AN50963

EZ-USB FX1/FX2LP Boot Options

Author: Prajith Cheerakkoda

Associated Project: Yes

Associated Part Family: CY7C68013A/14A/15A/16A, CY7C64713

Software Version: CyConsole Version 1.7.0.2

Related Application Notes: AN65209

To get the latest version of this application note, or the associated project file, please visit http://cypress.com/an50963.

More code examples? We heard you.

To access a variety of FX2LP code examples, please visit our USB High-Speed Code Examples webpage.

Are you looking for USB 3.0 peripheral controllers?

To access USB 3.0 product family, please visit our USB 3.0 Product Family webpage.

AN50963 describes in detail the boot options available in Cypress's EZ-USB® FX1™/FX2LP™ USB 2.0 peripheral controllers.

Contents

1 Introduction

Cypress' EZ-USB FX1/FX2LP offers a highly integrated solution for USB 2.0 peripheral devices. It comes with an integrated and high-performance CPU based on the industry-standard 8051 microcontroller. A soft (RAM-based) architecture allows flexible configuration and upgrades.

FX1/FX2LP offers the following boot options:

- I²C boot: Use an I²C EEPROM to download the complete firmware with Vendor ID (VID)/Product ID (PID) or only VID/PID/Device ID (DID) information to FX2LP internal RAM.

  Note: You must use a VID assigned to you by the USB Implementers Forum. It is not acceptable to use the Cypress VID and, in particular, the Cypress default VID/PID other than for testing purposes. That rule applies to your firmware VID/PID and EEPROM VID/PID.

- USB boot: Use the Control Center utility provided with the FX2LP Development Kit (DVK) to download the firmware image into FX2LP internal RAM. After the FX2LP device has enumerated, you can download the firmware to FX2LP from the USB Host. FX2LP can enumerate with the default Cypress VID/PID or with a custom VID/PID. If an I²C EEPROM with only VID/PID information is connected to FX2LP, that VID/PID will be used to enumerate. Then you can use a host application to download the firmware to FX2LP.

- Boot from external parallel memory: The EZ-USB FX2LP/FX1 chip executes the firmware from an external parallel memory connected to the address/data bus. Only 128-pin EZ-USB parts allow you to add off-chip data and program memory—100-pin and 56-pin EZ-USB chips do not support external parallel memory.

The EZ-USB FX2LP device (CY7C68013A/14A/15A/16A) supports both full- and high-speed modes. The EZ-USB FX1 device (CY7C64713) supports only the full-speed mode. The FX1/FX2LP CPU is an enhanced 8051 with fast execution and added features.

Note: In this application note, features described for FX2LP also apply to FX1, unless otherwise noted.

1.1 Bootloader Sequence

After reset, the FX2LP bootloader takes the following steps to select the boot option.

1. If no off-chip memory (either on the I^{2}C bus or on the parallel address/data bus) is connected to the EZ-USB, it enumerates as the default USB Device, with descriptors and VID/PID/DID supplied by hardwired internal logic.
2. If an EEPROM containing custom VID/PID/DID values is attached to the EZ-USB I^{2}C bus, EZ-USB also enumerates as the default USB Device, but it substitutes the VID/PID/DID values from the EEPROM for its internal values.
3. If an EEPROM containing EZ-USB firmware is attached to the I^{2}C bus, the firmware is automatically loaded from the EEPROM into the EZ-USB on-chip RAM. Then the CPU is taken out of reset to execute the bootloaded code.
4. EZ-USB begins executing the firmware from the off-chip memory if the following three conditions are present:
   - Flash, EEPROM, or other memory is attached to the address/data bus (128-pin package only).
   - A properly formatted EEPROM meeting the previous requirements is not present.
   - The EA (External Access) pin is tied high (indicating that EZ-USB starts code execution at 0x0000 from off-chip memory).

   ReNumeration is an important concept related to FX2LP boot. After the default USB Device enumerates and the USB Host downloads the firmware and descriptor tables to EZ-USB, it begins executing the downloaded code. This code execution electrically simulates a physical disconnect/connect from the USB Device and causes EZ-USB to enumerate again as a second device, this time taking on the USB personality defined by the downloaded code and descriptors. This patented secondary enumeration process is called “ReNumeration.”

   The EZ-USB Technical Reference Manual (TRM), chapter 3, “Enumeration and ReNumeration,” describes the FX2LP boot sequence in greater detail.

2 I^{2}C Boot

Use an I^{2}C interface to download the firmware or VID/PID/DID information to FX2LP. If an EEPROM containing FX2LP firmware or VID/PID/DID information is attached to the I^{2}C bus, it is automatically loaded from the EEPROM to FX2LP. If it is firmware, it is loaded to on-chip RAM. Then the CPU is taken out of reset to execute the bootloaded code. There are two types of loads through the I^{2}C bus. Refer to the TRM, section 3.2, “EZ-USB Startup Modes,” for more details.

2.1 C0 Load: Loading USB IDs from I^{2}C Device

At power-on reset, if the FX2LP device detects an EEPROM connected to its I^{2}C bus with the value 0xC0 at address 0, EZ-USB automatically copies the VID, PID, and DID from the EEPROM into internal storage. Table 1 shows the format of data for a C0 Load.

Table 1. C0 Load Format

EEPROM Address	Contents
0	0xC0
1	Vendor ID (VID) L
2	Vendor ID (VID) H
3	Product ID (PID) L

EEPROM Address	Contents
4	Product ID (PID) H
5	Device ID(DID) L
6	Device ID(DID) H
7	Configuration Byte

The eighth EEPROM byte contains configuration bits that control the following:

- I^2C bus speed: Default is 100 kHz.
- Disconnect state: Default is for EZ-USB to come out of reset connected to USB.

The TRM, section 3.4.2, “Serial EEPROM Present, First Byte is 0xC0,” contains a full description of the configuration bits.

FX2LP then supplies the EEPROM bytes to the host as part of its response to the host’s Get Descriptor-Device request. These six bytes replace only the VID/PID/DID bytes in the default USB Device descriptor. This causes a host driver matched to the VID/PID/DID values in the EEPROM to be loaded by the host OS.

2.2 C2 Load: Loading Firmware from I^2C Device

At power-on reset, if the FX2LP detects an EEPROM connected to its I^2C bus with the value 0xC2 at address 0, it downloads the firmware from the EEPROM. Table 2 shows the EEPROM image format for C2 load. Compared to C0 load, C2 load eliminates the need to download the firmware from the USB Host.

Note: The TRM, section 3.4.3, “Serial EEPROM Present, First Byte is 0xC2,” contains a full description of the configuration bits.

Cypress provides host application utilities, called CyConsole and Control Center. Use either one to program the EEPROM during development. For Linux developers, a host application is available in the FX3 SDK for Linux.

