Energy Micro University

UM001 - Introduction

This lesson is the first in the Energy Micro University series. It assumes no previous knowledge of MCUs, but some knowledge of computers and programming will be helpful. The aim of this first lesson is to give a bird's eye view of what MCUs are; applications as well as related software and hardware. Concrete examples will be given in later tutorials.

Concepts that will be introduced include:

- **MCU**
- **Registers**
- **Peripherals**
- **Clocks**
- **ADC and DAC**
- **C and assembly**
- **Interrupts**
- **Development kits**

This lesson includes:

- **This PDF document**

1 Introduction

1.1 About Energy Micro University

Energy Micro University is an introductary learning program which provides students with the most basic consepts of the EFM32 microcontroller. The program consists of 7 modules:

- UM001 - Introduction
- UM002 - Introduction to C
- UM003 - Setting Up Development Environments
- UM004 - Interrupts, DMA and PRS
- UM005 - Peripherals
- UM006 - Energy Modes
- UM007 - Energy Optimization

Each module starts with a breif introduction followed by a more detailed description of some selected topics. In some of the more detaled descriptions you will find some programming examples for EMF32 Tiny Gecko and Giant Gecko. You have to choose the correct IAR project, where the project name either ends with TG or GG. In the end og each module you will find excercises designed to test whether you have understood the key concepts in the module. These exercises are either theoretical, or pure programming tasks. Energy Micro University provides solution to all programmig excercises.

Some modules have additional projects without proposed solution. These are more time-consuming than the regular excercises.

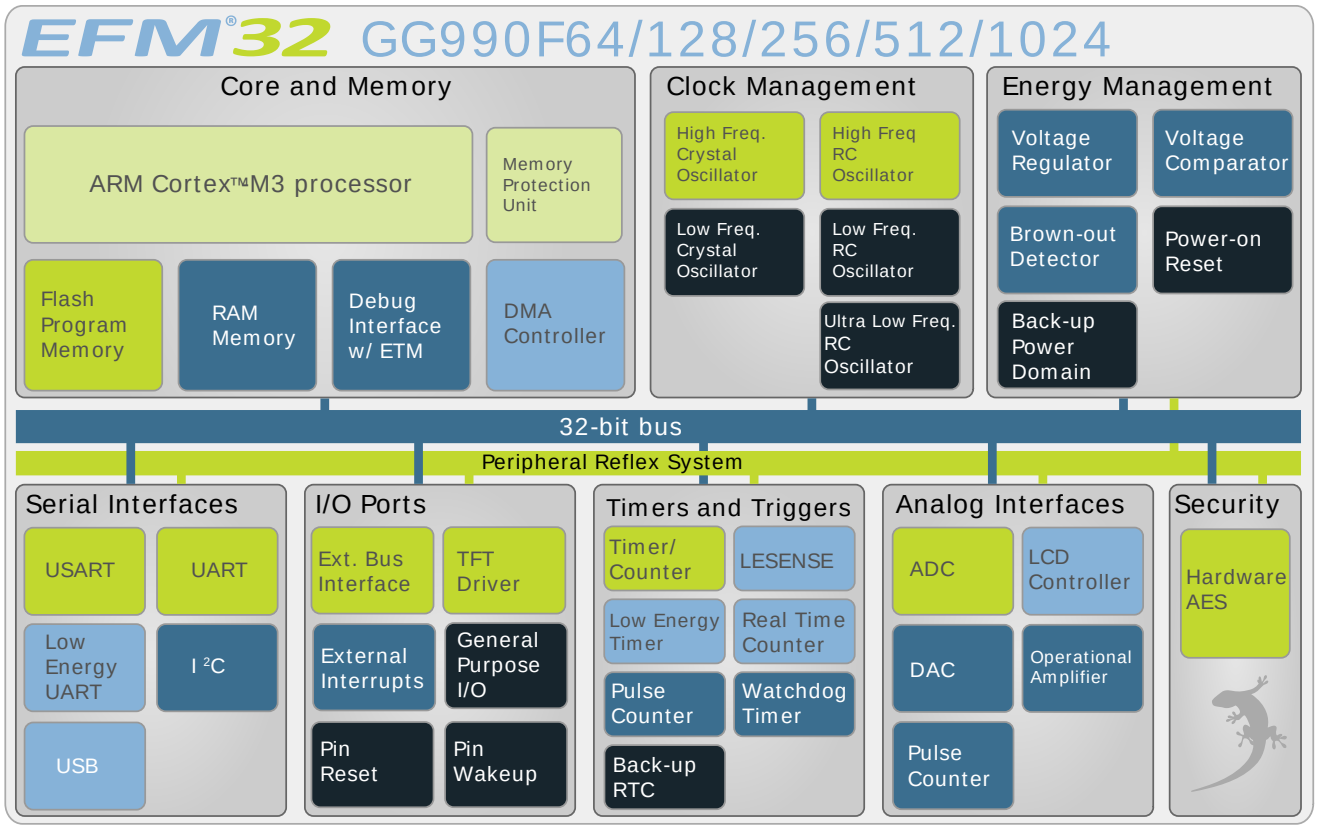
1.2 What is an MCU?

