AN0885: EFM8UB Low Energy USB Overview

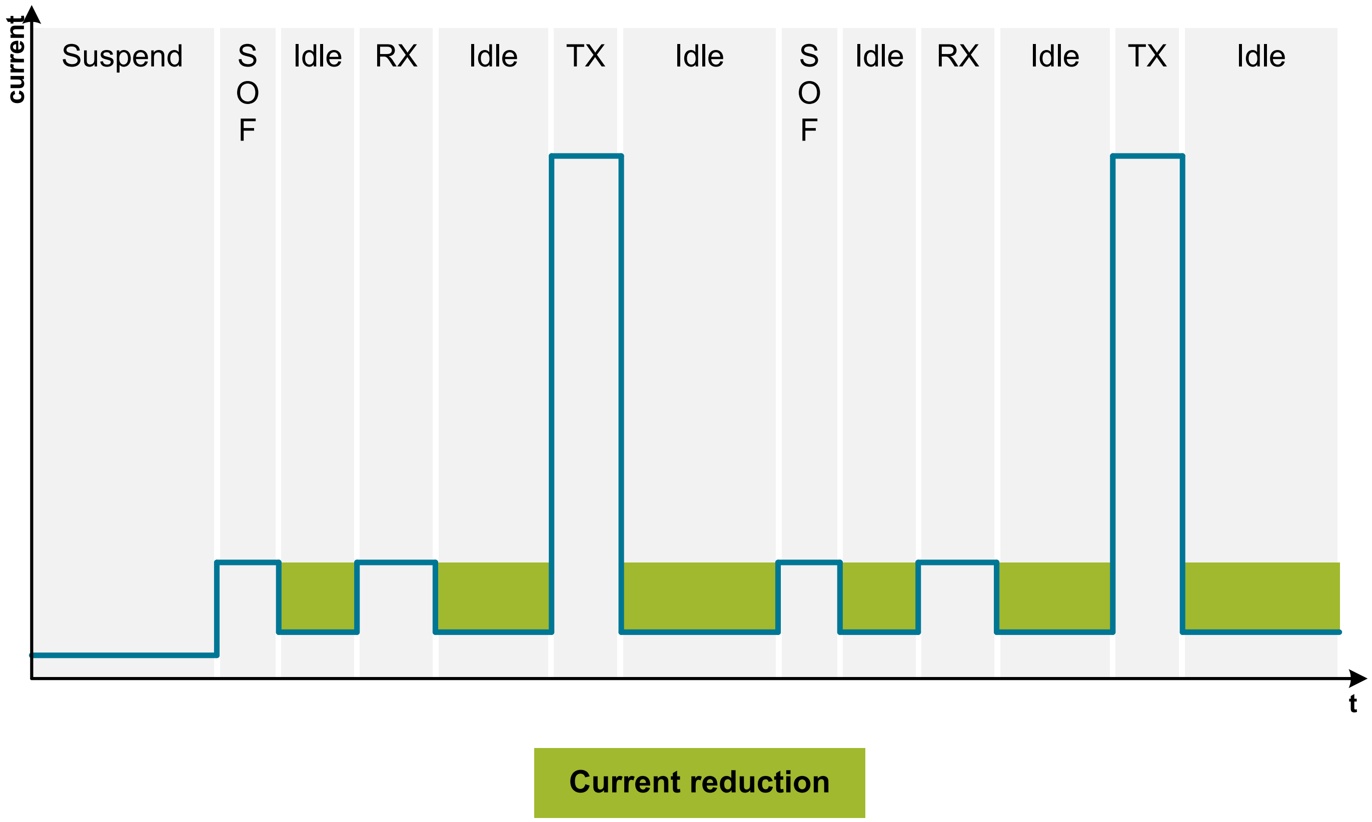
More USB peripherals are being connected to things with batteries or are powered from a battery.

Systems are not just a guaranteed 100 mA any longer. MCUs with good power begin to look more attractive in the USB space, especially when those USB interface MCUs aren't the only element in the system. In one of these systems, most of the power is consumed just communicating over USB. On the EFM8UB1 devices, Silicon Labs has implemented techniques to reduce the amount of power consumed by the USB module and system-level hooks that can reduce the overall power consumption of the entire device.

