Batteries worldwide celebrate the arrival of

EFM32®

… the world's most energy friendly microcontrollers

Introduction

The explosion in use of battery operated electronics is followed by the need for the battery to last longer for convenience, environmental and cost reasons. At the same time, applications become more and more complex and demand higher performance of the microcontroller, making the challenge of designing systems for low power even more difficult.

Energy Micro is proud to introduce the EFM32 microcontroller family, which has an energy consumption that is many times lower than any other microcontroller family today, whether you benchmark with leading 8, 16 or 32 bit microcontrollers. With the EFM32 you can easily obtain battery lifetime in the 20-25 years range while at the same providing the high performance of the 32-bit ARM Cortex-M3. The EFM32 is truly and by far the world's most energy friendly and energy efficient microcontroller.

Low energy consumption together with high performance makes the EFM32 a perfect match for energy sensitive application like energy metering systems, home and building automation, security and medical systems.

Figure 1: EFM32 Target Applications

Energy friendliness

For battery powered applications, the target is to make the battery last as long as possible. To achieve this you need the power consumption in your application to be as low as possible. Also, by finishing tasks faster through higher processing performance, the microcontroller can spend more time sleeping, thus increasing battery lifetime. In a typical battery powered application, the microcontroller will spend most of its time in sleep and having a low stand-by current is therefore essential.

Energy Micro addresses the increasing energy demands by introducing the term “Energy friendly”, which means that our focus is put on minimizing the energy needed to complete the tasks included in the application. In the end, less energy consumed per finished task, combined with ultra low deep sleep current, means that the finite amount of energy available in the battery will last significantly longer.

Active power consumption

One important factor to look at when you want to increase the battery lifetime in your application is the power consumption of the MCU when the processor is running code, which would in most cases be stored in Flash memory. When designing the EFM32 family, great care has been taken to ensure that the resources needed to run code, need as little power as possible. Some of the steps done to ensure this are:

- Power optimized oscillators and clock tree
- Optimized clock gating structures for all synchronous logic
- Minimum bit toggling in the bus system and memories (Flash and SRAM)

All of these measures put together enable the EFM32 devices to run code stored in Flash while consuming as little as 180 µA/MHz. This benchmark is achieved with a prime number search algorithm executing code from flash and accessing RAM for data. This is by far better than any 8-, 16- or 32-bit solution today and even without accounting for the differences in processing performance. When comparing results between different microcontroller vendors, one should also note that not all vendors include the power consumption from the Flash program memory in their datasheet, with misleading statements such as "while (1) running from Flash", where the single instruction needed to implement the infinite loop is read from a flash buffer instead of from Flash. Note that the 180uA/MHz benchmark is measured at 3.0V supply voltage opposed to many other benchmarks, where typically the lowest supply voltage possible is used as a reference. In a real life application however, the supply voltage is often dictated by the battery and the MCU must therefore be efficient at higher voltages as well.

The higher active power consumption of other 32-bit MCUs means that the peak power consumption of the device is high. Finding a battery which can deliver this much power will then be an issue. The solution is often to use a sophisticated battery type at a higher cost. The EFM32 devices have a very low peak power consumption, which means that a low cost coin cell battery can be used to drive the application. The current consumption out of reset is low enough to be supported by even small coin cell batteries.