A MicroController Unit (MCU) is much like a Personal Computer (PC) only smaller; everything is integrated on a single chip called an integrated circuit (IC). MCUs are all around us: they are embedded in household appliances, toys, cars, cell phones, water and gas meters, alarm and security systems, health and fitness devices and much, much more. The MCUs usually fulfill a very specific role, often as a part of a larger system, which allows them to be tailored to fit the requirements of the application, whether it is functionality, reliability, power consumption or even price. Birthday cards with music is an example of a product that is very price sensitive – the added feature of music is only worth so much to the consumer.

Figure 1.1. An MCU with the pins exposed on the back side

The typical household has usually at least ten times as many MCUs as PCs. Although PCs may be familiar programming ground to many, the MCU market is much larger and opens endless possibilities for innovation. Will you bring the next revolutionizing gadget to life? The continued progress in the MCU industry requires dedicated and talented people developing both the MCUs as well as the end products they are embedded in; there should be plenty of interesting opportunities.

Figure 1.2. A block diagram of an EFM32



1.3 MCU components

An MCU typically consists of a Central Processing Unit (CPU), memory, Input/Output (I/O) and other external modules (peripherals) like Analog to Digital Converters. A MCU product line usually consists of many different models, each with its own set of functionality and peripherals. An MCU vendor might design all the components of a MCU themselves or buy designs from others. The designs are referred to as Intellectual Property (IP).

1.4 Production

All the components of the MCU are combined into a single Integrated Circuit (IC). A batch of a few hundred MCUs can be manufactured on a single slice of semiconductor material called a wafer. The MCUs are then divided and packaged in small plastic containers, which only reveals the pins of the microcontroller, see Figure 1.1 (p. 2) .

2 MCUs vs. PCs

MCUs are smaller than PCs, both in term of physical size, performance and cost. This allows them to be used in countless applications not suited for PCs, but it also imposes many restrictions. Programmers coming from PCs might not have ever cared about code size or power consumption, both of which are very important concerns when developing programs for MCUs. When the application is running on battery power and the device is cost sensitive, minimizing power consumption and storage requirements might be more important than execution speed. Furthermore, reliability and responsiveness is important. Programmers used to floating point operations are not likely to find any support for this in hardware – Floating Point Units (FPUs) are not common. The architecture otherwise might also be totally different from the ones used by PCs. An executable produced on your PC is highly unlikely to run on an MCU, and while PCs use operating systems to handle hundreds of programs and processes, some MCUs only run a single program at a time, others run some kind of operating system aimed at MCUs including Real Time Operating Systems (RTOS).

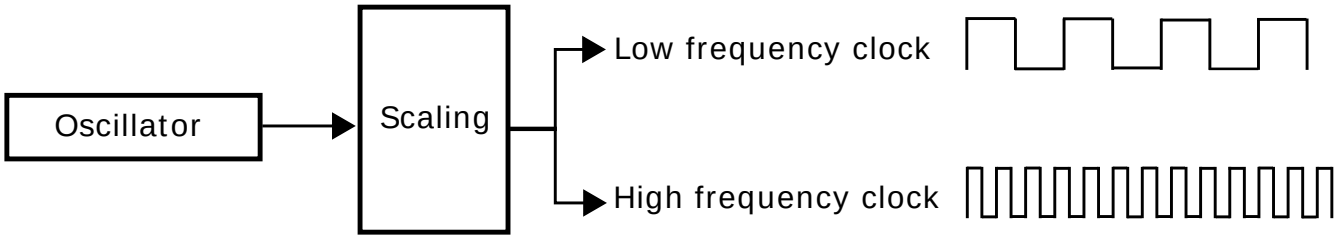
3 Peripherals

Peripherals add functionality to the MCUs – they can work alone 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. 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. This section describes some typically peripherals.

