AN58009

Serial (UART) Port Debugging of EZ-USB® FX1/FX2LP™ Firmware

Author: Praveen Kumar
Associated Project: Yes
Associated Part Family: EZ-USB® FX1, FX2LP™
Software Version: Keil µVision2
Related Resources: Click here

More code examples? We heard you.
For a consolidated list of USB Hi-Speed Code Examples, visit the Cypress webpage.

AN58009 explains the code implementation required for debugging EZ-USB® FX1/FX2LP™ firmware through a serial port (UART). This code enables you to print debug messages over the UART using any serial communication application.

Contents

1 Introduction....1
2 Hardware Requirements....1
3 Software Requirements....2
4 Serial Debug Files....2
4.1 FX2LPSerial.h....2
4.2 FX2LPSerial.c....2
5 Demonstration....3
5.1 Adding Files to the Target Code....3
5.2 Firmware Modifications....4
5.3 HyperTerminal Settings....5
5.4 Testing....7
6 Summary....9
A Cypress Design Resources....10
B Application Notes and Reference Designs....12
B.1 Application Notes....12
B.2 Reference Designs....13
Document History....14
Worldwide Sales and Design Support....15

1 Introduction

When developing firmware projects, designers must be able to debug code. The Keil tools supplied with the EZ-USB FX2LP family of development kits let you debug code using single-step, start, stop, and other features. For more details, see *GS51.pdf,* available at *C:\Keil\C51\HLP* (location varies based on installation) after you install Keil µVision2. However, this ability may not be useful when any of the code to be debugged is real time, as the delay caused by stopping or single-stepping the firmware execution causes failure. A typical example would be debugging within a USB ISR, which must finish execution within a specific time period as per USB specifications. Single-step debugging may violate the timing constraint.

Printing debug messages through the UART in the HyperTerminal program on the computer or capturing them in a file enables easy debugging and tracking of the firmware code flow. This application note discusses the code that you must add to any FX2LP firmware that requires this debugging feature.

Note: This application note uses the name “FX2LP” to refer to both FX1 and FX2LP unless noted otherwise.

2 Hardware Requirements

To perform UART debugging, you must have a serial cable, an FX2LP (or FX1) DVK, and a Windows computer with a serial port. Computers that do not have a serial port can use a USB-to-UART cable, which provides a virtual COM (serial) port on the computer. Most USB-to-UART cables come with a customized driver. Be sure to install those drivers and verify whether a COM port is available in the Device Manager.

There are two serial ports on the FX2LP development board: SIO-0 and SIO-1. Connect the serial cable to SIO-0 on the FX2LP DVK. (SIO-0 is configured in this example firmware.)

3 Software Requirements

The computer must have the HyperTerminal application installed to capture or print the debug messages.

HyperTerminal (Serial Communication software) is not present in Windows 7 and higher PC, by default. You need to install it manually for Windows 7 and higher PC. To download and install HyperTerminal, click here. Alternatively, any other serial communication software like TeraTerm can also be installed. TeraTerm is attached with this application note for installation.

You need the USB Control Center to download the firmware into FX2LP. It is available in EZ-USB FX3 SDK.

4 Serial Debug Files

This section describes the files that are used for debugging. They are provided with this application note in the folder FX2LPSerialDebug.

4.1 FX2LPSerial.h

The *FX2LPSerial.h* header file contains the function definitions for all the functions in *FX2LPSerial.c*. This header file must be included in the project to access the functions in *FX2LPSerial.c*.

4.2 FX2LPSerial.c

FX2LPSerial.c includes all the required functions for serial debugging.

4.2.1 FX2LPSerial_Init ()

This function initializes Timer2 as a baud rate generator and sets the values of the register RCAP2H/L for a baud rate of 38,400. Because the RCAP2H/L value has been calculated with the assumption that the CPU clock is running at 48 MHz, this function sets the clock frequency to 48 MHz by writing to the CPUCS register. Refer to chapter 14 of the EZ-USB Technical Reference Manual for more information on the configuration values based on the serial port used, timer used, and CPU clock frequency. Following is the code snippet of the function:

```
T2CON = 0x34 ;
RCAP2H  = 0xFF ;
RCAP2L = 0xD9;
SCON0 = 0x5A ;
TI = 1;
CPUCS = ((CPUCS & ~bmCLKSPD)|bmCLKSPD1);       //Setting up the clock frequency
FX2LPSerial_XmitString("\r\n->") ;             //Clearing the output screen
```

4.2.2 FX2LPSerial_XmitString (char *str)

This function calls the intermediate function FX2LPSerial_XmitChar(char ch) in a while loop to print the characters of the input string. FX2LPSerial_Xmitchar() prints one character at a time. Following is the code snippet of the FX2LPSerial_XmitChar() function:

```
while (TI == 0) ;
TI = 0 ;
SBUF0 = ch ; //print the character
```

Following is the code snippet of the FX2LPSerial_XmitString() function:

```
while (*str)
    FX2LPSerial_XmitChar(*str++) ;
```

4.2.3 FX2LP_XmitHex2 (BYTE Variable)

This function calls the intermediate function FX2LPSerial_XmitHex1(BYTE b) with the upper nibble as input and then the lower nibble as input to print the hex value of a variable of BYTE type. Following is the code snippet of the FX2LP_XmitHex1() function:

```
if (b < 10)
    FX2LPSerial_XmitChar(b + '0') ;
else
    FX2LPSerial_XmitChar(b-10+'A');
```

