AN740: Using the Ember® EM358x/EM359x USB

This application note provides key information for using the Ember® EM358x/359x USB. The goal of the EM358/359x USB is to act as a COM port replacement for UART-style functionality by enumerating as a communications data class (CDC) USB device.

KEY FEATURES

- Breakout board configuration
- Host PC driver
- Terminal programs
- On-chip firmware driver
- Basic serial port functionality
- USB certification

1 Breakout Board Configuration

The 710-0680-000 C breakout board (EM358x) has a USB connector located next to the RCM. To use USB on this breakout board, the J40 jumper next to the USB connector must be set in the “USB EN” state. With USB enabled, PA0, PA1, PA2, and PA3 are not functional anywhere else on the breakout board.

The EM359x USB connector is located on the RCM. When used with the 710-0682-000 A0 breakout board, the J2/J3 Vin selector should be disconnected.

1.1 GPIO Configuration and Control

The USB driver code will ignore any existing GPIO configuration by directly setting PA0, PA1, PA2, and PA3 as needed. PA0 and PA1 are the only pins that support USB data. The default configuration of the driver is to use PA2 for enumeration control and PA3 for VBUS monitoring. Board file abstraction allows PA2 and PA2 functionality to be changed to other GPIO.

The board file uses the macro USB_SELFPWRD_STATE to define if the system is bus-powered or self-powered by a source other than the USB connection. By default USB_SELFPWRD_STATE is set to 1 indicating the device is self-powered. When USB_SELFPWRD_STATE is set to 1, VBUS monitoring is required. To conform to USB Certification, when USB is not physically connected (no voltage on VBUS) enumeration control will be configured for de-enumeration and the on-chip driver will not allow enumeration. When USB is physically connected, VBUS monitoring will sense the VBUS change and enumeration control will be returned to the desired enumeration state.

1.1.1 PA0 and PA1

PA0 and PA1 are for D- and D+ bus signaling and configured as GPIOCFG_ANALOG.

Board file abstractions use USBDM for PA0 and USBDM for PA1 in the names.

1.1.2 PA2

PA2 is for enumeration control. The EM358x/359x must always be able to control enumeration. PA2 connects to the necessary pull-up resistor allowing software on the device to specifically enumerate or de-enumerate as needed. When enumerated, PA2 is set to GPIOCFG_OUT in a logic high state. When de-enumerated, PA2 is set to GPIOCFG_IN.

Board file abstractions use ENUMCTRL in the names.

1.1.3 PA3

PA3 is configured as GPIOCFG_IN for monitoring VBUS to know if the USB cable is physically connected. When a design is self-powered, VBUS monitoring is required. The 5 volt VBUS must be routed through a resistor-divider or similar circuit so that the EM358x/EM359x sees an acceptable voltage level on an input GPIO.

Board file abstractions use VBUSMON in the names.

1.1.4 PA3 interaction with PA2

For USB devices that are configured as self-powered, the USB specification requires that the enumeration pull-up resistor only be applied if VBUS is present. If PA3 does not sense VBUS, PA2 will be configured as de-enumerated. The on-chip driver will not allow PA2 to attempt enumeration until PA3 detects VBUS. The on-chip driver will monitor for any change of VBUS state. PA2’s actual configuration will appropriately track the desired enumeration state and the VBUS state.

1.1.5 VBUS Monitoring

To properly monitor a change in VBUS an IRQ must be used. The IRQ must be configured to monitor both a rising and falling edge of VBUS to allow PA2 to be configured properly. The default configuration is to use PA3 for VBUS monitoring which means a selectable IRQ must be used. The two available selectable IRQs are IRQC and IRQD. IRQD is the default choice. The behavior of the VBUS monitoring IRQ is derived from the behavior of buttons.

