and .NET-based applications to talk to the FX2LP and FX3 devices. For convenience, the files used in this application note are supplied in the accompanying zip file. To download the full tool suites, visit the Cypress website.

Figure 3 shows the application note folder layout.

- **Bulkloop_Solution**: This is a Microsoft .NET solution, which implements the bulkloop tester in three languages - Visual Basic, Visual C#, and Visual C++.
- **Drivers**: When you plug in an FX2LP or FX3 development board for the first time, Windows asks for the location of a suitable driver. This folder contains drivers for Windows XP ("wxp"), Windows Vista ("wlh"), and Windows 7 ("win7"). Vista and Windows 7 drivers are provided for 32-bit systems ("x86") and 64-bit systems ("x64"). The driver is called cyusb3.sys and works for both FX3 and FX2LP.
  Note ezusb.sys and cyusb.sys are the legacy drivers distributed by Cypress for FX2LP family of devices. We do not recommend ezusb.sys or cyusb.sys for new designs. For more details on migrating to cyusb3.sys, refer to Migration of an EZ-USB® FX2LP™ Driver from CyUSB.sys to CyUSB3.sys.
- **Firmware**: This folder contains BULK loopback firmware for the FX2LP and FX3 boards. The source code is included so you can study, modify, and compile the code after downloading the full development suites.
- **USB Control Center**: This Visual C# application, taken from the EZ-USB FX3 Software Development Kit (SDK), downloads FX2LP and FX3 firmware into the respective development boards. Preparing the USB Target

3.1 Development Boards

The FX2LP development board is included in the FX2LP Development Kit. The FX3 board is included in the FX3 Development Kit (CYUSB3KIT-001 or CYUSB3KIT-003). Before running the Windows code, a development board must be prepared to execute the loopback function shown on the right in Figure 3. 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*).

Before loading the loopback firmware, verify that the jumpers and switches are set according to Table 1 for the FX2LP board and Table 2/Table 3 for the FX3 board.

Table 1. FX2LP Evaluation Board Settings

Jumper/Switch	Purpose	Setting
JP1, JP10	3.3 V to EZ-USB chip	IN,IN
JP2	Vbus powered	IN
JP3	Activates 4 LEDS	IN(x4)
JP5	3.3 V 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)

Table 2. FX3 Evaluation Board (CYUSB3KIT-001) Settings

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

Table 3. FX3 Evaluation Board (CYUSB3KIT-003 Settings)

Jumper/Switch	Purpose	Setting
J3	Development board is powered from USB 3.0 VBUS	Short
J4	Select USB download	Short
J5	Deselect external SRAM	Open

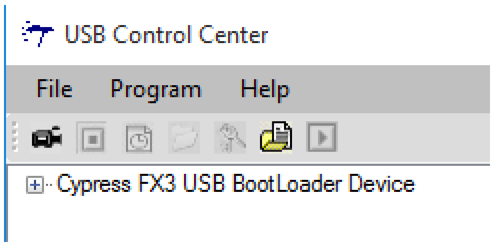
When the boards are configured as shown in the tables, FX2LP and FX3 use internal logic to enumerate as a USB loader device, ready to download user firmware over USB.

3.2 Download Firmware into Development Board Using USB Control Center

Double-clicking the *CyControl.exe* file in the USB Control Center folder brings up the USB Control Center console. Figure 4 shows how the FX2LP or FX3 boards appear in the left panel device list.

FX3 board appears as “Cypress USB Bootloader” (FX2LP board as “FX2LP Default Device: NO EEPROM”) if the *CyUSB3.inf* file provided in the attachment is used.

Figure 4. USB Control Center Device List



If you do not see anything in the left panel, the development board may be configured incorrectly or the Windows driver may not be installed. Refer to the Appendix for details.

Note Some versions of the USB Control Center may list all connected USB devices in the left panel. To simplify the display, click the **Device Class Selection** tab and deselect all but the **Devices served by the CyUSB.sys drive…** checkbox.

