Vendor Command Design Guide for FX2LP™

AN45471

Author: Rich Peng
Associated Project: Yes
Associated Part Family: CY7C68013A/CY7C68014A/CY7C68015A/CY7C68016A
Software Version: None
Associated Application Notes: None

Application Note Abstract

Vendor commands are used to issue commands to a device, by which tasks unique to an application are accomplished. This application note demonstrates how you can quickly design USB vendor commands to perform specific features of products. In addition, using the Cypress CyConsole utility to issue vendor commands is also explained.

Introduction

Most designs can follow a USB-IF defined class to fit their product. You can find a suitable USB class and then implement the pre-defined class commands into your design. However, if none of the USB-IF classes fit the product design, then you must use a vendor command to achieve the specified function. This application note helps you understand and use vendor commands. An example code is provided to help create customized vendor commands.

Vendor Commands

Table 1. USB Device Request Format

Byte	Field	Meaning
0	bmRequestType	Request type
1	bRequest	Actual request
2	wValue	Varies by request
4	wIndex	Varies by request
6	wLength	Number of data bytes

Table 1 shows the standard USB device request format. Each request consists of eight bytes of data. Request commands always use the control endpoint EP0.

The bmRequestType byte specifies the direction of data flow, the type of request, and the recipient. The bmRequestType field also contains one of three different command types: Standard, Class, and Vendor. They are defined in bit 5 and bit 6 of the bmRequestType byte. Bit 7 is used to define the command direction, host to device (OUT), or device to host (IN), as shown in Table 2.

Table 2. bmRequest Format

Bits	Field	Values
7	Direction	0: Host to Device (OUT) 1: Device to Host (IN)
6..5	Type	0: Standard 1: Class 2: Vendor 3: Reserved
4..0	Recipient	0: Device 1: Interface 2: Endpoint 3: Other 4-31: Reserved

The type field is used to define the command as Standard, Class, or Vendor. Standard type commands are always used during a USB device enumeration with a USB host. This is defined in the USB 2.0 Specification. If an application can use a defined USB class, then you may use the commands defined in that class to implement the required device features.

Class commands are not discussed in this application note. For more information on class commands and how to implement them, visit the USB-IF website at http://www.usb.org/developers/docs.

This application note focuses on the following issues:

- Differences between vendor commands and other commands
- When to implement a vendor command
- Using vendor commands in applications

[+] Feedback

In Table 2 on page 1, the vendor is defined in the Type field. Vendor commands follow the same basic format as the Standard and Class commands. The difference is that a vendor command is user-defined. This is because it is not possible to define every device type command that could be used by a USB application. Some applications may require specific software or a customized driver. As a result, you must use user defined commands for the device to function as required.

Vendor commands must be created in the following situations:

- If you cannot find any predefined commands to fit your application
- To initialize another chip or to communicate with another chip with a USB controller. If there is a specific software request, you must create vendor commands to accomplish this.

The following section uses the Cypress CY3684 FX2LP™ Development Kit to define a vendor command and exercise an actual example. This information can guide you in creating unique vendor commands.

Vendor Command Example Code

The following materials are required to understand this application note and exercise the example code:

- CY3684 Development Kit
- CY3684 Development Software (CyConsole)
- Keil C (full version is required to compile this firmware)

The example code project attached with this application note is Vend_Ax. Unzip Vend_Ax.zip. The Vend_Ax folder containing a Source and Driver folders opens. This driver supports Microsoft Windows XP system for 32/64 bit and Vista 32 bit version running in the XP compatibility mode. Refer the source code to create your own vendor commands.

The firmware project contains the following files:

- dscr.a51
- fw.c
- eeprom.c
- vend_ax.c
- EZUSB.LIB
- USBJmpTb.OBJ

DSCR.A51

This file defines USB descriptors and contains all USB related definitions. In this file, you can make the required modifications to configure USB resources. Typical modifications include changing the VID/PID, Endpoint Configuration, Interface Class, and the Strings section.

FW.C

This file is called the USB Frameworks. All USB standard commands are processed inside this file. Code is included to handle USB standard command sets to allow the device to enumerate. Usually it is not necessary to modify the fw.c frameworks file. However, you can trace the firmware in this file to see how the device complies, detailed in Chapter 9: USB Device Framework of the USB Specification.

EEPROM.C

This file includes all EEPROM related subroutines. This file shows how to perform an EEPROM Read/Write and get page size information.

VEND_AX.C

This is the main file in this firmware example. All vendor commands are implemented in this file. You can refer this file to learn how vendor commands work. It is used as a starting point to create vendor commands. Ten vendor commands are defined in this file. Note that Cypress reserves vendor commands 0xa0~0xaf.