Following is the code snippet of the FX2LP_XmitHex2() function:

```
FX2LPSerial_XmitHex1((b >> 4) & 0x0f) ;
FX2LPSerial_XmitHex1(b & 0x0f) ;
```

4.2.4 FX2LP_XmitHex4 (WORD Variable)

This function calls the FX2LPSerial_XmitHex1(BYTE b) function with the upper BYTE as input and then the lower BYTE as input to print the hex value of a variable of WORD type. Following is the code snippet of the FX2LP_XmitHex4() function:

```
FX2LPSerial_XmitHex2((w >> 8) & 0xff) ;
FX2LPSerial_XmitHex2(w & 0xff) ;
```

5 Demonstration

This section uses the bulkloop example firmware available at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.0\Firmware\Bulkloop* after installing the CY3684 EZ-USB FX2LP Development Kit (or CY3674 EZ-USB FX1 Development Kit for FX1) to demonstrate the debugging procedure discussed in this application note.

Modified bulkloop example code with the addition of relevant serial debug files and code is attached with this application note.

5.1 Adding Files to the Target Code

1. Add the file *FX2LPSerial.c* to the target Keil project.
2. In the project window on the left side of the µVision IDE, select “Target 1.”
3. In the project menu, select “Options for Target ‘Target 1’.” The **Options for Target ‘Target 1’** window appears.
4. Click the **C51** tab. In the **Include Paths** text box, add the path of the *FX2LPSerial.h* header file (see Figure 1 and Figure 2).

Figure 1. Selecting Options for Target ‘Target 1’ in Project Menu

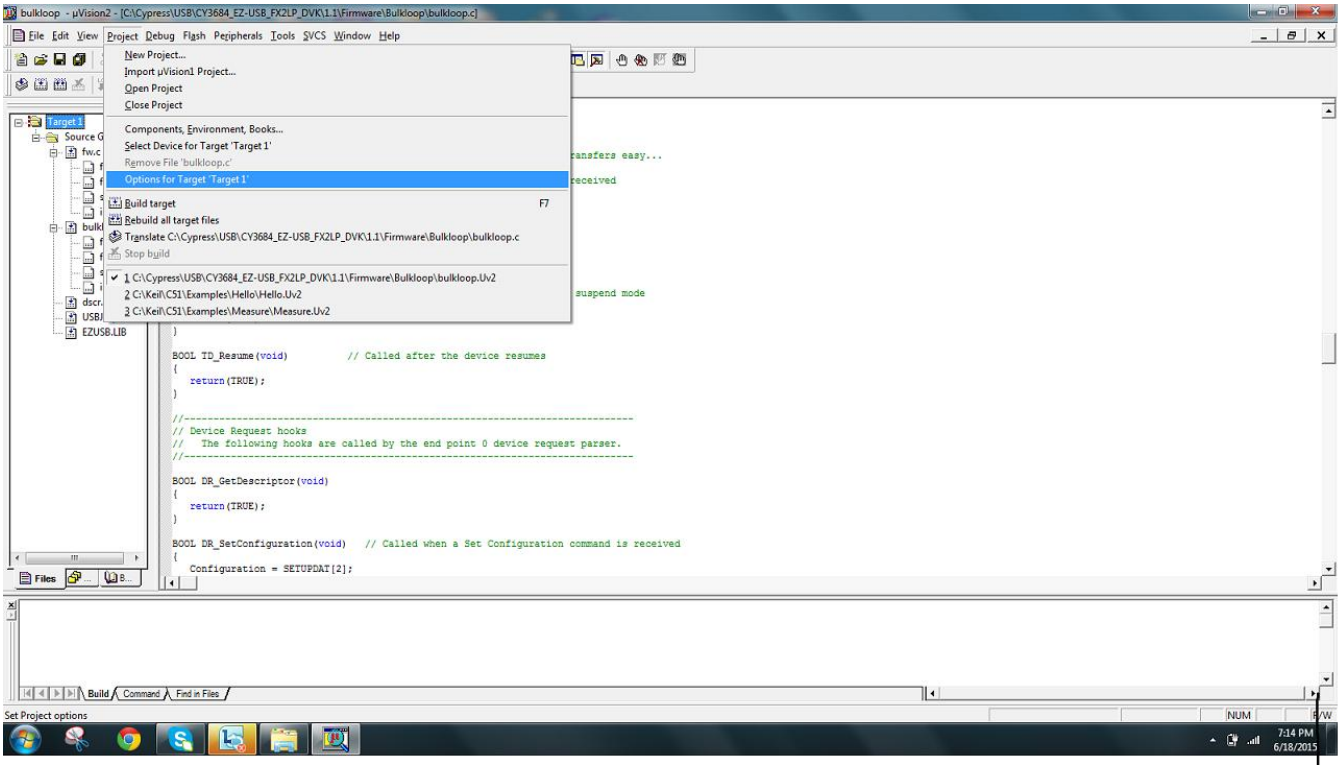
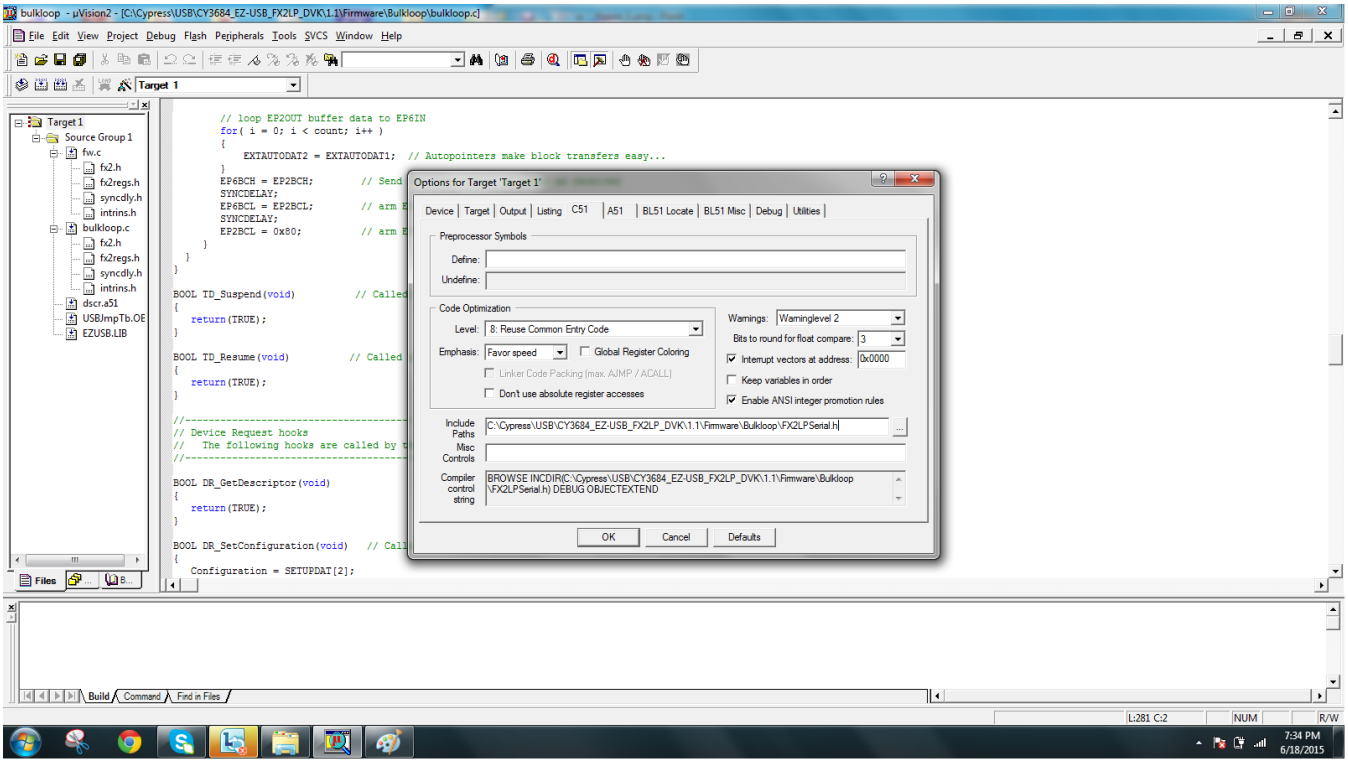