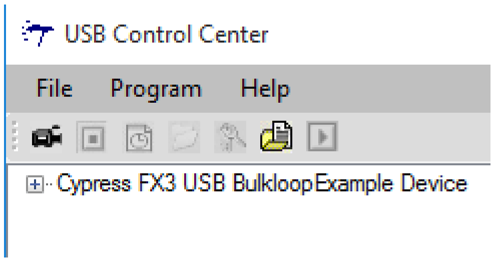
To download the bulkloop firmware into a development board, select the board you are using (Figure 4). Click the **Program** item, select either the FX2 or FX3 item, and select **RAM** to download the firmware into the on-chip program memory. Then navigate to the respective file as follows:

FX2: Firmware\Bulkloop_Fx2LP\bulkloop.hex

FX3: Firmware\Bulkloop_FX3\USBBulkLoopAuto\Debug\ USBBulkLoopAuto.img

Double-click the file to load the firmware. The USB loader device is replaced by the USB loopback device shown in Figure 5.

Figure 5. Bulkloop Device



This is EZ-USB ReNumeration in action. After the bulkloop firmware loads, the development board automatically disconnects from the USB and reconnects as the device implemented by the downloaded firmware. The FX2LP or FX3 development board is now ready to serve as the USB loopback peripheral for the PC example code in this application note.

4 Application Overview

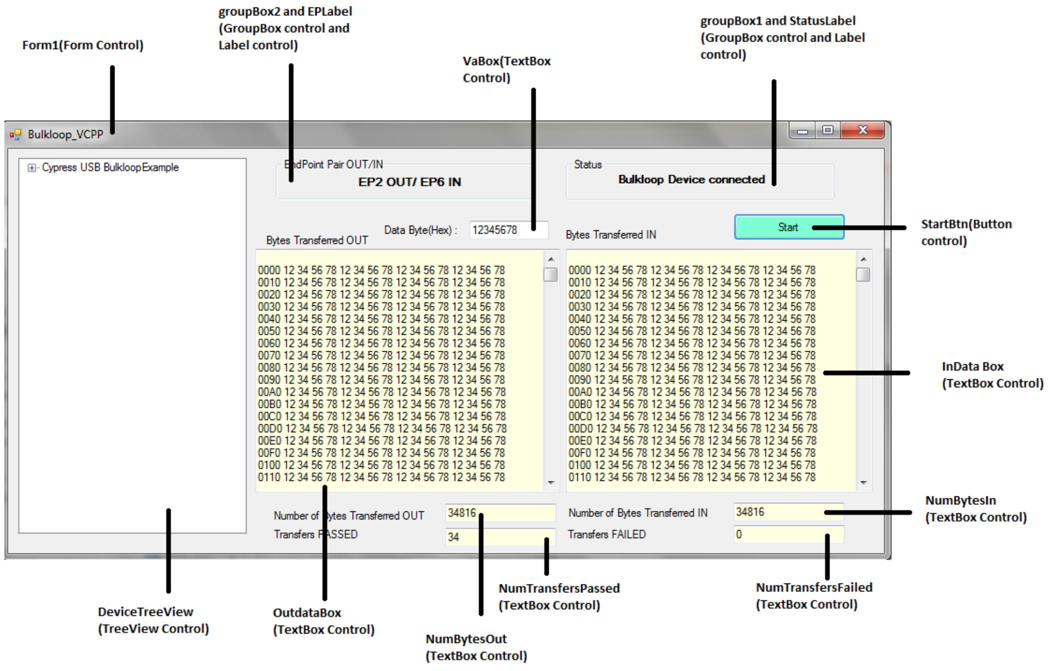
To test drive the application, double-click the executable file in the directory corresponding to your desired .NET language:

- VB_Application\Application\Bulkloop_VB.exe
- VC++_Application\Application\Bulkloop_VCPP.exe
- VCS_Application\Application\Bulkloop_VCS.exe

4.1 Features

Figure 6 shows the user interface of the Bulkloop application.

Figure 6. User Interface


Form1(Form Control)
groupBox2 and EPLabel (GroupBox control and Label control)
VaBox(TextBox Control)
groupBox1 and StatusLabel (GroupBox control and Label control)
StartBtn(Button control)
InData Box (TextBox Control)
NumBytesIn (TextBox Control)
NumTransfersFailed (TextBox Control)
NumTransfersPassed (TextBox Control)
NumBytesOut (TextBox Control)
OutdataBox (TextBox Control)
DeviceTreeView (TreeView Control)
Bulkloop_VCPP
Cypress USB BulkloopExample
Start