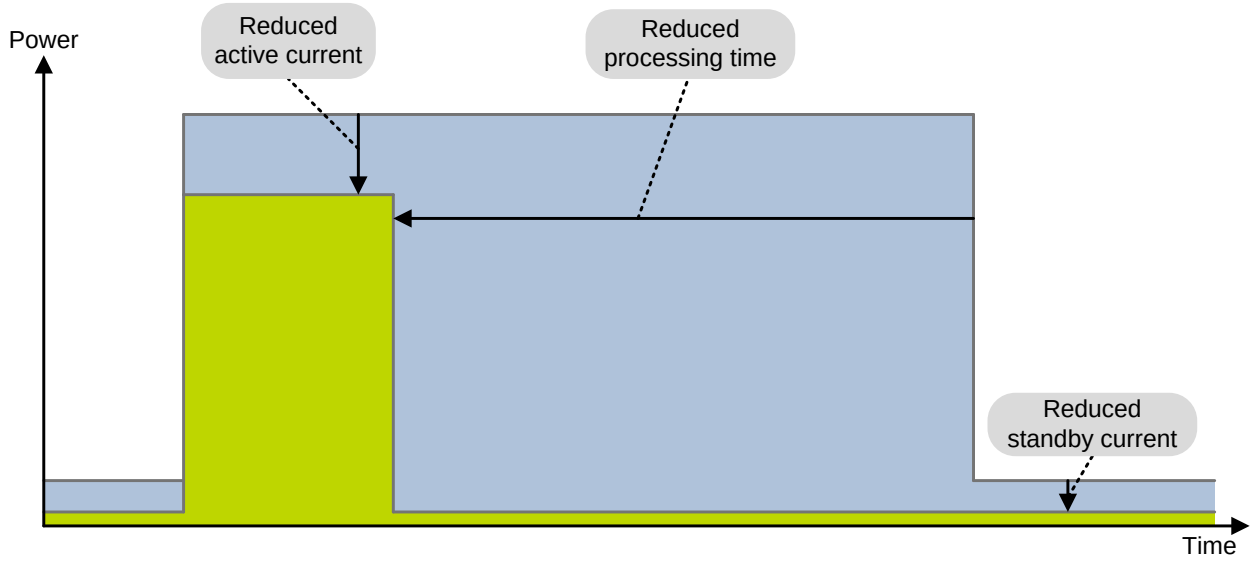

Figure 2: Reducing Active Energy Consumption

Processing power

To minimize the energy consumption it is important to spend as much time in deep sleep as possible. This means that tasks which require the processor to be awake must be solved as quickly as possible to keep the active-to-idle ratio low. The ARM Cortex-M3 processor architecture is being used in the EFM32 family because of its high processing performance combined with advanced energy saving features. The Cortex-M3 builds on ARMs long experience in processor design to achieve the best performance in the market. Where 8-bit and 16-bit MCUs (and even other 32-bit MCUs) must stay active for longer periods of time to finish the task, the performance of the Cortex-

M3 allows the EFM32 to solve even complex problems quickly, reducing the active-to-idle ratio dramatically.

The Cortex-M3 uses the Thumb-2 instruction set, which combines the compact 16-bit Thumb instructions seamlessly with the high performance 32-bit ARM instructions. While the 32-bit instructions can solve advanced tasks in a single cycle, around 90% of the code can be implemented using 16-bit instructions in a typical application. Because of this, 32-bit flash accesses will in most cases fetch two instructions at a time, which means a greatly reduced number of flash reads. The compactness of the 16-bit instructions combined with the performance of the 32-bit instructions then helps to greatly reduce the total active power consumption.

Energy Modes

Since the performance of the EFM32 allows the system to stay in deep-sleep for a majority of the time in a typical application, the always present baseline power consumption in voltage or bias current regulators, power-on reset, brown-out detection etc. becomes significant factors in the overall energy budget. In the EFM32, innovative techniques have been utilized to make these resources extremely energy efficient, while still meeting precision and reliability demands.

Even though many other MCUs have relatively low power consumption in deep sleep, the features available in these modes are often very limited. Since applications often require such features as Real-time Counters, Power-on Reset/Brown-out Detection or UART reception to be enabled at all times, many MCU systems are prevented from ever entering deep sleep, since these features are only available in an active state. In the EFM32 devices, the resources and thereby also the power consumption can be scaled in several levels which are called Energy Modes (Figure 3).

EFM32 @ 3 V	EM0 "Run Mode"	EM1 "Sleep Mode"	EM2 "Deep Sleep Mode"	EM3 "Stop Mode"	EM4 "Shutoff Mode"
Current Consumption	180 µA/MHz	45 µA/MHz	0.9 µA	0.6 µA	20 nA
Wake-up Time	-	0	2 µs	2 µs	160 µs
CPU	On	-	-	-	-
High Frequency Peripherals	Available	Available	-	-	-
Low Frequency Peripherals	Available	Available	Available	-	-
Asynchronous Peripherals	Available	Available	Available	Available	-
CPU and RAM Retention	On	On	On	On	-
Power-on Reset, Brown-out Detector	On	On	On	On	On

Figure 3: Energy Modes

While the Energy Modes constitute a coarse division, a more fine-grained tuning of the resources within each Energy Mode can be done by enabling/disabling different peripherals. This flexibility ensures that only the resources actually used are enabled at all times, so no energy is wasted. It is important to note that Power-on Reset and Brown-out Detection are included in all of the above power numbers and there is full system retention, including SRAM and CPU state, even down in Energy Mode 3. In Energy Mode 2 a running Real-Time Clock is also included in the power numbers.