Creating a vendor command is a simple process. The first step is to define the vendor command. For example, to define GETI2C_TYPE as a 0xA5 vendor command, use the following code:

```
#define VR_GETI2C_TYPE 0xa5 // 8 or 16 byte address
```

Next, add the code to be executed by the vendor command in the DR_VendorCmnd routine.

```
BOOL DR_VendorCmnd(void)
{
// Add your vendor command process in this function.
}
```

The following example shows how a vendor command works:

```
BOOL DR_VendorCmnd(void)
{
switch(SETUPDAT[1])
{
        case VR_GETI2C_TYPE:
            *EP0BUF = DB_Addr;
            EP0BCH = 0;
            EP0BCL = 1;
            EP0CS |= bmHSNAK;
break;
}
return(FALSE); // no error; command handled OK
}
```

Here the SETUPDAT buffer maps to the USB Request format, as shown in Table 3 on page 3.

Vendor Type and Direction are defined in SETUPDAT[0]. GETI2C_TYPE is defined with Request Code 0xA5 as the vendor command. The Switch SETUPDAT[1] in DR_VendorCmnd is used to decode the command. The remaining SETUPDAT bytes are determined by the created vendor command.

[+] Feedback

Table 3. SETUPDAT Structure

Byte	Field	SETUPDAT
0	bmRequestType	SETUPDAT[0]
1	bRequest	SETUPDAT[1]
2	wValue	SETUPDAT[2] SETUPDAT[3]
4	wIndex	SETUPDAT[4] SETUPDAT[5]
6	wLength	SETUPDAT[6] SETUPDAT[7]

The following section demonstrates an actual vendor command using the Vend_Ax project included with the FX2LP Development Kit. A Cypress host PC utility is used to exercise and test the vendor command.

Vend_Ax Project Demo

This section describes how to prepare the FX2LP development board to run this project. There are two methods to load and run this code. The first method is to download the HEX file into the FX2LP's internal RAM directly. The second method is to program an .IIC image of the hex file into the EEPROM. This is loaded into the FX2LP internal RAM automatically when connected to the USB bus. The Cypress utility program CYConsole is always used, regardless of the method. This example uses the CY3684 FX2LP development board to demonstrate these methods.

HEX File Download

1. Move the SW2 switch on the FX2LP development board to the NO EEPROM position.
2. Connect the USB cable to the PC and the FX2LP development board.
3. Follow the "Found New Hardware" wizard to install the driver, if necessary. The device must be identified as "Cypress EZ-USB® FX2LP - EEPROM missing".
4. Run CYConsole. The device must have the status "Cypress EZ-USB FX2LP - EEPROM missing" in the select device display. This indicates that the CYConsole recognizes the FX2LP board. Click Options from the top menu of the CyConsole window and then choose the EZ-USB Interface. A new application window opens.
5. Click the Download button of the EZ-USB Interface window. Navigate to and choose vend_ax.hex from the \Vend_Ax\Source folder provided with this application note. The display window shows the process of the hex file download.
6. After the download, the FX2LP enumerates with a VID/PID of 04B4/1008, as defined in DSCR.A51. As a result, the host asks you to install the driver for the EZ-USB Vendor Command device. In the project folder that accompanies this application note, a \Vend_Ax\Driver folder is included containing the INF file matching the firmware VID/PID.
7. After installing the driver, the CyConsole device description changes to show the EZ-USB Vendor Command.
8. The vendor command can now be tested.

In this method, you must download the hex file every time the FX2LP is disconnected and reconnected to the host PC to use the Vend_Ax project.

The Vend_Ax Hex file located at \Vend_Ax\Source as the example attached within this app note.

IIC File Programming

1. Set switch SW2 to the NO EEPROM position on the FX2LP development board.
2. Connect a USB cable to the PC and the development board.
3. Install the driver, if needed, as described in step 3 of the section HEX File Download.
4. Run CYConsole. The device must be displayed as "Cypress EZ-USB FX2LP - EEPROM missing" in the select device display. This indicates that the CYConsole recognizes the FX2LP development board.
5. Click Options from the top menu of the CyConsole window, and then choose the EZ-USB Interface. A new application window opens. Move the SW2 switch on the FX2LP dev board to the EEPROM position and SW1 to the LARGE EEPROM position.
6. Click the Lg EEPROM button of the EZ-USB Interface window, navigate to and choose vend_ax.hex from the \Vend_Ax\Source folder provided with this application note. The display window shows the process of IIC file being programmed into the EEPROM and displays "Downloaded Successful" when completed.
7. Press the RESET button on the FX2LP development board or disconnect and reconnect the USB cable. The FX2LP enumerates with a VID/PID of 04B4/1008, as defined in DSCR.A51. The host asks you to install the driver for the EZ-USB Vendor Command device. In the project folder that accompanies this application note, a \Vend_Ax\Driver folder is included containing the INF file that matches the firmware VID/PID. Navigate to the provided folder when prompted during driver installation.
8. CyConsole shows the device as "EZ-USB Vendor Command".
9. The vendor command can now be tested.

