Design Considerations in Adding USB Communications to Embedded Applications

Designing universal serial bus (USB) communications into an application enables a system to communicate with a variety of USB host devices and provides a convenient power option through the USB connection. Today's printers, cell phones, digital cameras, media players, external hard drives and gaming systems all transfer data using the USB protocol. Having both power and data communication capability in one cable adds convenience and flexibility to applications. USB communications can be designed into new systems or added to upgrade legacy systems through the addition of a fixed-function USB communication bridge or a USB microcontroller (MCU) with custom USB firmware. The main issues with USB development options are data throughput versus development time and whether USB expertise is required for implementation. Small USB MCUs and fixed-function USB communication bridges provide cost-effective solutions for adding USB communications to designs.

The USB communications interface contains four signals: D+, D-, Ground and VBUS. The D+ and D- signals are differential data lines, and the VBUS signal is a 5 V line provided by the USB host device. The VBUS signal is used to indicate the presence of a USB cable in a USB port but can also provide a system with up to 500 mA from a powered hub or 100 mA from an unpowered hub. An MCU or fixed-function USB communication bridge with an on-chip 5 to 3 V regulator can use the regulator output to power an entire application. The specification also supports a variety of USB connectors that vary in size and shape, including standard, mini and micro connectors. The variety of USB connector sizes gives developers many options for integrating USB communications into their applications.

In addition, the USB specification allows up to 127 devices to be attached to a single bus and supports a variety of device classes, including the human interface device (HID) class, which is natively supported by most operating systems and does not require driver installation. During the enumeration process, host machines determine the type of USB device connected. After plugging a USB device into a host, the USB device sends descriptors to the host that indicate the device type and which drivers to load.

Developers can upgrade legacy systems to add USB connectivity, or they can design new systems that include USB from the beginning. A USB MCU or a fixed-function USB bridge can address both scenarios. Table 1 shows options for adding USB communications to a system, as well as developer and end-user requirements.

Table 1. USB Connectivity Options

		Fixed-Function Virtual COM Port Bridge	Fixed-Function HID Bridge	HID USB MCU	Custom USB MCU
DEVELOPER TASKS	*Driver Development*	No; Driver supplied by manufacturer	No; Native support by OS	No; Native support by OS	Yes
	USB Firmware Development	No	No	Yes	Yes
	PCB Redesign	Yes	Yes	Yes	Yes
END-USER TASKS	*Driver Installation*	Yes	No	No	Yes

Choosing the right USB communications option depends on several factors, such as whether a developer is upgrading an existing system or creating a new system. First, let's examine how to design a new system with USB capability.

Developers creating a new system have flexibility in choosing the best method of adding USB communications. They can design the system around a USB MCU or a fixed-function USB communication bridge and change aspects of the design to fit the USB solution. For example, the initial printed circuit board (PCB) design contains all necessary components, including the USB device and USB connector, and the board designer can reposition them as needed. In addition, the method of interfacing USB communications to the system is unrestricted, and developers can choose any of the four USB communications options listed in Table 1.

Fixed-function USB communication bridges provide the easiest solution for adding USB communications to a new design but offer the least flexibility. They are available as HID or non-HID fixed-function USB communication bridges, such as the USB-to-UART virtual COM port (VCP) bridges. When using these communication bridges, USB expertise is not necessary because USB firmware and driver development are not required. For non-HID class devices, manufacturers provide the necessary drivers for supported operating systems. In addition, manufacturers often provide dynamic-link libraries (DLLs), which aid in the development of USB host applications. The lack of USB firmware, DLL and driver development reduces the application's time to market. With this approach, the USB interface is not directly connected to the target system. Instead, another bridge device interface, such as UART, serial peripheral interface (SPI), or inter-integrated circuit (I^2C), directly connects to the target application. In the following system, a USB-to-UART VCP bridge communicates with a target system through the UART interface.

Figure 1. USB-to-UART VCP Bridge

Developers using this option to add USB communications to a system must ensure that the target system can communicate using the UART interface and consider the throughput of the bridge device, which is usually limited by the UART communication speed. In addition, the developer will need to provide a driver and driver installation package to the end-user. The end-user will need to install the driver in order to use the device. In this example, the bridge device appears as a COM port to the USB host system. Developers desiring a fixed-function USB communication bridge that does not require host-side driver installation should consider an HID communication bridge.