This document discusses these features and how they can be leveraged to reduce system power consumption in a USB application. Note that this document assumes some familiarity with USB transfer types and terms.

KEY POINTS

- Most of the energy in low energy USB systems is consumed in the USB communcation.
- The low energy USB features of the EFM8UB1 Universal Bee family can dramatically reduce this USB energy consumption.
- The device can use low power modes when idle to further reduce device energy consumption.



1. Taking Advantage of Bus Idle Times

USB traffic is inherently bursty. Full-speed frames occur every millisecond, and the frame is shared amongst all the devices connected to the bus. From a single end-node's perspective, this generally means that much of the frame will be spent talking to other devices. Normal USB devices stay fully operational during this entire frame, even if they aren't actively communicating on the bus.

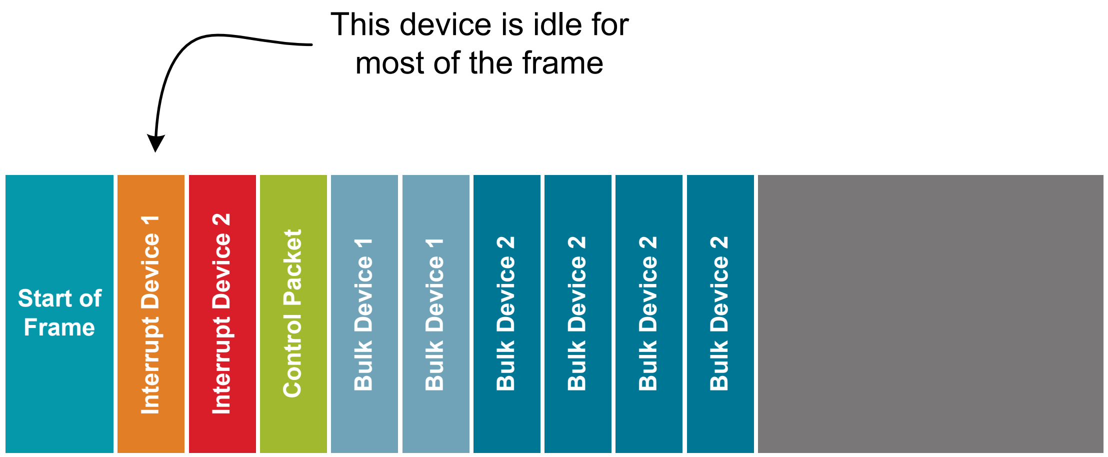


Figure 1.1. Example Full-Speed USB Frame

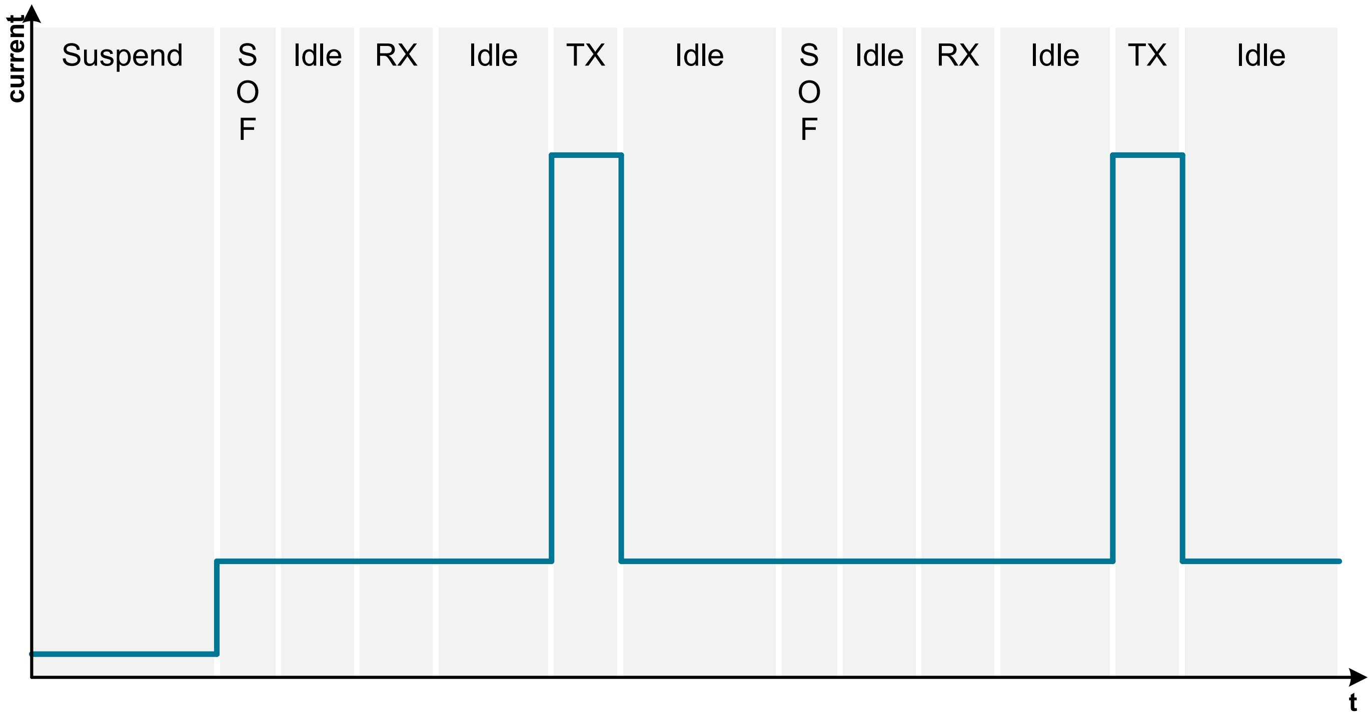


Figure 1.2. Example Standard Full-Speed USB Device Energy Consumption

Instead, the EFM8UB1 Universal Bee devices use two mechanisms to reduce the overall energy consumed during standard USB bus traffic whenever the device USB transceiver is idle:

- Reducing transceiver energy consumption.
- Gating the clock to the USB peripheral.

Additionally, the hardware has different options for when to move to a low-energy mode. The two options are:

- Move to a low-energy mode any time the bus is idle.
- Move to a low-energy state when NAKing an OUT packet.

A low-energy mode when the bus is idle saves the most energy in most systems. The second option of moving to a low-energy state when NAKing an OUT packet is mostly intended for situations where the device is actively receiving lots of data from the host in a Bulk application and having to NAK a lot of packets. The hardware can select either one of these options, none, or both.

As a result of these mechanisms, the EFM8UB1 Universal Bee family can lower the overall energy consumption of the system.