FX2LP interprets the format shown in Table 2; it decodes the format and downloads it to the appropriate memory locations. The Hex2bix utility can be used to generate the EEPROM image for C2 load. For more information on the Hex2bix utility, see the *readme.txt* file at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Utilities\Hex2Bix* after installing the FX2LP DVK.

Table 2. C2 Load Format

EEPROM Address	Contents
0	0xC2
1	Vendor ID (VID) L
2	Vendor ID (VID) H
3	Product ID (PID) L
4	Product ID (PID) H
5	Device ID(DID) L
6	Device ID(DID) H
7	Configuration Byte
8	Length H
9	Length L
10	Start Address H
11	Start Address L
–	Data Block
–	
–	Length H

EEPROM Address	Contents
–	Length L
–	Start Address H
–	Start Address L
–	Data Block
–	
–	0x80
–	0x01
–	0XE6
–	0x00
Last	00000000

EZ-USB uses the EEPROM address pins A2, A1, and A0 to determine whether it is connected to a small EEPROM or large EEPROM. Single-byte-address EEPROMs (small EEPROMs) must be strapped to address 000, while double-byte-address EEPROMs (large EEPROMs) must be strapped to address 001. Generally, a double-byte-addressed EEPROM is used for C2 load. See Section 13.6, “EEPROM Boot Loader” of the Technical Reference Manual (TRM) for further details on how the EZ-USB device determines if it is connected to small or large EEPROM.

Note: Although FX2LP can perform C2 load from EEPROMs as large as 64 KB, code can be downloaded only to on-chip RAM (16 KB) by the hardware bootloader. Using a bootloader, the firmware allows you to download to external RAM through a two-stage bootload. The Hex2bix utility has the bootloader embedded and can be invoked using the -e option. The -e option is used to create a file for the external RAM in IIC format.

A schematic of the DVK shows a sample EEPROM connection to FX2LP. The parts used for large and small EEPROMs are the 24LC128-I/P and 24LC00/P, respectively.

2.3 EEPROM Hardware Connection

Figure 1 and Figure 2 show connection diagrams for C0 load and C2 load, respectively.

An external pull-up resistor is required on both the SCL and the SDA lines. The recommended value is 2.2 kΩ. To make the connection, take the following steps:

- Ensure that the pull-up resistor is on the FX2LP side of the connection.
- Enable the SDA pin to be disconnected from FX2LP by adding a switch or jumper. This disables the EEPROM boot so that the EEPROM can be programmed in-system.

For C0 load, use the EEPROM address pins A2, A1, and A0 to indicate that the EEPROM is single-byte addressed (000).

Figure 1. EEPROM Connections for C0 Load

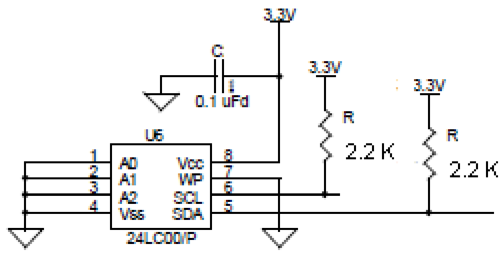
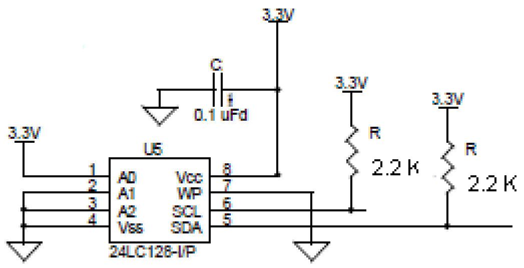


Figure 2. EEPROM Connections for C2 Load



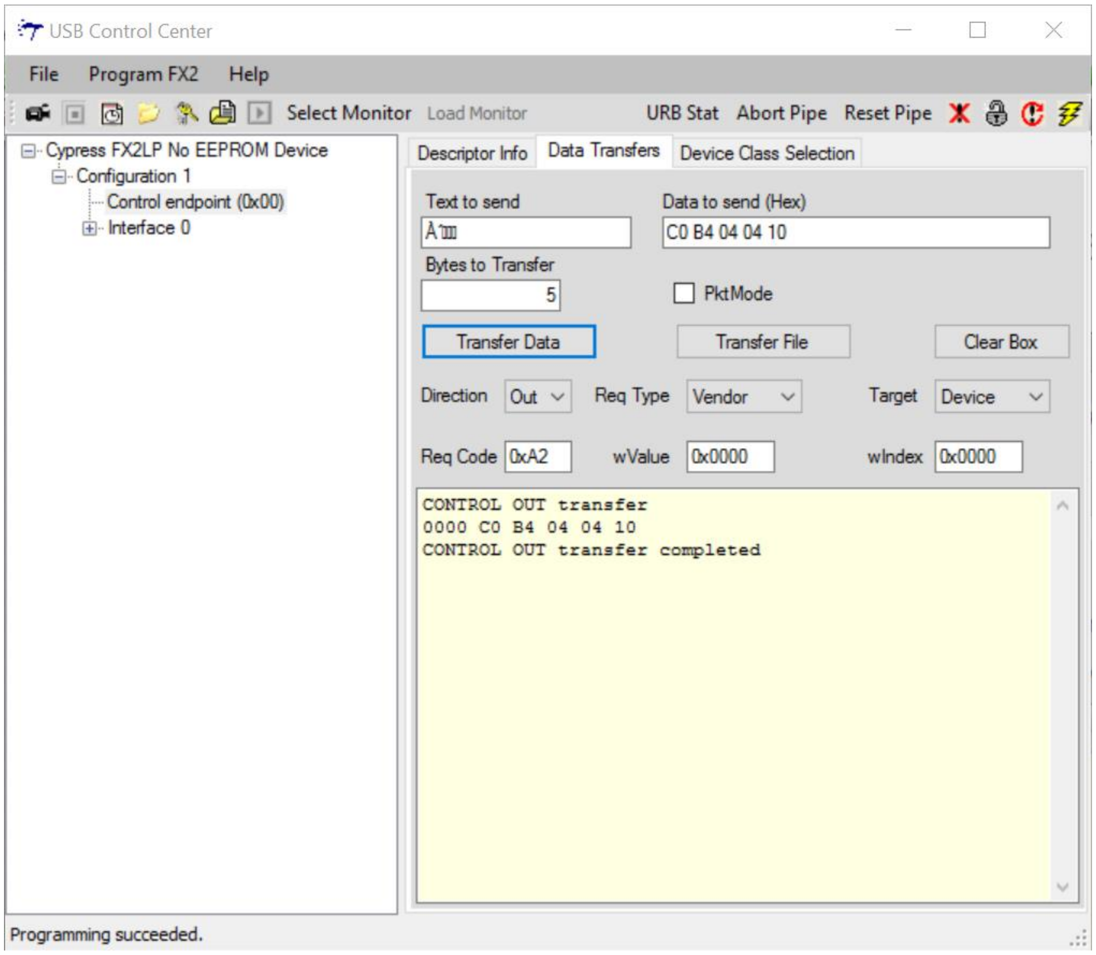
2.4 Programming I^{2}C EEPROM

There are two options for programming the EEPROM. Preprogram it and drop it into the system, or program it in-system using Control Center. The following steps demonstrate how to perform C0 load in the FX2LP DVK using Control Center:

1. Power on the FX2LP DVK board without the EEPROM connected, so that the part can enumerate with the default Cypress VID/PID. The device needs to be bound to *CyUSB3.sys* for access through Control Center. Refer to the "Matching Devices to Driver" section of the CyUSB3.sys Programmer's Reference.
2. After the board enumerates, set switch SW2 on the DVK to "EEPROM" and SW1 to "small EEPROM."
3. Open Control Center and navigate to the DVK firmware example Vend_ax in *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Firmware\Vend_ax* (may vary based on installation) and click Open. This downloads the *Vend_Ax.hex* file into the FX2LP internal RAM. For programming FX2LP using Control Center, refer to the *CyControlCenter.pdf* file at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Windows Applications\Application Source files\c_sharp* (after installing FX2LP DVK).
4. Select "Control endpoint (0x00)," enter "0xA2" into the **Req Code** field, enter "0x0000" into the **wValue** and **wIndex** fields, select "Out" for **Direction,** and select "Vendor" as the **Req type** and "Device" as the **Target.** Finally, enter the VID/PID combination into the **Data to send (Hex)** field, as shown in Figure 3. Table 3 describes the bytes entered in the **Data to send** field of Control Center.
5. Click **Transfer Data**. A successful EEPROM upload is shown in Figure 3.
6. To check the programmed VID/PID, unplug the device and then plug it back in. This time, the device will enumerate with the programmed VID/PID. Note that the DVK will renumerate in the Control Center utility only if it gets bound to the *CyUSB3.sys* driver. For example, passing 'C0 B4 04 04 10' makes it to enumerate as 'Cypress FX2LP Sample Device'.

Another way of doing a C0 load is with the **Small EEPROM** option in the Control Center utility. Open the utility, follow steps 1 and 2, click the **Program FX2** tab, and then select "small EEPROM**."** Browse to the IIC file, which contains the VID/PID information, and **Open** it. An example IIC file *small.iic* is provided along with the application note.

Figure 3. EEPROM Program



Text to send: À´ш

Data to send (Hex): C0 B4 04 04 10

Bytes to Transfer: 5

Direction: Out   Req Type: Vendor   Target: Device

Req Code: 0xA2   wValue: 0x0000   wIndex: 0x0000

```
CONTROL OUT transfer
0000 C0 B4 04 04 10
CONTROL OUT transfer completed
```

Programming succeeded.

Note: The format of the data transferred should follow the C0 load format as in Table 3.

Table 3. C0 Load Bytes

Byte Number	Parameter	Comment
Byte 0	C0	Indicates valid small EEPROM
Byte 1	VID L	Lower byte of VID
Byte 2	VID H	Upper byte of VID
Byte 3	PID L	Lower byte of PID
Byte 4	PID H	Upper byte of PID

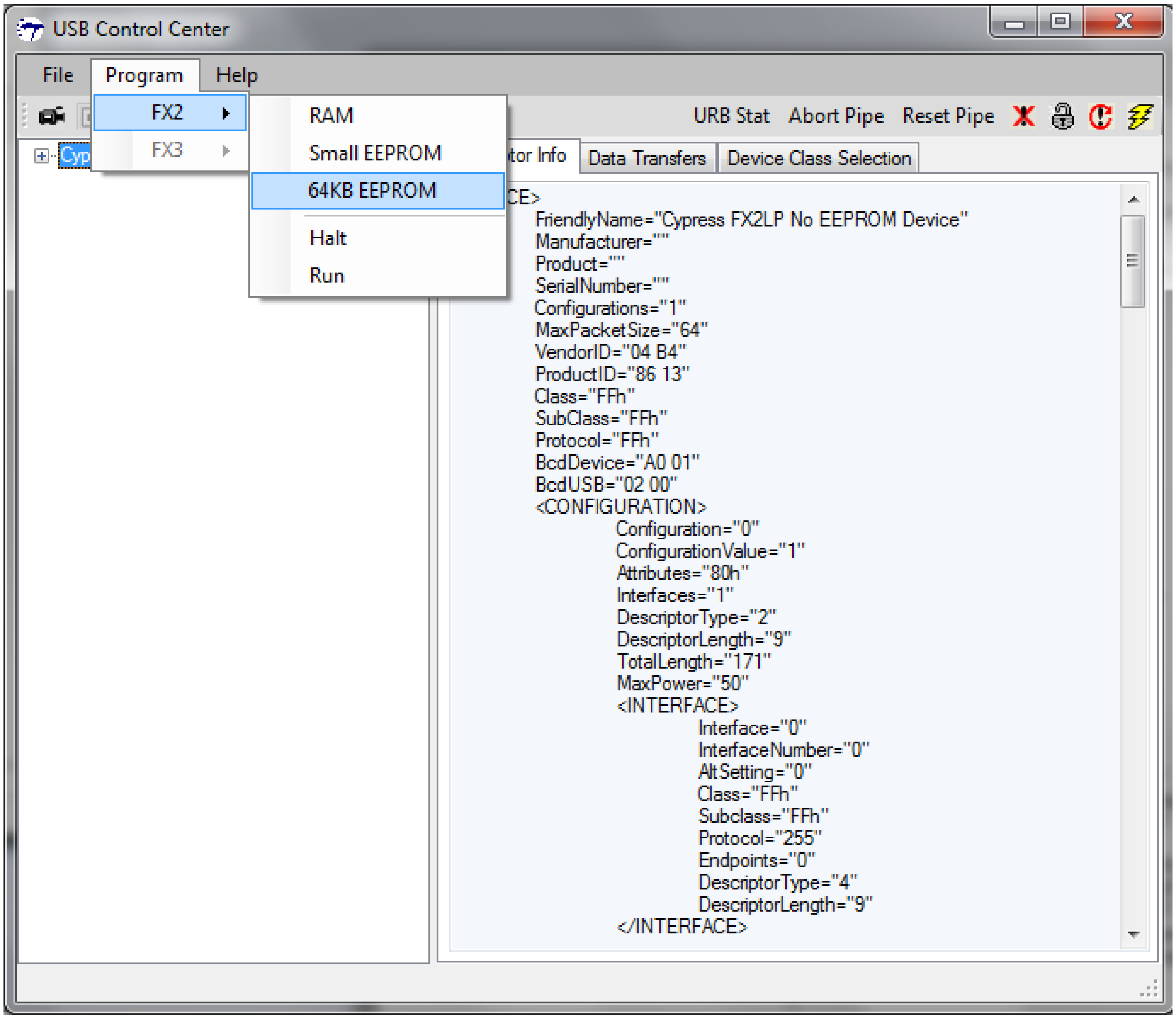
The following steps demonstrate how to perform a C2 load in the FX2LP DVK using Control Center:

1. On the FX2LP DVK board, select “SW2-NO EEPROM” and connect a USB A-to-B cable from the J1 connector on the board to a Windows PC USB Host controller port. The EZ-USB device enumerates with the default VID/PID.
2. Before programming the EEPROM image file (.*iic*), select “SW2-EEPROM” and “SW1-LARGE EEPROM” as switch settings to select the large EEPROM U5 on board.
3. Open Control Center from Windows: **Start** > **All Programs** > **Cypress** > **Cypress USBSuite** > **Control Center** (see Figure 4). Observe that EZ-USB FX2LP is listed as “Cypress FX2LP No EEPROM Device.”

4. Choose **Program** > **FX2** > **64KB EEPROM** in the EZ-USB interface window and browse to the project folder, as shown in Figure 4. Select the *bulkloop.iic* image at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Firmware\Bulkloop* or the corresponding installation path.
5. Press RESET button S1. The device will enumerate with the new VID/PID and will prompt the driver. To access the device through Control Center for testing bulk loop operation, it needs to be bound to *CyUSB3.sys*.

   Note: Cypress has migrated from *CyUSB.sys* to *CyUSB3.sys* for FX1/FX2. Cypress has stopped supporting *CyUSB.sys*.

Figure 4. C2 Load



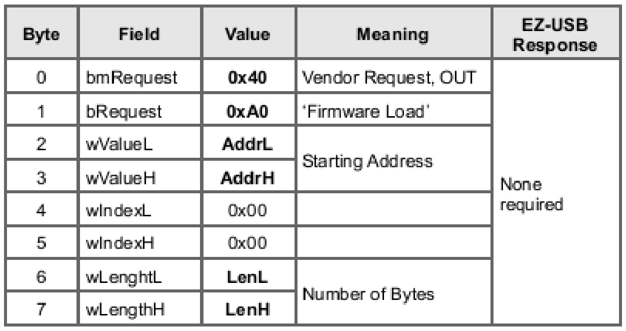
```
CE>
    FriendlyName="Cypress FX2LP No EEPROM Device"
    Manufacturer=""
    Product=""
    SerialNumber=""
    Configurations="1"
    MaxPacketSize="64"
    VendorID="04 B4"
    ProductID="86 13"
    Class="FFh"
    SubClass="FFh"
    Protocol="FFh"
    BcdDevice="A0 01"
    BcdUSB="02 00"
    <CONFIGURATION>
            Configuration="0"
            ConfigurationValue="1"
            Attributes="80h"
            Interfaces="1"
            DescriptorType="2"
            DescriptorLength="9"
            TotalLength="171"
            MaxPower="50"
            <INTERFACE>
                    Interface="0"
                    InterfaceNumber="0"
                    AltSetting="0"
                    Class="FFh"
                    Subclass="FFh"
                    Protocol="255"
                    Endpoints="0"
                    DescriptorType="4"
                    DescriptorLength="9"
            </INTERFACE>
```

3 USB Boot

A host application can use vendor commands to download the firmware to FX2LP. The USB endpoint zero protocol provides a mechanism for vendor-specific requests. Bits 6:5 of the bmRequestType field are set to “10” for a vendor request.

FX2LP provides native support for the 0xA0 vendor request to write/read the FX2LP internal memory. The host application can use the 0xA0 vendor request to download the firmware to FX2LP. In the 0xA0 vendor request, the wValue field specifies the address of the memory location to be accessed. The data phase of the 0xA0 vendor request is used to send the data to write the memory. The A0 vendor request format is shown in Table 4. For more details, refer to the TRM, section 3.8, “EZ-USB Vendor Request for Firmware Load.”

Table 4. A0 Vendor Request Format



Byte	Field	Value	Meaning	EZ-USB Response
0	bmRequest	**0x40**	Vendor Request, OUT	None required
1	bRequest	**0xA0**	‘Firmware Load’	
2	wValueL	**AddrL**	Starting Address	
3	wValueH	**AddrH**		
4	wIndexL	0x00		
5	wIndexH	0x00		
6	wLenghtL	**LenL**	Number of Bytes	
7	wLengthH	**LenH**		

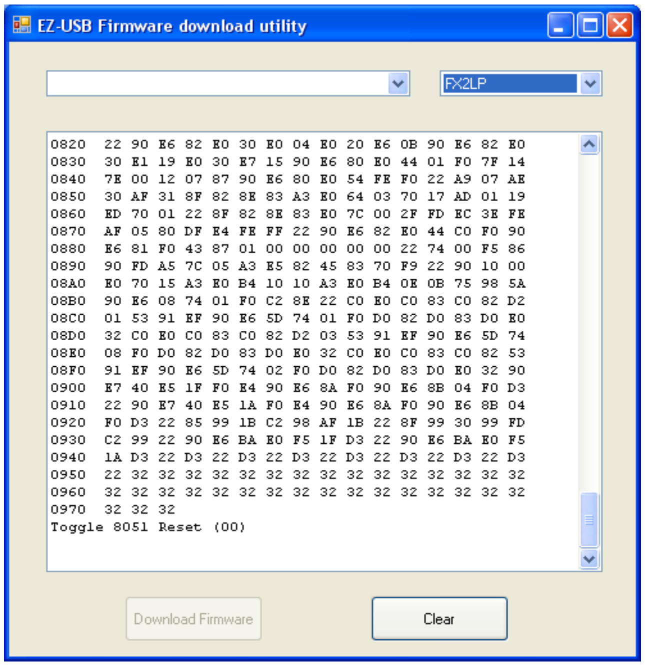
3.1 Example Host Application GUI

Attached to this application note is a *CyAPI.lib* (VC++)–based host application, “EZ-USB Firmware download utility,” developed using Visual Studio 2008. It demonstrates firmware download using the 0xA0 vendor request. CyAPI.lib is part of SuiteUSB 3.4 – USB Development Tools for Visual Studio. CyUSB.dll, the C# counterpart of CyAPI.lib, includes a LoadRAM() API, which is used to download the firmware to FX2LP.

