Energy Micro University

UM005 - Peripherals

In this lesson we look at how to control peripherals through manipulating registers and using library functions. Three peripherals are used as examples, but the techniques are applicable to other peripherals.

Concepts that will be introduced include:

- **Controlling peripherals through registers**
- **Controlling peripherals using library functions**
- **Analog to Digital Converter (ADC)**
- **Digital to Analog Converter (DAC)**
- **Universal Synchronous Asynchronous Receiver/Transmitter (USART)**

This lesson includes:

- **This PDF document**
- **Source files**
  - **Example C code**
  - **Multiple IDE projects**

ENERGY micro

1 Introduction

1.1 What are peripherals?

Peripherals add functionality to MCUs – they can work autonomously on their own or together with other peripherals. The peripherals are connected to each other, the CPU and memory through a bus, which is a mechanism that transfers data between components.

1.2 How to control peripherals through registers?

The peripherals are controlled by turning features on or off in their associated control registers. A register is a small amount of memory that can be read from and written to.

2 Analog to Digital Converter (ADC)

2.1 What is an ADC?

An ADC is as its name suggests, a device that converts analog signals into a digital representation. The input is a voltage that is compared to a reference voltage. The input has to be smaller or equal to the reference voltage, since the output is a digital representation of the ratio input/reference where the highest number represents that they are equal. For an 12-bit ADC the output value would be in the range 0 to 4095. The ratio input/reference would therefore be represented by output/4095.

2.2 Resolution and sampling rate

The ADC converts continuous analog signals to discrete digital representations. The resolution of the ADC is the number of discrete values it can produce over the range of analog values. The resolution is usually represented in bits.

The sampling rate is the number of new digital representations sampled from the analog signals per second.

The EFM32 has a Successive Approximation Resolution (SAR) architecture with a resolution up to 12 bits and a sampling rate of up to one million samples per second. A 12-bit resolution provides $2^{12} = 4096$ different discrete values. A SAR architecture uses a binary search algorithm to converge to the sampled input signal. The incoming analog signal is compared to a signal made by a Digital to Analog Converter (DAC) which is the converse operation of the ADC. The SAR register is updated based on whether the incoming analog signal is greater or smaller than the signal produced by the DAC. The reference signal from the DAC is also updated and the comparison is continued until convergence.

2.3 Conversion phases

The conversion consists of two phases; the acquisition phase where the input is sampled and the conversion to digital representation phase where the digital approximation is performed. The acquisition phase is needed in order to have a fixed voltage to perform the binary search on.

2.4 Input channels

The ADC is connected to 18 input channels; 8 external pins, that can be either configured as 8 single ended channels or 4 differential channels, and 6 internal single ended channels. The internal channels are channels within the MCU. When using single ended inputs the incoming analog voltage is compared to a reference voltage. Differential ended input means that two neighboring channels are connected, and the difference between these two voltages are measured.

2.5 Single and scan conversions

In single mode, a single conversion is performed per trigger. It can also be done repetitively if the REP bit in ADCn_SINGLECTRL is set. The result found in the ADCn_SINGLEDATA register will be overwritten if it is not being read. In scan mode a sequence of samples are done per trigger. This can also be done repetitively. The sequential samples are done via several channels. The samples is done one by one, channel by channel, but there is no need for a new trigger between each sample.

2.6 Reference voltages

The reference voltages can be selected from the following sources:

- 1.25 V internal bandgap
- 2.5 V internal bandgap

- V_{DD}
- 5 V internal differential bandgap
- External single ended input from pin 6
- Differential input, 2x(pin 6 - pin 7)
- Unbuffered 2xV_{DD}

Bandgap is a temperature independent voltage reference circuit. V_{DD} is the power supply of the MCU. Unbuffered voltage means that it is directly connected to the ADC supply voltage, which makes it more susceptible to supply noise.

2.7 Warm-up time

Before a conversion can take place the ADC needs to warm up. When enabling the ADC it must be warmed up for 1 µs. This is also the case if the reference voltage is changed between samples. If the bandgap is chosen as reference, the ADC needs an additional 5 µs warm-up time. The ADC is normally turned off until it receives a request, and the warm-up procedure starts. However, if short latency is required, the ADC and the bandgap reference warm-up can stay warm between the sample requests. The bandgap reference can only stay warmed up in scan mode.

