Guide to a Successful EZ-USB® FX2LP™ Hardware Design

Author: Rama Sai Krishna Vakkantula

Associated Project: Yes
Associated Part Family: CY7C68013A/14/15/16A
Software Version: NA
Related Application Note: AN65209
To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN15456.

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.

Building a USB device requires careful attention to design details beyond the USB specification. This application note discusses design topics common to any USB device, focusing on Cypress's EZ-USB® FX2LP™ devices. The information presented also applies to the older FX1 device and to USB devices in general. The note concludes with a schematic review checklist to help you make any USB hardware design a success, and a description of Cypress software that helps with device checkout.

Contents

1 Introduction

USB 2.0 brought a significant increase in bandwidth over the 1.1 specification. Offering 40x more bandwidth, it increased the importance of a good PCB design and a careful selection of components surrounding USB chips such as Cypress's FX2LP. This application note presents a host of USB design topics that apply to any USB device at any speed, but especially to the higher speeds of USB 2.0. Although the discussions are specific to Cypress devices, they also should be helpful in any USB peripheral design.

USB provides power in the cable, making self-contained devices practical. The Powering USB Devices section discusses how USB devices are powered and the special considerations that apply to self-powered devices.

USB is hot-pluggable, making USB chip reset circuits critical. The USB Reset Circuits section describes why the traditional RC reset circuit is inadequate and recommends a superior solution.

Cypress USB devices use high-speed on-chip RAM for program storage. This RAM can be bootloaded over USB or loaded from an external EEPROM at power on. The EEPROM Use Considerations section describes how to implement a successful EEPROM design that satisfies USB power and hot-plug requirements.

Every USB chip requires a precise clock, provided by a crystal. Recent developments in lower cost resonators (with included load capacitors) make them usable as USB clock sources. The Selecting a Resonator for USB section discusses these resonators and provides relevant specifications.

Good printed circuit board layout is critical to a successful USB peripheral. The High-Speed USB PCB Layout Recommendations section gives recommendations, including BGA requirements. Use the Schematic Review Checklist to help confirm that design requirements are met.

Finally, the Cypress Driver and USB Control Center section shows how to install the Cypress Windows driver and a utility program called USB Control Center, which eases the task of bringing up an FX2LP-based board for the first time.

2 Powering USB Devices

A USB device can be powered in three different ways:

- Bus powered: The device derives its power from the USB cable VBUS wire. This is by far the most common method of powering USB devices such as thumb drives, mice, keyboards, and newer disk drives.
- Self-powered: Some USB peripherals such as disk drives exceed the power provided by the cable. These peripherals use external power supplies, usually in the form of wall warts.
- Mixed power: A USB device can be both self-powered and bus powered. These devices are rare and usually require multiple power supplies.

This section concentrates on the first two powering methods since the third is a combination of them.

2.1 USB Power Specifications

Several important USB specifications guide the design of power circuitry. This section summarizes the important specs.

2.1.1 VBUS Power

A USB 2.0 host supplies two power levels on the 5-V VBUS wire: 100 milliamps at plug-in and up to 500 milliamps during operation. The USB specification gives a range for VBUS of 4.4 V to 5.25 V for a 100-mA load, and 4.75 V to 5.25 V for a 500-mA load. USB peripherals typically regulate this voltage down to 3.3 V to provide the cleanest power to the device.

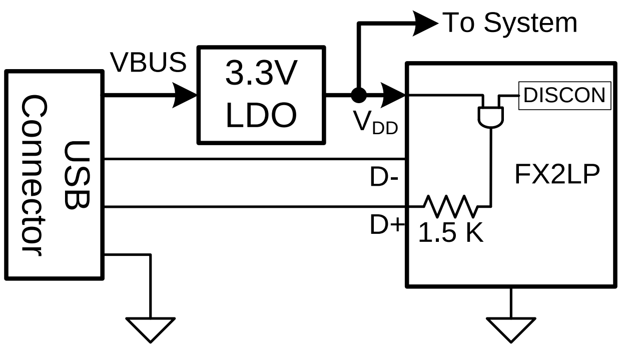
A bus-powered hub supports 100 mA to downstream ports in all cases. An externally powered hub can supply up to 500 mA to each of its downstream ports.

2.1.2 USB Power at Plug-in

A USB host regards a USB device that has just plugged in as unconfigured. By spec, an unconfigured USB device cannot draw more than 100 mA from the VBUS wire. During the enumeration process, the host determines the device's power requirements. If the host can meet the requirements, it moves the device to the configured state and supplies the required power. This helps to make USB safe—for example if a 400-mA device plugs into a bus-powered hub, the host does not configure or operate the device because the bus-powered hub supports only 100-mA downstream devices.

A full-speed or high-speed USB device indicates its presence to the host at plug-in by pulling up the D+ signal line with a 1.5-kΩ resistor to a voltage source between 3.0 V and 3.6 V. By USB spec, this resistor must never be powered if VBUS is not present. This is a consideration only for self-powered devices, since in a bus-powered device, VBUS itself supplies the pull-up voltage, usually through a 5-V to 3.3-V voltage regulator. So, bus-powered devices automatically meet the VBUS present spec (Figure 1).

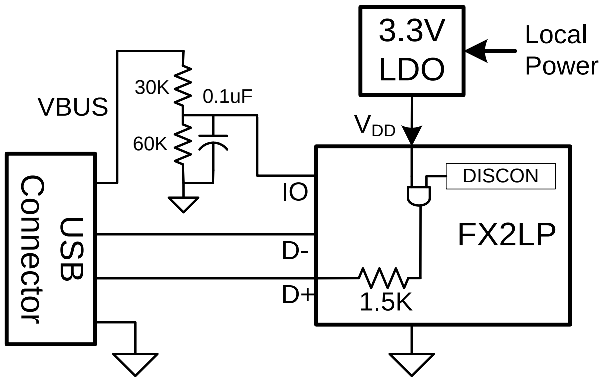
Figure 1. Bus-Powered Device



As Figure 1 illustrates, the VBUS wire supplies 5-V power to a 3.3-V regulator, which powers the FX2LP chip and other device circuits. FX2LP supplies the 3.3-V pull-up voltage using its DISCON bit. The absence of VBUS removes FX2LP power and therefore the pull-up resistor voltage.

A self-powered device is another matter, requiring careful consideration (Figure 2).

Figure 2. Self-Powered Device



If the self-powered device receives local power before USB attachment or during the attachment, or if the host turns off VBUS, it could power the pull-up resistor in the absence of VBUS, violating the USB spec. This is a specific USB compliance test item—the compliance test turns off VBUS and checks for any voltage on the D+ line.

As Figure 2 shows, the VBUS wire no longer supplies power, instead functioning as a signal input. By sensing the VBUS signal, FX2LP firmware can ensure that its DISCON bit powers the 1.5-kΩ D+ pull-up resistor only when VBUS power is present. The resistor divider drops the 5-V VBUS voltage to 3.3 V to make it compatible with the FX2LP I/O pin voltage levels. The resistor to ground also pulls the I/O pin low when the VBUS signal floats due to a mechanical disconnect. The capacitor provides transient suppression for connect and disconnect. Some FX2LP systems may use a coprocessor. In these systems, either the coprocessor or FX2LP may sense the VBUS signal, as long as the resulting action is to control the DISCON bit properly.

In most self-powered designs, any general-purpose input pin can be used to sense VBUS. For the fastest firmware response, you can use an FX2LP interrupt pin. If you want to put the FX2LP into a low-power (sleep) state until VBUS is reapplied, then you can use the FX2LP WAKEUP input pin to sense VBUS. By programming the WAKEUP pin to be active high, you can use the VBUS signal as the trigger to wake up the FX2LP MCU and resume USB signaling.

Note: Two FX2 variants, AT2LP and HX2LP, provide a dedicated VBUS sense pin.

2.1.3 GPIO VBUS Sense Example Code

Figure 3 is example C code to test the PORTA.7 I/O pin for the presence of VBUS and to set the bmDISCON bit if it is present.

Figure 3. C Code Using I/O Pin to Sense VBUS

```
if ( !(IOA & 0x80) )  // Test VBUS signal (using a PORTA.7 pin for example)
{
  USBCS |= bmDISCON;  // VBUS not present: disable D+ pullup
}
else
{
   USBCS &= ~bmDISCON; // VBUS is present: enable D+ pullup
}
```

2.1.4 WAKEUP Pin VBUS Sense Example Code

Figure 4 is example C code using the WAKEUP pin to detect the presence of VBUS.

Figure 4. C Code Using WAKEUP Pin to Sense VBUS

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

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

// enable WU active high (wait for VBUS ONLY)
WAKEUPCS = bmWU | bmWUPOL | bmWUEN;
// place processor in idle mode
// Code execution resumes here when WU pin goes active (VBUS detected)
EZUSB_Susp();
// enable WU active low and D+, too
WAKEUPCS = bmWU | bmDPEN | bmWUEN;
// connect
// application specific code           // restart normal operations
USBCS &= ~bmDISCON;
// enable interrupts
EA = 1;
// debounce delay
EZUSB_Delay(30);
}
```

The EZUSB_Susp() statement puts FX2LP into a low-power state by stopping its oscillator. Asserting the WAKEUP pin restarts the oscillator, waits for the clock PLL to stabilize, and then activates the WAKEUP interrupt service routine (ISR). The ISR clears the interrupt request bit and then executes a "reti" instruction to resume code execution at the instruction following the one that put FX2LP into the IDLE state.

If your code is based on Cypress's USB Firmware Frameworks, the code to enable and service the WAKEUP interrupt is written for you in *fw.c*, as shown in Figure 5. Because the resume_isr function is declared as an ISR, it automatically executes the reti instruction on exit.

Figure 5. WAKEUP Interrupt Code in fw.c