The host application has two drop-down lists: the device list and the target device list. The device list identifies the USB devices connected to *CyUSB3.sys* and is used to select the target device. The target device list is used to select whether the target device is FX2LP or FX/ANXX. It also includes a text box and two buttons. The text box displays the data being downloaded to the device. One of the two buttons is used for firmware download; the other is used to clear the text box. The values downloaded to the device are updated in the text box of the host application. Figure 5 shows the firmware download host application GUI.

Note: This utility has been tested successfully on the Windows XP, Windows Vista, Windows 7, Windows 8.1, and Windows 10 operating systems.

Figure 5. Firmware Download Host Application



```
0820  22 90 E6 82 E0 30 E0 04 E0 20 E6 0B 90 E6 82 E0
0830  30 E1 19 E0 30 E7 15 90 E6 80 E0 44 01 F0 7F 14
0840  7E 00 12 07 87 90 E6 80 E0 54 FE F0 22 A9 07 AE
0850  30 AF 31 8F 82 8E 83 A3 E0 64 03 70 17 AD 01 19
0860  ED 70 01 22 8F 82 8E 83 E0 7C 00 2F FD EC 3E FE
0870  AF 05 80 DF E4 FE FF 22 90 E6 82 E0 44 C0 F0 90
0880  E6 81 F0 43 87 01 00 00 00 00 00 22 74 00 F5 86
0890  90 FD A5 7C 05 A3 E5 82 45 83 70 F9 22 90 10 00
08A0  E0 70 15 A3 E0 B4 10 10 A3 E0 B4 0E 0B 75 98 5A
08B0  90 E6 08 74 01 F0 C2 8E 22 C0 E0 C0 83 C0 82 D2
08C0  01 53 91 EF 90 E6 5D 74 01 F0 D0 82 D0 83 D0 E0
08D0  32 C0 E0 C0 83 C0 82 D2 03 53 91 EF 90 E6 5D 74
08E0  08 F0 D0 82 D0 83 D0 E0 32 C0 E0 C0 83 C0 82 53
08F0  91 EF 90 E6 5D 74 02 F0 D0 82 D0 83 D0 E0 32 90
0900  E7 40 E5 1F F0 E4 90 E6 8A F0 90 E6 8B 04 F0 D3
0910  22 90 E7 40 E5 1A F0 E4 90 E6 8A F0 90 E6 8B 04
0920  F0 D3 22 85 99 1B C2 98 AF 1B 22 8F 99 30 99 FD
0930  C2 99 22 90 E6 BA E0 F5 1F D3 22 90 E6 BA E0 F5
0940  1A D3 22 D3 22 D3 22 D3 22 D3 22 D3 22 D3 22 D3
0950  22 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
0960  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
0970  32 32 32
Toggle 8051 Reset (00)
```

Note: This example host application works only with the *CyUSB3.sys* driver.

3.1.1 Implementing EZ-USB Firmware Download Utility

The host application that comes with this application note demonstrates firmware download to FX2LP from the USB Host. This method eliminates the need to store the firmware on a large EEPROM, enabling you to use a smaller, less expensive EEPROM.

In this case, the host first holds the CPU in reset and uses the EZ-USB default USB Device to download the firmware. Then the host takes the CPU out of reset so that it can execute the downloaded code. Firmware download to FX2LP is done using A0 vendor commands. The host must write to the CPUCS register to put the CPU in or take it out of reset. Use the 8051RES bit (bit 0) of the CPUCS register of FX2LP to reset or run the 8051. The USB Host can write to this bit through the 0xA0 vendor request. The host writes '1' to reset the 8051 and '0' to run the 8051.

The 0xA0 firmware load request may be used even after ReNumeration, but the request is valid only while the 8051 is held in reset.

If the application implements vendor-specific USB requests and the firmware load feature is not required, then do not use the bRequest value 0xA0 for your custom requests. To avoid future incompatibilities, vendor requests 0xA0 to 0xAF are reserved by Cypress.

Keil µVision2 is used to develop the firmware for FX2LP. It outputs the firmware in the Intel hex format (http://microsym.com/editor/assets/intelhex.pdf). This firmware file is parsed to obtain the firmware image, which is downloaded to FX2LP. The EZ-USB Firmware download utility parses the firmware file and downloads the firmware image to FX2LP.

Note: Because of the difference in address of CPUCS (0x1F92 for AN21xx, FX, FX1 and 0xE600 for FX2LP) and register addresses between the AN21xx, FX and FX1, and FX2LP series of chips, the Vend_ax firmware to be used differs between the two. The host application demonstrates firmware download to AN21xx and FX as well to expose the Vend_ax firmware in an easily usable ArrayList format. AN45471 – Create Your Own USB Vendor Commands Using FX2LP provides more details on vendor commands and how to test the ones included in Vend_ax, which defines several vendor commands.

3.1.2 Algorithm

A 0xA0 vendor request can access only the internal memory of FX2LP. When part of the firmware resides in external memory, then you need a method to access external memory. The Vend_ax example, which is part of the DVK, is used for this purpose. You can find it at *C:\Cypress\USB\CY3684_EZ-USB_FX2LP_DVK\1.1\Firmware\Vend_ax* (location may vary based on installation) after installing the DVK. This firmware implements the 0xA3 vendor request for read/write to the FX2LP external memory. This vendor request can access internal memory as well.

Standard Intel Hex Format

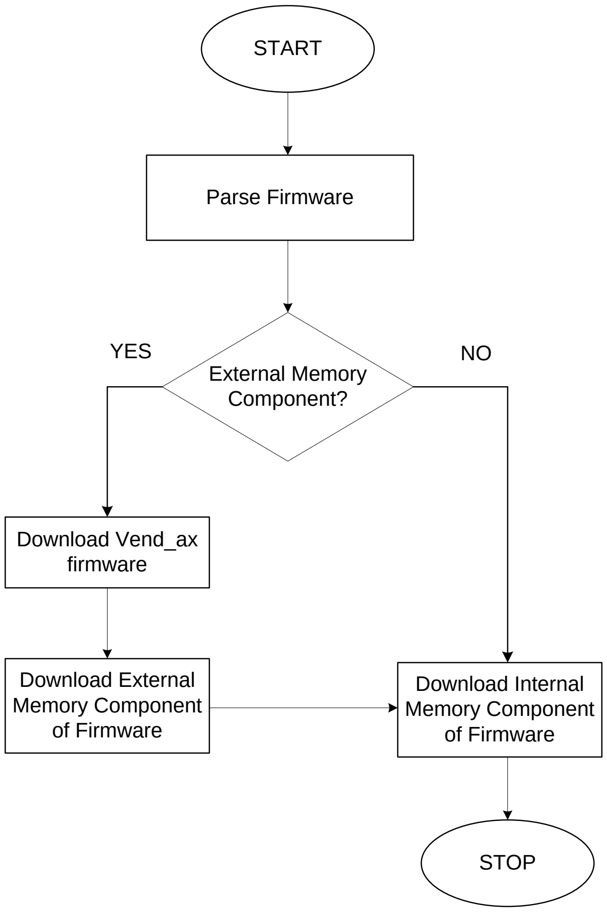
Each line in a hex file has the same basic structure and is called a “record.” Each record has six parts, as follows:

- **Start code**: One character, an ASCII colon ':'
- **Byte count**: Two hex digits, a number of bytes (hex digit pairs) in the data field
- **Address**: Four hex digits**,** a 16-bit address of the beginning of the memory position for the data
- **Record type**: Two hex digits, 00 to 05, defining the type of data field
- **Data**: A sequence of *n* bytes of the data themselves, represented by 2*n* hex digits
- **Checksum**: Two hex digits, the least significant byte of the two's complement of the sum of the values of all fields except fields 1 and 6 (start code ":" byte and two hex digits of the checksum)

Note: For more details on the hex file format, read the specification.

Figure 6 shows the algorithm used to download the firmware to FX2LP through the host application.

Figure 6. Firmware Download Algorithm


START
Parse Firmware
External Memory Component?
YES
NO
Download Vend_ax firmware
Download External Memory Component of Firmware
Download Internal Memory Component of Firmware
STOP

Functions in EZ-USB Firmware Download Utility

The functions in the host application are:

- **WndProc(message %m):** This user-defined function processes messages sent to a window.
- **CPU_Reset(bool ON):** This function resets or runs the FX2LP CPU based on a Boolean input. If input is true, then the CPU is placed in Reset; if input is false, the CPU is allowed to run.
- **GetDevice():** This function populates the Device_List drop-down list with the devices bound to *CyUSB.sys* and gets a handle on them.
- **Error(String^ err):** This function is used to print error messages.
- **InitVendAX(bool FX2LP):** This function is used to initialize an array (VendAX) that contains the Vend_ax firmware. A Boolean input to this function specifies whether the firmware contains an external memory component. The VendAX array is used to download the Vend_ax firmware to FX2LP.
- **Firmware_Download(String^ file):** This function uses the path of the firmware file as input to parse and downloads the firmware to FX2LP. When "VendAX" is passed as input, this function downloads the Vend_ax firmware to FX2LP. If the firmware has an external memory component, then this function is called recursively for downloading the Vend_ax firmware. The FX2LP internal RAM is 16 KB. It is 8 KB for the predecessors of FX2LP (AN21xx, FX, and so on). Because 0xA3 can access internal and external memory, the host application uses 0x2000 as the start location of external memory to simplify the algorithm.

3.2 Automated Firmware Download Using the Script File

Control Center can create scripts that can automatically download the firmware to FX2LP/FX1. To use this technique, FX2LP/FX1 is booted with a VID/PID through 'C0 Load', as explained in the section C0 Load: Loading USB IDs from I2C Device. The INF file binds this VID/PID (Bootloader VID/PID) to the script file that contains the recorded sequence for downloading the firmware. Upon enumeration, this firmware is downloaded to the device using the script file. The device then enumerates with the VID/PID contained in the downloaded firmware. The advantage of this method is that it eliminates the need to store the firmware on a large EEPROM. Therefore, a smaller and less expensive EEPROM can be used. It also automates firmware download from USB. The script feature is supported only by the Cypress-provided *CyUSB3.sys* driver.

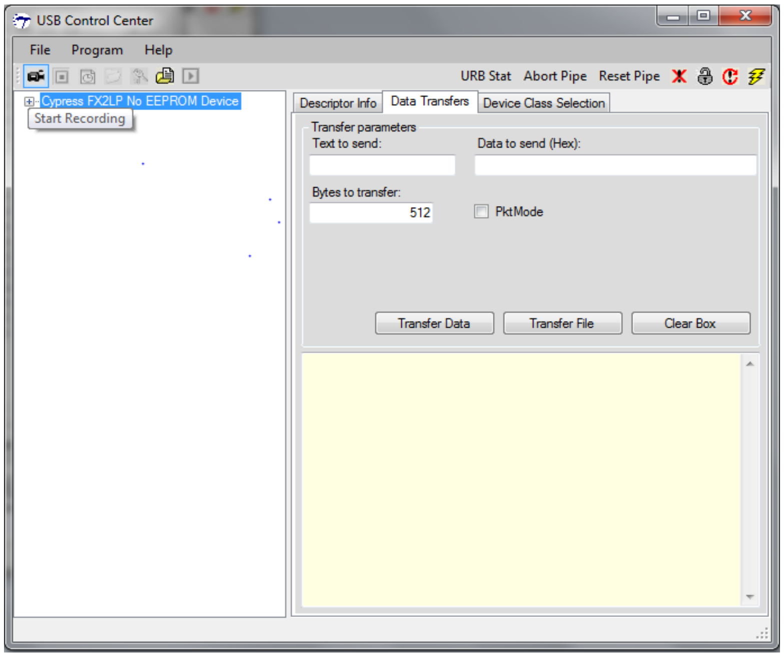
Note: The VID/PID value used in the downloaded firmware should be different from that of the bootloader VID/PID. Otherwise, a recursive download of the firmware will occur.

3.2.1 Creating the Script File

After you have working firmware, create the script to download the firmware using the FX2LP DVK:

1. Disconnect the SDA line from the FX2LP (SW2 to "No EEPROM" position).
2. Connect your board to the computer using a USB cable.
3. Allow the board to enumerate with its default settings. It must appear as "Cypress FX2LP No EEPROM Device" (VID/PID 0x04B4/0x8613) for FX2LP and "Cypress EZ-USB FX1 No EEPROM Device" (VID/PID 0x04B4/0x6473) for FX1 in Device Manager. The procedure for binding the driver to the device is available in section 7.2.4, "Binding Cypress USB Driver for the Downloaded Firmware Image," of the EZ-USB DVK User Guide, which is part of the DVK documentation.
4. Open Control Center.
5. Click the "Record" button in the top left corner of Control Center, as shown in Figure 7.