3.1 Clocks

Clocks are what drives the CPU, memory and peripherals and ensures cooperation between the different components by emitting electrical impulses at regular intervals. The clocks are produced by oscillators. Different clocks can be obtained by using different types of oscillators and by scaling.

Figure 3.1. Clocks produced by an oscillator and scaled into one low frequency and one high frequency clock.



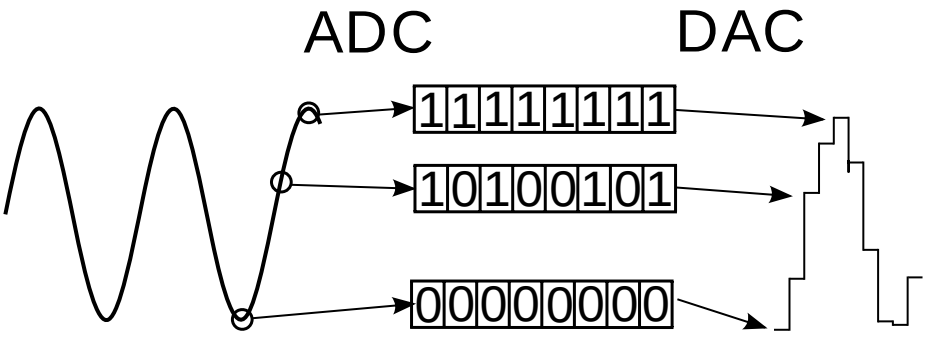
3.2 Timers

A Real Time Counter (RTC) can be used to keep track of time. It can both increment a count register based on a clock and to check if the count has reached some value and trigger an action.

3.3 ADC and DAC

There are ways to make the MCU interact with everyday continuous signals like for instance sound waves. Analog to Digital Converters (ADC) are used to discretize an analog signal to a digital (binary) value. An example could be to take the continuous variation in voltage produced by a microphone and store it as a sequence of discrete voltage values. The values can then be accessed from software. The converse operation, performed by Digital to Analog Converters (DAC), could be used to play a sound through speakers.

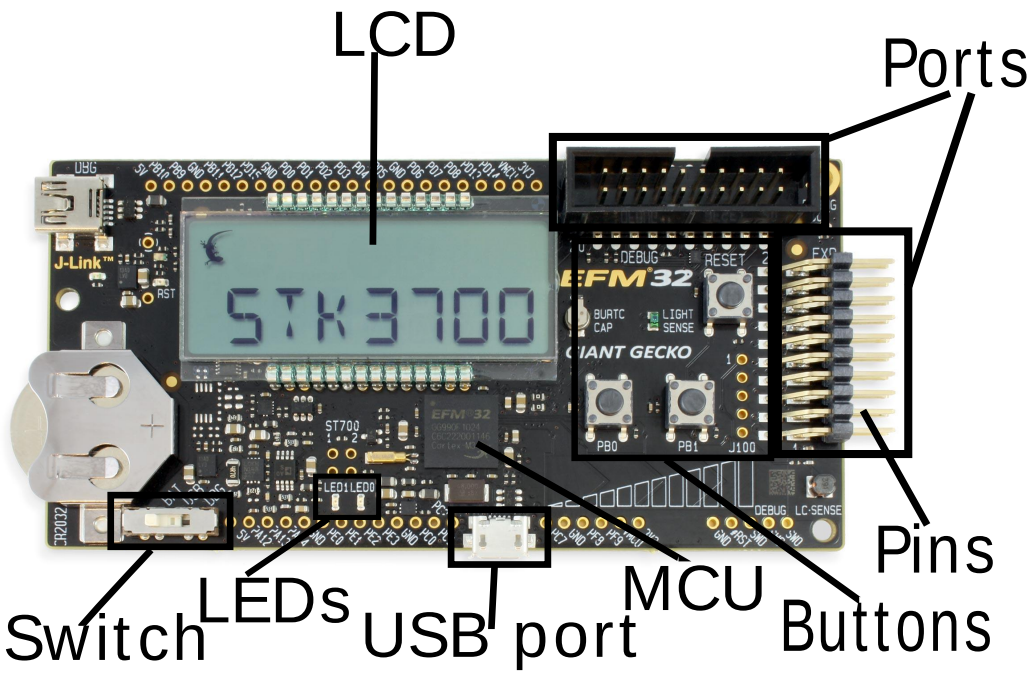
Figure 3.2. The analog signal is sampled at regular time intervals by the ADC. The sample size is 8 bit and corresponds to the unsigned voltage level at the given time step. The DAC converts the discrete values back into an analog signal.