In this method, the FX2LP development board always runs the Vendor Command firmware downloaded from the Large EEPROM. You are not required to reprogram the FX2LP until changes are made to the code and have to be tested. Follow the procedures discussed in this section to program the Large EEPROM and test the new code.

The Vend_Ax IIC file located at \Vend_Ax\Source as the example attached within this application note.

[+] Feedback

Vendor Command Testing

This project defines several vendor commands. The principle of operation for all vendor commands is the same. Therefore, as an example, the EEPROM read and write vendor command is exercised. The CyConsole utility is used to demonstrate testing a vendor command.

The IIC file programming method is used to program the FX2LP in the following demonstration. The FX2LP development board must have SW1 in the Large EEPROM position and SW2 in the EEPROM position.

The Large EEPROM Read and Write command is defined as 0xa9. You can choose the desired read or write length. The address to begin a read or write can also be selected.

Open CyConsole and select the EZ-USB Interface Window as described earlier.

The CyConsole has a Vend Req button and other control fields that allow you to key in the value of the defined vendor command inside the firmware. The Req field accepts the user input code that identifies the vendor command. The length field indicates the desired length of a Read or Write. The Dir field indicates the direction of the command, data out or data in. The Hex Bytes field allows you to enter hex bytes that are used as the data to be transferred in an OUT vendor command. The Value and Index fields are dependent on the created vendor command structure.

Figure 1. EEPROM Read Vendor Command

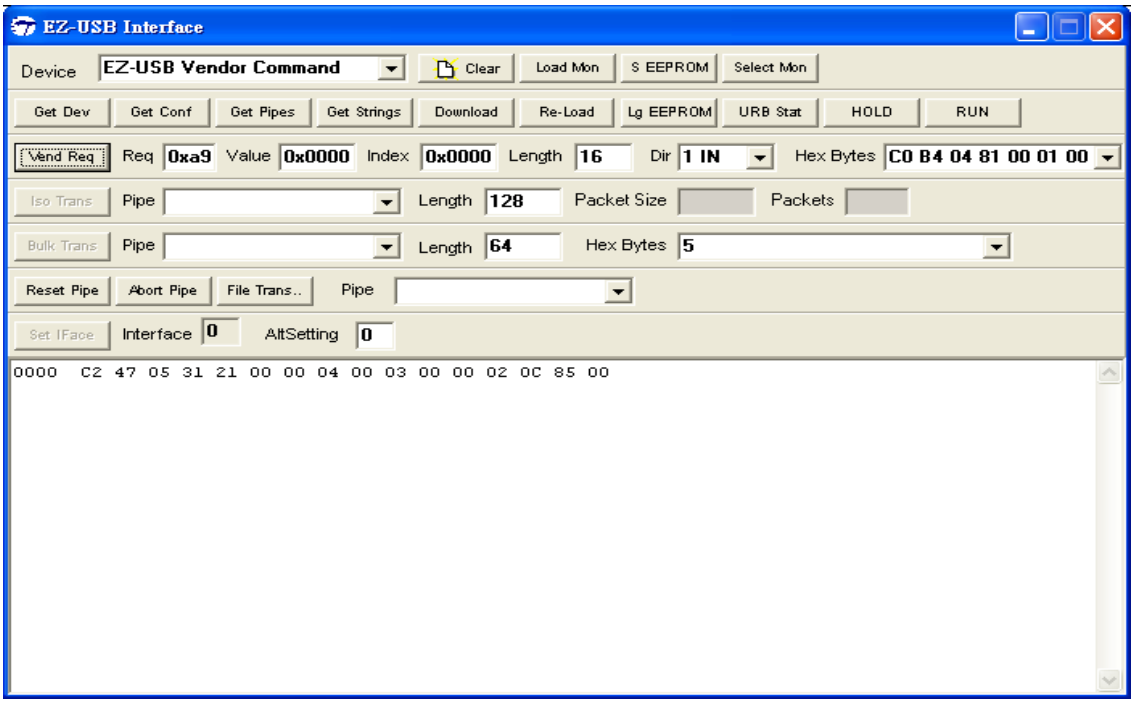

Figure 1 shows how to execute a vendor command to read from an EERPOM. First, the vendor command request code of 0xA9 is entered into the Req field, the Length field is set to 16 bytes, and the Dir field is set to 1 IN. The Value and Index fields are left at their default values of 0x0000.