Figure 2. Options for Target 'Target1' Window



5. Compile once and make sure that the file *FX2LPSerial.c* and the header file location are included in the project.

5.2 Firmware Modifications

This section describes various serial functions, which are added in the above bulkloop.c example project, which is present at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.0\Firmware\Bulkloop* after installing the CY3684 EZ-USB FX2LP Development Kit (or CY3674 EZ-USB FX1 Development Kit for FX1) . The function FX2LPSerial_Init() is called in the TD_Init() function in *bulkloop.c*. This ensures that the serial port is initialized and ready for use at any point in the code execution. The other required functions are called wherever necessary as described in sections below.

Please note that the modified Bulkloop example is included along with this application note.

5.2.1 TD_Init()

The following code snippet initializes the serial port and prints a string to reflect the same. Then the value of IFCONFIG is printed.

```
//CPUCS=((CPUCS & ~bmCLKSPD)|bmCLKSPD1); //Commented since the CPU frequency is
configured in FX2LPSerial_Init()
FX2LPSerial_Init();// Serial Debug Code Start
FX2LPSerial_XmitString("Serial port initialized\r\n");
// set the slave FIFO interface to 48MHz
IFCONFIG |= 0x40;
FX2LPSerial_XmitHex2(IFCONFIG); //Printing the value of IFCONFIG
FX2LPSerial_XmitString("\r\n");
```

5.2.2 TD_Poll()

The following code snippet prints a string to indicate the transfer of a packet from EP2 to EP6 and the byte count of the packet.

```
FX2LPSerial_XmitChar('P');
FX2LPSerial_XmitString("acket of Byte Count = ");
FX2LPSerial_XmitHex4(count);
```

```
FX2LPSerial_XmitString(" being moved to EP6\r\n");
```

5.2.3 ISR_Ures()

The following code snippet prints a string from the USB Reset ISR to indicate that the bus reset ISR was triggered.

```
FX2LPSerial_XmitString("USB Reset ISR triggered\r\n");
```