EndPoint Pair OUT/IN
EP2 OUT/ EP6 IN

Status
Bulkloop Device connected

Data Byte(Hex) : 12345678

Bytes Transferred OUT

```
0000 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0010 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0020 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0030 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0040 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0050 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0060 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0070 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0080 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0090 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00A0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00B0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00C0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00D0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00E0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00F0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0100 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0110 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
```

Bytes Transferred IN

```
0000 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0010 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0020 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0030 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0040 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0050 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0060 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0070 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0080 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0090 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00A0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00B0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00C0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00D0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00E0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00F0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0100 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0110 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
```

Number of Bytes Transferred OUT	34816	Number of Bytes Transferred IN	34816
Transfers PASSED	34	Transfers FAILED	0

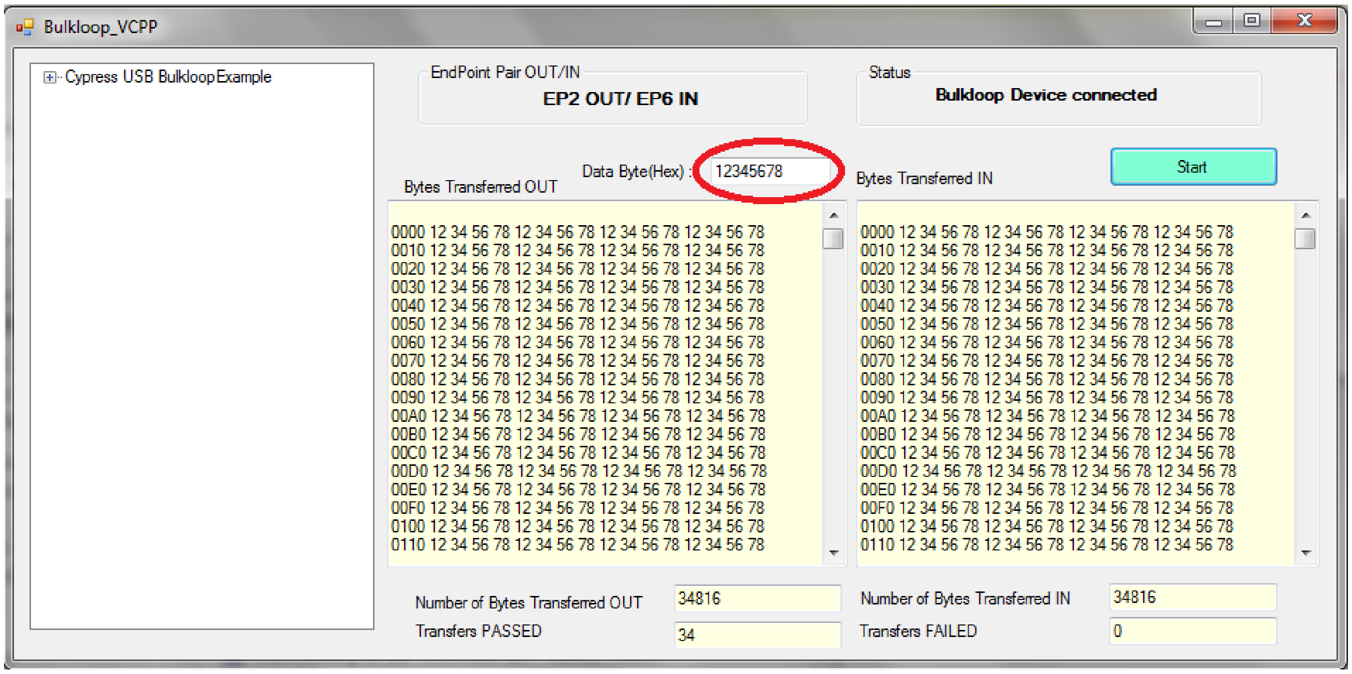
This application user interface is designed to show the following:

- **Status:** Reports whether the target 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 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 eight nibbles or four 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.

4.2 Operating Instructions

When a loopback device is detected, the Status box indicates “connected” and the left panel shows a tree indicating the internal organization of the USB device. You can enter a byte pattern to transfer or accept the default 0x12345678 byte pattern. Press the **Start** button; the loopback device sends and receives packets continuously.