```
EZUSB_ENABLE_RSMIRQ();  // Enable Wake-up interrupt
.....
// Wake-up interrupt handler
Void resume_isr(void) interrupt WKUP_VECT
{
   EZUSB_CLEAR_RSMIRQ();
}
```

2.2 USB Plug-in Timing

The USB specification imposes timing requirements on USB power up and RESET signaling.

2.2.1 D+ Pull-Up

A device must pull up D+ within 100 milliseconds of connection. The FX2LP default behavior is to power up with the DISCON bit cleared, automatically connecting the pull-up resistor and satisfying this requirement. (DISCON is bit 3 of the USBCS register.) FX2LP includes the option to override this behavior by setting a bit in the boot EEPROM.

FX2LP, like all EZ-USB family parts, uses volatile RAM for program storage. At power on, FX2LP can come up three different ways:

1. As a default USB device that enumerates with a fixed profile, characterized by vendor ID VID=0x04B4, denoting Cypress, and product ID PID=0x8613, denoting FX2LP. This default USB device contains circuitry to accept firmware downloads over USB, load the firmware into internal RAM, and execute the firmware code. Cypress provides Windows tools that bind a driver to this profile to download firmware over USB (see the Cypress Driver and USB Control Center section).
2. As in number 1, but with the option for the user to supply custom VID and PID values. The same default USB firmware loader device exists, but with non-Cypress VID and PID. This allows optional vendor customization of the PC bootloader. A small I^2C EEPROM supplies six bytes of ID information, plus a configuration byte that allows the DISCON bit default to be 1 (disconnected) instead of the 0 (connected) default value.
3. A full firmware image can be loaded from a large (64-KB) I^2C EEPROM. As in number 2, the first eight bytes contain a configuration byte, in which the DISCON pin power-up value can be changed.

Most designs keep the DISCON default to automatically connect to USB when plugged in. This easily meets the 100-millisecond time limit to connect. However, if your design overrides the DISCON default value of 0 using the EEPROM configuration byte, FX2LP powers up disconnected from USB. It is now the responsibility of your firmware to meet the 100-millisecond deadline to set DISCON=0. If you use method number 3, the firmware download time is added to the time your code takes to set DISCON=0. For this reason, you should set the FX2LP to its maximum I^2C clock rate of 400 kHz using another bit in the EEPROM configuration byte. Then once the downloaded code starts executing, be sure to set DISCON=0 very soon after coming out of MCU reset.

2.2.2 Bus Reset

A USB device must respond to a USB bus reset 100 milliseconds after connecting its D+ pull-up. FX2LP automatically meets this requirement because its internal circuitry handles a bus reset, requiring no firmware.

2.2.3 Other USB Timing

Once a USB device is operational, it must respond to host SETUP packets without a data stage within 50 milliseconds. Single-stage SETUP packets include the following device requests:

- SET_ADDRESS
- SET_CONFIGURATION
- SET_FEATURE
- CLEAR_FEATURE
- SET_INTERFACE

SET_ADDRESS is handled in FX2LP hardware. The other requests involve simply setting and clearing internal status bits, so the firmware response can be very quick. Normally the TD_Poll() function of the Cypress USB Firmware Frameworks checks for SETUP packets in its endless main loop. If your code takes excessive time away from this loop, a “Setup Data Available” interrupt request can minimize the SETUP response time.

SETUP packets with data stages can take up to 500 milliseconds to respond, which is normally plenty of time to retrieve the requested data such as USB descriptors.

2.3 MCU Power Conservation

The main FX2LP power consumers are the MCU and the general-purpose parallel interface (GPIF) unit. The MCU can be put into an idle state and restarted with a WAKEUP interrupt as previously described. If your design does not use the GPIF, do not configure the IFCONFIG register (0xE601). This saves the power used by running the GPIF at 48 MHz.

The MCU can run at three clock rates: 12 MHz, 24 MHz, and 48 MHz. The power-on default is 12 MHz, but most designs use the 48-MHz clock rate to ensure a speedy response to USB events. The MCU clock rate can be changed at any time.

2.4 USB Power Conservation

A USB host puts a device into a low-power standby state by suspending bus activity for 3 milliseconds. A USB device responds by entering a low-power state and monitoring the bus for activity to resume operation. FX2LP has interrupts for SUSPEND and RESUME. When SUSPEND is detected, the MCU does any required internal housekeeping, such as turning off power to peripheral units, and then enters its idle, or SUSPEND, state. The MCU can also command an FX2LP SUSPEND state independent of USB bus signaling. This is an important feature when using the WU2 pin for periodic wakeups.

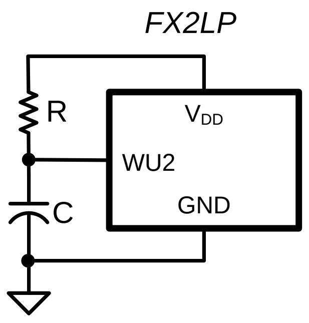
FX2LP can exit the SUSPEND state using three methods. Each method has its own enable bit.

1. The USB host resumes signaling.
2. The WAKEUP (WU) pin is asserted.
3. The secondary WAKEUP pin (WU2) is asserted.

The secondary WAKEUP (WU2) pin has special internal hardware. The WU2 pin is simultaneously a wakeup source, restarting the FX2LP oscillator, and a general-purpose output pin. If an RC network is connected, as in Figure 6, and the WU2 pin is programmed to be active high, FX2LP can periodically discharge the capacitor, enter the SUSPEND state, and be reawakened when the capacitor charges to approximately 2 V.

WU2 polarity is set to active high by setting the WU2POL bit in the WAKEUPCS (Wakeup Control and Status) register. The associated output bit is PORTA bit 3 (PA3). Set PA3 to zero, and set OEA.3 (Output Enable) to 1 to drive the WU2 pin low and discharge the capacitor. Then clear OEA to float the WU2 pin and allow the capacitor to charge.

Figure 6. WU2 Pin Can Trigger Periodic WAKEUP Events



3 USB Reset Circuits

Much of the popularity of USB is due to the fact that the USB cable supplies device power, enabling bus-powered devices. Even better, USB is “hot pluggable,” meaning that USB devices can be attached and detached from powered PCs without incident. However, the hot plug feature places special demands on the circuit used to reset a USB chip.

3.1 The Old Way

The RC network in Figure 7 is a very common reset circuit. Figure 8 illustrates a problem with this simple circuit for a bus-powered device (the overwhelming majority of USB devices). The waveforms show voltage waveforms as a USB device is connected and then disconnected and reconnected in quick succession. Chip power is red, the chip reset threshold voltage is blue, and the chip's active low reset input voltage is green. For RESET pin voltages below the blue line, the USB chip is held in reset, and above the blue line the chip is un-reset and operational.

Figure 7. Simple RC Reset Circuit

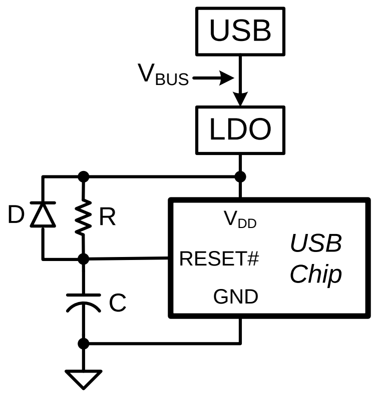
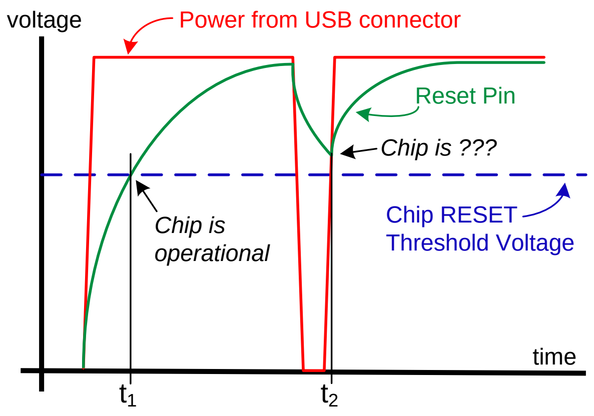


Figure 8. Worst-Case Waveforms for the Simple RC Reset Circuit



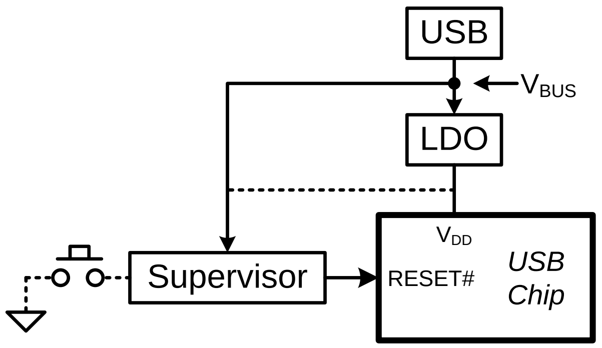
At power on, the capacitor is discharged to 0 V, holding the USB chip in RESET. The USB chip receives power from a regulator that converts the 5-V bus power from the USB connector to the 3.3-V V_{DD} pin. At plug-in, the LDO supplies chip power and begins charging the capacitor through the resistor. When the capacitor charges to the reset voltage threshold, the USB chip comes out of RESET at t_1 and starts operating. The critical concept here is that all internal chip circuits must be powered and stabilized before reset is released, which is true for any chip.

When the USB device disconnects, the chip loses power, but the capacitor is still charged to the chip's power supply voltage, slowly discharging through the resistor. What if the USB device is immediately replugged? If the capacitor has not discharged below the reset threshold, the chip receives power with the RESET pin voltage too high to reset the chip at t_2. A band-aid for this problem is to add the diode in Figure 7 to speed up the capacitor voltage discharge at power down, but this only shortens the critical disconnect-reconnect time. The RC discharge method depends on the LDO output going to 0 V when its input is disconnected, which may depend on the particular LDO and its external components such as bulk capacitors. The bottom line is that this circuit cannot guarantee a reliable reset under all circumstances, in particular during a quick USB disconnect-connect event.

3.2 A Better Way