3.4 Communication

Several communication solutions exist: Universal Serial Bus (USB), Serial Peripheral Interface (SPI), Universal Asynchronous Receiver/Transmitter (UART) and Universal Synchronous Asynchronous Receiver/Transmitter (USART). The UART can be used to communicate with a serial port on a PC.

Figure 3.3. A kit where an MCU is mounted on a printed circuit board together with external peripherals such as LEDs, buttons and an LCD display.



4 Application Development

4.1 C and assembly

In order to harness the power of MCUs one must both familiarize oneself with the available hardware as well as the development environment made available by the MCU vendor or third parties. The desired functionality is usually available through high level language, such as C, and by setting the appropriate values in the control registers of the peripherals, see Figure 4.2 (p. 8) .

Example 4.1. C code snippet, comments are surrounded with /* and */.

```
/* Wait until counter value of the
peripheral TIMER0 is over 1000 */
while(TIMER0->CNT < 1000){
/*Do nothing, just wait */
}
```

If a high level language is not an option or a more fine grained control of the program is desired, an assembly language can be used. Assembly languages are dependent on the architecture of the processor used as well as the compiler. Assembly is also more difficult to write and maintain than high level languages such as C. It is becoming increasingly rare to be used for application development. The compiler can often produce better results from a high level language than the programmer can write in assembly. Assembly language has a one to one correspondence with machine code, i.e. the instructions that are executed in hardware. If a high level language is used it is usually compiled into assembly language, which is then assembled and linked into machine code which can be run on the device.

Example 4.2. Assembly (Thumb-2) version of the code snippet from Example 4.1 (p. 7) .

```
    LDR.N     R0, TIMER_CNT
    MOV.W     R1, #1000
loop_start:
    LDR       R2, [R0]
    CMP       R1, R2
    BCC.N     loop_start
```

4.2 Registers

The functionality of a peripheral is controlled by setting bits in the appropriate registers, see Figure 4.1 (p. 7) . The word length of an architecture is the working unit, i.e. the number of bits of a number, or a pointer the architecture can naturally work with. Common word lengths for MCU's are 8-, 16- or 32-bit. The registers of the peripherals have the same size as the word length, although not all of the bits may be in use. A single attribute might be controlled by setting a single bit to 1 or 0, or by setting a bitfield consisting of two or more bits to the appropriate binary value.

Figure 4.1. An example of a 32-bit register with a byte and bitfields marked.

