EFM32
... the world's most energy friendly microcontrollers

EFM32 as USB Host

AN0801 - Application Note

This application note introduces the EFM32 USB Host stack and explains how to configure the EFM32 as a USB Host.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

ARM ZERO TINY GECKO LEOPARD GIANT WONDER

SILICON LABS

1 Introduction

Several EFM32 families include parts with integrated USB support. See the data sheet or product selector for information about which devices contain USB. The EFM32 USB can be operated in both host and device mode. This document discusses how to use the USB in *host* mode.

Some features of the EFM32 USB include:

- Fully compliant with USB Specification Revision 2.0
- Full-speed USB (12 Mb/s) host and device
- Support for Control, Bulk and Interrupt transfers
- Up to 12 configurable endpoints (6 IN, 6 OUT) in addition to EP0
- 2 kB of endpoint memory
- Low power mode with reset/resume detection

The EFM32 comes with a full USB Stack (software API). The USB stack is provided as C source files and full doxygen documentation is available.

1.1 More Information

In addition to this document, Silicon Labs provides several other resources on USB:

- usb Doxygen Documentation
- AN0042 USB-UART Bootloader
- AN0046 USB Hardware Guidelines
- AN0065 EFM32 as USB Device
- Kit examples
  - USB CDC Virtual COM port (usbdcdc)
  - USB HID Keyboard (usbdhidkbd)
  - USB Mass Storage Device (usbdmsd)
  - USB Vendor Unique Device (usbdvud)
  - USB Device Enumerator (usbhenum)
  - USB Host HID Keybard (usbhhidkbd)
  - USB Host Mass Storage Device (usbhmsdfatcon)

The reader should also be familiar with the USB Specification Revision 2.0 available from the Universal Serial Bus Implementers Forum (USB-IF).

There are several other good resoures on USB online. Two of them are listed here:

http://www.usbmadesimple.co.uk/

http://www.beyondlogic.org/usbnutshell/usb1.shtml

2 USB Host

When the EFM32 is operating as a USB Host it is responsible for identifying and enumerating connected devices, and initiate all communication over the USB Port.

2.1 Identification

When a USB Device is connected the USB Host will first read the device and and configuration descriptors in order to verify that it recognizes the device and knows how to communicate with it.

The descriptors provide information about the USB Device, including the manufacturer, product version, power consumption and which (if any) of the standardized USB classes the device implements.

As an example, if the USB Host should communicate with a memory stick it would verify that the device implements the Mass Storage Device (MSD) USB class. It can also put limits on specific vendors or products.

2.2 Enumeration

Enumeration is the process of assigning a unique ID to the device and selecting a configuration.

In a generic USB setup there can be multiple USB Devices connected to the same USB Host through one or several USB Hubs. It is therefore necessary for the Host to assign a unique ID to each Device. The EFM32 USB Host Stack does not support hubs and multiple connected devices and the device management is therefore simpler. However, in order to comply with the USB specification it must issue a device ID (different from zero) before any other communication can take place.

During enumeration the USB Host will read out the *configuration descriptor(s)* of the device and tell the device which configuration to activate. USB Devices can have multiple *configurations*, but only one can be active at any given time. Once the configuration has been set, the device is ready to communicate.

The USB Host also needs to map *endpoints* to *host channels* in order to complete the *pipes* (logical channels) used for communication.

2.3 Host Channels

The EFM32 USB allows software to use 12 (6 IN, 6 OUT) *host channels* (similar to *endpoints* for a USB Device) for communication (in addition to the control pipe). The host channels has 2 kB of shared FIFO RAM and are serviced by the internal USB DMA controller which minimizes CPU usage during USB transactions.

3 Power Requirements

If a connected USB Device is bus-powered (draws all its power from the USB bus), the USB Host is responsible for providing the power required by the USB Device. Regardless if the device is self-powered or bus-powered the USB Host is always responsible for providing 5V on the VBUS line.

Before a USB Device is configured it may not draw more than 100 mA. In the configuration descriptor, the USB Device specifies the maximum amount of current it will draw once configured up to a maximum of 500 mA. The USB Host can use this information to choose whether or not to configure the device.

3.1 Overcurrent Protection

The EFM32 USB Host stack provides a method to detect an overcurrent condition. This requires an external device to measure the current draw of VBUS and signal a GPIO interrupt to the EFM32 on the overcurrent condition. The GPIO interrupt can be configured to use any input pin on the EFM32. See the kit schematics for an example of the external circuit.

Overcurrent detection must also be enabled in software. See the usbconfig.h file on how to enable overcurrent detection. On an overcurrent event the USB stack will cut power to VBUS, thus disconnecting the USB Device.

3.2 Suspend and Resume

The USB Host may place the device in Suspend mode to reduce current draw from the device. When the device is in Suspend mode it may not draw more than 2.5 mA.

When the device is suspended, no communication takes place. In order to resume communication, the host must issue a Resume command to the device. The EFM32 USB Host Stack does not support 'Remote Wakeup' (which allows the device to initiate a resume).

Suspending the USB device limits the current draw from the device, but the USB Host must still be in energy mode EM0 or EM1 (because USB is active). Instead of suspending a device for longer periods it can be beneficial to disconnect and reconnect at a later time in order to achieve more sleep time.

4 Software Example

The software example in this application note is meant to be used together with the USB Device example in AN0065 EFM32 as USB Device. This requires two EFM32 Kits: one running the USB Host firmware in this application note and one running the USB Device firmware from AN0065. By connecting both kits with a USB cable this can demonstrate how to do generic (vendor specific) USB communication between the two EFM32s.