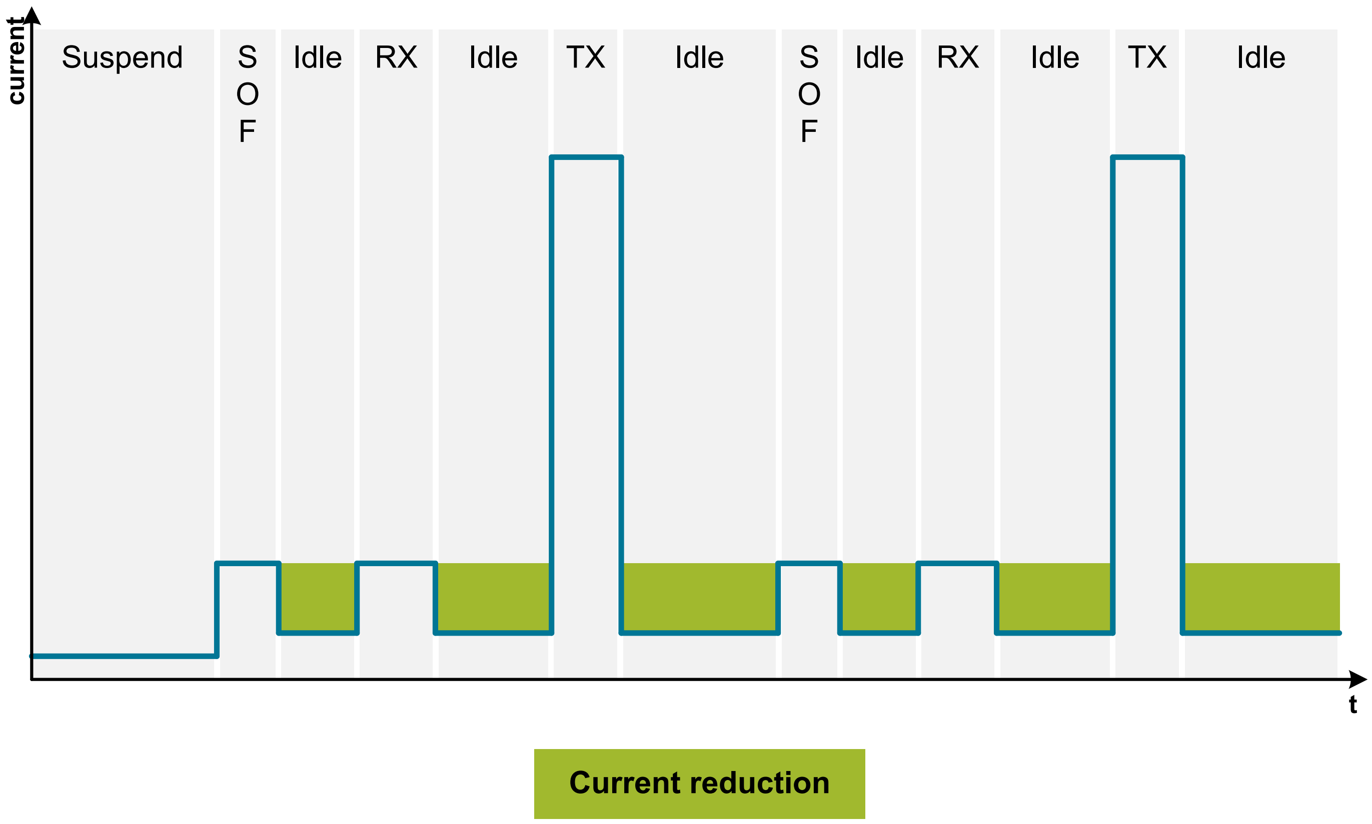


Figure 1.3. Example EFM8UB1 Universal Bee Full-Speed USB Energy Consumption

2. Bus-Loading Examples

The Low Energy USB features reduce more energy consumption the more the device is idle on the bus. This means that low transfer-rate USB transfer types like Interrupt transfers (i.e., HID) will conserve more energy with the Low Energy USB features enabled than high transfer-rate USB transfer types like Bulk transfers.

2.1 Bulk Transfers

Bulk transfers are usually the least energy-friendly, as there is always activity when a device is engaged with the driver. Typically, the low-level bulk driver will just fill the bus with requests, and using up the bandwidth available as a Bulk endpoint means little to no energy savings. However, Bulk data transfers tends to be bursty, and the low-energy techniques help applications during times when the device is active while there is no traffic on the bus.

