The Past, Present and Future of USB

Introduction

When the humble universal serial bus (USB) was first introduced, it took the electronics industry a few years to get on board, but now there are few connected devices that don't use USB interfaces. Data transfers of all types (device synching, stored data, etc.) and even charging are now thoroughly the province of the USB interface.

There's just one drawback when developing USB solutions for the embedded device market: power. The humble USB becomes a power-hungry tyrant when trying to design a low-power device that is battery operated and connected. This need for power, leads designers to use larger batteries than they would prefer, and to get poorer performance from those batteries.

So how do you then manage your power budget, when you need to upgrade your interface to a USB interface? The tradeoff is sometimes less than ideal, and you have to make hard decisions about battery size and the ultimate cost of the device to consumers. Is it ultimately worth it to make a device more expensive, or to use bigger batteries to preserve your energy budget? What if you didn't have to make those tradeoffs? In this paper, we'll explore the past, present, and future of the USB connection protocol.

The Past: USB’s Origin Story

In the bad old days, the back of a PC was, frankly, a mess. You had all different manner of ports—5-pin DIN, PS/2, serial port, parallel port, or even a SCSI (scuzzy) port. If you were a gamer, you might have had a game port on your sound card. In case you’ve blocked this memory, Figure 1 should remind you.

Fig 1: A 90’s-style PC tower with a multitude of ports.

In the mid-90s, the creators of USB saw a need to consolidate the pathways by which machines speak to each other (Machine to Machine or M2M). USB was so successful that today it is the default connection for everything from smartphones to mice to printers. Micro-USB and Mini-USB variants further expand the USB footprint.

Under the covers of the PC, even the touchpad and built-in keyboard and other built-in peripherals of your laptop communicate with the device via USB.

How does USB work?

Let’s step back and examine how USB functions so that we fully understand the implications of the technology as we progress to our next topic. It’s all about devices and hosts, as you can see in Figure 2.

Figure 2: The Device/Host topology.

Perhaps one of the most useful aspects of the USB topology is its ability to provide power to a device. You know this if you’ve ever used a USB cable to charge your smartphone. How much power can you move over a USB interface? The standard is at least 100 mA of current, but it could take as much as 500 mA. Sounds like there should be no power issues at all, right? Unfortunately, because USB requirements depend so heavily on having a reliable main power supply connected to an outlet, like the PC itself. There’s where we run into problems with M2M interfaces like those in the Internet of Things.

The Present: USB Today

One term that is crucial to understand as we move forward with our discussion of USB is “power budget.” A power budget represents the amount of energy a device can consume, and it is based off the battery size and required (or desired) battery life. In one example, you want to use a 250 mA battery and you want your device to have a 48 hour battery life. You’ll need to spread this power consumption across the entire device, encompassing all functionality from sensor acquisition to communication and displays. It’s very much a puzzle that each developer has to solve for every time they sit down to design.

As microcontrollers (MCUs) grew smaller and batteries improved over the last two or three decades, the market was flooded with a vast array of new portable devices. However, with the introduction of smart phones with quad-core gigahertz processors, we now see more portable devices being introduced as additions to smart phones since manufacturers no longer have to worry about processing power or user interfaces. This market trend is driving the proliferation of inexpensive add-ons like credit card payment processing and smartphone breathalyzers.

When designing a portable device, you want to maximize interoperability and user friendliness, so you want to choose an interface like USB. And when you incorporate USB, you also make your gadgets host-agnostic. It doesn’t matter whether your users connect to a PC, an iPhone, or an Android tablet. Therefore, when you want to connect all of these extra gadgets via USB to your battery-powered go-to mobile devices, what was never a concern in the original USB specification – power consumption – suddenly becomes a top priority when choosing a USB-based solution. You don’t want to waste the precious battery life of a tablet or laptop just to communicate with the onboard peripherals. And you don’t want to design a simple add-on application for a smart phone that quickly drains its battery.

By choosing the right USB-enabled hardware, you will be able to develop your device with a much smaller energy footprint since a universal M2M interface allows you to exclude almost all external components.

The Future: USB and the IoT

We've already discussed the very basic topology of USB, but to understand fully how USB can be improved to work with the next generation of IoT devices, we need to go a bit deeper. In general, only the host can initiate transfers. Even if there is no communication, the host sends keep-alive messages to the device every millisecond. If the device has data available, it will reply. In this active mode, the device has up to 100 mA of power, and the host expects the device to provide an immediate response to any request. When the host stops sending these keep-alive messages for 3 ms, the device should enter a suspend state and immediately reduce its current draw below 3 mA.