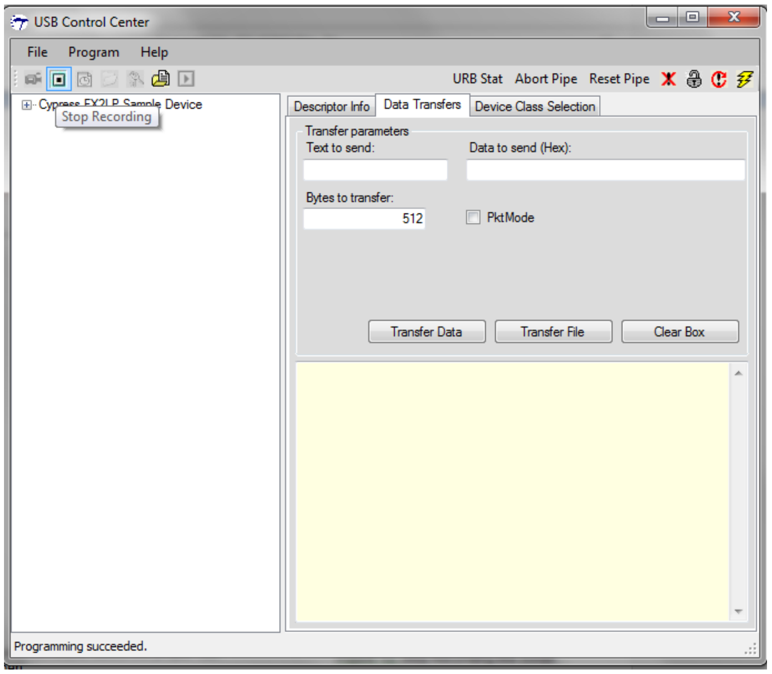
Figure 7. Start Recording the Script


USB Control Center
File Program Help
URB Stat Abort Pipe Reset Pipe
Cypress FX2LP No EEPROM Device
Start Recording
Descriptor Info Data Transfers Device Class Selection
Transfer parameters
Text to send:
Data to send (Hex):
Bytes to transfer:
512
PktMode
Transfer Data Transfer File Clear Box

6. Choose **Program** > **FX2** > **RAM** and select the firmware (hex file) for which the script is to be generated.
7. Once the program is successfully loaded into RAM, click the "Stop recording" button adjacent to the "Start recording button," shown in Figure 8. Save the script with an *.spt* extension.
8. Name your file "*MyDevice.spt*" and save it. The script is now created.

   Note: Sometimes script generation will fail because of an error generated by Control Center. If this occurs, close and open Control Center in admin mode and repeat the steps to create the script.

Figure 8. Stop Recording the Script


USB Control Center
File Program Help
URB Stat Abort Pipe Reset Pipe
Cypress FX2LP Sample Device
Stop Recording
Descriptor Info Data Transfers Device Class Selection
Transfer parameters
Text to send:
Data to send (Hex):
Bytes to transfer:
512
PktMode
Transfer Data Transfer File Clear Box

Programming succeeded.

3.2.2 Creating an INF File

To use the script created previously to download the firmware, you must create an INF file. This file recognizes the VID/PID that was programmed onto the small EEPROM of the device. Then it loads the firmware onto the device using the script file. A sample INF file for demonstration is included as part of the application note project attachment. Detailed steps to create an INF file for the script-based firmware download are available in Part II of the CyUSB3.sys Programmer's Reference.

1. Start with the example INF file *cyusb3.inf* that comes with this application note. This file is commented upon and set up to run the script as easily as possible.
2. Open *cyusb3.inf* with a text editor, such as Notepad.
3. In *cyusb3.inf*, replace the 04B4/8613 VID/PID, which is the FX2LP standard bootloader VID/PID, with the VID/PID that was downloaded to the small EEPROM.
4. Replace the 04B4/1004 VID/PID, which is the bulk loop firmware VID/PID, with the VID/PID from your firmware.
5. Near the bottom of the INF file, in the "strings" section, ensure that the VIDs/PIDs shown match the entries for the VID/PID before script download and the VID/PID the device will enumerate with after firmware download. Place some meaningful text between the quotation marks.
6. Save the new INF file. The INF file attached to this application note includes a digital signature. Note that any changes made to the INF file will invalidate the digital signature in the catalog file. You will have to get the INF file certified. Or to use it without a signature, you will need to disable the driver signature enforcement feature in Windows 8, Windows 8.1, and Windows 10. Windows 7, Windows Vista, and Windows XP will give you a warning but allow binding without disabling the driver signature.

3.2.3 Firmware Download Using the Script File

The files that come with this application note download the bulk loop firmware (with VID 04B4 and PID 1004) into FX2LP using the script. The VID/PID value downloaded to the small EEPROM through C0 load is 04B4/8613. Therefore, INF files included will have these IDs specified. These files are available along with this application note in separate folders based on the OS and CPU architectures. You can either experiment with these VID/PID values, in which case you can directly use the attached files without any changes. If you want to experiment with different VID/PID values, change the following: the IIC file used to download to small EEPROM, the firmware to be downloaded to the RAM, the script file, and the INF file.

The default VID/PID to be used for experimenting with these files is 0x04B4/0x8613. The firmware VID/PID is 0x04B4/0x1004.

1. After the script, INF file, and board are ready, download the firmware to the FX2LP device using the script.
2. Ensure that the script, driver files, and INF file are in the same folder.
3. Connect the board to the host and bind it manually to the driver provided in the "Drivers" folder of the application note attachment.
4. Different INF files are provided for Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 8.1 (32 bit and 64 bit) with this application note in the "Drivers" folder. Choose the INF file that corresponds to your OS.

4 Boot from External Parallel Memory

You can connect a parallel memory externally on the EZ-USB address/data bus. The firmware is not loaded in the EZ-USB internal RAM; instead, it is run directly from the external parallel memory. This memory can be flash, EEPROM, or any other parallel memory.

Note: Only 128-pin packages have an external address/data bus. Therefore, this application is limited to these parts: CY7C68013A-128AXC, CY7C68014A-128AXC, CY7C64713-128AXC, and CY7C64714-128AXC.

To meet the requirements of this mode, you need to make some modifications to the firmware and development boards.

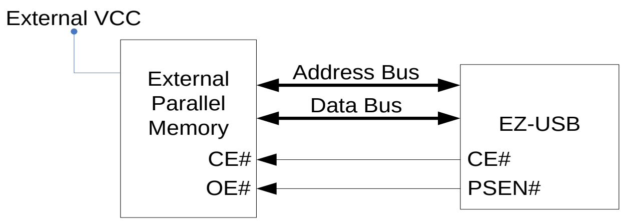
4.1 Boot Requirements

1. The code memory for this mode must be in the external parallel memory. If the EA input pin is set to logic HIGH on power-on reset, the memory map of EZ-USB FX2LP is modified so that the entire external 64-KB memory is mapped as code space, and the complete internal memory (main RAM) is used as data space.

   Note: Setting EA=1 puts EZ-USB in a Harvard architecture configuration, compared with the other Von Neumann architecture configurations of EZ-USB.