3 ADC Code Example

The single conversion examples configure the ADC to sample $V_{DD}/3$ with the 1.25 V bandgap reference. This setup makes it possible to measure the supply voltage of the chip. The voltage is calculated and displayed on the segment LCD which is updated 10 times every second. There are two equivalent versions of the example, where the second one is the exercise at the end of this module. The voltage is calculated using the following equation:

Example 3.1. Calculating the voltage

$(V_{DD}/3)/V_{ref}$ = (sample value)/4096

3.1 ADC example using register operations

In the attached `1_adc_register` example the registers are set in the main file. The registers that are set are the Control Register and the Single Sample Control Register. The following table is found in the EFM32GG Reference Manual, and consists of all the registers in the ADC.

Offset	Name	Type	Description
0x000	ADCn_CTRL	RW	Control Register
0x004	ADCn_CMD	W1	Command Register
0x008	ADCn_STATUS	R	Status Register
0x00C	ADCn_SINGLECTRL	RW	Single Sample Control Register
0x010	ADCn_SCANCTRL	RW	Scan Control Register
0x014	ADCn_IEN	RW	Interrupt Enable Register
0x018	ADCn_IF	R	Interrupt Flag Register
0x01C	ADCn_IFS	W1	Interrupt Flag Set Register
0x020	ADCn_IFC	W1	Interrupt Flag Clear Register
0x024	ADCn_SINGLEDATA	R	Single Conversion Result Data
0x028	ADCn_SCANDATA	R	Scan Conversion Result Data
0x02C	ADCn_SINGLEDATAP	R	Single Conversion Result Data Peek Register
0x030	ADCn_SCANDATAP	R	Scan Sequence Result Data Peek Register
0x034	ADCn_CAL	RW	Calibration Register
0x03C	ADCn_BIASPROG	RW	Bias Programming Register

It will be useful to look up the ADC registers to see how the values in the example code are found. We take a look at the Control Register:

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset					0x0							0x1F						0x00									0x0		0		0x0	
Access					RW							RW						RW									RW		RW		RW	
Name					OVSRSEL							TIMEBASE						PRESC									LPFMODE		TAILGATE		WARMUPMODE	

Some of the bitfields of a register give a distinct functionality to each value as in the OVSRSEL bitfield in the Control Register, see OVSRSEL (p. 6) Others like the TIMEBASE (TIMEBASE (p. 6))

register do not list each individual value because they follow a simple rule. In the case of TIMEBASE the value must be set equal to or higher than the number of clock cycles in 1 µs.

Bit	Name	Reset	Access	Description
31:28	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in*		
27:24	OVSRSEL	0x0	RW	**Oversample Rate Select**

Select oversampling rate. Oversampling must be enabled for each mode for this setting to take effect.

Value	Mode	Description
0	X2	2 samples for each conversion result
1	X4	4 samples for each conversion result
2	X8	8 samples for each conversion result
3	X16	16 samples for each conversion result
4	X32	32 samples for each conversion result
5	X64	64 samples for each conversion result
6	X128	128 samples for each conversion result
7	X256	256 samples for each conversion result
8	X512	512 samples for each conversion result
9	X1024	1024 samples for each conversion result
10	X2048	2048 samples for each conversion result
11	X4096	4096 samples for each conversion result

23:21	*Reserved*	*To ensure compatibility with future devices, always write bits to 0. More information in*		
20:16	TIMEBASE	0x1F	RW	**Time Base**

Set time base used for ADC warm up sequence according to the HFPERCLK frequency. The time base is defined as a number of HFPERCLK cycles which should be set equal to or higher than 1us.

Value	Description
TIMEBASE	ADC warm-up is set to TIMEBASE+1 HFPERCLK clock cycles and bandgap warm-up is set to 5x(TIMEBASE+1) HFPERCLK cycles.

Notice that the default/reset values do not need to be set. For a complete register map with descriptions please look in the EFM32GG Reference Manual.

4 Digital to Analog Converter (DAC)

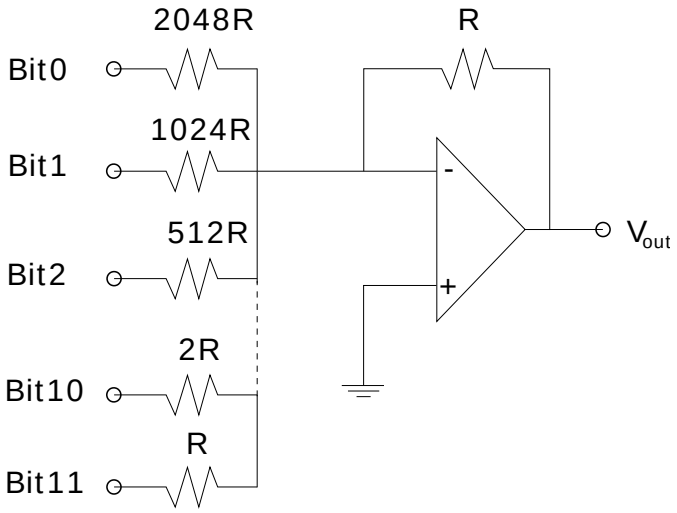
4.1 What is a DAC?

