How to Add USB Communications without USB Expertise

Introduction

Universal serial bus (USB) is the most widely used protocol to enable communications between embedded systems and computing platforms ranging from desktops to laptops to tablets. Despite the popularity and pervasive use of USB, embedded developers often face the challenge of learning how to use the USB protocol and properly design it into their embedded system. Due to the complex nature of USB and design time constraints, developers are turning to component vendors that offer fixed-function communication interface bridges designed to simplify the complexity associated with USB designs and save development effort and expense.

Three Easy Steps for Adding USB

An example of this type of communication interface product is the Silicon Labs CP2130 USB-to-SPI bridge controller, which enables USB connectivity by interfacing through the serial peripheral interface (SPI) port on a general-purpose MCU. Figure 1 shows how a communication bridge interfaces with an embedded system. In addition to USB-to-SPI bridge devices, the following communication bridges supporting other interfaces are available:

- USB-to-UART, 2x UART, 4x UART
- USB-to-SMBus/I²C
- USB-to-I²S

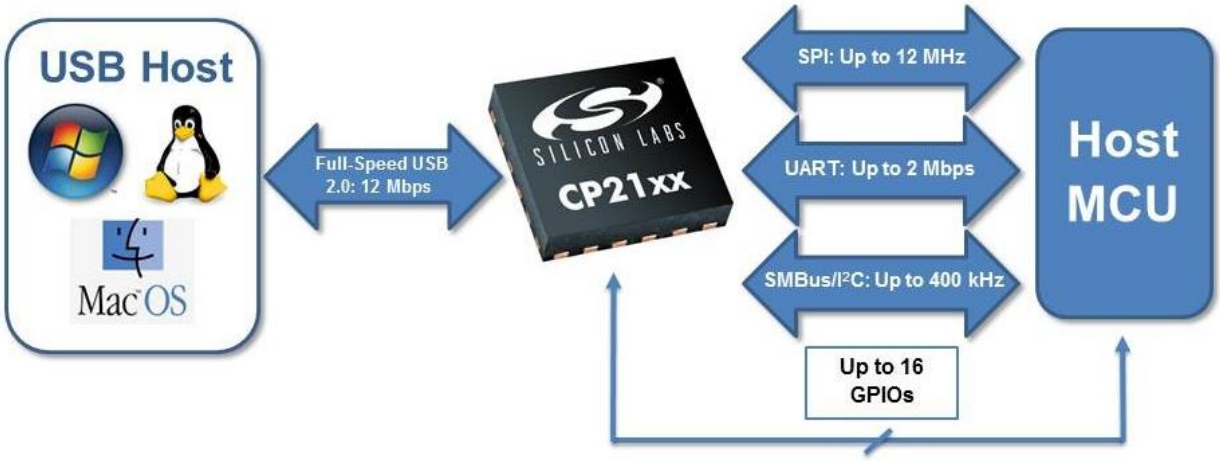

Figure 1. System-Level USB Host to MCU Host Connectivity

Embedded systems targeting USB communications can be divided into two categories: upgrading a legacy design to use USB or enhancing a new design by adding USB.

For either category, developers can follow three easy steps to quickly enable USB communications without requiring USB expertise:

- Identify the desired communication peripheral on a host MCU.
- Build a prototype using an evaluation kit and jumper wires.
- Create a custom schematic and layout with a communication bridge.

Additionally (to be discussed later), developers can create an application-customized part and driver. Now, let's take a closer look at each specific step and examine the pros and cons of several design choices along the way.

STEP 1: Identify the Target Communication Peripheral on the Host MCU

Whether upgrading a legacy design or enhancing a new design, developers must identify the host MCU's available communication peripheral(s). In the legacy design scenario, developers should look for any free peripherals on the host MCU. If there are none, the developer can use an addressable protocol such as SMBus/I^2C. For new designs, developers can select a host MCU to match the desired communication protocol. For example, if SPI communication is the desired protocol, developers should select a host MCU that has an additional free SPI port. It is important to also consider the maximum transfer rate requirements of the application when selecting a communication peripheral. If a large amount of data must be transferred at high speeds, SPI or UART is the best choice. If transfer speed is not important or multi-device bus connectivity with arbitration is important, SMBus/I^2C is the best choice, only requiring two pins.