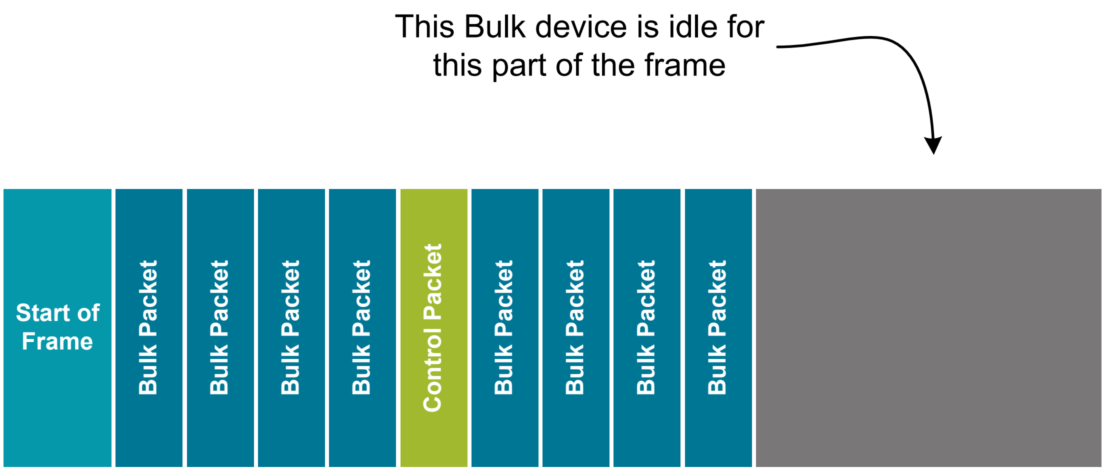


Figure 2.1. Example Bulk Device Traffic

Enabling low-energy when NAKing OUTs in the EFM8UB1 USB transceiver makes gains in some lower-bandwidth Bulk out scenarios. Reducing IN data requests from the host will also have a positive impact on energy consumption as well. In other words, if the host sends requests for the device less frequently when a data stream isn't active, this will help reduce the device energy consumption with the transceiver low energy features enabled. In PC-based systems, however, it may not be possible to control the number of IN packets sent to the device from the application level, as it is controlled by low-level host USB drivers.

2.2 Isochronous Transfers

The Low Energy USB benefit for Isochronous devices is potentially higher than Bulk devices. In this case, the energy savings will directly relate to the bandwidth used by the audio or video device. For example, a 16-bit 48 kHz audio device supporting two channels (left and right) and playback only will require 192 bytes per Full-Speed frame, or 3 packets. The device will be idle the rest of the frame and will be able to stay in a lower power mode.

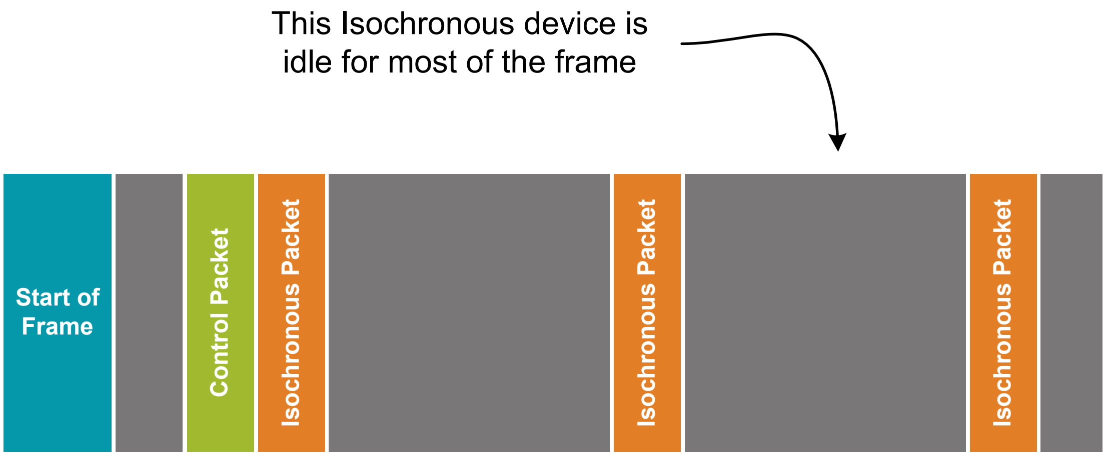


Figure 2.2. Example Isochronous Device Traffic

2.3 Interrupt Transfers

Interrupt transfers are the lowest bandwith of all the USB transfer types, with 64 bytes per millisecond in Full-Speed systems. These types of transfers are therefore able to make the most use of the Low Energy USB features, since the device is idle most of the time in each frame. Reducing the USB host polling rate of the device and the number of bytes sent per packet may also help reduce energy consumption.