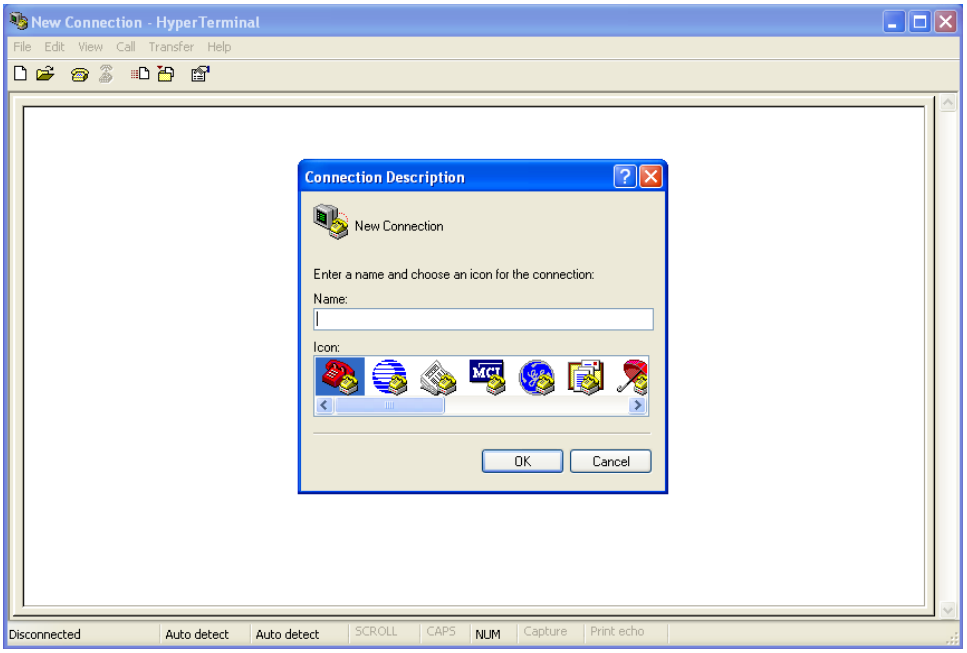
After the required debugging code is added to the source, compile the project. The target *.hex* file is generated for testing purposes.

Please note that if UART debug prints are added in USB ISRs, then it should be ensured that the UART prints will not result in the violation of the USB specification timings. To be compliant with USB specification timings, keep the messages as short as possible. Only flags can be set in ISR and UART messages can be printed in the main thread (TD_Pol()) if UART messages are long. In this way, ISRs will not be blocked for longer time.

5.3 HyperTerminal Settings

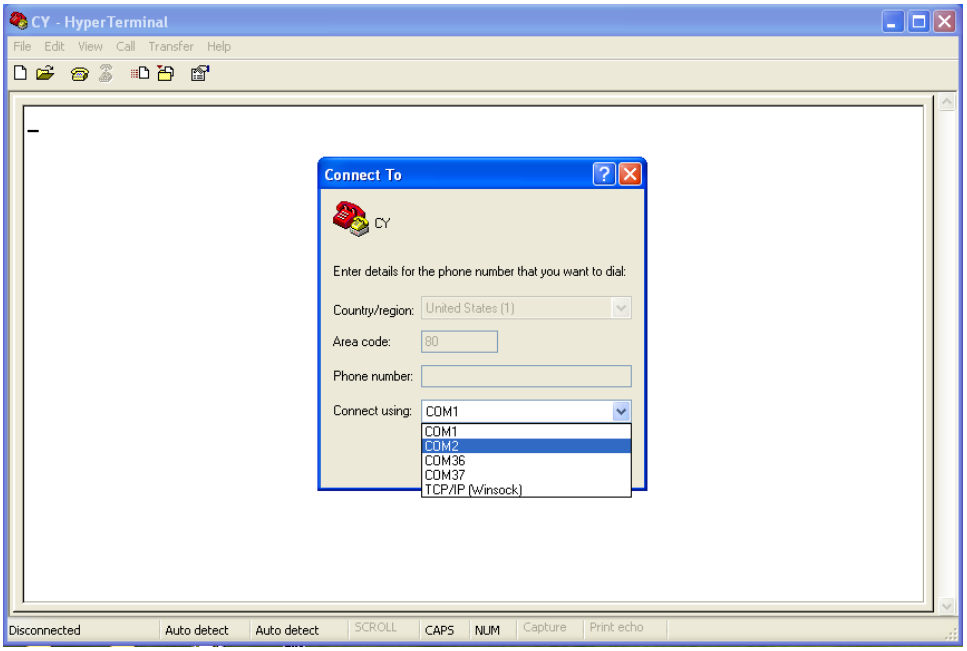
1. On the Windows computer, choose **Start** > **All Programs** > **Accessories** > **Communications** and then click **HyperTerminal**. As shown in Figure 3, a new connection opens and asks for the **Connection Description** in a separate dialog box.
2. Enter the name of the connection and click **OK**.

Figure 3. Entering the Connection Name



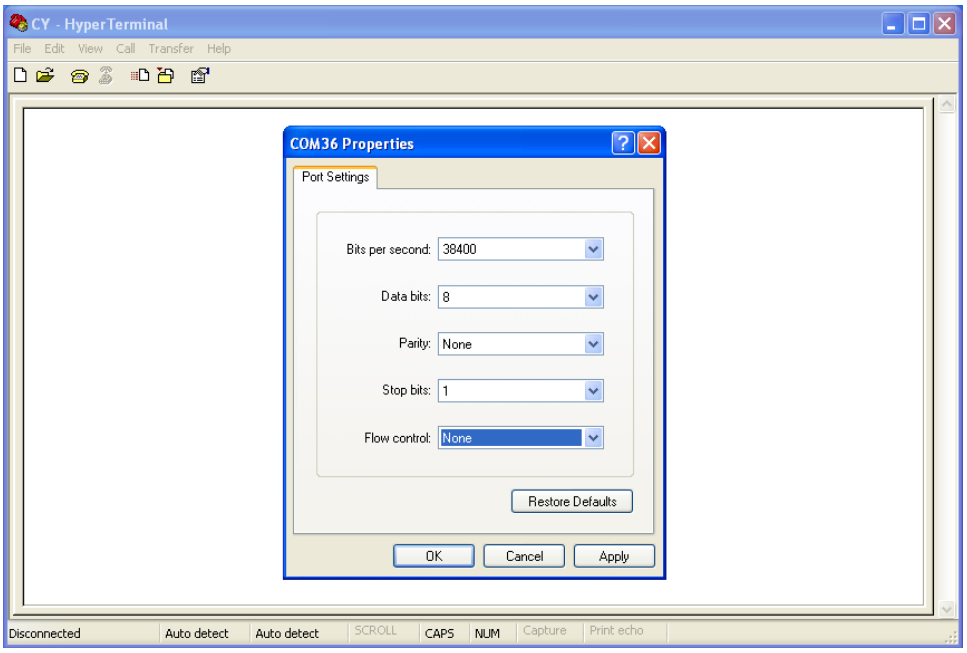
3. In the **Connect To** window, go to the **Connect using** drop-down list and select the appropriate COM port (the COM port to which the FX2LP DVK serial port is connected) and click **OK,** as shown in Figure 4.