In addition to selecting the communication peripheral, it is also important to consider the various driver options that are available. The most user-friendly option is a Human Interface Device (HID)-class communication bridge, which does not require a driver installation for use. Devices in the HID class use native drivers found on common operating systems. Simply plug the device into a USB port and begin using it. Other examples of driver options include virtual COM port (VCP) drivers, WinUSB/LibUSB drivers and vendor-specific drivers. All of these options require a driver installation and can typically achieve higher throughput than HID-class products.

STEP 2: Build a Prototype Using an Evaluation Kit and Jumper Wires

In the next step, the developer uses a communication bridge evaluation kit to build a prototype connecting the evaluation kit to a host MCU using jumper wires. The prototype is meant to verify communication between the two devices and serve as a starting point for the schematic. Generally, the evaluation board will be clearly labeled, which helps the developer determine where to connect jumper wires without having to consult documentation.

In the following example shown in Figure 2, we will connect a CP2130 USB-to-SPI evaluation board to a C8051F850 MCU card running SPI slave example code. A legacy design with SPI signal test points could be used in place of the C8051F850 MCU card in this example. In the figure below, the SPI header has been connected to the specific port pins of the MCU development board, which are connected to the internal SPI peripheral.

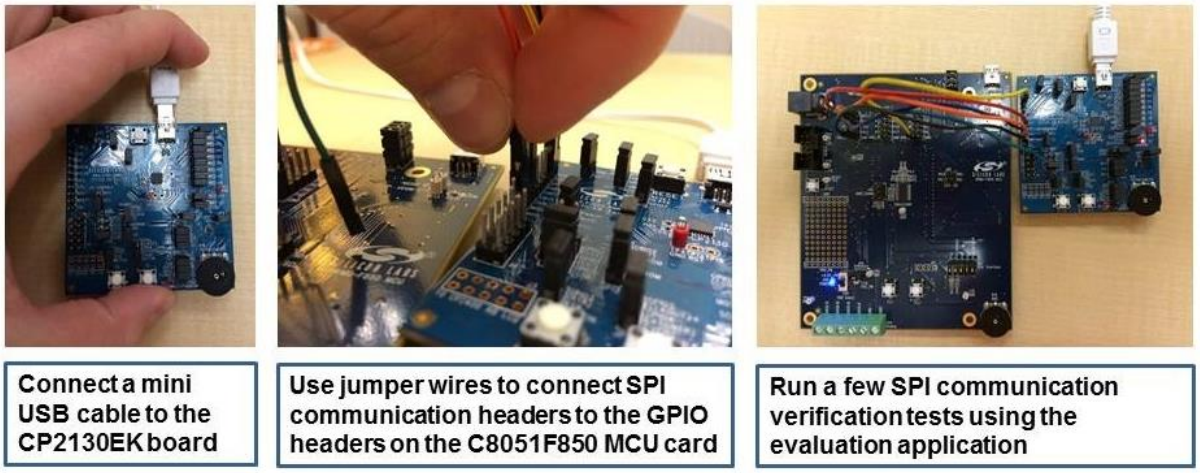

Figure 2. Example of a Bus-Powered Prototype System

Now that the hardware is ready, you can begin to prepare your USB host machine by installing demonstration applications and a driver (if necessary). All required software can usually be found on the manufacturer's web site. This option is typically preferred, as it will include the most up-to-date software and documentation. The CP2130 bridge device requires a driver installation before use and can use an evaluation application for USB communications, both of which are available in the CP2130 software package that comes with the evaluation kit. After finishing the installation, the evaluation application can be used to read and write SPI data to the C8051F850 MCU via USB. Now it's time to run a few verification tests of reads and writes to verify operation.