In the suspend state, most of the device can be switched off, and usually we can switch off the most power-hungry parts of the PHY. Even though a 3 mA suspend current should be easily achievable by any modern MCU, there is no reason to keep it that high. MCUs with well-thought-out energy modes, like the Silicon Labs EFM32 Happy Gecko MCU, should be able to achieve less than 3 µA in this mode, including the current draw of the PHY.

However, in active mode, when inspecting the USB communication of a regular keyboard device, active mode is still not very active; most of the time, the device is just waiting for the host to send data. However, whenever the host requests a response from the device, the response must be immediate; that is why most implementations keep the USB peripheral running at 48 MHz at all times to allow sufficient response time. In this particular example, the lines are idle for 97 percent of the time, even though we are enumerated and active.

A USB implementation that decides when the clock is needed and for how long is uniquely optimized for battery-powered applications. Silicon Labs now has two patents pending for designs to make the USB interface truly usable in today's battery-powered IoT world.

Energy-efficient communications, even in active mode, are enabled by using crystal-less USB oscillators and by disabling the power-hungry part of USB connectivity between packets, as shown in Figure 3. This innovative approach greatly reduces system-level power consumption and creates a truly universal M2M interface offering exceptional energy efficiency.

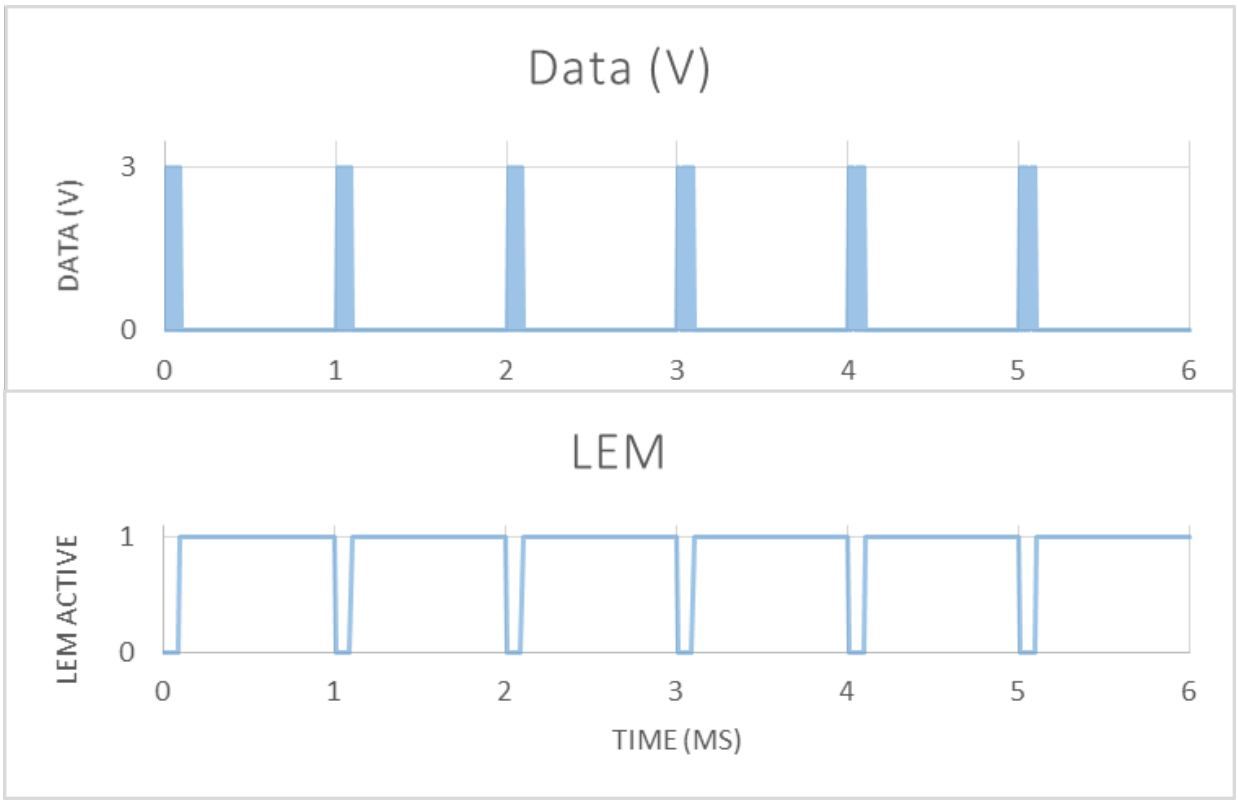

Figure 3: Overview of bus traffic for a keyboard with the low-energy mode (LEM) active signal indicating when the power-hungry parts of the USB interface are disabled.