Figure 4. Selecting the COM Port



4. In the **Port Settings** tab of the **COM Properties** window (Figure 5), change the **Bits per second** to “38400” and the **Flow control** to “None,” click **Apply,** and then click **OK**.

Figure 5. Select the Serial Port Configuration



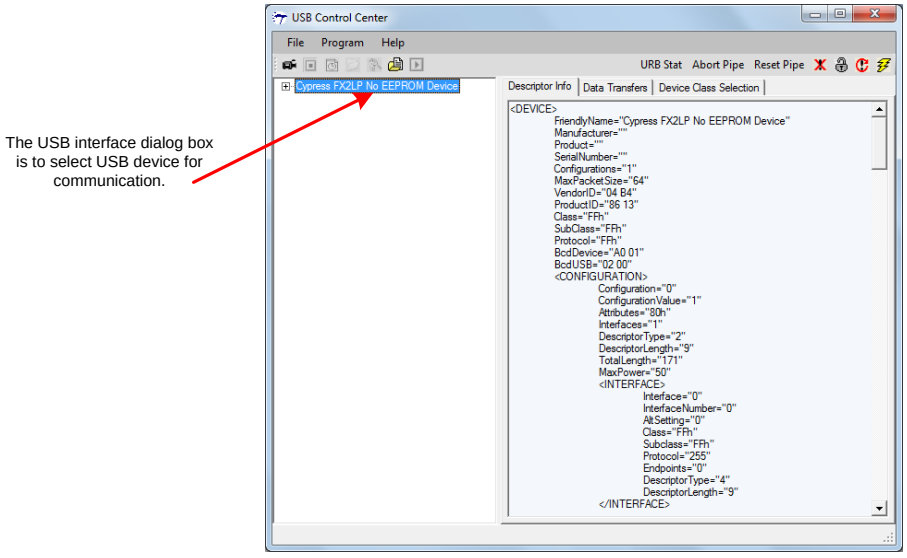
The HyperTerminal is now ready to print the debug messages received through the serial port.

5.4 Testing

The USB control center uses USB to download the compiled *.hex* file. Following are the steps to test the example firmware:

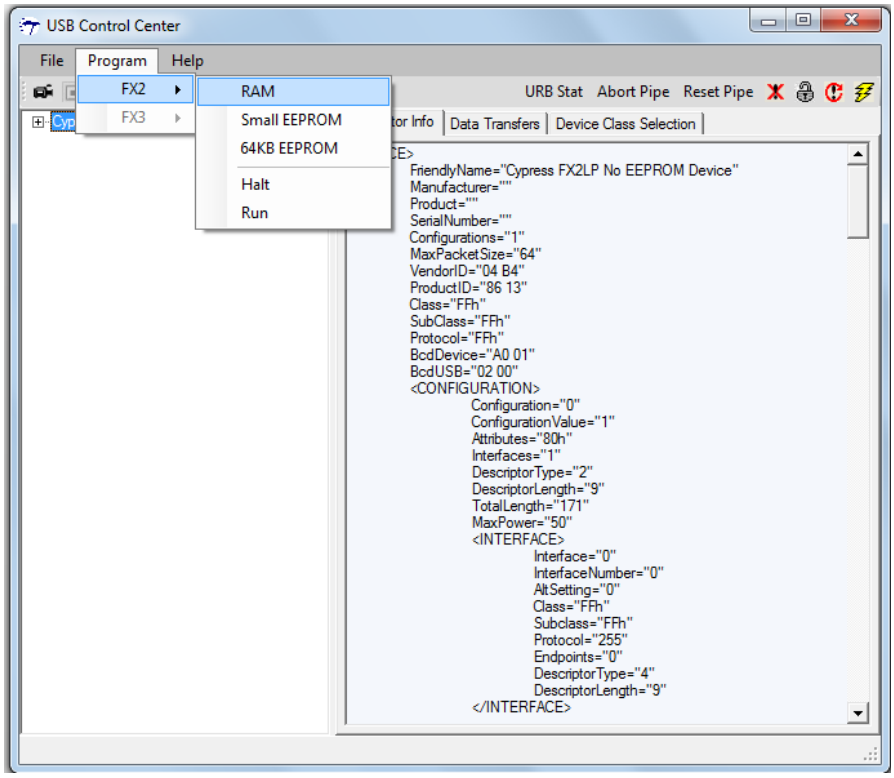
1. Plug the CY3684 EZ-USB FX2LP DVK board into the computer using a USB cable and bind it to the driver *CyUSB.sys*. Download and install EZ-USB FX3 SDK to launch USB Control Center.
2. Choose START >> All Programs >> Cypress >> EZ-USB FX3 SDK >> Cypress USBSuite >> Control Center
3. The application launches, and the window shown in Figure 6 appears.