The DAC performs the converse operation of the ADC, namely converting digital values to analog signals; it is a voltage generator. Using DMA and a timer the DAC can be used to generate waveforms without any CPU intervention. If connected to a speaker this would produce sound.

4.2 Conversions

The DAC has two channels (channel 0 and channel 1) which have separate 12-bit data registers. These are used to produce two single ended outputs or one differential output. A simple DAC can be constructed by using a summing amplifier and set of resistors as input. DACs usually have a more advanced construction.

Figure 4.1. Showing the summing amplifier for a simple DAC



The resistors are of magnitude R, 2R, 4R,...,2048R, each to represent one of the 12 different input bits. The lowest value of R belongs to the most significant input bit. The output voltage becomes

$V_{out} = -V_{ref}((1/1)Bit11 + (1/2)Bit10 + (1/4)Bit9 + ... + (1/2048)Bit0)$

Bit0, Bit1, ... and Bit11 can take the values 0 or 1, and hence the DAC can produce 4096 different voltages.

4.3 Conversion modes

The DAC supports three different conversion modes; continuous mode, sample/hold mode and sample/off mode:

- In the continuous mode the two channels drive the output continuously without stop.
- In sample/hold mode the DAC core is turned off between samples, and the output is held in a sample/hold element. The output has to be refreshed by writing a new value to the DACn_CHxDATA register. The DAC core is woken up to perform the refresh. The refresh can also be done automatic by setting the bit REFREN in DACn_CHxCTRL.
- In sample/off mode both the DAC core and the sample/hold element is turned off between samples, which means that the output voltage must be held externally.

The two last modes consume less power than the first.

4.4 Reference voltages

There are three voltage references available which are selected by setting the REFSEL bits in the DACn_CTRL register:

- 1.25 V internal bandgap
- 2.5 V internal bandgap
- V_{DD}

4.5 Warm-up time

The reference selection can only be changed while both channels are disabled. Just as for the ADC the references for the DAC need to be enabled for some time before they can be used. This is called the warm-up time, and starts when one of the channels is enabled. For a bandgap reference, this period is 5 DAC_CLK cycles while the V_{DD} reference needs 1 DAC_CLK cycle.

5 DAC Code Example

5.1 Creating a programmable voltage source

The `3_dac_adc` example configures the DAC in continuous mode and converts a single value continuously. The value is then sampled by the ADC and displayed on the LCD to verify that the DAC outputs the correct voltage. The example uses an internal connection between the DAC and the ADC, but by changing `adcSingleInpDACOut0` to `adcSingleInpCh3`, pin D3 will be sampled by the ADC instead. The DAC is already configured to output both to the B11 pin and the ADC, so after reconfiguring the ADC a piece of wire can be used to connect B11 and D3 to obtain the same result as the internal connection. Since the DAC is operating in continuous mode, the output voltage will be stable even if the output is loaded, i.e. if the DAC is to power some component. The fact that the DAC is continuously working will be reflected in the power consumption. The DAC output is measured by the ADC, and the result is printed on the LCD.

5.1.1 Looking in register maps

In the first ADC code example register operations were used. In this example we use the emlib functions. It is nevertheless useful to look in the register maps to see the description of the different bitfields and which values they can take.

In this example it is important to set the OUTMODE bitfield in the DACn_CTRL - Control Register. The default setting makes it possible for the DAC to give output to pins, but not directly to the ADC. The bitfield should be set to PINADC to allow output to the ADC.

Offset	Bit Position																													
0x000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset											0x0			0x0									0x0		0	0	0x1		0x0		0	0
Access											RW			RW									RW		RW	RW	RW		RW		RW	RW
Name											REFRSEL			PRESC									REFSEL		CH0PRESCRST	OUTENPRS	OUTMODE		CONVMODE		SINEMODE	DIFF

Bit	Name	Reset	Access	Description
5:4	OUTMODE	0x1	RW	**Output Mode**

Select output mode.

Value	Mode	Description
0	DISABLE	DAC output to pin and ADC disabled
1	PIN	DAC output to pin enabled. DAC output to ADC and ACMP disabled
2	ADC	DAC output to pin disabled. DAC output to ADC and ACMP enabled
3	PINADC	DAC output to pin, ADC, and ACMP enabled

Using the default value will then lead to wrong measurement of the voltage. When trying to set the INPUTSEL bitfield in the ADCn_SINGLECTRL Register to DAC0OUT0, i.e. choosing channel 0 from the DAC as input for the ADC, it will fail. The program will still compile, but the ADC is not measuring the right voltage.