2. EZ-USB first tries to boot from the serial EEPROM on the I^{2}C bus. It checks for 0xC0 and 0xC2 as first bytes to determine which boot method to use. To avoid a serial EEPROM bootload, there should be no serial EEPROM on the I^{2}C bus. If an external serial EEPROM is connected, the first byte must not have the values 0xC0 or 0xC2.
3. Satisfying these two requirements ensures that EZ-USB looks for firmware in the external memory on the address/data bus and starts executing code from address 0x0000 on power-on reset if the jumper setting on the FX2LP DVK is MM0=IN and MM1=OUT. Refer to section 4.8, "Memory Maps," of the EZ-USB DVK User Guide for details on the memory map on the FX2LP DVK.
4. The WinBond W27C512-45Z parallel EEPROM used for testing has an access time of 45 ns. For more information on the W27C512-45Z, see the W27C512 datasheet. A possible replacement part is the Atmel AT27C512R. For more information, see the AT27C512R datasheet. Figure 9 shows the connection details. Make certain that the chip has sufficient power; some chips may require more power than that which the DVK supports.[1]

Figure 9. Connection Details for External Parallel Memory



The external parallel memory should have the following characteristics:

- Fast enough access time to enable EZ-USB to read code. See the FX2LP datasheet for details on access time tACC1.
- The raw binary image of the firmware should be loaded on the external parallel memory from location 0x0000 without any start bytes, unlike the IIC file image that goes on the serial EEPROM. The hex file generated by the Keil software is in the Intel hex format. It cannot be loaded directly. Convert it to the raw binary file format before loading it on the external parallel memory, unless the programmer utility does it automatically. The ALL-100 programmer has been tested successfully for programming the WinBond W27C512-45Z parallel EEPROM on the FX2LP DVK. Its software package has a utility to convert the Intel hex file into a raw hex file. The IIC file generated by the hex2bix utility inserts the start byte in the firmware image; it should not be used to load on the external parallel memory.

5. The setup data pointer can access only the internal memory locations from 0x0000 to 0x3FFF and 0xE000 to 0xE1FF. If the firmware uses SUDPTRH:L for pointing to the descriptors for enumeration, the descriptors must be stored in the internal data space. In this boot method, the descriptors are stored with the firmware in the code memory that is completely external to EZ-USB. Therefore, the descriptors should be unconditionally copied from the external memory to the internal memory at startup.

4.2 Firmware Modifications

You can modify any firmware based on the EZ-USB framework as part of the CY3684 DVK contents to run from external parallel memory. As explained in the section Boot Requirements, make sure that the descriptors are unconditionally copied into the EZ-USB internal memory (main RAM). The modifications in the EZ-USB framework follow.

1. In *fw.c*, there is a descriptor setup section after TD_Init() in the main function. In this section, remove the if condition "if ((WORD)&DeviceDscr & 0xC000)", and whatever code (mostly descriptor copying) resides in this if condition is carried out unconditionally. It ensures that the standard USB descriptor (Device/Configuration Descriptors) values are always copied into the internal memory (main RAM).
2. Change the line that copies these standard USB descriptors from external memory to internal memory to let the compiler know that the descriptors stored in the external memory are actually in code memory: Change "*((BYTE xdata *)IntDescrAddr+i) = *((BYTE xdata *)ExtDescrAddr+i);" to "*((BYTE xdata *)IntDescrAddr+i) = *((BYTE code *)ExtDescrAddr+i);".

[1] Cypress has not tested the Atmel AT27C512R. The AT27C512R is a possible replacement part based only on datasheet review. This is an OTP part. Reprogrammable parts with slower access times work with FX2LP running at 24 MHz.

3. Include class-specific descriptors in the external-to-internal copying for applications in which class-specific descriptors are defined. For the HID KB example, three descriptor pointers—pHIDDscr, pReportDscr, and pReportDscrEnd—are now updated in the main function. Updated offsets are applied to them and in the main function. Their re-evaluation in the DR_GetDescriptor function (function in *periph.c*) is removed.

4.3 Hardware Modifications

The most necessary hardware modification is to set the EA pin to logic 1. However, when using the DVK boards, certain hardware connections must be modified to run EZ-USB from the external parallel memory. These modifications apply to both the CY3686 DVK and CY4611B DVK.

4.3.1 Board Modifications

1. The DVK boards come with a SRAM (U3 on CY3686 and U4 on CY4611B) on the address/data bus. To prevent bus contention, remove or disable this chip. Setting SRAM chip enable (Pin 5, RAMCE#) to logic 1 disables the chip.
2. The control signals to the SRAM are routed through the GAL22LV10C chip (U2), which translates the external parallel memory access signals before sending signals out to the SRAM. Remove GAL22LV10C from the socket or reprogram it to prevent these signals from going through the chip.
3. Make sure that the serial EEPROM on the I^{2}C bus is disconnected or disabled. To disable it, set SW2 to “No EEPROM” on CY3686 and remove jumper J22 on CY4611B.

5 Summary

This document described the boot options available in FX1/FX2LP, their significance, advantages, and disadvantages. Using this information, you will be able to choose the best boot option to use for your end application.

About the Author

Name:	Prajith Cheerakkoda
Title:	Applications Engineer Sr.

Document History

Document Title:AN50963 - EZ-USB FX1/FX2LP Boot Options

Document Number: 001-50963

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2631792	KFR / PYRS	01/07/2009	New application note.
*A	3022458	AASI	09/03/2010	Major content update
*B	3121045	AASI	12/27/2010	Added demonstration files for XP 64-bit environment
*C	3210023	AASI	03/30/2011	Added link to firmware download through host application code example.
*D	3443980	AASI	11/21/2011	Modified entire document to discuss all four methods of firmware download. Updated template according to current CY standards.
*E	3757038	PRJI	09/27/2012	Major content update.
*F	3848173	PRJI	12/20/2012	Updated Programming I2C EEPROM (Updated 2nd point only in page 4) under I2C Boot.
*G	4333400	PRJI	04/04/2014	Updated in new template.
*H	4609884	PRJI	12/29/2014	Sunset review
*I	4684656	AMDK	03/16/2015	Complete AN revamp. Updated documentation and attached project files to work with latest CyUSB3.sys driver.
*J	4825914	NIKL	07/06/2015	Updated template
*K	5732688	AESATP12	05/16/2017	Updated logo and copyright.
*L	6162207	HENA	05/01/2018	Added links to the app note webpage, USB High Speed Code examples webpage and USB 3.0 Product family webpage. Updated sections 2.2, 2,4, 3.1 and 3.2. Redrew Figure 6. Added files to be attached with the app note.

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 MCU

Cypress Developer Community

Community | 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