Figure 6. USB Control Center Window



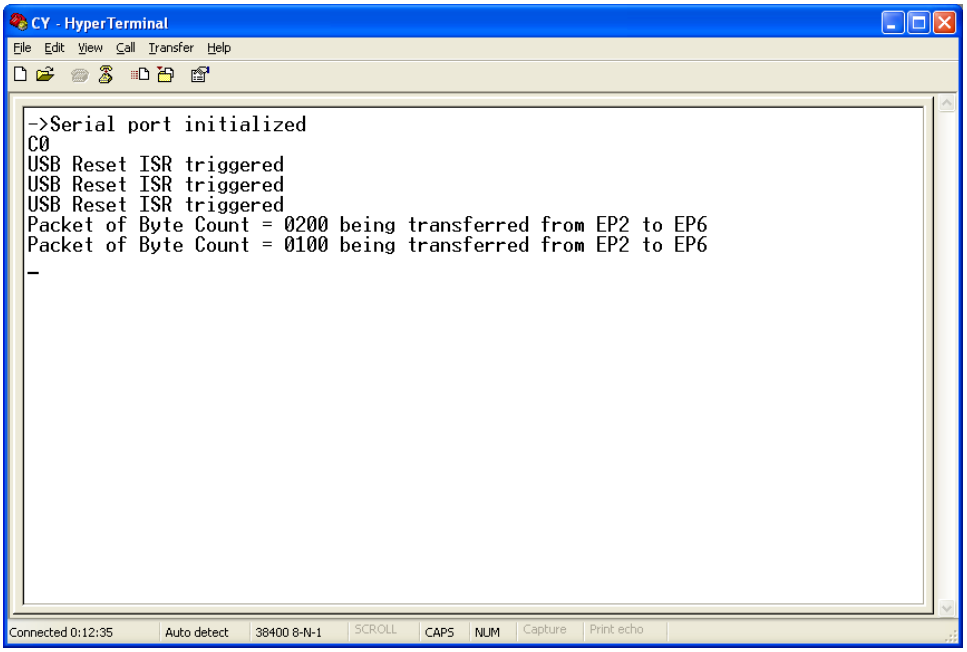
4. Click Program >> FX2 >> RAM, as shown in Figure 7 . In the pop-up window that appears, navigate to and select *bulkloop.hex*, which is available in the Bulkloop folder in the attachment that comes with this application note.

Figure 7. Download the Firmware



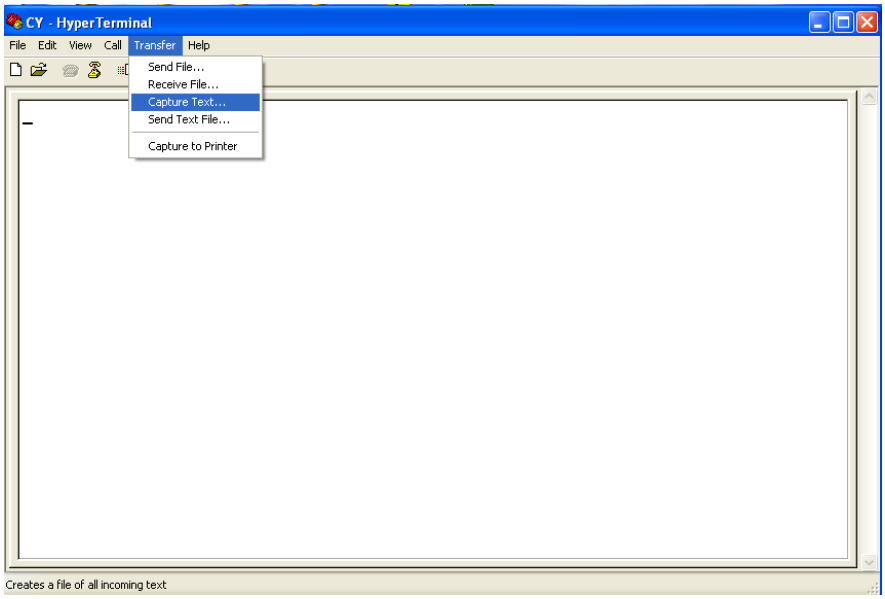
As the code is executed by FX2LP, debug messages are printed to the HyperTerminal, as shown in Figure 8.