Offset	Bit Position																													
0x00C	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset		0x0						0	0x0					0x0							0x0						0x0			0	0	0
Access		RW						RW	RW					RW							RW						RW			RW	RW	RW
Name		PRSSEL						PRSEN	AT					REF							INPUTSEL						RES			ADJ	DIFF	REP

Bit	Name	Reset	Access	Description
11:8	INPUTSEL	0x0	RW	**Single Sample Input Selection**

Select input to ADC single sample mode in either single ended mode or differential mode.

DIFF = 0		
Mode	Value	Description
CH0	0	ADCn_CH0
CH1	1	ADCn_CH1
CH2	2	ADCn_CH2
CH3	3	ADCn_CH3
CH4	4	ADCn_CH4
CH5	5	ADCn_CH5
CH6	6	ADCn_CH6
CH7	7	ADCn_CH7
TEMP	8	Temperature reference
VDDDIV3	9	VDD/3
VDD	10	VDD
VSS	11	VSS
VREFDIV2	12	VREF/2
DAC0OUT0	13	DAC0 output 0
DAC0OUT1	14	DAC0 output 1
DIFF = 1		
Mode	Value	Description
CH0CH1	0	Positive input: ADCn_CH0 Negative input: ADCn_CH1
CH2CH3	1	Positive input: ADCn_CH2 Negative input: ADCn_CH3
CH4CH5	2	Positive input: ADCn_CH4 Negative input: ADCn_CH5
CH6CH7	3	Positive input: ADCn_CH6 Negative input: ADCn_CH7
DIFF0	4	Differential 0 (Short between positive and negative inputs)

For a complete register map with descriptions please look in the EFM32GG Reference Manual.

6 Universal Synchronous Asynchronous Receiver/ Transmitter (USART)

6.1 What is an USART?

The Universal Synchronous Asynchronous Receiver/Transmitter (USART) is a communication module that supports both synchronous and asynchronous communication (explained below). Several external peripherals like color LCDs and digital sensors can be controlled by the USART. It can also be used to communicate with a PC using a serial port or a serial to USB converter. The USART will only be briefly introduced here, for more information please see the Reference Manual and Application Note 8.

6.2 Synchronous mode (SPI)

The Serial Peripheral Interface Bus (SPI), see Figure 6.1 (p. 11), has a separate clock signal (SCLK) transmitted with the data. This allows the two ends, master and slave to know when to write and read data. Master writes to MOSI and reads from MISO, and slave does the opposite. Master initiates a transaction by setting SS_n low and ends it by setting SS_n high. Data is sent both ways at the same time as shown in Figure 6.2 (p. 11) .

Figure 6.1. Typical Serial Peripheral Interface Bus (SPI) setup

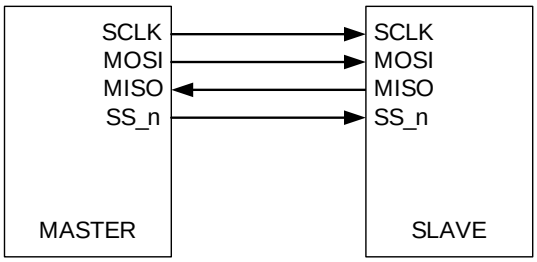
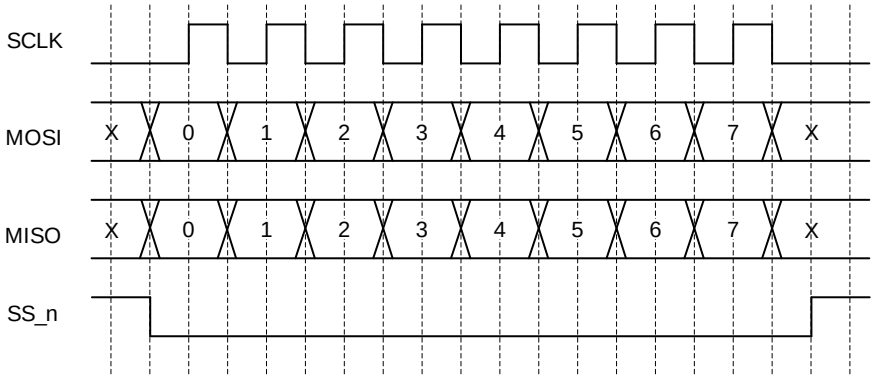


Figure 6.2. Typical SPI transaction



6.3 Asynchronous mode (UART)

An Universal Asynchronous Receiver/Transmitter (UART) does not have a separate clock signal, so the receiver has to determine when to sample. This is achieved by adding synchronization bits to the data. The transmitter and receiver must also transmit at the same baud rate, i.e. number of symbols to transmit/receive per second. A symbol can for instance be 1 byte that represents a letter.