When starting the example it will initially stay in EM2 to save power when USB is not in use. When PB1 is pressed or a USB cable is plugged in it will attempt to start a USB session. If a compatible device is connected to the USB port it will connect, enumerate the device and start sending periodic messages to the device. It will also listen for messages sent from the device. These can be generated by pressing the push buttons on the kit acting as the USB device.

The example also demonstrates how to suspend/resume the USB Device.

Figure 4.1. USB host state machine

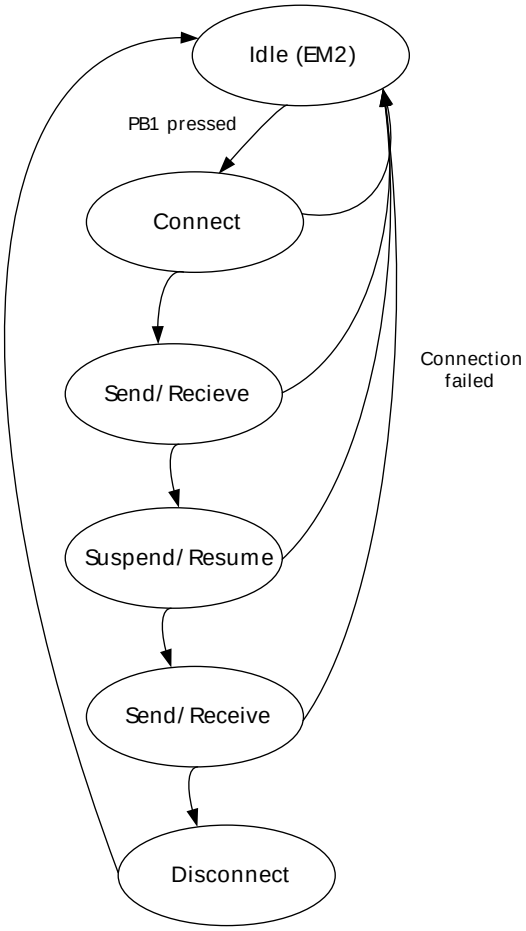
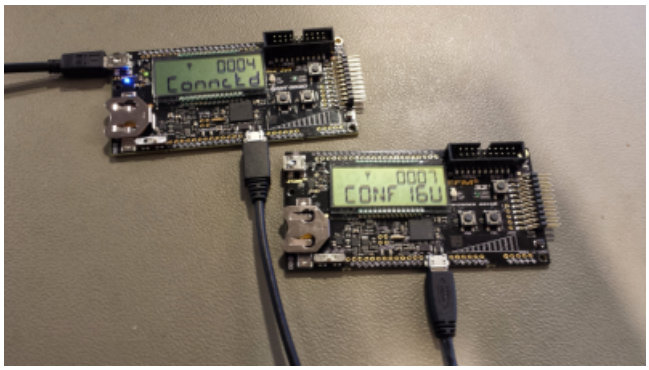


Figure 4.2 (p. 6) shows two EFM32GG-STK3700s connected over USB. One acting as a USB Host (top left), the other as a USB Device (bottom right). The LCD display shows the current state and the number in the top right of the display indicates how many messages have been received.

Figure 4.2. Two EFM32GG-STK3700s connected over USB



Note

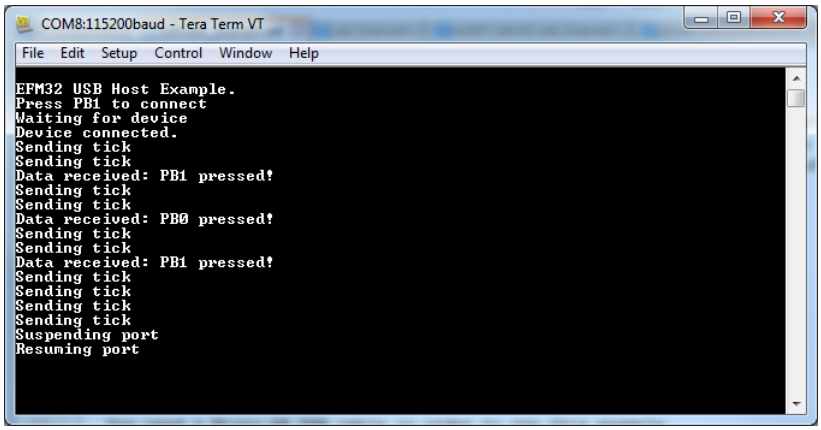
You need a Micro-AB USB cable in order to run this example (Micro-A plug at one end and Micro-B plug at the other).

4.1 Debug output

The example code and USB stack provides debug output to visualize the communication and give information about errors. When running the DK example, the output is printed to the TFT display. On the STK, the output is sent over UART. The output can be read with a UART adapter and a terminal emulator on the PC such as Tera Term.

The default configuration uses the PD0 pin (pin 4 on the STK expansion header) as TX (connect to RX on the UART adapter). The UART configuration is 115200 baud, 8 data bits, no parity and one stop bit.

Figure 4.3. Debug output in Tera Term



5 Revision History

5.1 Revision 1.02

2014-05-07

Updated example code to CMSIS 3.20.5

Changed source code license to Silicon Labs license

Added project files for Simplicity IDE

Removed makefiles for Sourcery CodeBench Lite

5.2 Revision 1.01

2013-12-19

Changed application note numbering to 4 digits.

5.3 Revision 1.00

2013-11-04

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

B Contact Information

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

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4