Figure 8. Debug Messages Printed to HyperTerminal



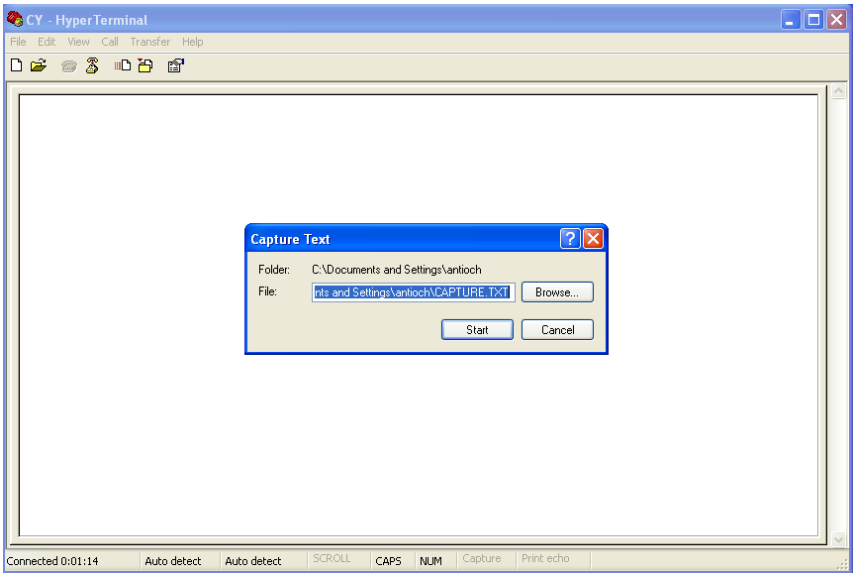
5. The debug messages can be captured in a separate text file instead of displaying them using the HyperTerminal. To capture the debug messages in a text file, go to the **Transfer** tab and select “Capture Text,” as shown in Figure 9.

Figure 9. Configuring HyperTerminal to Capture Text



6. Enter the path and name of the target file and click **Start,** as shown in Figure 10.

Figure 10. Entering File Name for Capture Text



The debug messages are captured in a separate *.txt* file and saved for analysis.

6 Summary

This application note explained the procedure for serial debugging, using the bulkloop example project as a reference. The *FX2LPSerial.c* and *FX2LPSerial.h* files attached to this application note can be added to any other project for debugging purposes.

A Cypress Design Resources

Cypress FX2LP design resources include datasheets, application notes, evaluation kits, reference designs, firmware examples, and software tools. The resources are summarized in Table 1.

Table 1. FX2LP Resource Summary

Design	Available Resources	Where To Find Resources
Hardware	Development Board – Schematic, Board files and documentation	Development Kit (DVK) Schematic Board files available with FX2LP DVK installation DVK User Guide DVK Quick Start Guide
	Hardware design guidelines including recommendations for crystals, decoupling capacitors for power supplies and PCB layout	Application note – AN15456
	IBIS model	http://www.cypress.com/?id=193&rtID=114
FX2LP Firmware	Free version of Keil IDE (up to 4KB of code size)	Available with FX2LPDVK installation
	Firmware examples	
	Sync Slave FIFO firmware example in a complete design with FPGA	Application note - AN61345
Firmware Debug	Setting Up, Using, and Troubleshooting the Keil Debugger Environment	Application note - AN42499
Code Example	A comprehensive list of all USB high speed (FX2LP) code examples	USB Hi-Speed Code Examples
Host PC Software	USB2.0 driver – cyusb.sys	Available with Suite USB installation. This Suite USB installation file (.exe) is also available with the FX2LP DVK installation. Suite USB for Mac OS is also available. Use FX3 SDK for Linux to get the host application similar to Control Center for Linux platform.
	Host application examples – Control Center and Streamer applications	
GPIF Interface Design	GPIF Designer Tool that enables you to design a GPIF waveform and generate code to be integrated into firmware	Available with GPIF Designer installation. The GPIF Designer installation file (.exe) is also available with the FX2LP DVK installation.
	Examples of popular GPIF implementations	Application notes - AN57322 – Interfacing SRAM with FX2LP over GPIF AN66806 – EZ-USB® FX2LP™ GPIF Design Guide AN63787 – EZ-USB® FX2LP™ GPIF and Slave FIFO Configuration Examples using 8-bit Asynchronous Interface
	Documentation on GPIF and instructions for using the tool	GPIF Designer's User Guide – available with GPIF Designer Tool

Other Collateral	
FX2LP Datasheet	http://www.cypress.com/?rID=38801
FX2LP Technical Reference Manual	http://www.cypress.com/?rID=38232
Application Notes	http://www.cypress.com/?id=193&rtID=76
Reference Designs	http://www.cypress.com/?id=193&rtID=201
Knowledge Base Articles	http://www.cypress.com/?id=193&rtID=118
Material on USB 2.0	http://www.beyondlogic.org/usbnutshell/usb1.shtml AN57294 - USB 101: An Introduction to Universal Serial Bus 2.0
Third-party Development Kits	http://www.ztex.de/usb-fpga-1 http://www.opalkelly.com/products/xem6010/

B Application Notes and Reference Designs

B.1 Application Notes

- AN65209 - Getting started with FX2LP™

  AN65209 introduces you to the EZ-USB® FX2LP™ USB 2.0 device controller. This application note helps you build a project for FX2LP and explore its various development tools, and then guides you to the appropriate documentation to accelerate in-depth learning about FX2LP.

- AN15456 - Guide to Successful EZ-USB® FX2LP™ and EZ-USB FX1™ Hardware Design and Debug

  This application note identifies possible USB hardware design issues, especially when operating at high-speed. It also facilitates the process of catching potential problems before building a board and assists in the debugging when getting a board up and running.

- AN50963 - EZ-USB® FX1™/FX2LP™ Boot Options

  This application note discusses the various methods to download firmware in to FX1/FX2LP.

- AN66806 - EZ-USB® FX2LP™ GPIF Design Guide

  This application note describes the steps necessary to develop GPIF waveforms using the GPIF Designer.