Figure 6.3. Typical UART setup

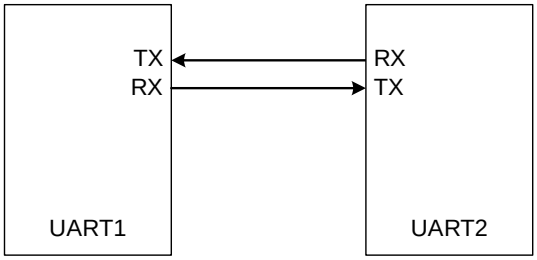
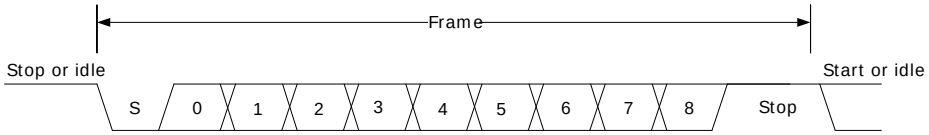


Figure 6.4. Typical UART transaction

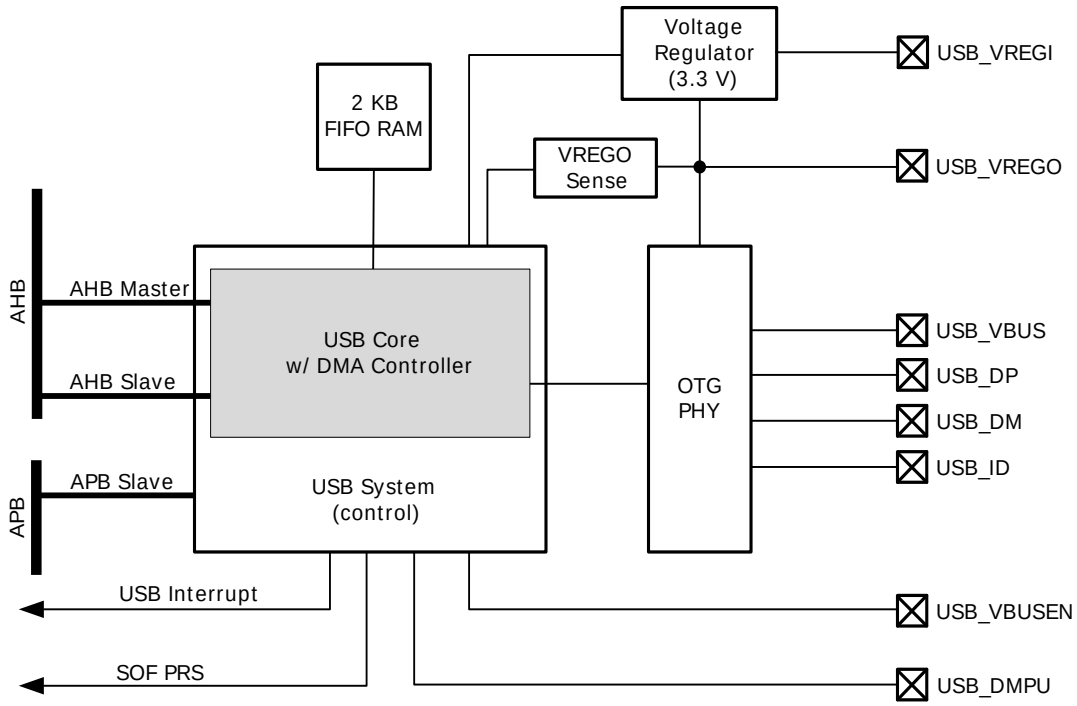


7 Universal Serial Bus (USB)

7.1 What is USB?

The Universal Serial Bus (USB) is an external bus standard that supports data transfers rates of 12 MBit/s (full-speed) and 1.5 MBit/s (low-speed). USB was designed to standardize the connection of computer peripherals such as keyboards and mass storage devices, both to communicate and supply electric power. The USB can be used in Device, On-the-go (OTG) Dual Role Device (explained below) or Host-only configuration. The on-chip 3.3V regulator delivers up to 50 mA and can also be used to power external components, eliminating the need for an external voltage regulator.