The HID device class is gaining acceptance as a general connection option for embedded systems because of its flexibility, overall throughput and lack of driver installation. Because the HID class is natively supported by most operating systems, driver development is not required, and end-users can plug in a device right out of the box and begin using it. There is no need for driver installation by the end-user. In the previous USB-to-UART VCP example, the bridge device can be replaced with an HID USB-to-UART device, as shown in Figure 2.

Figure 2. HID USB-to-UART Device

The majority of the design considerations for the HID bridge are identical to the VCP bridge example, but there are a few design differences between the HID and VCP USB-to-UART bridge example. With the HID configuration, the limit of the bridge device's throughput is the maximum HID throughput, which is 64 kilobytes per second. Also, the device will not appear as a COM port to the USB host, but instead as an HID-class device. HID fixed-function communication bridges provide a drop-in solution for developers who want to minimize the overall USB development time while adding USB communications to a system. If the throughput or general functionality of a fixed-function USB communication bridge is insufficient for an application, developers should instead consider adding a USB MCU.

USB MCUs provide the most flexibility and control over the USB communication interface but require the most design effort. Developers must generate all USB firmware, and if a non-HID class device is created, they must write device drivers. This requires some USB experience since writing USB firmware and device drivers is not a trivial exercise. Since all of the MCU firmware is customizable, the USB MCU can perform additional tasks as needed. This gives added flexibility that is not available with a communication bridge. For example, if the USB MCU has an analog-to-digital converter (ADC), the developer can add firmware to configure the ADC and take measurements as needed. The USB descriptors are also fully customizable in firmware. USB hosts determine if a device is an HID or non-HID device through the descriptors received from it during the enumeration process.

When using a USB MCU, the USB communications provides a direct interface to a target system, and the system can be built around the USB MCU.

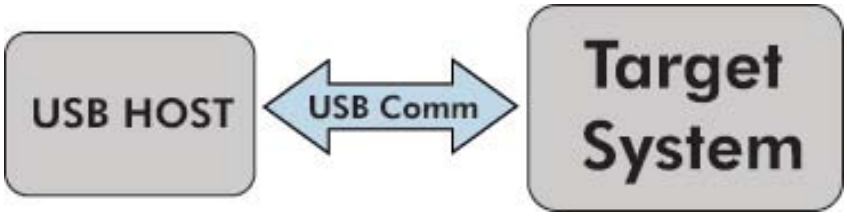

Figure 3. USB Host System Configuration

In addition to increased development time, developers should also consider the required throughput. The throughput limit of an HID class device is 64 kilobytes per second, or 512 kilobits per second. The throughput limit of a non-HID class device is 12 Megabits per second, or 12,000 kilobits per second. The non-HID class device can achieve much higher throughput than the HID device but also requires the development of a custom driver and installation of a driver by the end-user. This will add to the application's overall development time. Driver development and installation can be avoided by using an HID-configured USB MCU but only if the throughput of HID is sufficient for the application.

Creating a system containing a USB MCU provides flexibility in changing aspects of the design to fit the best USB solution as needed. For example, developers designing a medical device that sends measurements to a host using USB communications can change the USB MCU's data transfer type to meet the throughput limits of a desired USB MCU solution or implement a multi-interface device, such as a device with an isochronous and HID interface. When designing a new USB application, developers can analyze the requirements of each USB option and choose the best fit. Next, let's examine the scenario of upgrading a legacy design with USB communications.

Developers upgrading a legacy system with USB communications can choose any of the four options available for new designs, but they must select a USB solution that fits an existing application instead of designing an application to fit a USB solution. In this case, developers need to consider the current means of communication, the required USB data throughput and the available PCB space for additional components. Legacy designs have an established means of communicating with host systems. The addition of a fixed-function USB communication bridge is only an option if the interface used to communicate with a host is available in a bridge device. In most applications, this will be the UART interface. For these applications, a USB-to-UART communication bridge chip can be added to a design. Figures 4 and 5 show how the addition of a bridge device fits into a legacy design.