Low-energy USB should be implemented in a way that is completely transparent to developers and to end users. What will be noticeable is significantly reduced power

consumption through low-energy modes (LEM), as shown in Figure 4. When this technology is combined with other space- and cost-saving features such as crystal-less USB implementations and clock recovery, developers can realize a truly ultra-low-power universal M2M interface without the need for additional external components.

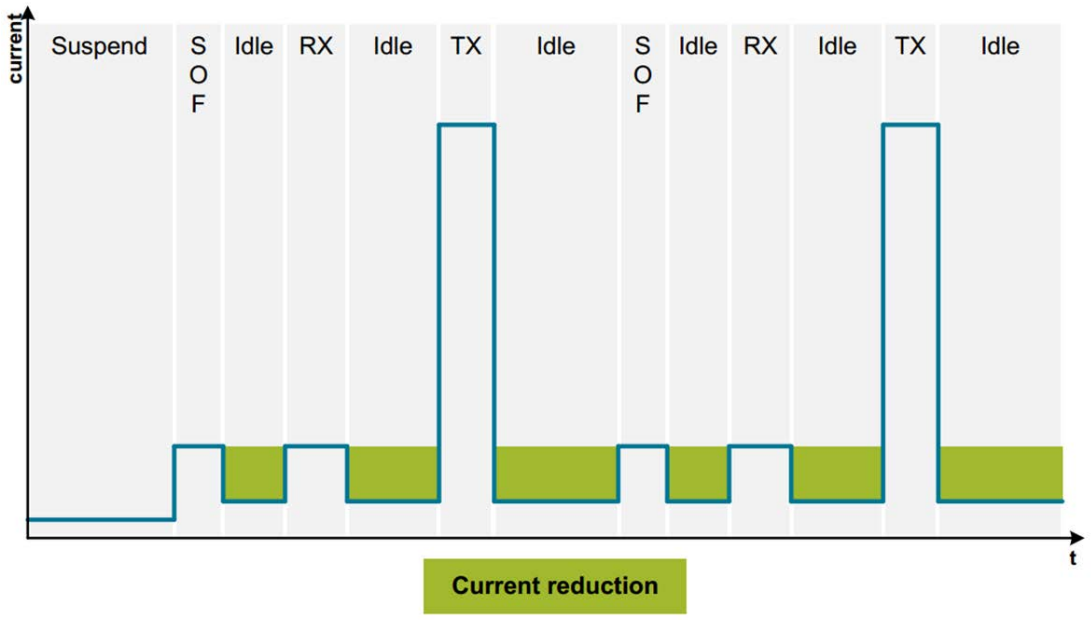

Figure 4: A typical USB transceiver stays in “receive” mode when idle, wasting 3–5 mA. With LEM techniques, the transceiver is kept in a low current mode similar to suspend.

EFM32 Happy Gecko—the USB MCU for the Future of the IoT

When examining the evolution of the USB interface, it’s clear that the next step is to make USB the universal and power-friendly solution for battery-powered devices. MCUs like Silicon Labs’ EFM32 Happy Gecko make the minute decisions necessary to reduce power consumption dramatically, enabling USB to penetrate markets where it has not yet succeeded to its fullest potential.

Making Electronics Smart, Connected, and Energy Friendly

Silicon Labs (NASDAQ: SLAB) is a leading provider of silicon, software and system solutions for the Internet of Things, Internet infrastructure, industrial control, consumer and automotive markets. We solve the electronics industry's toughest problems, providing customers with significant advantages in performance, energy savings, connectivity and design simplicity. Backed by our world-class engineering teams with unsurpassed software and mixed-signal design expertise, Silicon Labs empowers developers with the tools and technologies they need to advance quickly and easily from initial idea to final product.