Competing solutions in many cases have limited SRAM and CPU state retention in sub-µA standby modes, if available at all. Other solutions need to turn off or duty-cycle its brown-out and Power-on reset detectors in order to save power, at the cost of reduces reliability in the power supervision.

The EFM32 uses a superior power monitoring solution that combines nA current consumption with fast response time, without sacrificing the safety in the application.

Wake-up time

When an MCU goes from a deep sleep state, where the oscillators are disabled, to an active state, there is always a wake-up period, where the processor must wait for the oscillators to stabilize before starting execution of code. Since no processing can be done in this period, the energy spent while waking up is wasted. Hence, reducing the wake-up time is important to reduce the overall energy consumption. Also, MCU applications impose real-time demands, which often means that the wake-up time must be kept at a minimum to enable the MCU to respond to an event within a set period of time. Since the latency demanded by many applications is lower than the wake-up time of traditional MCUs, the device is often inhibited from going into deep sleep at all. For EFM32 devices the wake-up time from deep sleep to program execution has been lowered to 2 µs, which means that tight real-time latency demands can be met while still allowing the device to enter deep sleep. Figure 4 shows the high frequency oscillator starting in only 424 ns after a pin interrupt is triggered in this case.

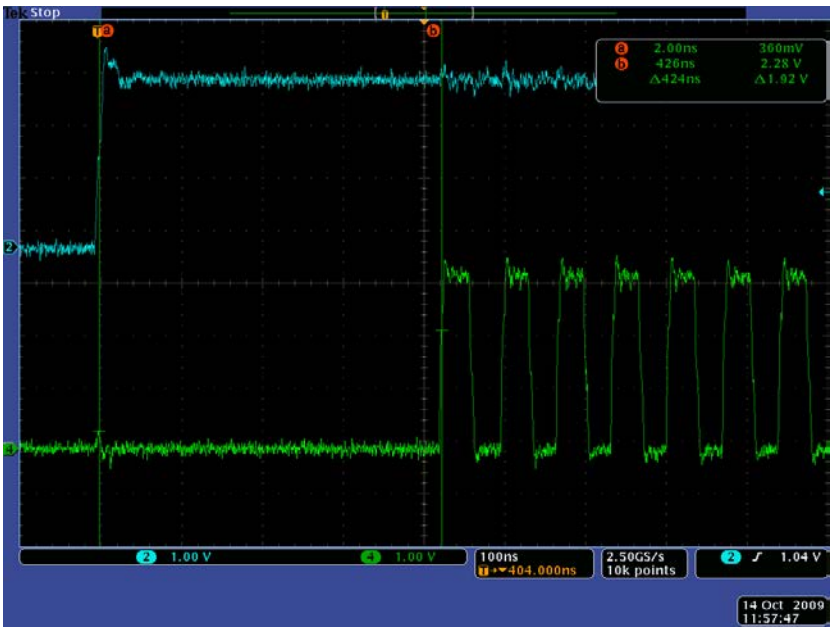

Figure 4: Fast oscillator start-up

Clock system

In an MCU, the power consumption typically consists of two parts:

- A baseline power consumption, which may come from modules such as voltage or current bias generators, voltage regulators, etc. This contribution is often independent of the operating frequency.
- A frequency dependent component, which is roughly proportional to the frequency of the clock signal, hence reducing the necessary clock frequency is important to meet the energy budget.

When operating at a low clock frequency, many microcontrollers suffer from high baseline power consumption which results in high total power consumption. A majority of MCUs achieve their optimal power consumption per MHz at a high frequency, performing significantly worse at a lower frequency. The EFM32 devices have an extremely low baseline power consumption, which makes it possible to lower the frequency while still achieving a low number of µA/MHz.

An application often imposes different demands for speed in the CPU and the peripherals. While highly processing intensive tasks are solved by the CPU, needing a high clock frequency, the peripherals might only need to run at a fraction of the CPU's clock frequency. In the EFM32 devices, the clock to the core and peripherals can be individually pre-scaled, hence the dynamic power consumption of the different parts is kept at a minimum. However, in some cases, the peripherals might need a higher clock frequency than the core, which is why the EFM32 devices allow any of the two clock domains to have the higher frequency.