Figure 4. Legacy UART Design

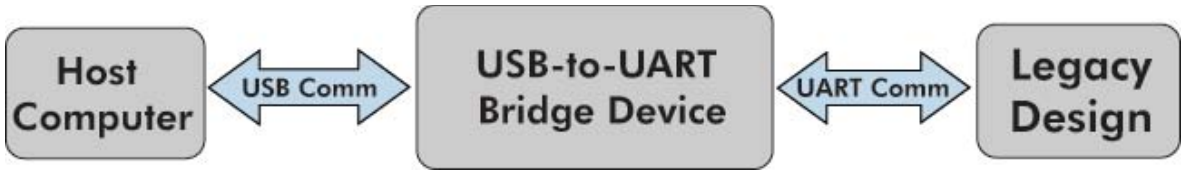

Figure 5. Legacy Design USB-to-UART Upgrade

From a hardware perspective, the existing PCB needs a redesign to allow the USB device and USB connector to fit on the existing board. From a software perspective, the manufacturer of the USB-to-UART device often provides a VCP driver to the developer; therefore, no driver development is required. In this example, the throughput limit of the bridge device is the baud rate of the UART interface. As long as the bridge device supports the application's required baud rate, throughput should not be an issue. The device still appears as a COM port to the USB host, which allows legacy host applications to function correctly without modification. The main difference between the legacy design and the upgraded design is the provision of an interface to the host through USB and the need for driver installation by the end-user.

If a driverless option is needed, an HID fixed-function USB communication bridge is a potential solution. With this choice, the same design considerations are necessary as with the VCP fixed-function communication bridge, but the bridge's throughput is limited to 64 kilobytes per second, which is the maximum HID throughput. In the previous legacy design upgrade example, developers could instead use an HID USB-to-UART bridge, but the device would not appear as a COM port to the host system. The device would instead appear as an HID. As a result, the legacy host application would not function correctly without modifications. Although a driver installation is not required with this solution, existing host applications will need to be modified to talk to the HID OS application programming interface (API) instead of the COM port API. In the majority of legacy design upgrades, fixed-function USB communications bridges are ideal because they provide the easiest solution for adding USB communications to a design without the need for USB expertise.

For legacy designs that require higher throughput, additional functionality or custom USB firmware, a USB MCU is the best choice. Many of the same design considerations apply in this scenario as in the new design scenario. This option requires some USB expertise because developers must write all USB firmware. Driver development and installation is also required for VCP USB devices. The USB MCU must have a means of communicating to the existing legacy application through GPIO pins or a peripheral interface, such as a system management bus (SMBus) or SPI, on the USB MCU.

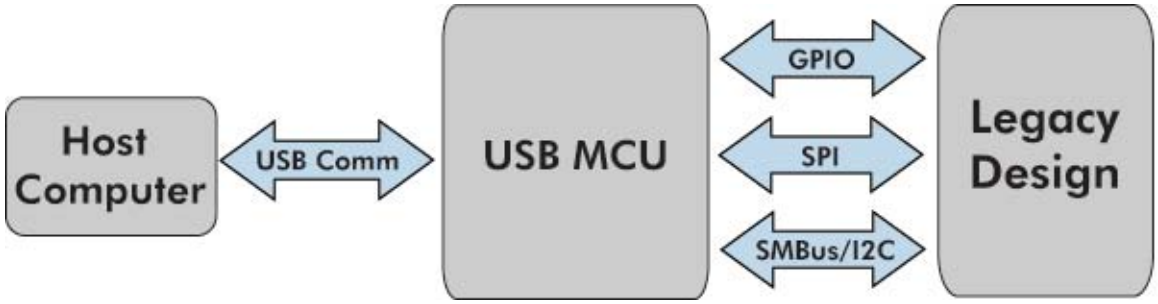

Figure 6. USB Communications with Legacy Application

In addition, the existing PCB will need a redesign with additional components. Upgrading a legacy application with a USB MCU is the best choice for developers who need higher throughput than bridge devices can achieve or use a communication method that is not available in a bridge device.

Choosing whether to add a fixed-function communication bridge or a USB MCU to a design depends on the target application, the developer's USB experience and the amount of development time available. Using a USB MCU provides the most flexibility but also requires USB expertise and possible driver development. Choosing a fixed-function USB communication bridge does not require any USB firmware or driver development, which reduces overall development time. It is the easiest way to add USB to a system with minimal redesign.

Adding USB functionality to a system adds convenience and flexibility by enabling communications with a wide array of USB host devices as well as adding a power option that can provide up to 500mA in a single cable. Silicon Labs offers a wide variety of USB MCUs as well as fixed-function USB communication bridges, including USB-to-UART, HID USB-to-UART and HID USB-to-SMBus bridge devices. Small USB MCUs and fixed-function USB communication bridges provide a cost-effective way to add USB communications to new designs or legacy systems.