Figure 7. Data Transfer from Bulkloop Application



EP2 OUT/ EP6 IN

Bulkloop Device connected

Data Byte(Hex) : 12345678

Bytes Transferred OUT

```
0000 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0010 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0020 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0030 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0040 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0050 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0060 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0070 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0080 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0090 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00A0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00B0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00C0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00D0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00E0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00F0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0100 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0110 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
```

Bytes Transferred IN

```
0000 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0010 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0020 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0030 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0040 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0050 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0060 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0070 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0080 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0090 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00A0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00B0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00C0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00D0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00E0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
00F0 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0100 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
0110 12 34 56 78 12 34 56 78 12 34 56 78 12 34 56 78
```

Number of Bytes Transferred OUT: 34816
Transfers PASSED: 34
Number of Bytes Transferred IN: 34816
Transfers FAILED: 0

Press **Stop** to terminate the transfers. The byte counters can be used to measure USB transfer bandwidth for this application using various host controllers and PCs.

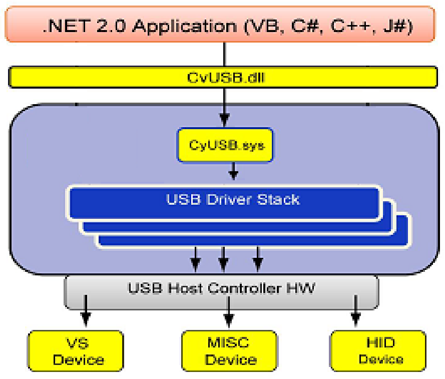
The remainder of this document describes the .NET language applications in detail.

5 Using the Cypress .NET Library

5.1 Adding a Reference to CyUSB.dll

CyUSB.dll is a managed Microsoft .NET class library that provides a high-level programming interface to USB devices. 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 8. Application Development with CyUSB.dll



To use the Cypress library, perform these two steps:

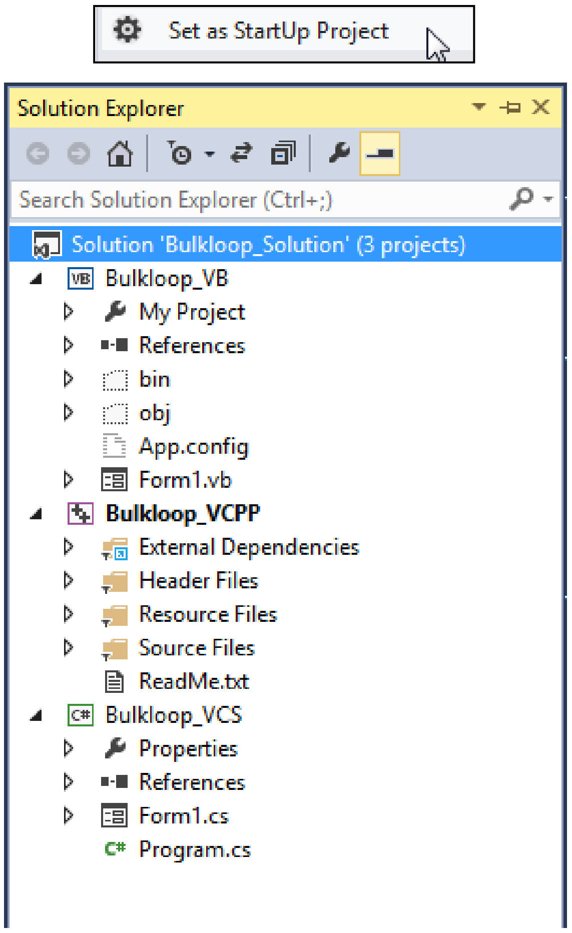
1. Add a library reference to *CyUSB.dll* in your project.
2. Add a line in your source file to include the library in your program.

Figure 9 shows where the library references are listed in the Visual Basic, VC++, and VC# projects in the Bulkloop_Exerciser solution. If expanding the reference item does not show CyUSB, right-click the project name, select the “reference” item, and browse to the folder containing *CyUSB.dll*. The projects supplied with this application note use the path C:\AN70983\CyUSB.dll. If you install the application note files elsewhere, these references need to be updated to match your file location.

All the projects in the solution compile when you click the **Build** button; any error messages apply to all three projects. You can limit error messages to the project you are using by excluding the other two projects from the ‘build’ operation. In the Solution Explorer panel, right-click the solution name, select **Properties**, and uncheck the projects you wish to exclude from the build operation.

If you have multiple projects in the ‘build’ operation, select the language version by right-clicking the project name and set it as the startup project. This is a quick way to switch between languages.