Limiting the current consumption in the oscillators themselves is an important energy saving feature. All 6 oscillators in EFM32 are designed using highly innovative techniques to achieve ultra low power while still maintaining fast wake-up times and high accuracy. The on-chip high speed RC oscillators for example starts in 0.5 µs, and is trimmed during production to an accuracy of 1%, allowing UART communication to be run without the need for a crystal. Still, it consumes only 90 µA at 14 MHz, and has a wide selection of frequency bands available.

On-chip clock distribution is often a dominating factor in the active power consumption of a microcontroller. Clock gating is a widely used technique where the clock to flip-flops which do not need to update their state is blocked by gating modules. This reduces the dynamic power dissipation in seldom updated flip-flops. Clock gating will however not remove the idle current consumption completely, since flip-flops which cannot be gated efficiently and clock gates still will consume some power when receiving a clock signal. Also, the clock tree and buffers themselves consume power when switching even when the clock is gated before reaching any flip-flops. The most effective way is then to cut off the clocks to unused parts of the system at the root. In EFM32, the clock to each subsystem can be shut off individually in the Clock Management Unit, which ensures that no energy is wasted in the clock tree or the unused modules. Effective clock distribution further reduces the power consumption when a clock is not gated.

Autonomous Peripherals

The peripherals in the EFM32 devices are designed with energy consumption as a top priority from day one. A key criterion for the peripherals is therefore their ability to operate with minimum intervention from the CPU. When the peripherals mind themselves, the CPU can either solve other higher level tasks or fall asleep, saving energy either way. With advanced sequence programming, routines for operating the peripherals normally controlled directly by the CPU can now be handled by the peripherals themselves. The 160-segment LCD controller can for instance be programmed to run custom animations on a set of LCD segments, without run-time control from the CPU, while consuming as little as 550nA.

All EFM32 devices include a DMA controller which is a key factor in autonomous peripheral operation. The DMA controller helps to offload the CPU by handling data transfers between memory

and communication or data processing interfaces. Highly flexible DMA modes enable the peripherals with the help of the DMA to handle tasks that would normally need additional CPU assistance. An example of this is the EFM32's Low Energy UART (LEUART), which is an extreme low power asynchronous communication interface. Running off a 32kHz oscillator, the LEUART can be enabled even in deep sleep, adding only 150nA when waiting for incoming data. When the LEUART receives data the DMA can be requested to fetch the data and store it somewhere else, typically in RAM. The LEUART can receive a larger blocks of data, all while the CPU is in its deepest beauty sleep. The LEUART does not need a dummy "wakeup word" to wake up, as is necessary with competing solutions.

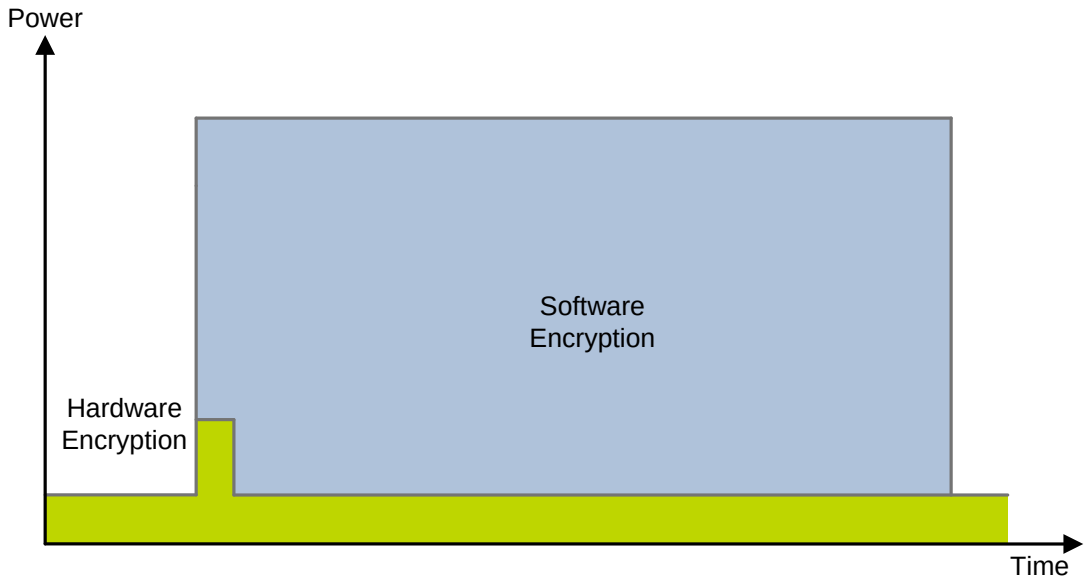

Figure 5: Hardware encryption (green) vs software encryption (blue)