The solution is to use an external chip called a power supervisor, also known as a power-on reset (POR) chip or power management IC (PMIC). These chips monitor a power supply voltage and supply a predictable RESET signal to the USB chip, as shown in Figure 9. They are designed to operate reliably down to voltages as low as 1 V. They use a precision comparator and incorporate an internal time delay to ensure that RESET is held active long enough after power on to guarantee proper chip operation. Some supervisors include a watchdog timer, which an MCU pulses every so often (such as every second) to keep the RESET signal from asserting. Another supervisor option is an input pin that allows a RESET pushbutton also to reset the MCU. Supervisor RESET outputs are available as active high, active low, or open drain. These chips are plentiful and low cost, in the 50-cent range. One popular web distributor lists 43,196 entries in the category “PMIC supervisors.”

Figure 9. Ideal RESET Solution



The supervisor reset point should be referenced to the highest voltage rail in the system to ensure that FX2LP is held in reset while *any* device in the system has not reached its minimum supply voltage. (See the EEPROM Use Considerations section for more information.) The worst-case USB VBUS voltage is 4.35 V, which applies when a device is plugged into a bus-powered hub (USB 2.0 spec, Figure 7-47). Therefore, a supervisor that monitors VBUS using a reset voltage of approximately 3.5 V to –4.0 V would be a reasonable choice. Whichever voltage you choose, be sure to test it thoroughly (see the Testing section).

If it is impractical to monitor VBUS (for example, in a self-powered device), the supervisor can monitor the 3.3-V rail. Table 1 shows the key supervisor specifications for FX2LP when monitoring the 3.3-V supply. For example, the TPS3820-33 from Texas Instruments is designed to monitor a 3.3-V supply, making it suitable for an FX2LP reset circuit. Its RESET output becomes active (low) at 1.1 V, and it asserts RESET until the 3.3-V supply voltage reaches 2.93 V. Thereafter, it keeps RESET asserted for at least 5 milliseconds. This guarantees that a quick USB disconnect-then-connect sequence triggers the internal comparator and provides the necessary RESET signal until the supply voltage again stabilizes.

Table 1. Key Supervisor Specifications

Specification	**Value**
Supply Voltage to Monitor	3.3 V
Reset Threshold Voltage (typ)	2.93 V
Reset Active Time (min)	5 ms

3.3 Testing

Any USB reset circuit should be thoroughly tested. The following tests are recommended:

- Cold power up, plugged into USB
- Cold power up, unplugged from USB
- Hibernate/resume, plugged into USB
- Power cycle, plugged into USB
- Power cycle, unplugged from USB

- Power cycle, plugged into five tiers of hubs (connect five hubs together and plug into the furthest one from the host)
- Unplug/replug the five tiers of hubs

Note After an erratic RESET# signal, FX2LP exits the reset mode and immediately starts to communicate with the EEPROM. In this event, the SCL/SDA may go LOW or HIGH intermittently. When RESET# is asserted again (due to the presence of the erratic signal), the SCL/SDA remains LOW and FX2LP does not enumerate. As a result, when these lines are externally forced HIGH, FX2LP resumes as expected.

Cypress recommends that you connect the reset circuitry of the FX2LP RESET# pin to the EEPROM RESET/POWER pin. This resets the EEPROM each time there is a reset on FX2LP, which in turn resets the SCL/SDA lines to HIGH, eliminating any erroneous state on these lines.

4 EEPROM Use Considerations

The EZ-USB family of devices uses internal RAM for program storage. This RAM can be loaded at power on using the USB cable, or from an I^{2}C EEPROM attached to the FX2LP chip. The EEPROM bootloading method requires special attention to the supply voltages as power is applied and removed. Under supply voltage transient conditions, the FX2LP core may be operational while the EEPROM is not. I^{2}C traffic from FX2LP in a voltage transient condition can corrupt the EEPROM.

This section discusses four methods to protect against this type of spurious activity. While the focus is on EEPROM devices, the same principles can be applied to other peripheral devices that are susceptible to the same conditions. The four methods, in order of preference, are:

1. Use an external power supervisor chip.
2. Use a GPIO pin to write-protect the EEPROM.
3. Permanently write-protect the EEPROM.
4. Control power supply ramp-down with a bleed resistor.

4.1 Use a Power Supervisor Chip

As the USB Reset Circuits section recommends, an external power supervisor chip is the best way to provide a reliable FX2LP reset. It inherently solves the EEPROM low-voltage issue, since an FX2LP in reset cannot initiate I^{2}C traffic to the EEPROM. When choosing the supervisor reset voltage, be sure to set it to a voltage compatible with all circuits in the system. Newer EEPROMs operate at low voltages; for example, most 64-KB EEPROMs operate down to 2.5 V, so the 2.93-V FX2LP threshold voltage recommended in the USB Reset Circuits section is a good choice for self-powered devices. For bus-powered devices, the VBUS voltage is a good one to monitor since it is guaranteed to be valid down to 4.35 V, so a reset threshold slightly below this (for example, 20 percent) guarantees operation for FX2LP and EEPROMs.

4.2 Connect a GPIO Pin to the EEPROM WP Pin

EEPROMs have a WP (Write Protect) pin to guard against inadvertent writes. By pulling this pin to the voltage level that asserts the WP function, FX2LP firmware can selectively enable writes. For example, an active high WP pin should be connected to the 3.3-V supply through a 10K (typical) resistor and to an FX2LP GPIO pin. Because the FX2LP powers up with its GPIO pins floating (OEA-OEE bits low), the logic high supplied by the pull-up resistor is valid until FX2LP firmware makes the pin an output and drives it low.

4.3 Permanently Write-Protect the EEPROM

In some FX2LP designs, the EEPROM contents never change, so tying the WP pin to its active state prevents EEPROM writes. If your design uses a small (16-byte) EEPROM only to supply custom vendor ID information, this solution probably applies. But keep in mind that permanently write-protecting a large EEPROM used to store code disables any reprogramming tools such as the USB Control Center, supplied by Cypress.

4.3.1 Use a Bleed Resistor

This is a last-ditch remedy, useful only in designs that are in production and require an add-on solution. The idea is to accelerate the ramp-down of the supply voltage to minimize the time that FX2LP and the EEPROM may be operating at incompatible voltages. By placing a power resistor between the main supply voltage and GND, the power-down time can be reduced. The resistor should be chosen to drain the onboard bulk capacitors in about 20 milliseconds, and it must have a suitable power rating. This obviously adds to the device power consumption.

5 Selecting a Resonator for USB

USB relies on a precision clock in the USB device. The USB specification dictates frequency deviation to be ±500 ppm at most. The FX2LP datasheet tightens this to ±100 ppm. The resonator is a small package, which also reduces the external part count. The resonator has internal load capacitors and therefore saves the additional space of the load capacitors that crystals require as external components.

5.1 Resonator Requirements

The FX2LP crystal requirements are as follows:

- 24 MHz
- Parallel resonant
- Fundamental mode
- 500-µW drive level
- 12-pF (5 percent tolerance) load capacitor

Cypress specifies resonator frequency tolerance as ±100 ppm to take into account other circuit factors that can add to the USB ±500-ppm tolerance budget such as circuit board layout, temperature, load capacitors, and crystal aging. One resonator manufacturer, Murata, offers CERALOCK® series resonators with initial accuracies of 70 ppm and 100 ppm. For FX2LP, a Murata part number may be encoded according to Table 2.

Table 2. Murata Part Number Construction

Designator	Meaning
CS	Ceramic resonator
T	Built-in load capacitors
CE	Small-cap SMT package (3.2 × 1.3 × 1.1 mm)
24M0	Frequency (“M” indicates decimal point)
G	Fundamental mode (thickness shear mode)
1	100 ppm (“H” for 70 ppm)
2	10-pF load capacitance*
*Specifying a 10-pF load capacitance allows for about 2 pF of PCB trace capacitance.	

6 High-Speed USB PCB Layout Recommendations

High-speed USB operates at 480 Mbps with 400-mV signaling. For backwards compatibility, devices that are high-speed capable must be able to communicate with full-speed USB products at 12 Mbps using 3.3-V signaling. High-speed USB hubs are also required to talk to low-speed products at 1.5 Mbps. Designing PCBs that meet these requirements can be challenging.

This section details guidelines for designing controlled-impedance, high-speed USB PCBs to comply with the USB specification. It applies to all Cypress high-speed USB solutions.

High-speed USB PCBs are typically four-layer or higher boards. Cypress does not recommend using a two-layer board for any high-speed USB PCB design. PCB design influences USB signal quality test results more than any other factor. This section addresses five key areas of high-speed USB PCB design and layout:

- Controlled Differential Impedance
- USB Signals
- Power and Ground
- Crystal or Oscillator
- Troubleshooting

6.1 Controlled Differential Impedance

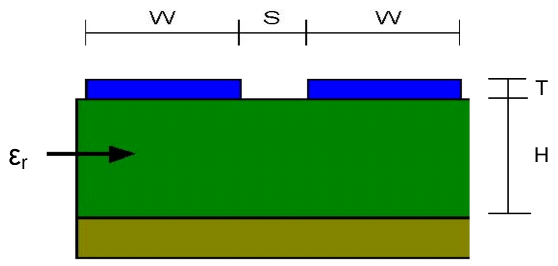
Controlled differential impedance of the D+ and D– traces is important in USB 2.0 PCB design. The impedance of the D+ and D– traces affects signal eye pattern, end-of-packet (EOP) width, jitter, and crossover voltage measurements. It is important to understand the underlying theory of differential impedance to achieve the specified 90 Ω ± 10 percent impedance.

6.1.1 Theory

Microstrips are the copper traces on the outer layers of a PCB. A microstrip has an impedance, Z_0, that is determined by its width (W), height (T), distance to the nearest copper plane (H), and the relative permittivity ($\mathcal{E}_r$) of the material (commonly FR-4) between the microstrip and the nearest plane. When two microstrips run parallel to each other, cross-coupling occurs. The space between the microstrips (S) as related to their height above a plane (H) affects the amount of cross-coupling that occurs. The amount of cross-coupling increases as the space between the microstrips is reduced. As cross-coupling increases, the microstrips' impedances decrease. Differential impedance, Z_{diff}, is measured by measuring the impedance of both microstrips and summing them.

Figure 10 shows a cross-sectional representation of a PCB showing (from top to bottom) the differential traces, the substrate, and the GND plane.

Figure 10. Microstrip Model of Differential Impedance



Equation 1 and Equation 2 provide the formulas necessary for estimating differential impedance using a 2D parallel microstrip model. Table 3 provides the definition of the variables. These formulas are valid for the ratios 0.1 < W / H < 2.0 and 0.2 < S / H < 3.0. Commercial utilities can obtain more accurate results using empirical or 3D modeling algorithms.

Equation 1. Differential Impedance Formula

$$Z_{diff} = 2 \times Z_0(1 - 0.48e^{-0.96S/H})$$

Equation 2. Impedance of One Microstrip

$$Z_0 = (87/(\mathcal{E}_r + 1.41)^{0.5}) \ln(5.98H/0.8W + T)$$

Table 3. Definition of Differential Impedance Variables

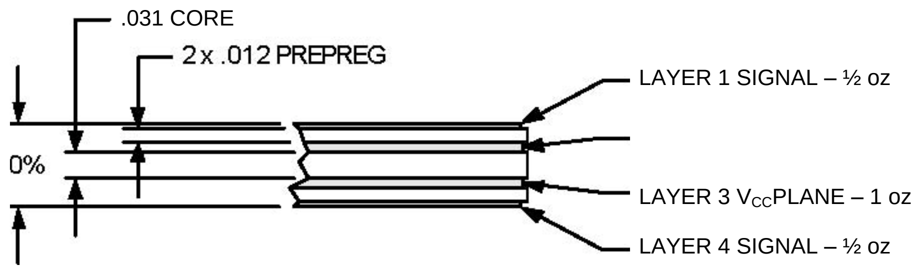
Variable	Definition
Z_{diff}	Differential impedance of two parallel microstrips over a plane
Z_0	Impedance of one microstrip over a plane
W	Width of the traces
H	Distance from the GND plane to the traces
T	Trace thickness (1/2 oz copper ≅ 0.65 mils)
S	Space between differential traces (air gap)
$\mathcal{E}_r$	Relative permittivity of substrate (FR-4 ≅ 4.5)

6.1.2 Typical 62-Mil, 4-Layer PCB Example

Figure 11 shows the recommended stackup for a standard 62-mil (1.6-mm) thick PCB. When this stackup is used with two parallel traces each with width, W, of 16 mils and spacing, S, of 7 mils, the calculated differential impedance, Z_{diff}, is 87 Ω.

With the same stackup, it is possible to achieve a 90 Ω ± 10 percent differential impedance on D+ and D– using other combinations of variables.

Figure 11. Typical Stackup for a 62-Mil, 4-Layer PCB



NOTE: .016 TRACES ARE 90-OHM DIFFERENTIAL IMPEDANCE

6.1.3 Recommendations

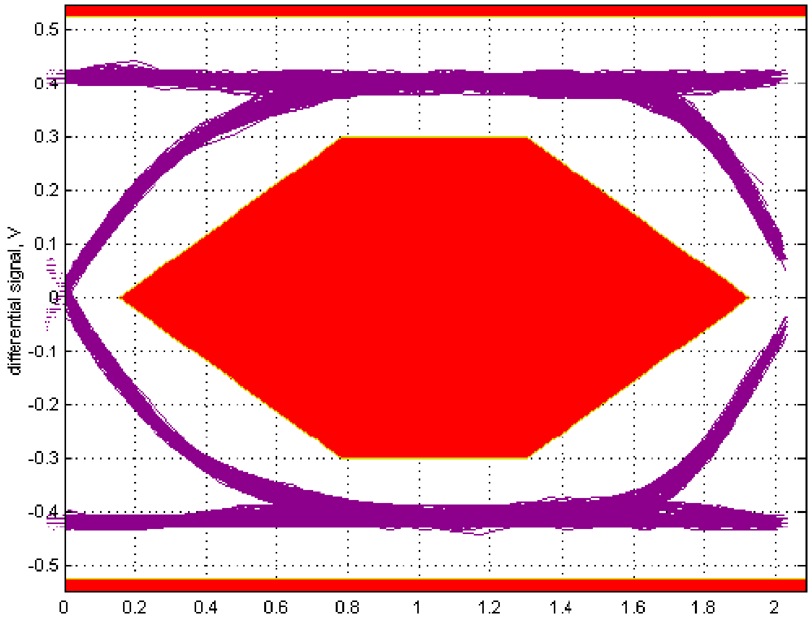
The recommendations to achieve proper differential impedance are the following:

- Consult with the PCB manufacturer to obtain the necessary design parameters and stackup for a 90-Ω ± 10 percent differential impedance on D+ and D–.
- Set the correct trace widths and trace spacing for the D+ and D– traces in the layout tool.
- Draw the proper stackup on the PCB fabrication drawing and require the PCB manufacturer to follow the drawing. See Figure 11.
- Annotate the PCB fabrication drawing to indicate which trace widths require controlled differential impedance. Also indicate what impedance and tolerance is required.
- Request differential impedance test results from the PCB manufacturer.

6.1.4 Eye Diagram

One key measurement of the USB data signal quality is the eye pattern. The eye pattern is a representation of USB signaling that provides minimum and maximum voltage levels as well as signal jitter. Section 7.1 in the USB 2.0 specification provides a detailed explanation of and the requirements for a compliant eye pattern. Figure 12 is an eye diagram of high-speed signaling as measured on the EZ-USB FX2LP component.

Figure 12. Eye Diagram of High-Speed Signaling



The purple lines are superimposed traces of many FX2LP D+/D– transitions on the bus. If any of the purple traces touch red, the USB signaling specification is not met. Notice how the FX2LP traces never touch the central six-sided red area or the voltage extremes at the top and bottom indicated by red lines. Excessive data jitter would make the purple lines fuzzy, with some of them touching red. A noncompliant eye can indicate jitter, mismatched impedance, or improper EMI filtering. The amount of white space between purple and red is a good indication of design margin. As Figure 12 illustrates, the FX2LP USB transmitter is extremely clean.

6.2 USB Signals

There are five USB signals: VBUS, D+, D–, GND, and SHIELD. Their functions are described in Table 4.

Table 4. USB Signals

Signal	Description
VBUS	Device power, +5 V, 500 mA (max)
D+ and D–	Data signals, mostly differential
GND	Ground return for VBUS
SHIELD	Cable shielding and receptacle housing

6.2.1 D+ and D–

Properly routing D+ and D– leads to high-quality signal eye pattern, EOP width, jitter, crossover voltage, and receiver sensitivity test results. The following recommendations improve signal quality:

- Place the Cypress high-speed USB chip on the signal layer adjacent to the GND plane.
- Route D+ and D– on the signal layer adjacent to the GND plane.
- Route D+ and D– before other signals.
- Keep the GND plane solid under D+ and D–. Splitting the GND plane underneath these signals introduces impedance mismatch and increases electrical emissions.
- Avoid routing D+ and D– through vias; vias introduce impedance mismatch. Where vias are necessary (for example, using a mini-B connector), keep them small (25-mil pad, 10-mil hole) and keep the D+ and D– traces on the same layers.

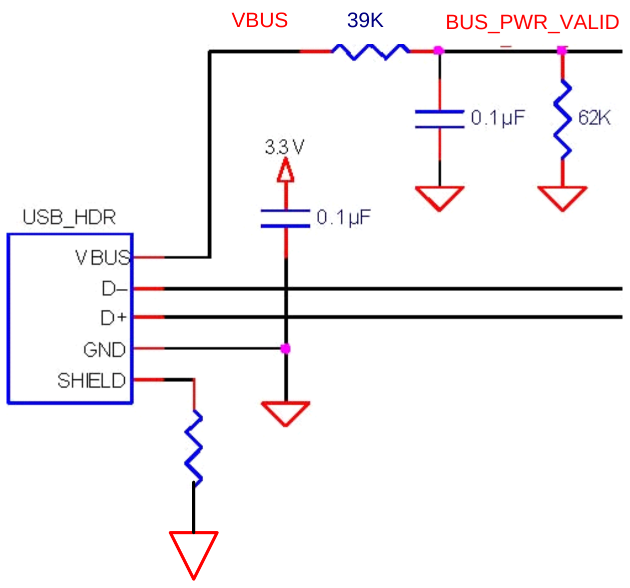
- Keep the length of D+ and D– less than 3 inches (75 mm). A 1-inch length (25–30 mm) or less is preferred.
- Match the lengths of D+ and D– to be within 50 mils (1.25 mm) of each other to avoid skewing the signals and affecting the crossover voltage.
- Keep the D+ and D– trace spacing (S) constant along their route. Varying trace separation creates impedance mismatch.
- Keep a 250-mil (6.5-mm) distance between D+ and D– and other nonstatic traces wherever possible.
- Use two 45° bends or round corners instead of 90° bends.
- Keep five trace widths minimum between D+ and D– and the adjacent copper pour. When placed too close to these signals, copper pour affects their impedance.
- Avoid common mode chokes on D+ and D– unless required to reduce EMI. Common mode chokes typically provide little benefit for high-speed signals and can adversely affect full-speed signal waveforms.

6.2.2 VBUS, GND, and SHIELD

These recommendations for the VBUS, GND, and SHIELD signals improve inrush current measurements and reduce susceptibility to EMI, RFI, and ESD.

- Route VBUS on the signal layer adjacent to the V_{CC} plane. This prevents it from interfering with the D+ and D– signals.
- If you use VBUS to detect power in a self-powered device, filter VBUS to make it less susceptible to ESD events. A simple RC filter works well. See Figure 13 for details. The filter should be placed closer to the USB connector than the USB chip.
- Use 10 µF or less of capacitance on VBUS to prevent violating the USB inrush current requirements.
- Connect the SHIELD connection to GND through a resistor. This helps isolate it and reduces EMI and RFI emissions. Keep this resistor close to the USB connector. Some experimentation may be necessary to obtain the correct value.
- Provide a plane for the USB shield on the signal layer adjacent to the V_{CC} plane that is no larger than the USB header.

Figure 13. Schematic Showing the VBUS Filter, USB SHIELD-to-GND Resistor, and Decoupling Capacitor

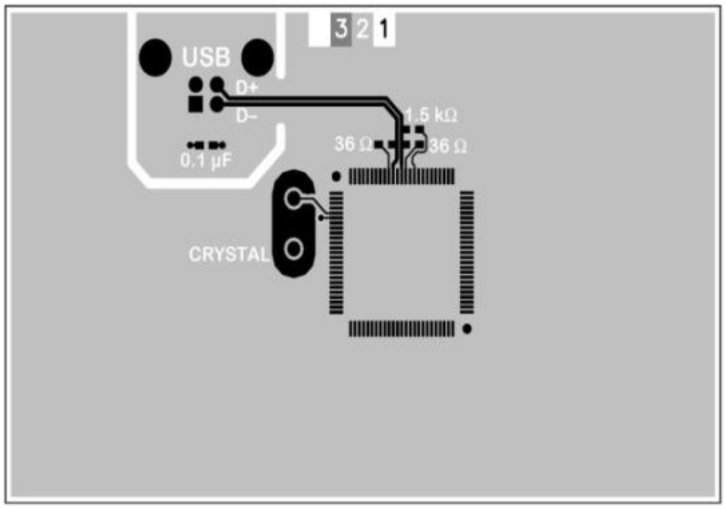


6.2.3 USB Peninsula

If the location of a USB connector is near the edge of the PCB, as shown in Figure 14, consider placing it on a “USB peninsula” as follows. EMI and RFI are decreased by reducing noise on the V_{CC} and GND planes, as they are partially isolated from the rest of the board.

- Make a cut in the V_{CC} and GND planes around the USB connector, leaving a 200-mil (5-mm) opening for D+ and D– to preserve their differential impedance.
- Use a 0.1-µF capacitor to decouple the V_{CC} and GND planes on the USB peninsula.