[+] Feedback

The vendor command as configured earlier indicates that the first 16 bytes of a Large EEPROM are to be read. Pressing the Vend Req button executes the vendor command and the information retrieved from the EEPROM is displayed in the window, as shown in the following figure.

.

Figure 2. EEPROM Write Vendor Command

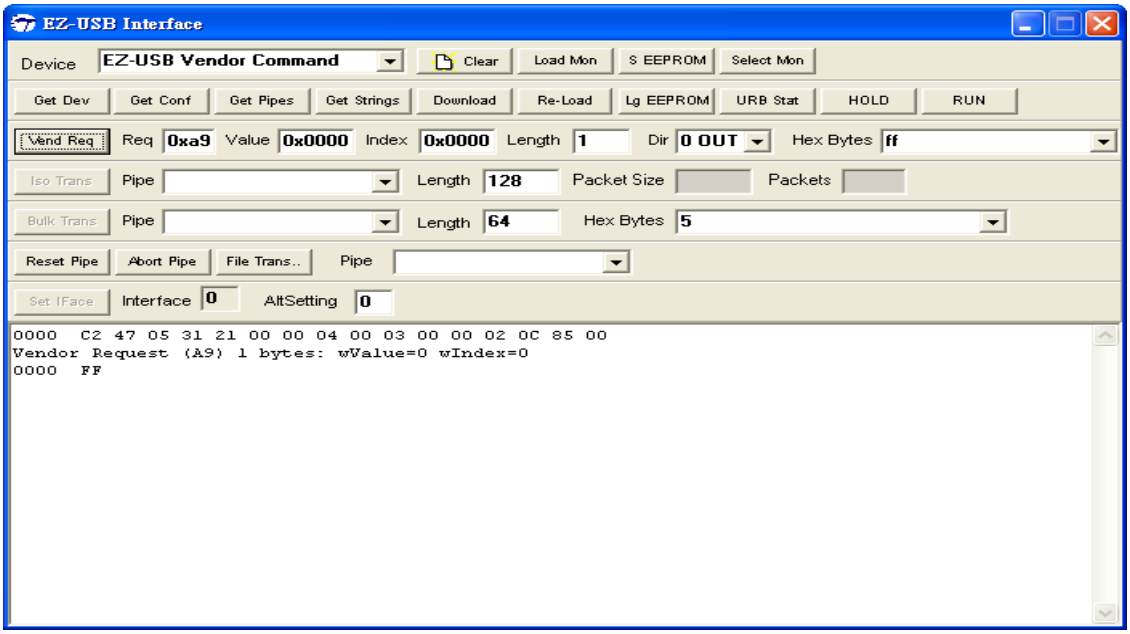

Figure 2 shows how to execute a vendor command to write to an EEPROM. First, the vendor command request code of 0xA9 is entered into the Req field, the Length field is set to 1 byte and the Dir field is set to 0 OUT. The Value and Index fields are left at their default values of 0x0000. The vendor command as configured earlier indicates that 1 byte must be written to the first address location of a Large EEPROM. Pressing the Vend Req button executes the vendor command and the information written to the EEPROM is displayed in the window.

Note that the FX2LP only recognizes an EEPROM first byte of C0 or C2. Therefore, resetting or reconnecting the development board to a host PC, after writing “FF” to the first byte of an EEPROM, causes the FX2LP to enumerate in the No EEPROM configuration.

Resources and Additional Information

- For more various Windows OS system driver support, you can download the SuiteUSB3.4 from Cypress Website, this driver package cover the 32 and 64 bit of Windows XP and Vista and 7.
- You also can find very good reference resource and various examples of USB Vendor Command Design and HID Class Usage in Jan Axelson’s Lakeview Research website.
- The very useful book to reference in USB Vendor Command Design – USB Complete by Jan Axelson, it describes full knowledge about USB technology and provide the various examples in firmware and software as well.

Conclusion

You can create unique vendor commands with the FX2LP chip. The example firmware provided with this application note also allows testing the Cypress FX2LP chip and using the I2C bus of FX2LP to control an EEPROM. The CyConsole provided with the FX2LP development kit is also used to demonstrate the testing and execution of a vendor command.

About the Author

Name: Rich Peng

Title: Application Engineer Sr Staff

Contact: lip@cypress.com

[+] Feedback

Document History

Document Title: Vendor Command Design Guide for FX2LP™ - AN45471

Document Number: 001-45471

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2505120	LIP	01/26/2009	New application note.
*A	3155612	LIP	01/27/2011	Added Resource and Additional Information sections. Updated links.

EZ-USB is a registered trademark of Cypress Semiconductor Corp. FX2LP is a trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein may be the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/

© Cypress Semiconductor Corporation, 2008-2011. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

[+] Feedback