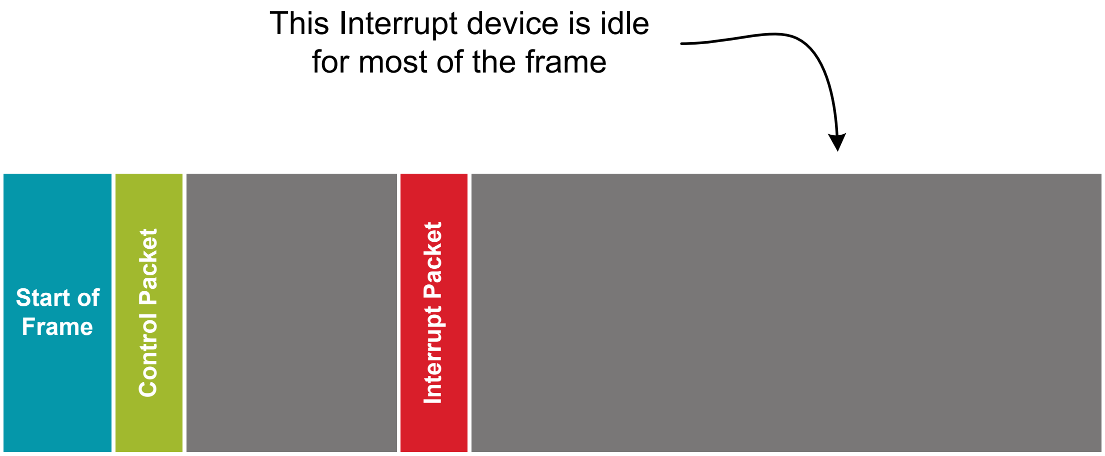


Figure 2.3. Example Interrupt Device Traffic

3. System-Level Savings

In addition to the Low Energy USB features themselves, the EFM8UB1 devices have additional ways to save power in the system.

3.1 Oscillator Management

The USB oscillator is a high-frequency, precision oscillator that runs at 48 MHz. Using this oscillator only for USB in the system will reduce the overall energy consumption of the device, since the core accesses flash at a slower rate. This can save energy in systems that do not need the extra instructions per second when running on a faster clock to perform other tasks.

3.2 Power Mode Management

If the core is not performing other tasks in between USB packets, the system can take advantage of the lower power modes available on the EFM8UB1 devices to reduce system energy consumption. For example, the core can switch to Idle mode in the main loop to reduce the device current consumption by approximately 2 mA in between packets. This can be a significant percentage of the device or system current consumption.

4. Low Energy USB Example

The HID Joystick demo for the EFM8UB1 devices demonstrates the Low Energy USB features discussed in this document. The example uses all three hardware features in the USB transceiver to reduce power consumption. In addition, the example switches to Idle mode running from the Low Frequency Oscillator (80 kHz) in the main loop whenever it's not sampling the joystick or buttons (every 1 ms) to further reduce the device power consumption.

Table 4.1. USB Joystick Current Consumption Comparison

Example Settings			Current Consumption (mA)	% Energy Consumed Compared to No Low-Energy Options
Low Energy USB	Switching to LFOSC0 in Main Loop	Idle in Main Loop		
Off	Off	Off	9.34	—
Off	Off	On	7.08	75.8%
Off	On	Off	4.7	50.3%
Off	On	On	4.7	50.3%
On	Off	Off	5.59	59.9%
On	Off	On	3.94	42.2%
On	On	Off	1.55	16.6%
On	On	On	1.55	16.6%