Figure 7.1. USB Block Diagram



The On-The-Go Physical Layer (OTG PHY) is explained in the next section.

7.2 On-the-Go (OTG)

USB On-The-GO (OTG) allows two USB devices to talk to each other without requiring the services of a personal computer. The OTG Dual Role Device is a device capable of funtioning as either host or peripheral. In OTG mode the host and peripheral can exchanges role if necessary. The PHY chip is integrated in the USB controller, and provides a bridge between the digital and the modulated parts of the interface.

Standard USB uses a master/slave architecture. This means that a USB host acts as the protocol master i.e has undirectional control over the slaves, and a USB device acts as the slave i.e performes the work demanded by the master. In OTG mode a device can be either a host(acting as the link master) or a peripheral (acting as the link slave).

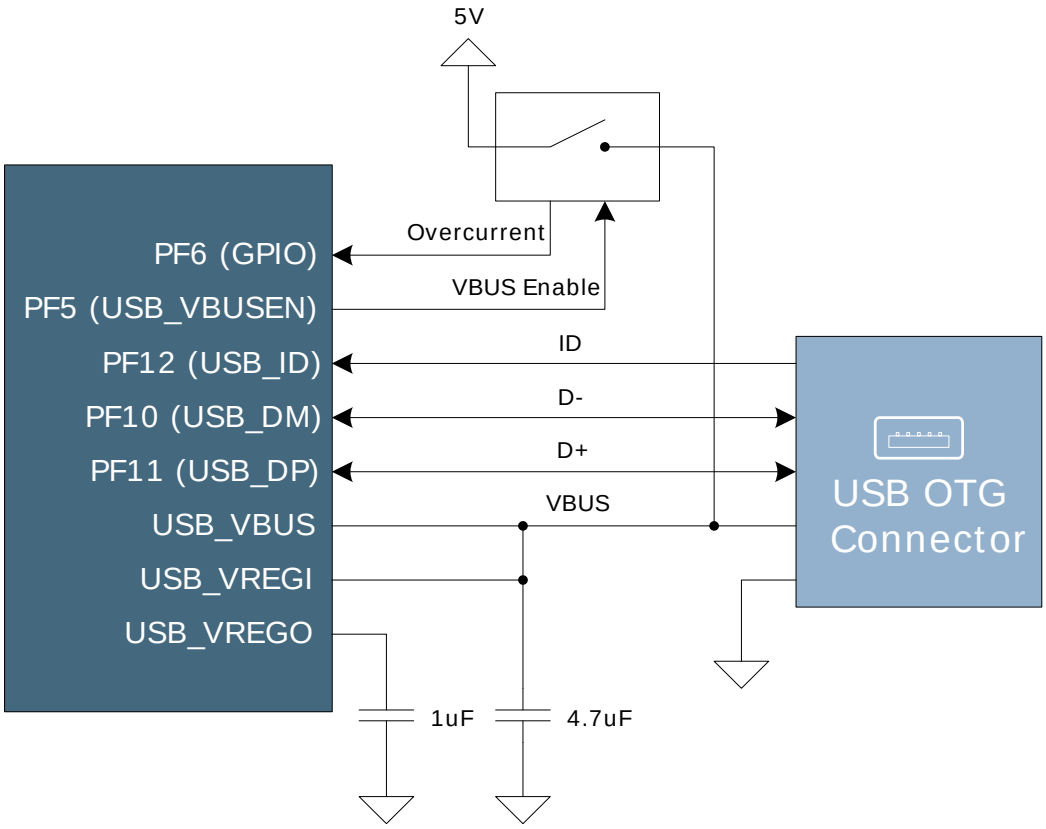
7.3 Pins

There are 8 pins associated with the USB.

- USB_VBUS should be connected to the VBUS (5V) pin on the USB receptacle. It is connected to the voltage comparators and the current sink/source in the PHY.
- USB_DP and USB_DM are the USB D+ and D- pins. These are the USB data signaling pins and a portable device can recognize the type of USB port from the way the D+ and D- pins are connected.

- USB_ID is the OTG ID pin used to detect the device type. The core enters Host mode(master) when an "A" plug is connected, and Device mode(slave) when a "B" plug is connected.
- USB_VREGI is the input to the voltage regulator and USB_VREGO is the regulated output.
- USB_VBUSEN is used to turn on and off VBUS power when operating as host-only or OTG A-Device.
- USB_DMPU is used to enable/disable an external D- pull-up resistor. A pull-up resistor weakly "pulls" the voltage of the wire it is connected to towards its voltage source level when the other components on the line are inactive. This is needed for low-speed device only.