2 Host PC Driver

2.1 Windows

The EM358x/359x USB device is designed to work with the Silicon Laboratories USB CDC driver, which builds upon the Windows USB CDC driver (usbser.sys). This driver is digitally-signed by WHQL and can be optionally installed alongside Simplicity Studio. Installers for 32-bit and 64-bit PCs can be run to install the driver without using Windows Device Manager. Alternatively, when the EM358x/359x is connected to the computer via USB, direct Windows Device Manager to tool/EM358CDCDriverInstaller/SiLabs-CDC.inf to install the driver software. After installation, Windows Device Manager should now list **Silicon Labs CDC Serial port** under **Ports (COM & LPT)**. Nothing more is needed to configure this new device.

The USB serial number the EM358x/359x device reports to Windows is the EUI64 of the chip. That means that if the physical USB port is changed on a given machine the EM358x/359x will enumerate as the same COM port number and two EM358x/359x devices on the same machine won't result in the same COM port number.

2.2 Mac OSX and Linux

CDC USB devices on Mac OSX and Linux do not require any additional driver software to enumerate.

3 Terminal Programs

With the EM358x/359x acting as a COM port, any terminal program, such as PuTTY (Windows), HyperTerminal (Windows), or miniterm (Mac OSX/Linux), or scripting languages that can interact with a COM port should work. Classic UART configurations such as baud rate, parity bits, and flow control do not have any effect in these terminal programs and can be ignored.

4 On-Chip Firmware Driver

The USB driver files can be found at hal/micro/cortexm3/usb/ and should be included in any application wanting to use USB in its build. The USB driver source code does not have to be modified for basic serial port functionality, but for custom USB devices, refer to the USB sections of the Reference Manual and the Application Framework API. Changes to which GPIO are used for ENUMCTRL and VBUSMON should be done in the board file. The USB data lines are fixed to PA0 and PA1.

5 Basic Serial Port Functionality

Because the USB driver is designed to work below the UART and serial library, the serial library will function as it does for the UART. While the UART uses port 1, the USB uses port 3.

5.1 Build Options

The following build options should be used:

```
EMBER_SERIAL3_MODE=EMBER_SERIAL_FIFO
EMBER_SERIAL3_TX_QUEUE_SIZE=128
EMBER_SERIAL3_RX_QUEUE_SIZE=64
EMBER_SERIAL3_BLOCKING
```

The TX and RX queue sizes are best left at 128 and 64, respectively, or greater for optimal throughput with USB. Smaller values could result in the USB getting locked up, which could trigger a watchdog reset.

The USB implicitly blocks serial port behavior as dictated by the host system so the serial build options should indicate this.

5.2 Initialization

```
emberSerialInit(3, 0, 0, 0);
```

Because the last three parameters are ineffective with USB, they can simply be left as 0. Initialization not only sets up the serial library but also initializes the chip's USB driver and causes the device to enumerate. Refer to section **GPIO Configuration and Control** for more detail as to what state the GPIO are set.

5.3 Reading and Writing

All serial functions, such `emberSerialPrintf(3, "foo");` and `emberSerialReadByte(3, *dataByte);` behave as they do with the UART.

`emberSerialGuaranteedPrintf` will also function but being on USB, timing is not guaranteed. Due to the fact that both Printf are blocking, it is safe to interleave uses of `emberSerialGuaranteedPrintf` and `emberSerialPrintf` knowing that one will run to completion before the other is executed.