- AN61345 - Implementing an FX2LP™- FPGA Interface

  This application note provides a sample project to interface an FX2LP with an FPGA. The interface implements Hi-Speed USB connectivity for FPGA-based applications such as data acquisition, industrial control and monitoring, and image processing. FX2LP acts in Slave-FIFO mode and the FPGA acts as the master. This application note also gives a sample FX2LP firmware for Slave-FIFO implementation and a sample VHDL and Verilog project for FPGA implementation.

- AN57322 - Interfacing SRAM with FX2LP over GPIF

  This application note discusses how to connect the Cypress CY7C1399B SRAM to FX2LP using the General Programmable Interface (GPIF). It describes how to create read and write waveforms using GPIF Designer. This application note is also useful as a reference to connect FX2LP to other SRAMs.

- AN42499 - Setting Up, Using, and Troubleshooting the Keil Debugger Environment

  This application note is a step-by-step beginner's guide to using the Keil Debugger. This guide covers the serial cable connection from PC to SIO-1/0, the monitor code download, and required project settings. Additionally, it gives guidelines to start and stop a debug session, set break points, step through code, and solve potential problems.

- AN4053 - Streaming Data through Isochronous/Bulk Endpoints on EZ-USBR FX2 and EZUSB FX2LP

  This application note provides background information for a streaming application using the EZ-USB FX2 or the EZ-USB FX2LP part. It provides information on streaming data through BULK endpoints, ISOCHRONOUS endpoints, and high bandwidth ISOCHRONOUS endpoints along with design issues to consider when using the FX2/FX2LP in high-bandwidth applications.

- AN58069 - Implementing an 8-Bit Parallel MPEG2-TS Interface Using Slave FIFO Mode in FX2LP

  This application note explains how to implement an 8-bit parallel MPEG2-TS interface using the Slave FIFO mode. The example code uses the EZ-USB FX2LP at the receiver end and a data generator as the source for the data stream. Hardware connections and example code are included.

- AN58170 - Code/Memory Banking Using EZ-USB

  The EZ-USBFX2 family of chips contains an 8051 core. The 8051 core has 16-bit address lines and is able to access 64KB of memory. However, some applications require more than 64KB. This application note describes methods of overcoming this 64KB boundary.

- AN1193 - Using Timer Interrupt in Cypress EZ-USB FX2LP Based Applications

  This application note helps EZ-USBR FX2LP firmware developers to use timer interrupts in their applications.

- AN63787 - EZ-USB® FX2LP™ GPIF and Slave FIFO Configuration Examples using 8-bit Asynchronous Interface

  This application note discusses how to configure the General Programmable Interface (GPIF) and slave FIFOs in EZ-USB FX2LP in both manual mode and auto mode to implement an 8-bit asynchronous parallel interface. This application note is tested with two FX2LP development kits connected back-to-back; the first one operating in master mode and the second operating in slave mode.

- AN61244 - Firmware Optimization in EZ-USB

  This application note describes firmware optimization methods in EZ-USB. Some of these methods are common to any processor and some are specific to the 8051 core of EZ-USB FX2LP.

- AN74505 – EZ USB FX2LP - Developing USB Application on MAC OS X using LIBUSB

  This application note describes a host application built on the MAC OS platform that uses libusb. The host application (Cocoa application) communicates with the BULK IN and BULK OUT endpoints ofFX2LP, using the interfaces provided by the APIs of libusb. This host application implements the transfer with devices that pass the particular VID/PID (=0x04B4/0x1004) identification.

- AN58764 - Implementing a Virtual COM Port in FX2LP

  This application note explains how to implement a virtual COM port device using the standard Windows driver in FX2LP. This information helps to migrate from UART to USB.

- AN45471 - Vendor Command Design Guide for FX2LP

  This application note demonstrates how to code USB vendor commands to perform specific product. In addition, the note explains how to use the Cypress CyConsole utility to issue vendor commands.

B.2 Reference Designs

Several reference designs of FX2LP for popular applications are available. The reference designs include demonstration source code, reference schematics, and a BOM, where appropriate, for the design.

The reference designs available on the Cypress website are:

- CY4661 - External USB Hard Disk Drives (HDD) with Fingerprint Authentication Security

  The CY4661 reference design kit from Cypress and UPEK provides customers with a turnkey solution for an external USB hard disk drive (HDD), with fingerprint authentication, and security to protect and authenticate data. The reference design uses UPEK's Touch Strip Fingerprint Authentication Solution (TCS3 swipe fingerprint sensor and TCD42 security ASIC).

- FX2LP DMB-T/H TV Dongle reference design

  This reference design kit is based on the Cypress FX2LP and Legend Silicon’s chipset. A captured and demodulated RF signal converted to an MPEG2 TS stream by the Legend Silicon chipset is sent to the PC through an FX2LP.The PC plays these streams using a media player. This is a complete design, including all required files.

Document History

Document Title: AN58009 - Serial (UART) Port Debugging of EZ-USB® FX1/FX2LP™ Firmware

Document Number: 001-58009

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2824643	PRKU	12/09/2009	New application note
*A	3167023	AASI	02/08/2011	Major content update Name change Added bulkloop example with debug prints to demonstrate the debugging methodology.
*B	3841353	PRJI	12/14/2012	Updated to new template.
*C	4810700	MVTA	07/27/2015	Updated to new template. Added Appendix A and Appendix B.
*D	5194845	MVTA	03/29/2016	Updated abstract Added link to code examples Updated template Removed reference of CyConsole and SuiteUSB 3.4.7 Added reference of USB Control Center and EZ-USB FX3 SDK Updated figures 6 and 7
*E	5688160	AESATMP8	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

© Cypress Semiconductor Corporation, 2009-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.