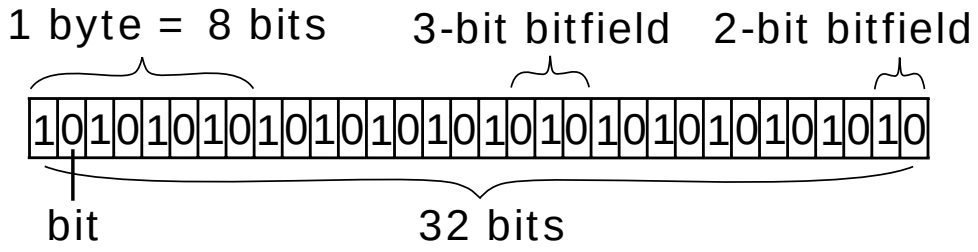
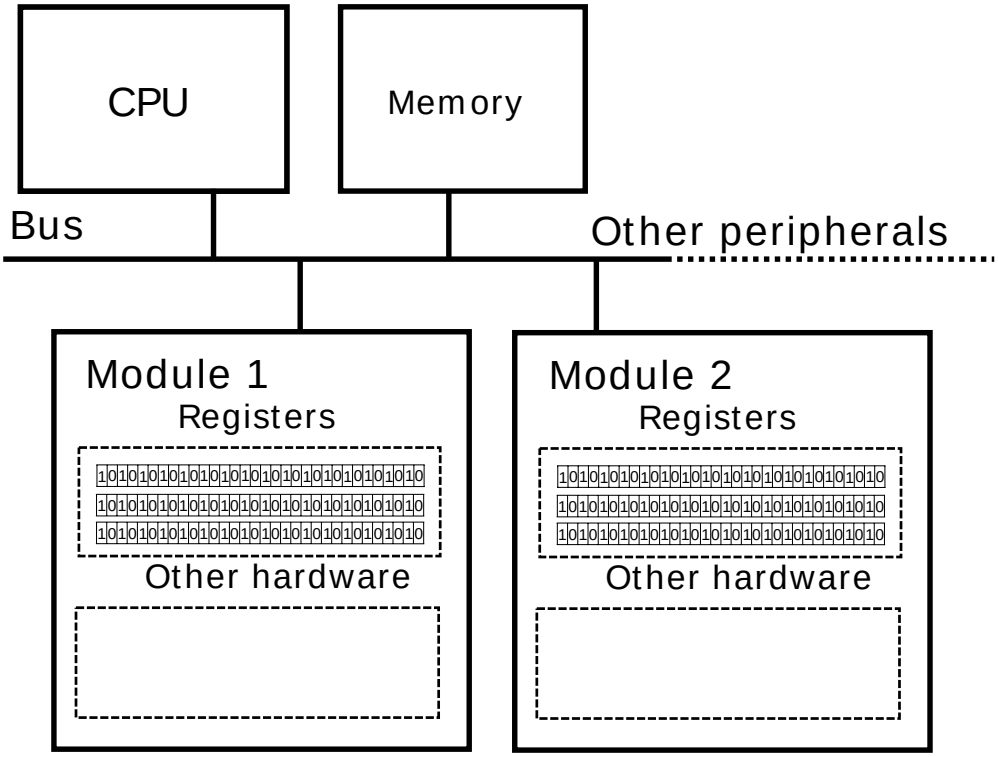


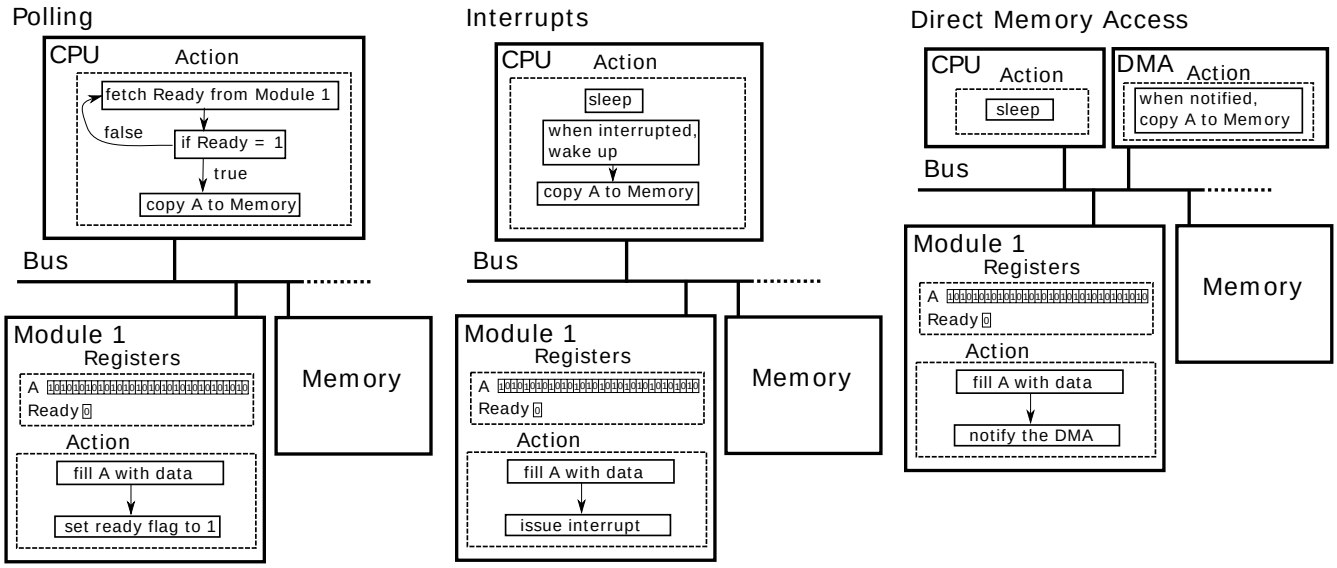
Figure 4.2. The figure shows how the registers are placed in the modules.