Figure 9. Add CyUSB.dll Reference to All Three Projects



Depending on the language, add a reference to the code at the top of the file according to Table 4.

Table 4. Reference Syntax for .NET Languages

Language	File	Added line
Visual Basic	form1.vb	Imports CyUSB
Visual C#	form1.cs	Using CyUSB
Visual C++	form1.h	#using "..\..\..\Application\CyUSB.dll" (or the respective path in your computer)

5.2 Source Code Overview

The following table explains the functionalities and important steps of the host application code.

Table 5. Code Overview

Action/Function	Description	CyUSB.dll API
Object initialization	■ Creates and initializes 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. It populates itself with USBDevice objects representing all 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 are initialized correctly 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 connection between the host and the target hardware 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. ■ Enables or disables the status label and the Start/Stop button. ■ If the device of interest is found, instantiates the “BulkOutEndpt” and “BulkInEndpt” with endpoints EP2 and EP6, respectively, because the bulkloop firmware in this example uses the same endpoints. ■ Clears and updates the tree view 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 the 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 communication is actually carried out. ■ The other parent thread only 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 set up the data for OUT transfers. ■ After the thread is started, data transfers in the OUT 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 (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.	—

5.3 Asynchronous Communication

This application uses synchronous communication to move 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 is blocked until the full data transfer completes. After that, the next data transfer operation starts. This method is suitable for most applications. But with applications that need to use the full data bandwidth available for transfers, the more advanced asynchronous communication method is used. Asynchronous transfers are more complex, suitable for advanced users. The 'Streamer' example that comes with the Cypress USBSuite shows how to use the asynchronous communication method.

6 Firmware

The firmware configures the endpoints and the interface to carry out bulk data loopback operation. Two endpoints are configured to handle bulk transfer: the OUT endpoint and 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.

6.1 FX2LP Bulkloop Firmware

The following code snippets explain important code functionalities of the FX2LP bulkloop firmware.

Action	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 maxpacket size	—	—	dscr.a51
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. ■ 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

	■ Checks Endpoint 2 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), checks the capability of Endpoint 6 to receive the data. 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))`
		`  {`
		`  }`
		`}`		

	■ To transfer 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 EP2BCL and EP2BCH registers 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;`		

	■ Because auto pointers are 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 repeatedly to transfer each byte from Endpoint 2 to 6.	`for( i = 0x0000; i < count; i++ )`
		`{`
		`  EXTAUTODAT2 =`
		`  EXTAUTODAT1;`
		`}`		

	■ After the data is transferred, Endpoint 2 must be re-armed to accept a new packet from the host. Endpoint 6 is “committed”, that is, the FIFO buffers are made available to the host for reading data from Endpoint 6	`EP6BCH = EP2BCH;`
		`SYNCDELAY;`
		`EP6BCL = EP2BCL;`
		`SYNCDELAY;`
		`EP2BCL = 0x80;`
				—

6.2 FX3 Bulkloop Firmware

The following code snippets explain 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 I/O 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()` `CyU3PDeviceConfigureIOMatrix()` `CyU3PKernelEntry()`	main()	cyfxbulklpauto.c
	■ FX3 application definition function 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()	
	■ Starts the USB device mode driver in the USB 3.0 platform and creates the DMA channels required for the control endpoint ■ Registers 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. ■ Registers a USB 3.0 LPM request handler callback. ■ Registers 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 eight different string descriptors. The driver only stores the descriptor pointers that are passed into this function; it does not make copies of the descriptors. Therefore, the	■ `CyU3PUsbStart()` ■ `CyU3PUsbRegisterSetupCallback()` ■ `CyU3PUsbRegisterEventCallback()` ■ `CyU3PUsbRegisterLPMRequestCallback()` ■ `CyU3PUsbSetDesc()` ■ `CyU3PConnectState()`	CyFxBulkLpApplnInit()	

	Description	FX3 APIs	Function Name	File name
	caller 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.			
	■ CyFxBulkLpApplnStart () is called upon Set Configuration event, which in turn configures USB endpoint's properties.	■ `CyU3PSetEpConfig()`	CyFxBulkLpApplnStart ()	
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.	■ `CCyU3PDmaChannelCreate()` ■ `CyU3PDmaChannelSetXfer()`	CyFxBulkLpApplnStart ()	

