

AN15813

Author: Rich Peng

Associated Project: No

Associated Part Family: CY7C68013A, CY7C68014A, CY7C68015A, CY7C68016A, CY7C64713

[GET FREE SAMPLES HERE](#)

Software Version: None

Associated Application Notes: None

Application Note Abstract

This application note describes the purpose and methods of monitoring VBUS from the upstream connector using the EZ-USB FX2LP™.

Introduction

A USB-IF specification requirement is to never drive the USB pins when they are not connected to the bus; they may only be driven when VBUS is present. Refer to section 7.2.1 of the *Universal Serial Bus Specification*, Revision 2.0. Violating this requirement may cause several failures to the system. One documented failure is reset problems on the upstream devices, which in turn may result in PC cold boot problems, or it causes hubs to fail to enumerate downstream devices. Other failures include the inability to properly resume from a suspend state and forcing other enumerated devices off the bus.

The method used to detect if a device is connected to the bus is to monitor the VBUS signal. A bus-powered design does not require VBUS monitoring because it cannot drive the bus when disconnected. A self-powered design uses the power input to detect connection; it knows when to assert the D+ or D- pull-up resistor or when it must disconnect. This application note is intended to help the designer understand the methods of monitoring the VBUS inside an application.

When the application is for a self-powered device, VBUS monitoring is very important. Do not drive the D+/D- bus before VBUS is present or after it is removed. This means that the D+/D- signal is only driven while the USB device is electronically connected to the USB host.

VBUS Function Overview

Figure 1. USB Physical Connection

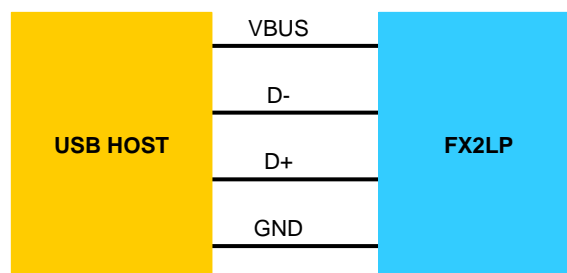


Figure 1 illustrates the physical USB connections between device and host. The FX2LP is the device in the system; it is never the host. Based on this concept, it is easy to understand how VBUS monitoring works in the USB system.

According to the USB specification, a USB device has two power configurations: bus-powered and self-powered. In the bus-powered device, the device power relies on the USB host to provide a limited amount of power over the cable. Because VBUS provides power for the device to function, the D+/D- is not driven before power is provided to the device.

For a self-powered device design, make sure that the device monitors VBUS and disables or enables the pull-up on the data line accordingly. In the self-powered device, the power is typically provided by a power adaptor or a battery, not the VBUS. What problems can possibly happen in this type of design? The USB specification states "Devices may not provide power to the pull-up resistor on D+/D- unless VBUS is present (see section 7.1.5 of *USB 2.0 Specification*). When VBUS is removed, the device must remove power from the D+/D- pull-up resistor within 10 seconds". For more details refer to the *USB 2.0 Specification*, section 7.2.1.

After the reset period, the FX2LP device comes up connected. This means when the FX2LP comes out of reset, D+ is driven. Per the USB 2.0 Specification, a maximum debounce interval of 100 ms is allowed to ensure that all connections are stable before any requests are sent to the device. If VBUS is not processed carefully, the design may fail both the USB-IF Compliance Testing and the Microsoft Windows WHQL and DTM tests.

The USB-IF verifies compliance by checking the back-voltage on the D+, the D-, and VBUS pins into a 15K ohm load. Refer to section F (Back Voltage Testing) of the *Universal Serial Bus Implementers Forum Full and Low Speed Electrical and Interoperability Compliance Test Procedure*, which is available on <http://www.usb.org>.

This document explains why a design must comply with the back-voltage test. It also explains the errors that occur if the device does drive D+/D- prior to seeing VBUS. Failures that occur are as follows:

- The PC fails to cold boot due to the back voltage affecting the motherboard reset sequence.
- The hub fails to enumerate downstream devices due to reset anomalies.
- The motherboard fails to properly resume from a suspend state.
- Introduction of a device or hub forces one or more upstream devices off the bus.

These are only a few of the reported failures or anomalies.

From this brief introduction, it is clear why VBUS must be monitored if the product is a self-powered device. If VBUS is not monitored in this type of design, then more effort is spent in debugging the board, resulting in wasted time and money. The suggestions presented here must be considered at the very beginning of a USB self-powered device design.

VBUS Monitor Methodology

When designing with the FX2LP as a self-powered device, there are two different design modes. One mode is a standalone design with the FX2LP and the other uses a coprocessor to control the FX2LP. Regardless of the mode used, the VBUS monitor feature must be implemented in the design. Since the FX2LP is a programmable chip, the firmware implements the VBUS monitor feature. Inside the FX2LP, the DISCON bit controls the D+ pull-up resistor to enable or disable the connection. Use this bit to control the connection when monitoring VBUS.

Figure 2 shows the standalone mode and Figure 3 illustrates the coprocessor mode. VBUS is monitored using a GPIO pin or the WAKEUP pin in standalone mode, while only the GPIO pin approach is used in a coprocessor design. In either case, the DISCON bit is enabled or disabled to match the VBUS status.