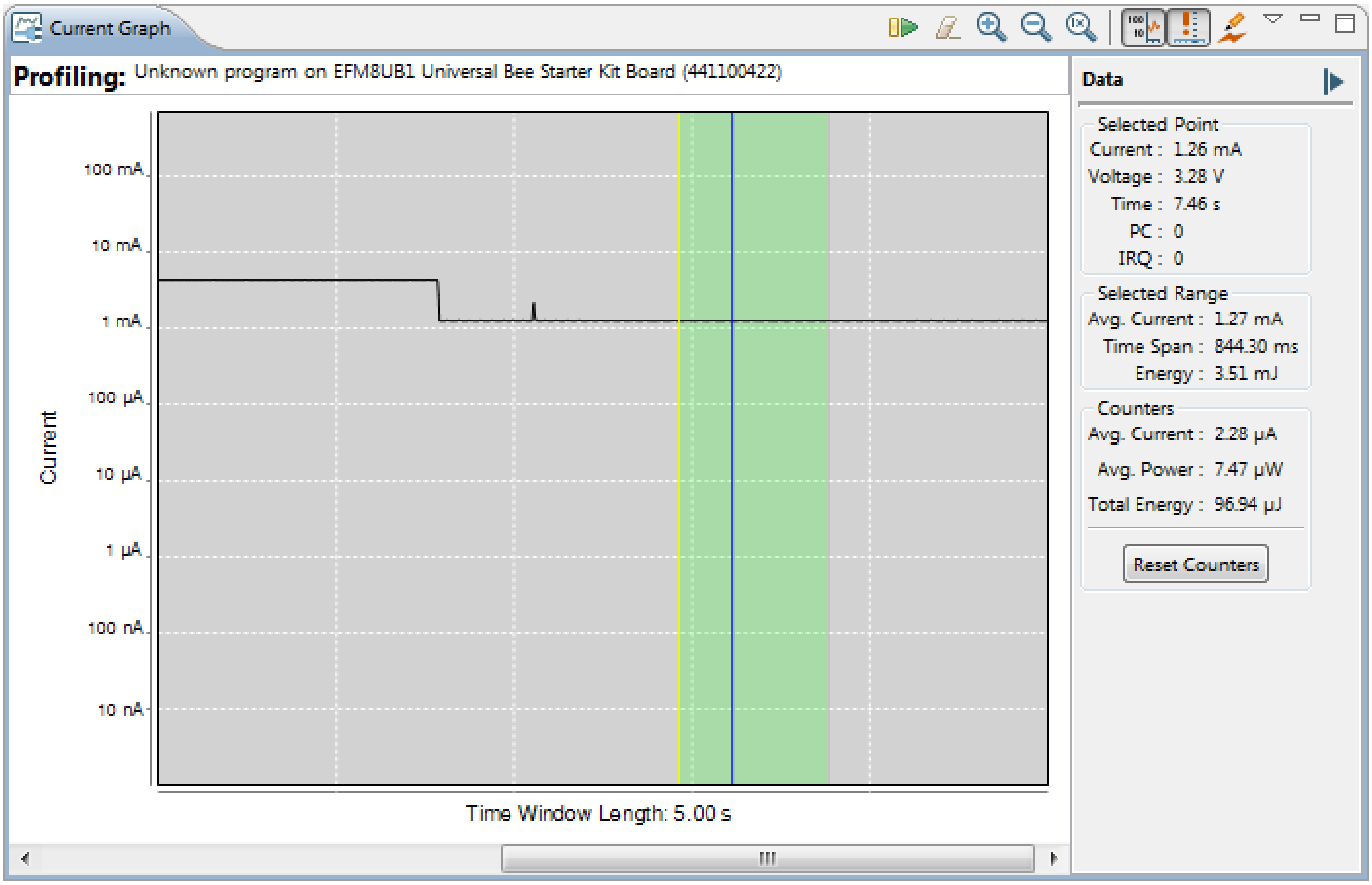


Figure 4.1. Using Advanced Energy Monitoring (AEM) with the EFM8UB1 USB Joystick Demo

After downloading the latest version of Simplicity Studio and installing using the Full or 8-bit MCUs options:

1. Provide power to the board by connecting the DBG USB connector to the PC using the provided USB cable.
2. Move the switch to the AEM position.
3. Click the [**Refresh detected hardware**] button and select the EFM8UB1 Universal Bee Starter Kit under [**Detected Hardware**].
4. Click the [**Demos**] tile under [**Tools**] to load the available demos.
5. Click the [**USB Joystick**] demo and click [**Start**] to download and run the demo.

To view and debug the source code for the USB Joystick demo:

1. Click the [**Software Examples**] tile under [**Software and Kits**].
2. In the wizard, select the EFM8UB1 Starter Kit and click [**Next**].
3. Select the [**USB Joystick**] from the list under the [**Demos**] folder and click [**Next**].
4. Click [**Finish**].
5. Click the [**Debug**] button in the IDE to build and download the code to the hardware.
6. Click the [**Resume**] button to start running the example.
7. Connect the device USB connector on the board to the PC. The board can now be used as a joystick or game pad on the PC.

Simpilcity Studio

One-click access to MCU tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

www.silabs.com/simplicity

MCU Portfolio
www.silabs.com/mcu

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com