STEP 3: Create a Custom Schematic/Layout with a Communication Bridge

The prototype from Step 2 can help create the schematic for your final design. First, locate the schematic for the communication bridge evaluation kit. For the CP2130 USB-to-SPI bridge controller, this schematic can be found in the CP2130-EK User Guide from www.silabs.com/USB-Bridge.

The evaluation kit schematic shows which components are necessary for USB operation. It is important to mention that some communication bridges integrate functionality that eliminates external components, reduces BOM costs and simplifies the design. Check to verify that your communication bridge supports the following features:

- Crystal-less USB: Does the communication bridge support USB communications without requiring an external crystal?

- Internal 5V regulator: Can the bridge power the system through the USB connection without requiring external components, and what is the maximum output current? (This feature is primarily applicable to USB bus-powered applications.)
- In-system programming memory: Does the bridge device contain integrated programmable memory that allows customization via USB?
- Small package options: Smaller packages enable developers to create small, highly-portable solutions.

Selecting a communication bridge that includes these features will result in a simplified design. Next, translate your prototype connections to schematic connections for the SPI signals (SCLK, MOSI, MISO, SS, GND) between the CP2130 device and host MCU. Before proceeding to layout, send your schematic to the Silicon Labs' support team at www.silabs.com/contactsupport for a complimentary review to verify that the device's schematic configuration is correct. The support team will review the schematic and provide recommendations if changes are needed. Now the board can be sent to manufacturing. If your product design requires a custom part and driver, please keep reading. If not, start communicating!

What if You Want to Create a Custom Part and Driver?

Communication bridges and drivers often come with vendor-specific standard USB descriptors and strings right out of the box. The USB Vendor ID (VID), Product ID (PID) descriptors and serial string are used by the operating system to map drivers to connected devices. As a result, it is highly recommended that the combination of descriptors and strings be unique to prevent errors when two devices with identical information are connected to a system. Creating a custom driver and device enables developers to use product-specific strings and device descriptors. Figure 3 shows the default strings displayed in Windows when installing a non-customized CP2130 device.

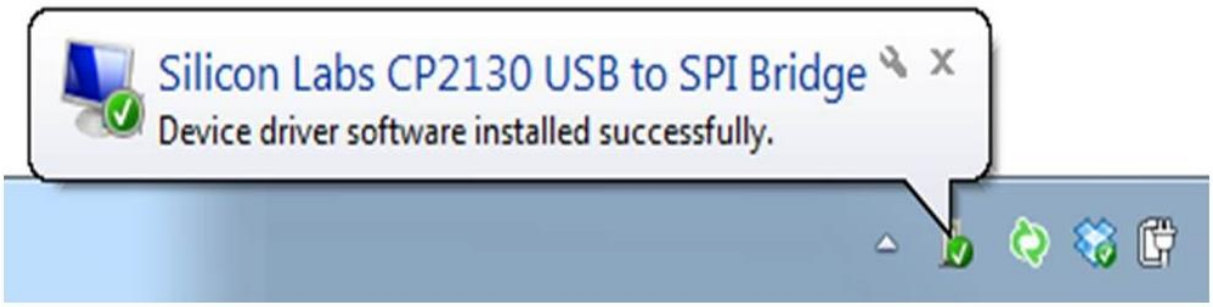

Figure 3. Default Strings Displayed in Windows during CP2130 Installation

To successfully customize a product, the communication bridge and the associated driver (if applicable) must be updated. Communication bridges have corresponding graphical customization utilities that allow customization through the USB connection. Bridge manufacturers also typically provide a driver customization utility that can serve as a step-by-step guide to creating a custom driver. After customizing a device, the device will not be able to communicate with the host system unless the associated custom driver has also been installed on the host.