- Place the SHIELD-to-GND resistor on the peninsula. If necessary, use a second set of pads that connects SHIELD to the GND plane off the peninsula.
- Place a common mode choke (if used, though not recommended) at the opening for D+ and D–, near to the connector.

Figure 14. USB Chip Layout Showing D+/D– Traces, Series Termination Resistors, USB Peninsula, and Crystal



6.3 Power and Ground

You need to provide adequate power and ground for high-speed USB designs. PCB layout is important.

- V_{CC} and GND planes are required for high-speed USB PCB design. They reduce jitter on USB signals and help minimize susceptibility to EMI and RFI.
- Use dedicated planes for V_{CC} and GND.
- Use cutouts on the V_{CC} plane if more than one voltage is required on the board (for example, 2.5 V, 3.3 V, 5.0 V).
- Do not split the GND plane. Do not cut it except as described in USB Peninsula. This reduces electrical noise and decreases jitter on the USB signals.

6.3.1 Power Traces

In some situations, it is not necessary to dedicate a power plane to a voltage rail, for example, a limited run of a 5-V or 12-V power trace. The following are guidelines for power traces:

- Keep the power traces away from high-speed data lines and active components.
- Make power traces at least 40 mils wide to reduce inductance.
- Keep power traces short. Keep routing minimal.
- Use larger vias (at least a 30-mil pad, 15-mil hole) on power traces.
- Provide adequate capacitance (see Decoupling and Bulk Capacitance).
- Use a chip filter, if necessary, to reduce noise.

6.3.2 Voltage Regulation

The following are guidelines for voltage regulators to reduce electrical emissions and prevent regulation problems during USB suspend.

- Select voltage regulators whose quiescent current is appropriate for the board's minimum current during USB suspend.
- Select voltage regulators whose minimum load current is less than the board's load current during USB suspend. If the current draw on the regulator is less than the regulator's minimum load current, then the output voltage may change.
- Place voltage regulators so they straddle split V_{CC} planes; this reduces emissions.

6.3.3 Decoupling and Bulk Capacitance

The following are guidelines for connecting the decoupling and bulk capacitors to the power input pins.

- Provide 0.1-µF ceramic capacitors to decouple the device power input pins. Place one cap per pin. Keep the distance from the pad to the power input pin less than 2.0 mm, where possible.
- Place bulk capacitors near the power input and output headers and the voltage regulator(s).
- Provide 10–20-µF capacitance for the Cypress USB chips. Ceramic or tantalum capacitors are recommended. Electrolytic capacitors are not suitable for bulk capacitance.
- Filter power inputs and outputs near the power headers to reduce electrical noise.
- Follow chip-specific guidelines to properly isolate AV_{CC} from V_{CC} and AGND from GND.
- Follow chip-specific guidelines to provide enough bulk and decoupling capacitance for AV_{CC}. Use ceramic or tantalum capacitors.

6.4 Crystal or Oscillator

A crystal or oscillator provides the reference clock for the Cypress high-speed USB chip. It is important to provide a clean signal to the USB chip and to not interfere with other high-speed signals such as D+ and D–.

- Use a crystal or oscillator whose accuracy is 100 ppm or less.
- Use a crystal whose first harmonic is either 24 MHz or 30 MHz (depending on the high-speed USB chip).
- Place the crystal or oscillator near the crystal pins.
- Keep the traces from the crystal or oscillator to the USB chip short.
- Keep the crystal or oscillator traces away from D+/D–.
- Use ceramic capacitors that match the load capacitance of the parallel-resonant crystal.

6.5 Troubleshooting

The USB electrical compliance tests often show mistakes in PCB layouts. The type of failure can point to the cause. Table 5 shows some common problems and their possible causes for boards that fail high-speed or full-speed signal integrity or high-speed receiver sensitivity tests.

Table 5. Troubleshooting High-Speed USB PCBs

Common Problem	Possible Causes
The high-speed or full-speed signal integrity tests show excessive jitter.	There is an impedance mismatch on D+ and D–.
	A noisy trace is located too close to D+ and D–.
	A common mode choke is interfering.
	An active component (such as a voltage regulator, SRAM, and so on) is not properly decoupled.
	AV_{CC} and AGND are not properly isolated or may not have enough bulk capacitance with a low ESR.

Common Problem	Possible Causes
The EOP is not detected or is out of spec during high-speed or full-speed signal integrity testing.	A common mode choke is interfering with the EOP.
The crossover voltage is out of the specified range.	The trace lengths of D+ and D– are not matched.
	There is an impedance mismatch on D+ and D–.
The voltage level at the beginning of the high-speed chirp is too high when coming out of suspend.	The voltage regulator is unable to maintain 3.3 V at 100 µA.
Receiver sensitivity is below the acceptable limit.	There is a split in the GND plane underneath D+ and D–.
	A common mode choke is interfering.
	AV_{CC} and AGND are not properly isolated or may not have enough bulk capacitance with a low ESR.
Inrush current is above the acceptable limit.	Reduce bulk capacitance on VBUS. If designing a bus-powered solution, employ a soft-start circuit so all of the capacitance is not filled at once.

6.6 Ball Grid Array Considerations

The 56-ball VFBGA version of the FX2LP (CY7C68013A) or FX2LP18 (CY7C68053) is a smaller package version of the QFN package. The 56-ball package meets the needs of space-sensitive PCB designs. This section provides guidelines for designing a PCB with these VFBGA parts.

6.6.1 PCB Layout Terminology

This section defines common terms used in PCB layout design with the Cypress FX2LP (CY7C68013A)/FX2LP18 (CY7C68053) 56-ball VFBGA.

6.6.1.1 Escape Routing

Escape routing is the method used to route each signal from the package to another element(s) on the PCB.

6.6.1.2 Multilayer PCB

Multilayer PCB is an industry-standard method to allow escape routing for high pin count packages, such as VFBGAs. This is achieved by routing signals on various numbers of PCB layers.

6.6.1.3 Via

Vias, or plated through holes, are used in multilayer PCBs to electrically connect signals between layers. Common via types are the following:

- Through via: connects signals between top and bottom layers
- Blind via: connects signals from either the top or bottom layer to an inner PCB layer
- Embedded via: connects signals between inner PCB layers, and is not accessible from the top or bottom of the board

6.6.1.4 Via Capture Pad

Vias are connected electrically to PCB layers through via capture pads that surround each via.

6.6.1.5 Microvia

Microvias are defined as vias that are equal to or less than 0.15 mm (6 mils) in diameter and have a target via capture pad equal to or less than 0.36 mm (14 mils). They are defined by the IPC-2315 and IPC-6012A standards.

6.6.1.6 Surface Land Pad

A surface land pad is the area on the PCB to which a VFBGA solder ball adheres. The size of these pads affects the space available for vias and for the escape routing. Generally, there are two different surface land pad designs: non-solder mask defined (NSMD) and solder mask defined (SMD).

6.6.1.7 NSMD Pad

For NSMD pads, the solder mask opening is larger than the copper pad. The copper surface of the land pad is completely exposed, which provides a greater area for the VFBGA solder ball to adhere.

6.6.1.8 SMD Pad

For SMD pads, the solder mask overlaps the copper surface of the land pad. This overlapping provides greater adhesion strength between the copper pad and the PCB’s epoxy or glass laminate, which is important under extreme bending and during accelerated thermal cycling tests. However, the solder mask overlap reduces the copper surface area for the VFBGA solder ball adhesion.

6.6.1.9 Stringer

Stringers are interconnecting segments that electrically connect via capture pads and surface land pads.

6.7 56-Ball VFBGA Package Details

Table 6 summarizes the dimensions of the 56-ball VFBGA package. These dimensions are crucial to get a precise PCB layout design. Detailed package information is available in the device datasheet.

Table 6. CY7C68013A/CY7C68053 56-Ball VFBGA Dimensions

Parameter	Size
Package size	5 x 5 x 1 mm
Ball count	56 (8x8 matrix)
Ball pitch	0.50 mm
Ball diameter	0.30 mm
Ball pad diameter	0.30 mm

6.7.1 Layout Guidelines

The following factors are important in the PCB design of VFBGA packages:

- Surface land pad dimension
- Via layout and dimension
- Signal line space and trace width

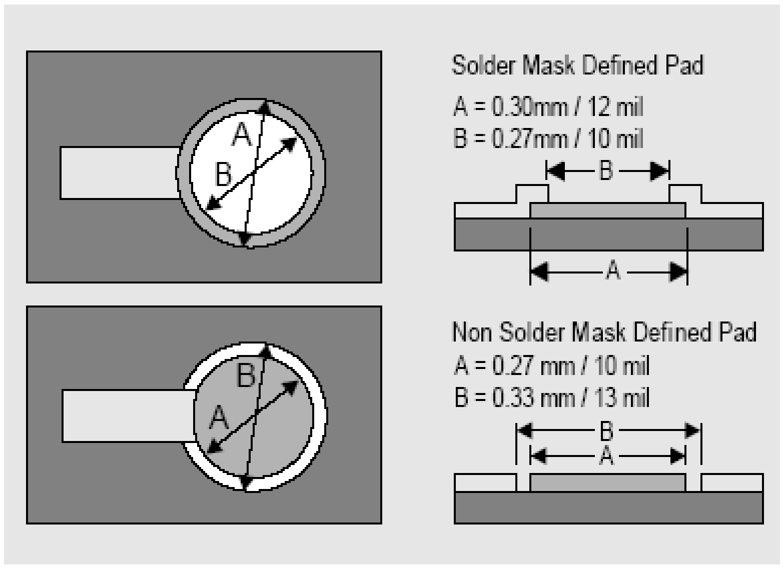
6.7.1.1 Surface Land Pad Dimension