Figure 2. Standalone Mode of FX2LP Design

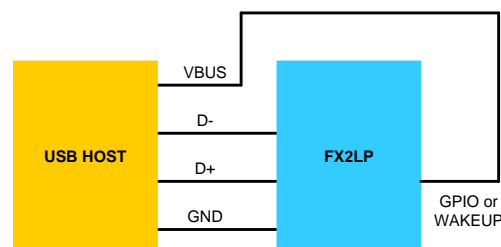
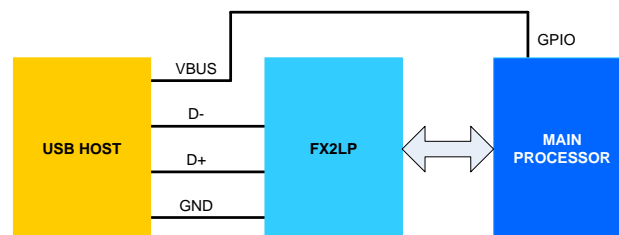
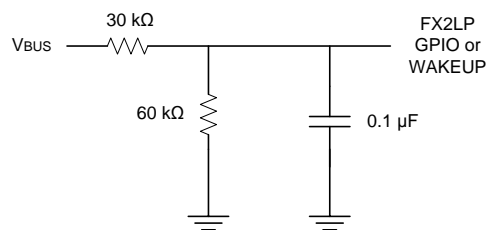


Figure 3. Coprocessor Mode of FX2LP Design



VBUS is connected through a resistive bridge and the firmware monitors the status of the selected line. The resistive bridge adds a bleed path for the VBUS signal. The monitoring pin (GPIO or WAKEUP) consumes less than 10 μA ; so when the cable is unplugged, the input stays high for some time. A path of approximately 100K ohms causes this charge to dissipate within a short period of time. Typically, a 30K ohm resistor between the VBUS pin and the monitor pin with a 0.1 μF capacitor in parallel with a 60K ohm resistor from the monitor pin to ground accomplishes this, as shown in Figure 4.

Figure 4. Sample Schematic



Firmware for GPIO Approach

The firmware to monitor the GPIO status is placed in the foreground loop (TD_Poll). This loop executes frequently and keys in on this as an event trigger. To accomplish this task, the firmware must configure any GPIO port pin as an input and read the status in the loop. TD_Poll firmware has something similar to the example in Code 1.

This code shows how to implement the VBUS monitor in the FX2LP standalone mode design. In some designs, other items may need to be considered such as in ATA applications (see the firmware in the CY4611 RDK).

If the application is in coprocessor mode, the connections are almost the same. The only difference is to make the VBUS detection with the main processor chip GPIO, and follow the firmware design flow implemented in the main

chip side. Next, send the command to control the FX2LP DISCON bit through the Slave FIFO interface. The application in coprocessor mode then has the same behavior as the standalone mode.

Code 1. Configure GPIO Port Pin for VBUS Monitor

```
if ( !(IOA & 0x80) ) // if VBUS not present (using a PORTA.7 pin for example)
{
    // take action and set the DISCON bit to disconnect FX2LP from USB
    USBCS |= bmDISCON; // Setting DISCON bit disables the pull up on D+
}
else
{
    USBCS &= ~bmDISCON; // Clear DISCON
    // Proceed to do other tasks
}
}
```

Firmware for WAKEUP Approach

The WAKEUP pin is polled to determine a change in VBUS status. Add firmware in the foreground loop (TD_Poll) to check the status of the wakeup pin and disconnect from the USB when VBUS is removed. The wakeup control register (WAKEUPCS) is written to clear

the built in latch and allow an accurate read of the current pin state. Code 2 gives an example of testing the WAKEUP pin. When VBUS is removed, disconnect by setting the DISCON bit. Perform any necessary housekeeping, then suspend and wait for VBUS to reconnect. When VBUS is reconnected, clear DISCON and resume normal operations.

Code 2. Configure WAKEUP Pin for VBUS Monitor

```
WAKEUPCS = bmWU | bmDPEN | bmWUEN; // clear built-in latch
WAKEUPCS = bmWU | bmDPEN | bmWUEN; // write again in case polarity was modified

if(WAKEUPCS & bmWU)
{
    // when WU low:
    USBCS |= bmDISCON; // disconnect
    // application specific code // shutdown normal operations
    EA = 0; // disable interrupts
    EZUSB_Delay(30); // debounce delay

    WAKEUPCS = bmWU | bmWUPOL | bmWUEN; // enable WU active high (wait for VBUS ONLY)
    EZUSB_Susp(); // place processor in idle mode

    WAKEUPCS = bmWU | bmDPEN | bmWUEN; // enable WU active low and D+, too
    USBCS &= ~bmDISCON; // connect
    // application specific code // restart normal operations
    EA = 1; // enable interrupts
    EZUSB_Delay(30); // debounce delay
}
}
```

Summary

This application note provides two approaches to monitor VBUS for self-powered devices. A GPIO pin or WAKEUP pin is used to detect a change in VBUS status. It is critical to monitor this signal to eliminate any back-voltage problems and to create a USB compliant device.

About the Author

Name: Rich Peng
Title: Application Engineer Sr Staff
Contact: lip@cypress.com
+886-27255515#207

All 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. 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.