4.3 Interrupts

There are several ways of triggering actions based on events. The first is to write a piece of code that constantly check some conditions, i.e. the value of a given bitfield in a register, and trigger actions based on that value. This is called polling, and is not ideal since the CPU is constantly doing work even though there might be nothing happening. A better approach is to use interrupts. Interrupts can be set in hardware to trigger a piece of code, called an interrupt handler, given some event. This means that a small piece of hardware can be awake checking if the conditions are met, while the CPU and other peripherals are asleep, i.e. turned off to save power. When the event occurs the interrupt is generated, which wakes up the CPU and execute the interrupt handler. An even better way is to use Direct Memory Access (DMA) to do work without having to wake up the CPU.

Figure 4.3. The figure shows how to trigger an event by use of polling, interrupts and DMA.



4.4 Kits

To help development of MCU based products and to facilitate learning, the MCU vendors often produce development kits, see Figure 3.3 (p. 6) They consist of the MCU attached to a printboard together with common I/O devices such as buttons and a display. It is also easy to attach custom hardware, and to connect the kit to a PC in order to download software as well as to debug software running on the MCU. There are many available kits for the EFM32, but in Energy Micro University, we will only consider the EFM32 Giant Gecko Starter Kit and EFM32 Tiny Gecko Starter kit.

4.5 PCB

MCUs are usually mounted on a Printed Circuit Board (PCB) together with other components, as shown in Figure 3.3 (p. 6) . The MCU connects external peripherals through its pins. Some features might need more than one pin – the related pins are referred to as a port. Some pins may have a fixed functionality, e.g. ground, power supply or controlling a Light-Emitting Diode (LED) or a Liquid Crystal Display (LCD). Others are more generic, these are called General Purpose Input/Output (GPIO).

5 Summary

This paper has given a glimpse of what a microcontroller is, including the construction, application development and an overview of the basic terms. Further reading can be found in reference manuals, datasheets, application notes and white papers of the microcontrollers that are on the market. Reference manuals explain all the features of a product family. Datasheets list the features of individual products. Application notes give examples of how to use certain features. The internet is also a great source for exploring the world of MCUs.

6 Exercises

1. Why are MCUs useful?
2. Name three applications of MCUs.
3. Name three central parts of an MCU
4. What is RTC?
5. What are ADC and DAC?
6. Why is polling not optimal, what are the alternatives?

7 Dictionary

Table 7.1. Dictionary

Word	Abbreviation	Description
Analog Comparator	ACMP	Comparing a voltage to a programmed treshold.
Analog to Digital Converters	ADC	Converts analog signals into digital representations.
Advanced Energy Monitoring	AEM	Visualizes energy consumtion of a system and relate it to the software running on the Microcontroller.
Assembly language		A low-level programming language in which each statement corresponds to a single machine language instruction.
Asynchronous signal		Singal that suddenly can arise without any clock signal telling it to.
Bandgap		Temerature independent voltage reference circuit.
Binary digit	Bit	Variable or computed quantity which represents either 1 or 0.
Binary digit field	Bit field	Consists of two ore more bits.
BSS		Area of memory containing uninitialized global variables.
Bus		A subsystem that transfers data between components inside a computer or between computers.
Byte		Consists of 8 bits.
Cortex Microcontroller Software Interface Standard	CMSIS	Coding standard for all ARM Cortex devices. Library containing header files, defines, startup files and commonly used functions.
Clock Management Unit	CMU	Responsible for controlling the oscilliators and clocks on-board the EFM32.
Central Processing Unit	CPU	Hardware whitin a computer which carries out the instructions of a computer program.
Crystal oscillator		An electronic oscillator that uses mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. The crystal oscillators are external.
Digital to analog converters	DAC	Converting digital values to analog signals.
DATA		Area of memory used for initialized global variables.