7 Troubleshooting

1. The host application will fail to compile and the CyUSB.dll APIs will not be accessible if the CyUSB.dll is not properly added and included for reference. Follow the instructions provided in Adding a Reference to CyUSB.dll.
2. If any of the CyUSB APIs fail, check the return code for the respective API. The return code values and their meaning is documented in the CyUSB.dll documentation located at "<InstallDirectory>\Cypress Suite USB 3.4.x\CyUSB.NET\CyUSB.NET" after installing SuiteUSB.
3. If the XferData/BeginDataXfer APIs fail, use the "UsbdStatus" or "UsbdStatusString()" to get the error code from the last call to the XferData or BeginDataXfer APIs
4. If APIs are unable to access the device, ensure the device handle is properly assigned.
5. If the project does not open in Visual Studio 2010 (VS 2010), create a new project in VS 2010, move the source files, and add the *CyUSB.dll* library to the project. All existing features will work for VS 2010 and above versions.

8 System Requirements

8.1 Hardware

8.1.1 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.

8.1.2 FX3

An FX3 DVK (CYUSB3KIT-001 or CYUSB3KIT-003) board can be 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 appropriate FX3 DVK).

8.2 Development Software

- Microsoft .NET framework 4.5
- 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 or 2012.

9 Related Resources

- UsbSuiteQuickStartGuide 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.
- More application examples are provided with the Cypress USB Suite and FX3 SDK 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. This application note is provided only for reference. Cypress also provides Linux SDK which can be found at FX3 SDK web page.
- 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. This application note is provided only for reference. Cypress also provides MAC SDK which can be found at FX3 SDK web page.
- For complete list of USB SuperSpeed Code Examples, visit http://www.cypress.com/101781 and for complete list of USB Hi-Speed Code Examples, visit http://www.cypress.com/101782.

10 Summary

This application note discusses the various aspects to build host PC applications using the Cypress library for Microsoft .NET languages to implement to communicate with Cypress's FX2LP and FX3 devices.

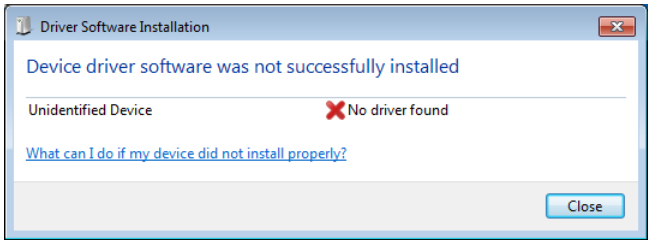
11 Glossary

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 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 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 IDE from Microsoft for its COM programming model
CYy3684 DVK	Cypress development kit for the EZ-USB FX2LP family. For more information, see the Cypress website.
CYUSB3KIT – 001/ CYUSB3KIT – 003 (EZ-USB® FX3™ SuperSpeed Explorer Kit)	Cypress development kit for the EZ-USB FX3. For more information, see CYUSB3KIT-001 or CYUSB3KIT-003.
EZ-USB	Generic name for all Cypress high-speed USB products such as FX2LP.
FX2LP	Cypress high-speed USB controllers. For more information, see the CYUSB3KIT-001 or CYUSB3KIT-003.
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 website.
Cypress USB Suite	Cypress USB development tools for Visual Studio. For more information, see the Cypress website.
CyUSB.dll	Cypress 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 Windows driver for USB communication. For more information, see the Cypress CyUsb3.sys Programmers Reference.
USB Control Center	Cypress application for general-purpose USB functionality such as device identification and code 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 comprises the Eclipse base platform and the CPP feature. This firmware development environment helps you to develop, build, and debug firmware applications for FX3.

A Appendix: Windows Install of FX2LP/FX3 DVK Driver

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

Figure 10. Driver Software Installation



Driver Software Installation

Device driver software was not successfully installed

Unidentified Device ✗ No driver found

What can I do if my device did not install properly?

Close the message box and navigate to the Windows Device Manager. To do this, click the **Start** button, right-click **Computer** in the right panel, and select **Properties** to bring up System Information. Click **Device Manager** in the left panel. In the Device Manager, FX2LP shows up as “Unknown Device” and FX3 as “WestBridge”.