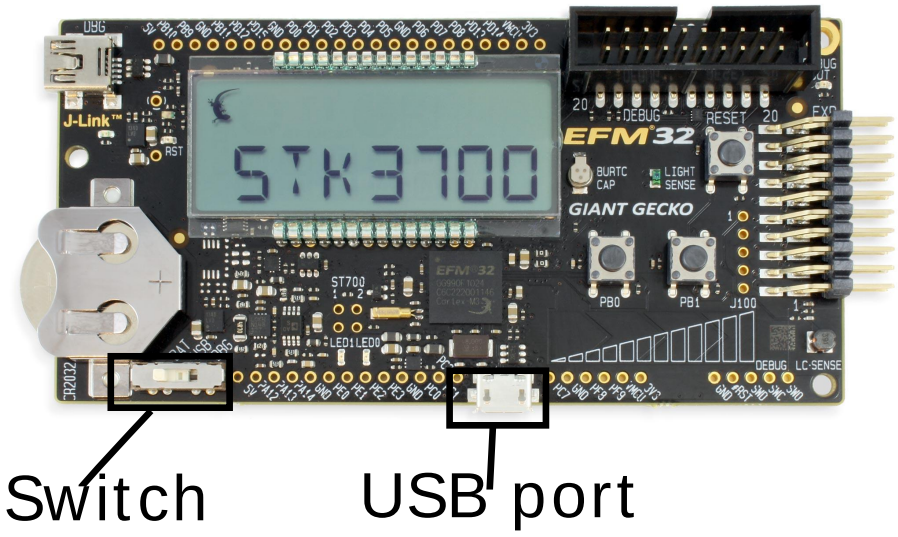
Figure 7.2. EFM32 USB Connector



8 USB Code Example

In this example we implement a keyboard device. The keyboard consists of one button, Push Button 0. When pressing the button, one by one letter in the string "Energy Micro AS -" will be written. The text will show in any text editor. The kit must be connected to the PC with a USB cable. The USB port and the switch are marked in Figure 8.1 (p. 15) .The switch should be in the right position, DBG, when compiling the code. Then the USB cable in the upper left corner, the on-board debugger will be used as power supply. When switching to the mid position, USB, the EFM32's own USB regulator will be used as power supply. Note that the Advanced Energy Monitor can only measure the current consumption of the EFM32 when the power selection switch is in the DBG position.