Other Useful Tips

- Once a driver has been customized in any way, an additional certification step may be required by operating system vendors. An example of this is Microsoft's Windows Hardware Quality Labs (WHQL) testing. Windows 7 (x64) and Windows 8 (x64) will not allow an uncertified driver to be installed. Other Windows versions will allow installation, but will display a warning message to the user. *Application Note 220: USB Driver Customization* provides instructions for creating a custom driver, and *Application Note 807: Recertifying a Customized Windows HCK Driver Package* provides instructions for recertifying a customized driver for Windows. Both application notes can be found at www.silabs.com/interface-appnotes.
- For production-quantities of customized communication bridges, manufacturers can provide preprogrammed devices for use in end systems that do not require the graphical customization utility.
- Before shipping your product to customers, remember to include a USB cable, as well as a flash drive, CD/DVD or web address for the custom driver.

Additional Benefits and Closing Thoughts

Although the primary goal of this three-step approach to adding USB is to easily enable USB communication in embedded systems, these bridge devices also have GPIO pins with additional special functions that can integrate functionality, eliminate external components and further reduce BOM cost. Examples of these special functions include:

- USB suspend indicators
- Clock output
- Toggle LED on data traffic
- Count pulses or edges
- Remote wake

In evaluating your USB bridge chip options, look for a semiconductor vendor that offers a comprehensive portfolio of bridge solutions that support all major communication interfaces. Be sure that the fixed-function communication bridge you choose includes such features as crystal-less USB operation, an internal voltage regulator, in-system programming memory and small-footprint package options. Selecting the right bridge chip for your next embedded application ultimately can save you considerable development effort and expense and help speed your design to market.

Table 1 compares the features of nearly a dozen Silicon Labs bridge products that use the most common communication interfaces.

Part Number	Interface	Driver Types	Operating Systems	GPIO Pins	Transfer Rate (Max)	Crystal Not Required	5V Reg Output	Program Memory	Package
CP2102*	USB to UART	VCP, USBXpress	Windows, Linux, OSX	-	1 Mbps	✓	3.3V, 100mA	Flash	28-pin 5x5 QFN
CP2103*	USB to UART	VCP, USBXpress	Windows, Linux, OSX	4	1 Mbps	✓	3.3V, 100mA	Flash	28-pin 5x5 QFN
CP2104	USB to UART	VCP, USBXpress	Windows, Linux, OSX	4	2 Mbps	✓	3.45V, 100mA	EPROM	24-pin 4x4 QFN
CP2105	USB to 2x UART	VCP, USBXpress	Windows, Linux, OSX	5	2 Mbps	✓	3.45V, 100mA	EPROM	24-pin 4x4 QFN
CP2108	USB to 4x UART	VCP, USBXpress	Windows, Linux, OSX	16	2 Mbps	✓	3.3V, 150mA	Flash	64-pin 9x9 QFN
CP2109*	USB to UART	VCP, USBXpress	Windows, Linux, OSX	-	1 Mbps	✓	3.45V, 100mA	EPROM	28-pin 5x5 QFN
CP2110	HID USB to UART	HID	Windows, Linux, OSX	8	1 Mbps	✓	3.45V, 100mA	EPROM	24-pin 4x4 QFN
CP2112	HID USB to SMBus/I^2C	HID	Windows, Linux, OSX	8	1 Mbps	✓	3.45V, 100mA	EPROM	24-pin 4x4 QFN
CP2114	USB Audio to I^2S	USB Audio	Windows, Linux, OSX	12	≤48 kHz 16bit stereo	✓	3.45V, 100mA	EPROM	32-pin 5x5 QFN
CP2130	USB to SPI	WinUSB LibUSB	Windows, Linux, OSX	11	6.6 Mbps	✓	3.45V, 100mA	EPROM	24-pin 4x4 QFN

* For new designs, customers should design-in using the lower-cost CP2104

Table 1. Common Communication Bridges

# # #

Silicon Labs invests in research and development to help our customers differentiate in the market with innovative low-power, small size, analog intensive, mixed-signal solutions. Silicon Labs' extensive patent portfolio is a testament to our unique approach and world-class engineering team. Patent: www.silabs.com/patent-notice