As communication integrity and protection of intellectual property is becoming increasingly important, encryption in microcontrollers is also becoming widespread. The Advanced Encryption Standard (AES) is one of the most used symmetric ciphers, replacing its predecessor from the 1970s, the Data Encryption Standard (DES). The AES algorithm is traditionally implemented in software, imposing a large memory footprint (several KBs) and longer periods of intensive processing, resulting in a high energy cost per encrypted byte. Energy Micro introduces energy friendly encryption with a hardware AES accelerator capable of encrypting or decrypting a 128-bit block in as little as 54 cycles using a 128-bit key or 75 cycles when using a 256-bit key. This is 20 to 40 times faster than a software implementation on the Cortex-M3 and 80 times faster than software solutions on an 8-bit controller. Since the AES accelerator only uses 20% of the power consumed by a traditional 32-bit CPU, the total energy per encrypted block is cut by 99% compared to other Cortex-M3 software solutions (Figure 5). Compared to 8-bit software solutions, the savings are even higher. In addition, flexible DMA interaction mechanisms, makes the AES accelerator able to implement the most common cipher modes like Cipher Block Chaining, Output Feedback, etc. without CPU intervention.

Peripheral Reflex System

In a microcontroller system, there are many scenarios which require interaction between peripherals. A typical example would be an ADC conversion triggered by a timer on a periodic basis. Traditionally, such an interaction would be handled by the timer triggering an interrupt, in which the CPU triggers the ADC conversion. Most other MCUs include a number of predefined and hard coded connections between a subset of the available peripherals, reducing the available flexibility for the application developer to select the optimal implementation for its application.

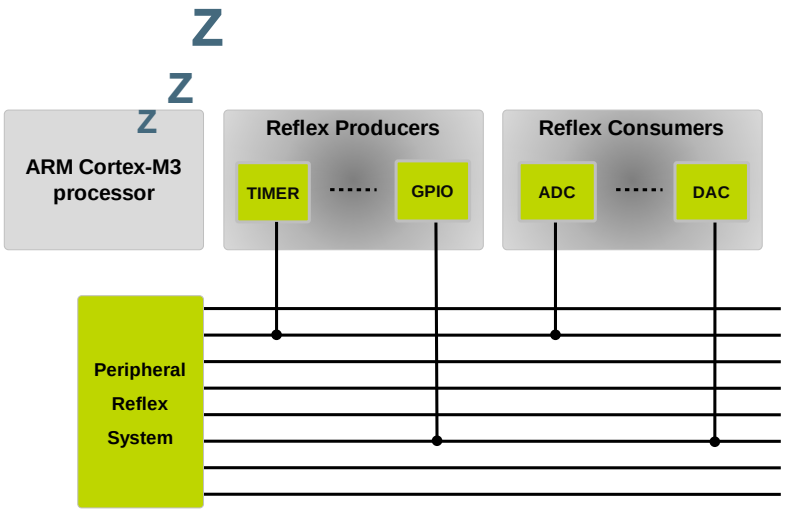

Figure 6: Peripheral Reflex System

In the EFM32 devices, the Peripheral Reflex System (PRS) allows the peripherals to communicate directly with each other without involving the CPU, just like human reflexes allow different body parts to trigger reactions in other body parts without involvement from the brain. Also, with the help of the DMA controller to transfer data (ADC samples etc.), many of the interactions earlier handled by the CPU can now be handled autonomously by multiple peripherals through direct interaction, reducing CPU active time and energy consumption. Each of the PRS channels, 8 in total, can be individually set up to relay a Reflex signal from a peripheral. Examples of Reflex signals can be pulses generated on timer overflow, or the level output from an analog comparator. Peripherals which are capable of performing actions based on incoming Reflex signals can then choose to listen to the PRS channel which carries the Reflex signal they are interested in. This generic structure of the PRS allows the peripherals to be connected in a great number of ways, adding flexibility to low energy design.

Energy Efficient Peripherals

Even though the CPU and memory often consumes the most power in the MCU, the peripherals will also contribute significantly to the overall power consumption. Therefore, normally power hungry peripherals like ADC, LCD etc. have been custom designed for the EFM32 devices with extreme low power in mind. This enables 12-bit ADC conversions at 10 ksamples/s with a current consumption of only 9 µA. Sample rates up to 1 Msamples/s is also available, still at 12 bits and with only 220 µA. In addition, the ADC can be put in different idle levels in between conversions, which enables it to only

consume 70 µA between conversions while still being able to wake up within 1 µs to do a conversion with full 12-bit resolution using an internal bandgap reference.