Figure 8.1. Showing the USB port an the switch of the Giant Gecko STK



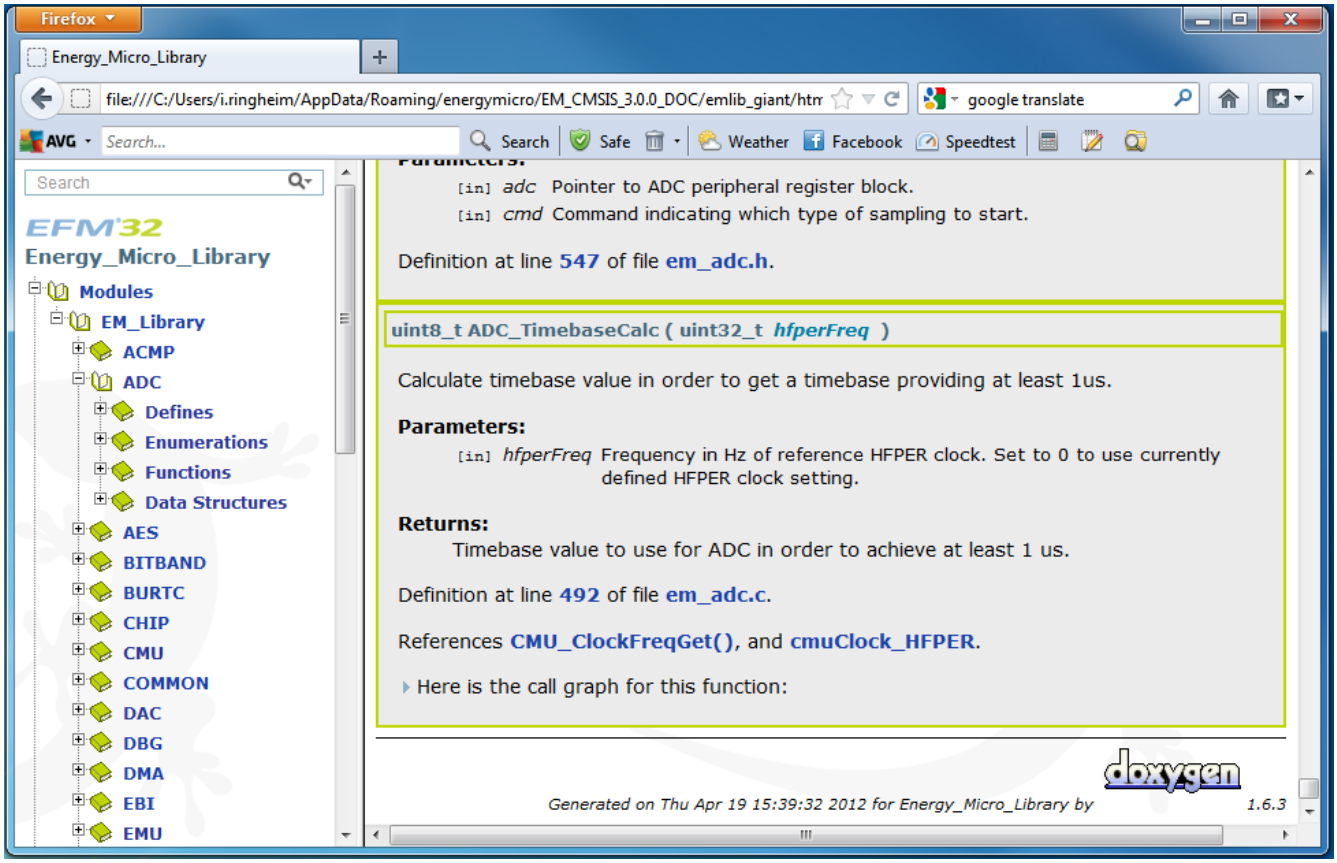
9 Summary

In this lesson we have looked at how to use the ADC, DAC and USART. These are only a small fraction of the peripherals available on a common MCU. You should, however, be able to use the peripherals by reading the reference manual, to discover features and how to control them with the registers, and the CMSIS documentation where you can find the appropriate library functions.

10 Exercises

In this exercise we will look at the same example as in `1_adc_register`. In stead of using direct register manipulation, you should use already defined functions which do the register operations. The emlib functions are described in Energy Micro CMSIS (Cortex Microcontroller Interface Standard). The resulting code should be equivalent and give the same result as `1_adc_register`. Your task is to fill in code in the function `ADCConfig()`. Structs for the ADC control register and the ADC_SINGLE control register are already declared and initialized with default values. You have to use emlib functons or enum defines to set some of the components in the struct that should have other values than the default value. When this is done, you have to initialize the ADC, in this case ADC0, by using the functions `ADC_Init()` and `ADC_InitSingle()`.

Figure 10.1. Showing an example of a function description in Energy Micro CMSIS



11 Projects

This is a bit more challenging and time consuming excercise. Use the EFM32 USB Device protocol stack to implement the a Mass Storage Class device (MSD). Feel free to use either the internal FLASH or SRAM. The task is to implement a program which measures the temperature and writes this data (i.e time and temperature) to a file which you store in the MSD. Hint: To see how to implement a MSD, look at the usbdmsd example in in Simplicity Studio.

12 Revision History

12.1 Revision 1.00

2011-06-22

Initial revision.

12.2 Revision 1.10

2012-07-27

Updated for Giant Gecko STK.

A Disclaimer and Trademarks

A.1 Disclaimer

Energy Micro AS 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 Energy Micro 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. Energy Micro 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. Energy Micro 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 Energy Micro. 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. Energy Micro products are generally not intended for military applications. Energy Micro 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

Energy Micro, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Energy Micro AS. ARM, CORTEX, THUMB are the registered trademarks of ARM Limited. Other terms and product names may be trademarks of others.

B Contact Information

B.1 Energy Micro Corporate Headquarters

Postal Address	Visitor Address	Technical Support
Energy Micro AS P.O. Box 4633 Nydalen N-0405 Oslo NORWAY	Energy Micro AS Sandakerveien 118 N-0484 Oslo NORWAY	support.energymicro.com Phone: +47 40 10 03 01

www.energymicro.com
Phone: +47 23 00 98 00
Fax: + 47 23 00 98 01

B.2 Global Contacts

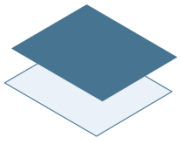
Visit **www.energymicro.com** for information on global distributors and representatives or contact **sales@energymicro.com** for additional information.

Americas	Europe, Middle East and Africa	Asia and Pacific
www.energymicro.com/americas	www.energymicro.com/emea	www.energymicro.com/asia

Table of Contents

List of Figures

List of Examples



ENERGY®
micro

Energy Micro AS
Sandakerveien 118
P.O. Box 4633 Nydalen
N-0405 Oslo
Norway

www.energymicro.com