Figure 11. FX2LP as Unknown Device

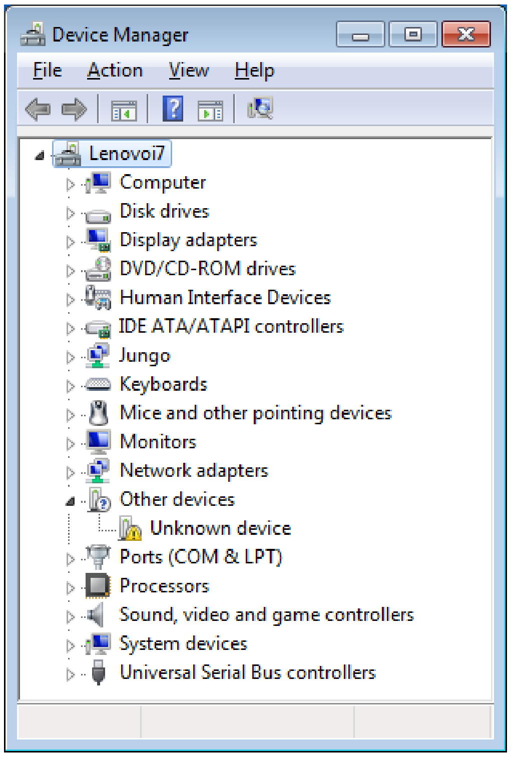
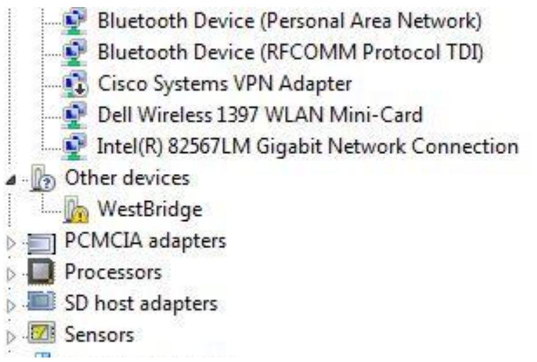


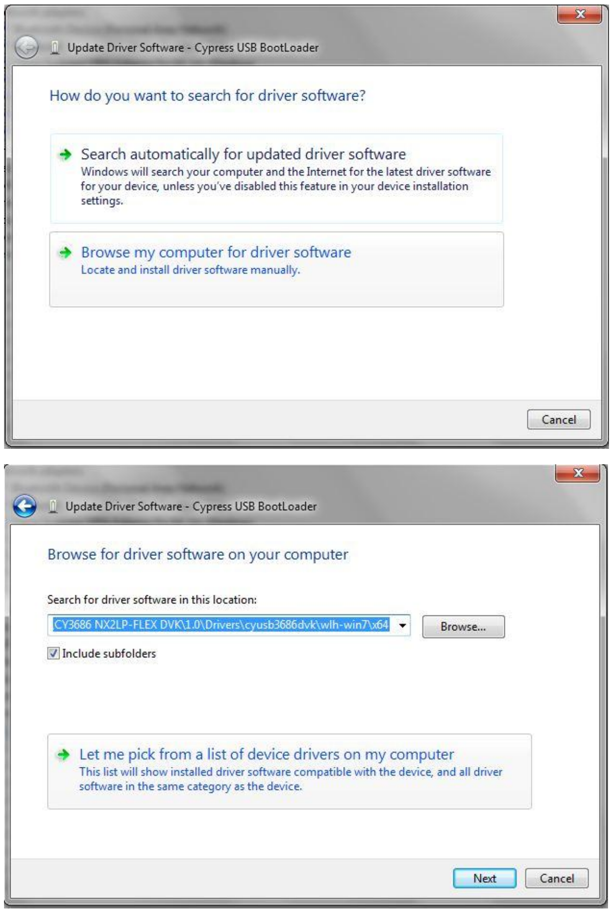
Figure 12. FX3 as WestBridge


Bluetooth Device (Personal Area Network)
Bluetooth Device (RFCOMM Protocol TDI)
Cisco Systems VPN Adapter
Dell Wireless 1397 WLAN Mini-Card
Intel(R) 82567LM Gigabit Network Connection
Other devices
WestBridge
PCMCIA adapters
Processors
SD host adapters
Sensors

Right-click the device (with the exclamation mark) and select **Update Driver**.

Select **Browse my computer for driver software**. Browse to select the driver software folder and click **Next**.