5.4 Suspend, Resume, Wakeup

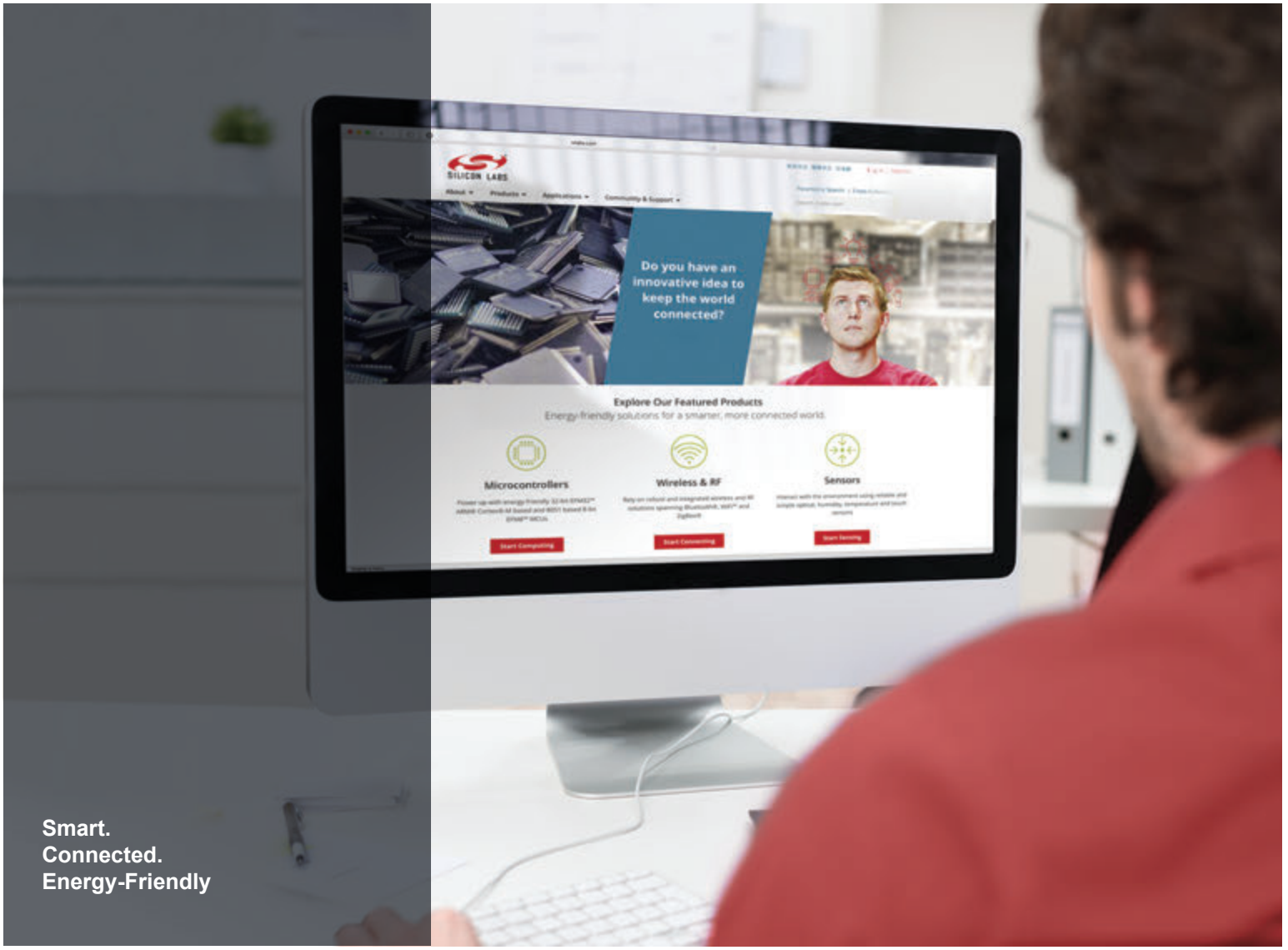
In order to properly suspend and resume the device, the USB device driver provides a delayed service routine which will reduce the main clock by ¼ when the device has detected a suspend signal. To enable suspend in an application, simply include `usbSuspendDsr();` in the main loop. Note that this function will abruptly shut down the stack and all peripherals in order to reduce power consumption to match bus powered USB certification specifications (<2.5 mA for remote wakeup enabled devices).

Remote wakeup can be enabled using the USB_REMOTEWKUPEN_STATE parameter found in the board header. The function `USBD_RemoteWakeup` will trigger a remote wakeup from the device.

6 USB Certification

The USB driver provides a simple interface for an application to support suspend, resume, or remote wakeup, but is not guaranteed to meet USB specification for all applications. Additionally, the clocking requirements necessitate reducing the frequency of all on-chip clocking by a factor of 4, which impacts timing dependent peripherals such as but not limited to Serial Controller 1s UART mode or the General Purpose Timers.

Note: The integrated USB on the EM358x and EM359x may fail to enumerate. For more information including effected conditions and the work-around, see the EM358x/EM359x Errata.



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

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

http://www.silabs.com