Word	Abbreviation	Description
Direct Memory Access	DMA	Hardware system that can transfer data, i.e reading and writing independently of the CPU.
Embedded-Applicationg Binary Interface	EABI	Standard specifying the layout of object files and how to use runtime resources like registers and the stack.
Energy Management Unit	EMU	Handles the different low energy modes in the EFM32 Microcontrollers.
General Purpose Input/Output	GPIO	A module which allows the configuration of the I/O pins, which are used by the microcontroller to interact with any external system components.
Ingerated Circiut	IC	An electronic circuit manufactured by lithography, or the patterned diffusion of trace elements into the surface of a thin substrate of semiconductor material.
Inter-Ingerated Circiut	I²C	A multi-master serial single-ended computer bus that is used to attach low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device.
Input/Output	I/O	
Integrated Development Environment	IDE	A software application that provides comprehensive facilities to computer programmers for software development.
Interrupt Mask Register	IMR	A registers which specifies which interrupts are to be ignored and not acknowledged.
Interrupt		Asynchronous signal sent to the processor telling it that some action has to be done.
Interrupt Service Routine	ISR	The same as interrupt handler.
Interrupt Request	IRQ	
Liquid Crystal Display	LCD	
Light-Emmiting Diode	LED	
Low Energy Universal Asynchronous Receiver/ Transmitter.	LEUART	see UART.
J-Link		Interface used to communicate between kits and PCs.
MicroController Unit	MCU	

Word	Abbreviation	Description
Module		see Peripherals.
Nested Vector Interrupt Controller	NVIC	Integrated part of the ARM Cortex-M processor, supporting both Cortex-internal interrupts and up to 240 peripheral interrupt requests.
Printed Circiut Board	PCB	
Peripherals		Any auxilliary device that connects to and works with the MCU in some way.
Pin		Connects the peripherals to the MCU.
Polling		The case when the CPU constantly checks for an event to happen in some hardware.
Port		Some features might need more than one pin – the related pins are referred to as a port.
Peripheral Reflex System	PRS	A network that allows the peripherals to communicate with each other without involving the CPU.
RC oscillators		An oscillator circuit which uses a combination of resistors and capacitors. The RC oscillators are internal.
Register		Small amount of memory that can be read from and written to.
Real Time Counter	RTC	Keeps track of time.
Real Time Operating System	RTOS	The main purpose of the RTOS is to schedule several smaller programs instead of having only one large program.
Successive Approximation Register	SAR	An analog voltage comparator that compares V_{in} to the output of the internal DAC outputs the result of the comparison to the SAR.
System Control Register	SCR	A processor register which changes or controls the general behavior of a CPU.
Serial Peripheral Interface Bus	SPI	A synchronous serial data link standard, that operates in full duplex mode.Devices communicate in master/slave mode where the master device initiates the data frame.
Serial Wire Debug	SWD	Used to debug the MCUs.
Starter Kit	STK	

Word	Abbreviation	Description
Universal Asynchronous Receiver/ Transmitter	UART	An individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port.
Universal Synchronous Asynchronous Receiver/ Transmitter	USART	The USART handles high-speed UART, SPIbus, SmartCards, and IrDA communication.
VDD		Power supply for the MCU.
Volatile		Variable declaration in C. Tells the compiler that the variable may change at any time.
"Wait for Event"	WFE	Program instruction. Turns off the CPU.
"Wait for Interrupt"	WFI	Program instruction. Turns off the CPU.

8 Revision History

8.1 Revision 1.00

2011-06-22

Initial revision.

8.2 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.

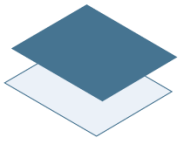
Americas	Europe, Middle East and Africa	Asia and Pacific
www.energymicro.com/americas	www.energymicro.com/emea	www.energymicro.com/asia

Table of Contents

List of Figures

List of Tables

List of Examples



ENERGY®
micro

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

www.energymicro.com