Figure 13. Update Driver



Update Driver Software - Cypress USB BootLoader

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

Update Driver Software - Cypress USB BootLoader

Browse for driver software on your computer

Search for driver software in this location:

CY3686 NX2LP-FLEX DVK\1.0\Drivers\cyusb3686dvk\wlh-win7\x64

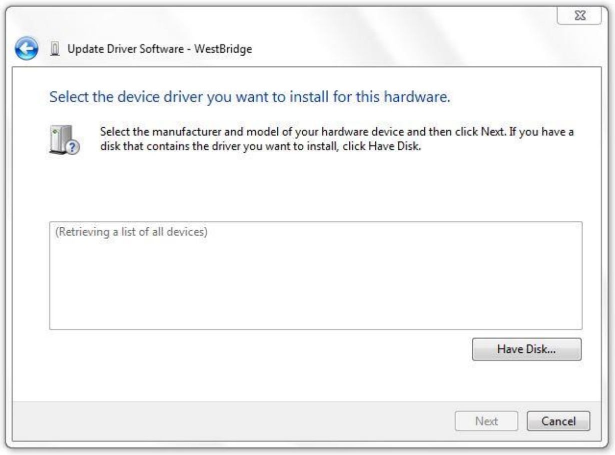
[✓] Include subfolders

Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Click the **Have Disk** button, browse to the location *AN70983_Project\Drivers* to choose *CyUSB3.inf*, and click **OK**. Use the 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
Windows 7 32-bit	win7\x86
Windows 7 64-bit	win7\x64
Windows Vista 32-bit	wlh\x86
Windows Vista 64-bit	wlh\x64

Figure 14. Click Have Disk to Select CyUSB3.inf

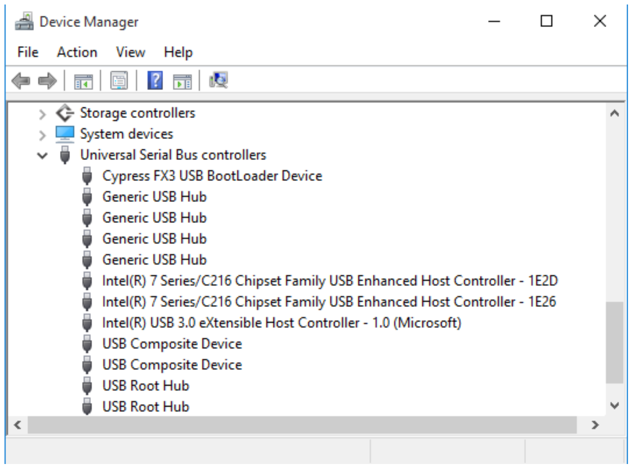

Update Driver Software - WestBridge
Select the device driver you want to install for this hardware.
Have Disk...
Next
Cancel

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

(Retrieving a list of all devices)

Click **Next** and continue to install the driver. After installing the driver, the Device Manager window should remove the entry with the yellow exclamation mark and show the recognized devices, successfully bound to the driver.

Figure 15. Cypress Device after Binding Driver



Storage controllers
System devices
Universal Serial Bus controllers
Cypress FX3 USB BootLoader Device
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
Intel(R) 7 Series/C216 Chipset Family USB Enhanced Host Controller - 1E2D
Intel(R) 7 Series/C216 Chipset Family USB Enhanced Host Controller - 1E26
Intel(R) USB 3.0 eXtensible Host Controller - 1.0 (Microsoft)
USB Composite Device
USB Composite Device
USB Root Hub
USB Root Hub

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.
*C	4218438	RSKV	12/12/2013	Restructured the Microsoft .NET code into a single solution containing three projects, one for each language. Major rewrite of the document.
*D	4690485	GAYA	09/16/2015	Updated the document as per new template. Updated Abstract. Added hyperlink to refer the complete list of USB SuperSpeed Code Examples and complete list of USB Hi-Speed Code Examples, in Abstract (page 1). Added a note for FX3 DVK in Functional Overview. Updated Figure 1. System Block Diagram. Added a note in Application Note Folder Layout. Updated FX3. Updated Table 2 title. Added Table 3 Updated the Term “CYUSB3KIT-001” to “CYUSB3KIT-001/CYUSB3KIT-003 (EZ-USB® FX3™ SuperSpeed Explorer Kit) and its Meaning in Glossary. Updated Related Resources.
*E	5601614	GAYA	01/25/2017	Updated figures 4, 5, 9, 5 Updated template
*F	5701881	BENV	04/19/2017	Updated logo and copyright

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

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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