It is desirable to maintain a 1:1 ratio between the package pad and the PCB land pad joint area to balance the stress during temperature cycling. For an NSMD pad, ensure that you have a clearance around the copper pad and the solder mask. This is to account for mask registration tolerances (typically 0.060 mm to 0.075 mm) and to avoid any overlap between the solder joint and the solder mask.

The SMD pad is recommended for handheld applications because it has a stronger pad adhesion that provides higher thermal and mechanical reliability.

Figure 15 illustrates the recommended dimensions of the SMD and NSMD pads for the PCB layout.

Figure 15. SMD and NSMD Pad Dimensions



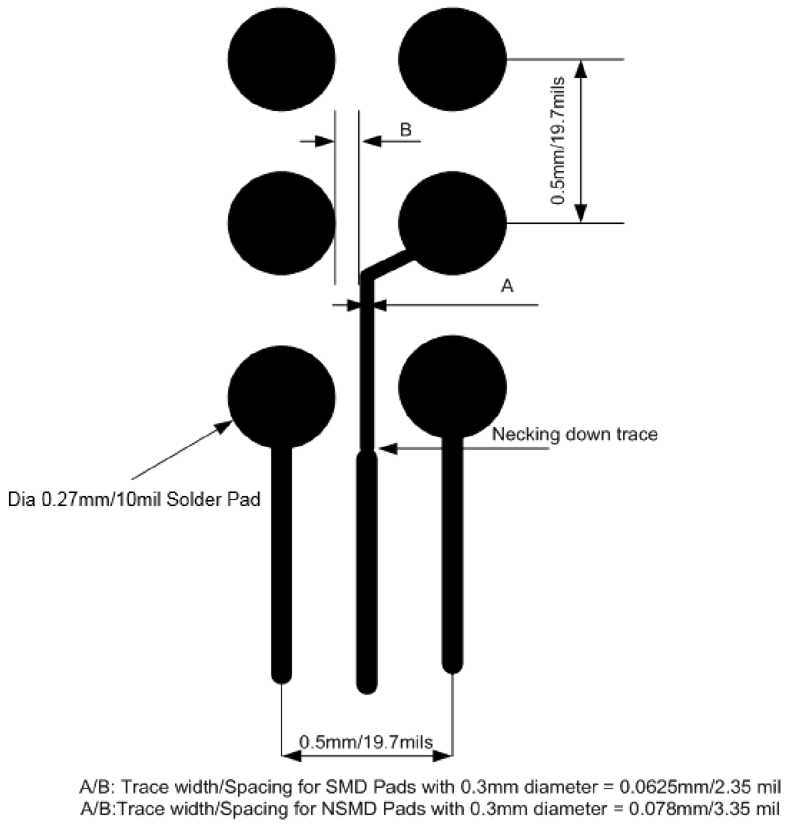
6.7.1.2 Via Layout and Dimension

The size and layout of the via and via capture pads affect the amount of space available for escape routing. For the FX2LP VFBGA, a microvia in the center of the surface land pad (via-in-pad technology) can be used. The microvia drill hole diameter should be equal to or less than 0.50 mm (19.7 mils). The drill hole should be plugged and planarized to create a flat surface. This prevents the solder from wicking through the holes during the assembly process.

6.7.1.3 Signal Line Space and Trace Width

The ability to perform escape routing is defined by the width of the trace and the minimum space required between traces. For the 56-ball VFBGA, a 0.0625-mm (2.35-mil) trace for SMD and a 0.078-mm (3.35-mil) trace for NSMD are preferable. Only one trace can be routed between surface land pads or via capture pads. Figure 16 shows the recommended line space and trace width.

Figure 16. Line Space and Trace Width for SMD Pads


B
0.5mm/19.7mils
A
Necking down trace
Dia 0.27mm/10mil Solder Pad
0.5mm/19.7mils
A/B: Trace width/Spacing for SMD Pads with 0.3mm diameter = 0.0625mm/2.35 mil
A/B:Trace width/Spacing for NSMD Pads with 0.3mm diameter = 0.078mm/3.35 mil

The 2.35/3.35-mil trace is applicable only to the VFBGA area that has routing space constraints. Otherwise, use necking down traces; this makes the PCB easier to manufacture and ensures better yields. As illustrated in Figure 17, after the trace exits the VFBGA pads, standard trace and space geometry is used. The trace layout is shown in Figure 18.

Figure 17. Necking Down Traces

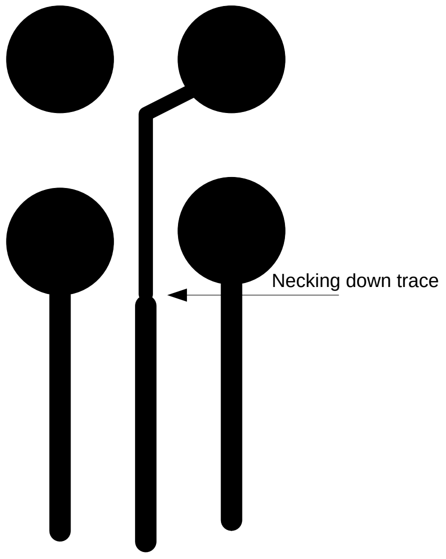
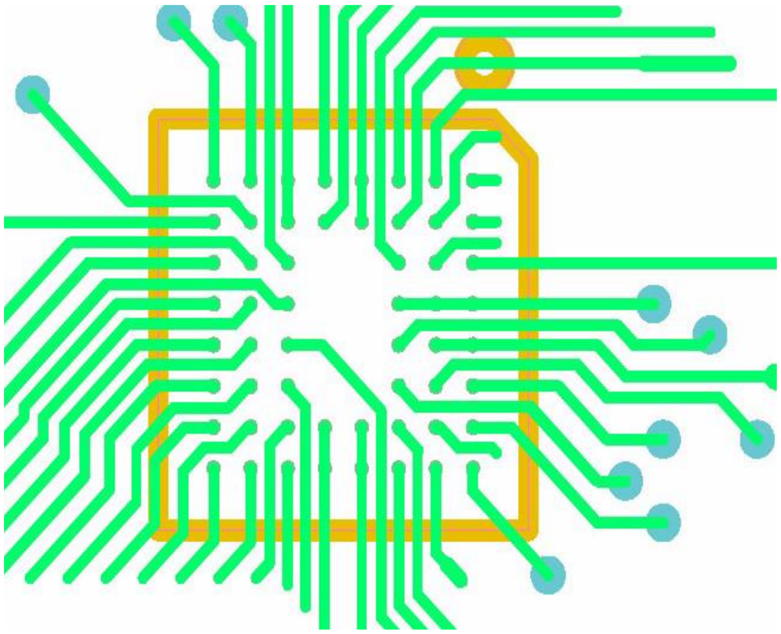


Figure 18. Trace Layout



High-speed USB PCBs must be designed to meet USB electrical requirements. This is best achieved by using controlled impedance PCBs, properly laying out D+ and D–, and adequately decoupling the VCC and GND planes to keep them electrically quiet.

Cypress provides a variety of high-speed USB development and reference design kits. These give helpful design examples and contain chip-specific design guidelines.

7 Schematic Review Checklist

Bringing up a new USB 2.0 design can be tricky due to the numerous details that need to be right (see High-Speed USB PCB Layout Recommendations). This section helps catch potential problems before building a board. The goal is to get the board ready to download firmware and begin testing the firmware.

Note: FX2LP refers to both FX2LP and FX1 devices, except where noted.

7.1 Creating a PCB

The following is a list of items that are critical for the successful operation of an FX2LP design. Read this checklist before creating a PCB using the FX2LP. If a board is built already and is not behaving properly, go through this list to verify that all the items are being implemented correctly on the target.

- All power pins (AV_{CC} or V_{CC}) are powered to a proper voltage level (3.0 V to 3.6 V).
- V_{CC} ramp-up time is at least 200 µs, with a maximum ramp rate of 18 V per millisecond.

  Most boards have enough capacitance on V_{CC} to meet this need. Many regulators also ramp up V_{CC} slowly. However, occasionally a board has an especially fast ramp-up time. In such a case, extra capacitance on V_{CC} slows it down to meet the requirements.
- When using a crystal, the RESET pin remains asserted for at least 5 ms past the point that V_{CC} reaches 3.0 V. When using an external clock source, this hold time is at least 200 µs. If RESET is asserted when the FX2LP is already powered, the reset is held asserted for at least 200 µs.

  The oscillator on the FX2LP requires a crystal or resonator with a load capacitance of 12 pF and a frequency of 24 MHz (±100 ppm). The crystal is required to handle a drive level of at least 500 µW. Drive level is the maximum power dissipation the crystal is expected to withstand. Using a lower drive level crystal may work, but exceeding the crystal's maximum drive level can lead to faster aging or loss of accuracy. In extreme cases, it can damage the crystal. Two crystals used successfully with the FX2LP are the eCera FX2400026 and the Ecliptek EC-12-24.000M. Load capacitors of 12 pF are required between each crystal pin and ground.

- The RESERVED pin is grounded.

  The RESERVED pin is a test mode pin. If it is not grounded, the FX2LP is placed into a test mode and does not operate correctly.
- The SCL and SDA pull-up resistors are installed, even if no EEPROM is used.

  If there are no pull-ups on SCL and SDA, FX2LP hangs immediately on booting. The FX2LP looks for a serial EEPROM on SCL and SDA when booting to determine the mode of operation. If the lines are pulled up with no EEPROM present, the SIE enumerates as the default USB device that enables connection to the USB Control Center application. However, with no pull-up resistors, the FX2LP believes the bus is controlled by another master and waits indefinitely for the other master to release the bus.

  The recommended value of the pull-up resistors is 2.2 kΩ. A method of temporarily breaking the connection between SDA of the FX2LP and SDA of the EEPROM should be included on the board, especially during development. This enables you to reprogram a corrupt EEPROM without having to remove it from the board. This disconnect can be a jumper, switch, or a removable 0-Ω resistor.
- EA is tied to ground unless external flash memory or ROM is connected to the external memory bus.

  If EA is tied high, the FX2LP immediately attempts to boot from code stored in external memory. If there is no code in external memory, the device cannot boot.
- The WAKEUP# pin is tied high or low and is not left floating.

  A floating WAKEUP# pin causes erratic suspend behavior. Holding the WAKEUP# pin low inhibits the FX2LP from entering suspend mode when no USB traffic is detected. WAKEUP# is typically tied high with a 10K or 100-kΩ resistor.
- FX2LP can bootload firmware from an I^2C EEPROM.

  A small EEPROM (address contained in a single I^2C byte) must have its address pins A[2:0] connected to 000. A large EEPROM (two address bytes) must have its address pins A[2:0] connected to 001.
- The FX2LP reset is reliable in a bus-powered design that is detached and quickly reattached.
- In self-powered designs, the USB device must monitor the VBUS wire and disable the D+ pull-up resistor (D– for low-speed devices) without VBUS per the USB specification. Self-powered designs using FX2LP can use a GPIO pin or the WAKEUP pin as explained in this note.
- Multiple Vdd lines are routed using individual bypass capacitors.
- The AV_{CC} pin is routed separately with 2.2-µF and 0.1-µF capacitors in parallel.
- Components (choke, resistor, and so on) are not connected to the USB data lines; such connection is not recommended as it can cause signal quality issues.
- All unused FX2LP I/Os are tied to a valid logic level for lowest current consumption.

7.2 Bringing Up the Board

When bringing up the board of a new FX2LP design, start with a board that has only a minimum set of components installed. Populate the board with a minimum number of components as follows:

- FX2LP or FX1 chip
- 3.3-V regulation circuit connected to all AVDD and AVCC pins
- Pull-up resistors (2.2 kΩ or other value compliant with the I^2C specification) on the SCL and SDA lines
- EEPROM blank or not installed
- Connect the WAKEUP# line to logic high or low to ensure it is not floating. Pulling it high with a 10K or 100-kΩ resistor is preferred for easier debugging.
- EA pin connected to logic low
- RESERVED pin(s) connected to ground
- Reset circuit that provides the required minimum reset timing
- Crystal oscillator circuit (including crystal and load capacitors) or external clock source

The purpose of initial checkout is to get the FX2LP default device to communicate with the USB Control Center application running on a PC (see the Cypress Driver and USB Control Center section). This application verifies USB communication of your design and downloads firmware into FX2LP for testing and development.

When the board is ready for testing, follow the steps in Cypress Driver and USB Control Center. If your device does not enumerate, the following procedure can help diagnose the problem.

Use an oscilloscope to look for a 12-MHz signal on the CLKOUT pin. A non-enumerating device may exhibit four CLKOUT scenarios:

7.2.1 No 12-MHz Signal on CLKOUT

In this situation, FX2LP is either not powering up correctly, or it is not receiving a reference clock. Check the V_{CC}/AV_{CC} pins, V_{CC} ramp rate, crystal oscillator circuit, and POR timing. Also make sure that the RESERVED pin is grounded.

7.2.2 CLKOUT Outputs 12 MHz, Which Disappears After a Short Period of Time

FX2LP is entering suspend mode. Make sure that the WAKEUP# pin is not floating. Suspend mode happens if the FX2LP is not seeing the regular start of frame (SOF) bus signals from the host and WAKEUP# is high. Make sure the board unit is plugged into the USB of the host and that D+ and D– connect from the host to the device with the correct polarity and with no external series resistors. Suspend mode can also occur if the reference clock (either driven externally or from the crystal oscillator) is outside the tolerated accuracy range. Check the frequency of CLKOUT to make sure it is 12 MHz ±100 ppm.

7.2.3 CLKOUT Continues to Output 12 MHz, But Enumeration Fails

- If the WAKEUP# pin is tied low (inhibiting suspend), the device may not be seeing the signals from the host. Check all the items in the previous case where CLKOUT is active but quickly turns off.
- If the WAKEUP# pin is tied high (or if it is tied low and you have double-checked all the previous items), make sure there are pull-ups on SCL and SDA and no EEPROM is connected. Also make sure that EA is tied low.
- If these items are verified and all the other checklist items from the first section are correct, then there may be a problem with the operating system driver. The Cypress Driver and USB Control Center section explains how to clear out an old driver and install the current one.

7.2.4 CLKOUT Outputs 48 MHz, and Device Enumerates But Does Not Connect to USB Control Center

This also can be a symptom of an incorrect operating system driver. The Cypress Driver and USB Control Center section explains how to clear out an old driver and install the current one.

8 Cypress Driver and USB Control Center

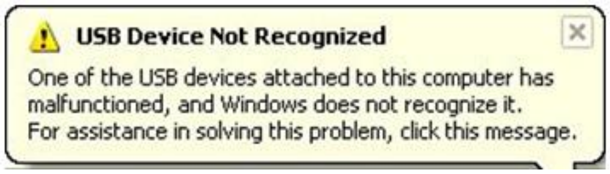
When an FX2LP powers up with no EEPROM connected to its I^2C pins (but with I^2C pull-up resistors installed), it enumerates as a Cypress default device with vendor ID VID=0x04B4 (Cypress) and product ID PID=0x8613 (FX2LP). An FX1 enumerates as VID=0x04B4 and PID=0x6473. This is a fully functional USB device that creates the resources (endpoints and transfer logic) required to load FX2LP code into its internal RAM over USB. This previously made USB device (no code required) simplifies the USB checkout of a new design by removing any user code issues from the initial testing.

To allow a PC host to communicate with an FX2LP based device, Cypress provides a Windows driver called *cyusb3.sys* and a companion *cyusb3.inf* file that binds the driver to the default FX2LP device. These files and a Windows utility that uses the driver, called USB Control Center, are included as a companion zip file to this app note. The first time you plug your device into USB for initial testing, you can use Cypress software tools to check out the design as long as no EEPROM is connected, enabling the default Cypress device at plug-in.

8.1 Driver Installation

The first time you plug your device into USB, Windows reads the VID/PID values from your device and searches for a compatible driver. Since this is the first time, you will see a series of windows asking to locate a compatible driver. If the host PC has a problem retrieving the VID/PID information, you will see the error message shown in Figure 19.

Figure 19. Windows Error Message for Unrecognized USB Device



USB Device Not Recognized

One of the USB devices attached to this computer has malfunctioned, and Windows does not recognize it.
For assistance in solving this problem, click this message.

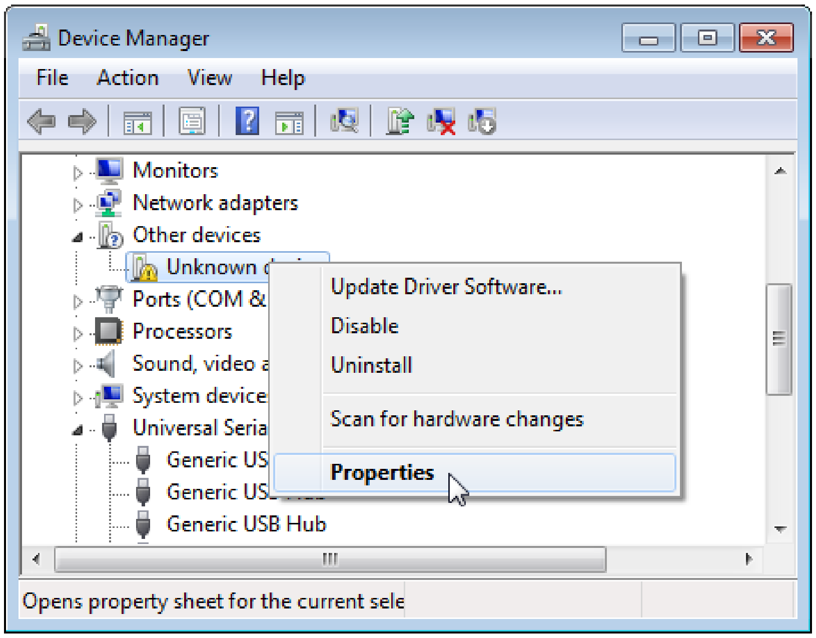
If you see this message, you know the host PC detected your device, so the D+ pull-up is working. But when it tried to retrieve the VID/PID data using a GET_DESCRIPTOR(Device) request, it received all-zero VID and PID values. This problem can be caused by the hardware issues mentioned in previous sections. Some quick checks you can perform are to verify power supply voltages and confirm that the oscillator CLKOUT pin has a 12-MHz output signal.

The next message you see for a new USB device indicates that Windows recognized the USB device, but it did not find a compatible driver. An easy way to install the driver is to dismiss any warning messages and then start up the Windows Device Manager. (You will get to know Device Manager very well as you check out a USB design.) Device Manager can be started in multiple ways, including the following two:

- Click the Windows Start menu button, right-click “Computer,” select “Properties,” and then select “Device Manager” in the top left column labeled “Control Panel Home.”
- Type “Devmgmt.msc*”* in the **Run…** box in the Start menu.

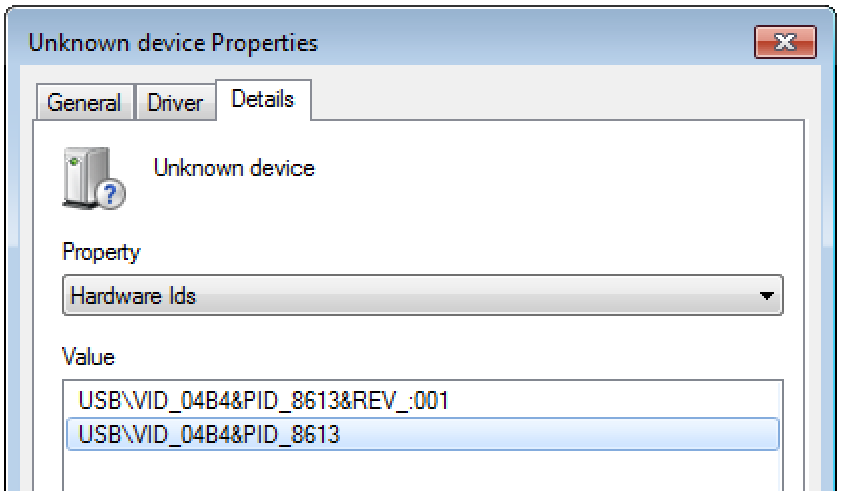
Your unidentified device will appear somewhere in the device tree. Figure 20 shows a screen shot from a Windows 7 64-bit machine. To make sure you are updating the correct device, right-click the device name and then select **Properties**.

Figure 20. Device Recognized as Unknown (No Driver Yet)



In the **Properties** drop-down box, select **Hardware Ids** and you will see the VID and PID for the device. As Figure 21 shows, the “Unknown Device” is the generic FX2LP chip with VID=04B4 and PID=8613.

Figure 21. Windows Finds Default FX2LP VID and PID


Unknown device Properties

General | Driver | Details

Unknown device

Property

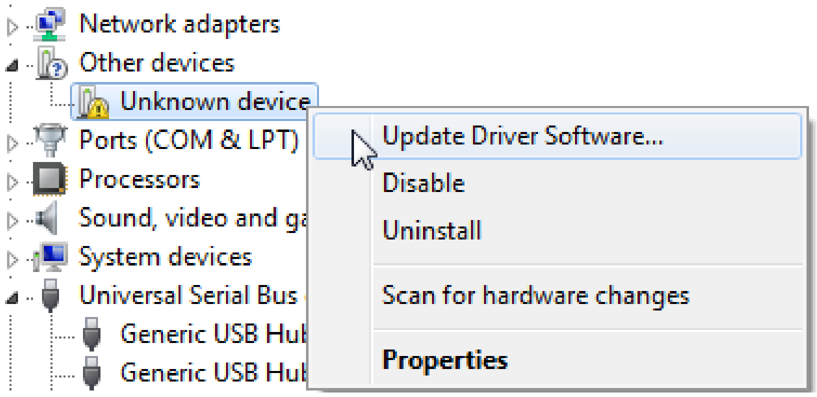
Hardware Ids

Value

USB\VID_04B4&PID_8613&REV_:001
USB\VID_04B4&PID_8613

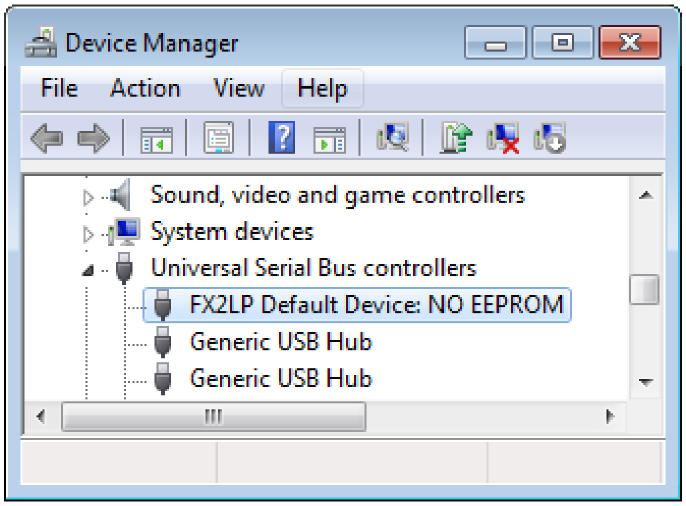
Back in Device Manager, right-click the device and select **Update Driver Software…** (Figure 22).

Figure 22. Update Driver Software



Choose **Browse my computer…** and navigate to the Cypress Driver folder for your OS: “win7” for Windows 7, “wlh” for Windows Vista (“Longhorn” was the Windows Vista code name), or “wxp” for Windows XP. Thirty-two bit versions are identified as “x86,” and 64-bit versions are identified as “x64.” You may get a security check message, which can be ignored. You should see a success message, and then Device Manager updates to indicate the correctly installed driver (Figure 23).

Figure 23. Successful Driver Installation

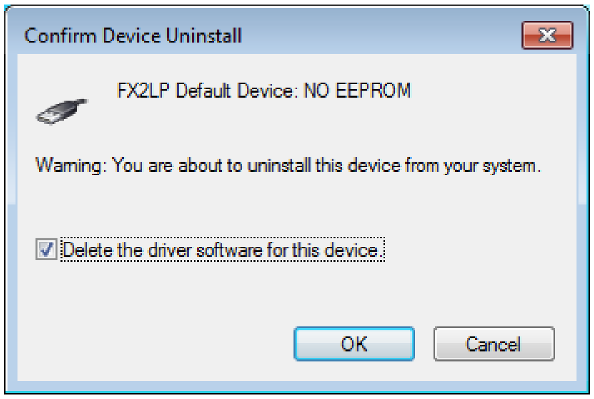


Sound, video and game controllers
System devices
Universal Serial Bus controllers
FX2LP Default Device: NO EEPROM
Generic USB Hub
Generic USB Hub

Note: You can also install the driver using an FX2LP development board (available here) with its EEPROM ENABLE slide switch set to the NO EEPROM position. This enables the default device with VID=0x04B4 and PID=0x8613.

In rare cases, the driver may be associated with a wrong device. You can remove this association by right-clicking the device in Device Manager and selecting **Uninstall**. Then select the box for **Delete the driver software for this device** and click **OK** (Figure 24).

Figure 24. Removing an Old Driver



FX2LP Default Device: NO EEPROM

Warning: You are about to uninstall this device from your system.

[✓] Delete the driver software for this device.

8.2 USB Control Center

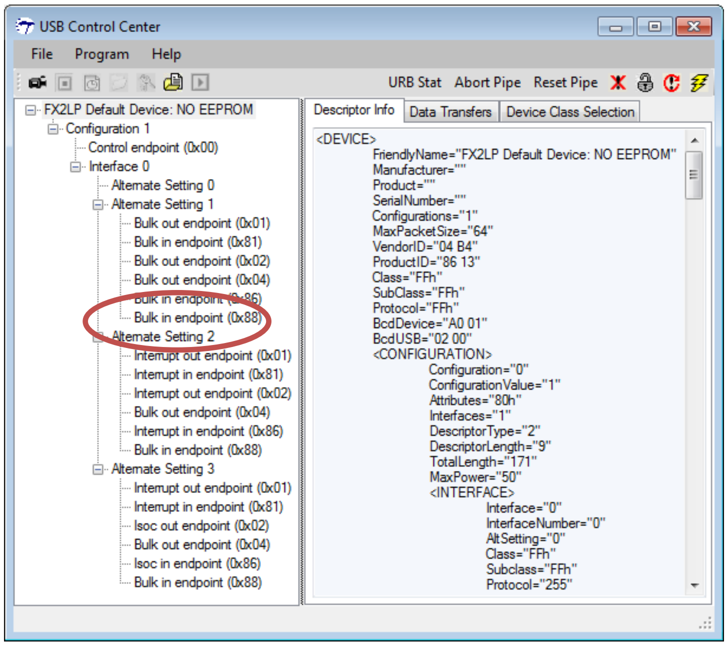
Double-click the *CyControl.exe* file in the folder accompanying this note to start the USB Control Center.

Note: The *CyControl.exe* file is included in the folder accompanying this note for convenience. By downloading a USB development suite from the Cypress website, you have full access to the Microsoft .NET source code for education or modification. Cypress also offers many app notes covering code development, for example AN70983 – Designing a Bulk Transfer Host Application for EZ-USB FX2LP/FX3.

Figure 25 shows the USB Control Center with an FX2LP development board plugged into USB. If your prototype is wired correctly and there is no boot EEPROM, you should see the same device—the Default FX2LP USB device. The left panel shows all the details of the device it found at VID=0x04B4 and PID=0x8613 (red circle).

As soon as your device prototype passes this verification step, you are ready to start testing your own USB code. You will find the USB Control Center to be a valuable tool for firmware checkout—note the **Data Transfers** tab. This tab can be used to schedule individual USB transfers in and out of your device and to check the results. Although it is no substitute for a full USB bus analyzer, it does provide some of the basic debug features of an analyzer.

Figure 25. USB Control Center Device Visibility


USB Control Center
File Program Help
URB Stat Abort Pipe Reset Pipe
Descriptor Info Data Transfers Device Class Selection

```
FX2LP Default Device: NO EEPROM
  Configuration 1
    Control endpoint (0x00)
    Interface 0
      Alternate Setting 0
      Alternate Setting 1
        Bulk out endpoint (0x01)
        Bulk in endpoint (0x81)
        Bulk out endpoint (0x02)
        Bulk out endpoint (0x04)
        Bulk in endpoint (0x86)
        Bulk in endpoint (0x88)
      Alternate Setting 2
        Interrupt out endpoint (0x01)
        Interrupt in endpoint (0x81)
        Interrupt out endpoint (0x02)
        Bulk out endpoint (0x04)
        Interrupt in endpoint (0x86)
        Bulk in endpoint (0x88)
      Alternate Setting 3
        Interrupt out endpoint (0x01)
        Interrupt in endpoint (0x81)
        Isoc out endpoint (0x02)
        Bulk out endpoint (0x04)
        Isoc in endpoint (0x86)
        Bulk in endpoint (0x88)
```

```
<DEVICE>
        FriendlyName="FX2LP Default Device: NO EEPROM"
        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"
```

9 Summary

This application note touched on several considerations for designing a successful FX2LP based USB peripheral. Key points covered were USB power distribution, achieving a reliable reset in a hot-plugged device, EEPROM and resonator design details, and PCB layout. Because FX2LP can power up as a fully functional "default" USB device, you can use Cypress software tools to check out your hardware design without writing a single line of code. This very usefully separates hardware checkout from firmware debug.

10 Related Documents

- AN65209 – Getting Started with FX2LP
- Larry W. Burgess & Paul D. Madden, "Designing and Fabricating Multi-Depth Via-in-Pad PCBs," IPC Printed Circuits Expo, April 1998
- Yuan Li. Anil Pannikkat, Larry Anderson, Tarun Verma, Bruce Euzent, "Building Reliability into Full-Array BGAs," 26th IEMT Symposium, December 2000
- AN70983 – Designing a Bulk Transfer Host Application for EZ-USB FX2LP/FX3

About the Author

Name:	Rama Sai Krishna Vakkantula.
Title:	Applications Engineer Staff
Background:	Rama Sai Krishna holds an M.Tech in Systems and Control Engineering from IIT Bombay. He is currently working on Cypress USB peripherals.

Document History

Document Title: AN15456 - Guide to a Successful EZ-USB® FX2LP™ Hardware Design

Document Number: 001-15456

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1429925	KLY	08/30/2007	New application note.
*A	3023712	AASI	09/06/2010	Added information on crystal drive level. Emphasized assumptions on the driver side. Textual changes to emphasize other collaterals available.
*B	3341713	AASI	08/10/2011	Renamed “Hardware checklist” section to “Schematic Review Checklist” and updated the same section.
*C	4227164	RSKV	01/05/2014	Merged all application notes (AN15813, AN064, AN5078, AN6070, AN1168, and AN43841) related to FX2LP hardware design guidelines into this application note. Changed Title of this application note from “Guide to Successful EZ-USB FX2LP and EZ-USB FX1 Hardware Design and Debug” to “Guide to a Successful EZ-USB® FX2LP™ Hardware Design.” Updated in new template.
*D	4526931	PRJI	10/07/2014	No technical updates. Completing Sunset Review.
*E	4963434	PRJI	10/14/2015	Added a note to connect the reset circuitry of the FX2LP RESET# pin to the EEPROM RESET/POWER pin and updated the trace width in Figure 16 to 4 mils. Updated template
*F	5700340	AESATP12	04/26/2017	Updated logo and copyright.
*G	5821072	HPPC	07/12/2017	Added the **More code examples** section.

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

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