Figure 7: EFM32 directly driving a 160 segment LCD display

The LCD driver can serve up to 160 segments at only 550 nA, excluding the LCD display itself. Since many LCD displays cannot operate at voltages below e.g. 3 volts, the LCD driver has an integrated voltage boost function. So when the supply voltage drops below 3 volts, the voltage booster can be enabled on-the-fly to boost the outputs to the LCD display up above 3 volts. This is a necessary feature when running off a battery since the voltage of the battery will often drop below 3 volts towards the end of the battery's lifetime.

The EFM32 also includes two extremely efficient analog comparators which can compare two analog inputs at only 100 nA. The comparators provide fully integrated voltage references, and have capacitive sense capability which enables sensing of up to 16 capacitive buttons.

Development Tools

When developing energy sensitive applications, the designer must make many implementation choices that can impact energy consumption. While the MCU plays a central role, choosing the right external components and integrating these in a sensible way is also of great importance. Being able to identify and remove energy drains at an early stage of prototype development can significantly reduce the energy consumption of the end product.

The EFM32 Development Kit addresses this with the Advanced Energy Monitoring (AEM) system, which gives the designer instant feedback on the power consumption of the prototype application. The Development Kit comes with a separate MCU board and prototyping board which are plugged into the motherboard and the current running in to the application can be watched live on the AEM screen. This way the total energy consumption of the prototype can be optimized, avoiding the suboptimal solutions of fractioned development.

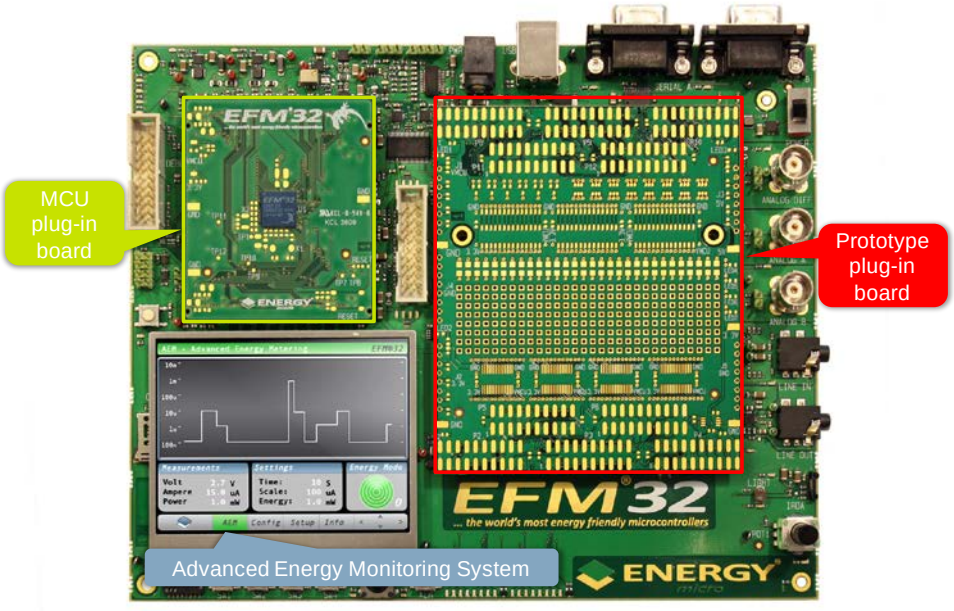

Figure 8: EFM32 Development Kit with Advanced Energy Monitoring System

The development kit additionally supports a USB connector that can be used to power the board and provide a live debug session to IDEs such as IAR Embedded Workbench and Keil µVision. A debug out port is even available to be able to debug the EFM32 in a user designed application.

Comparison

The MSP430 have for a long time been viewed as the leading low power microcontroller in the market. However, the introduction of the EFM32 has changed this by giving you 32-bit performance in addition to a drastically reduced energy footprint compared to the MSP430. Figure 7 below shows the EFM32G890F128 (green) compared with the MSP430F51x device (blue) with the following parameters:

- 3 V supply voltage
- 25 MHz clock frequency in active mode
- Standby mode with Real Time Counter, Power-on Reset, Brown-out Detector and full RAM and CPU retention

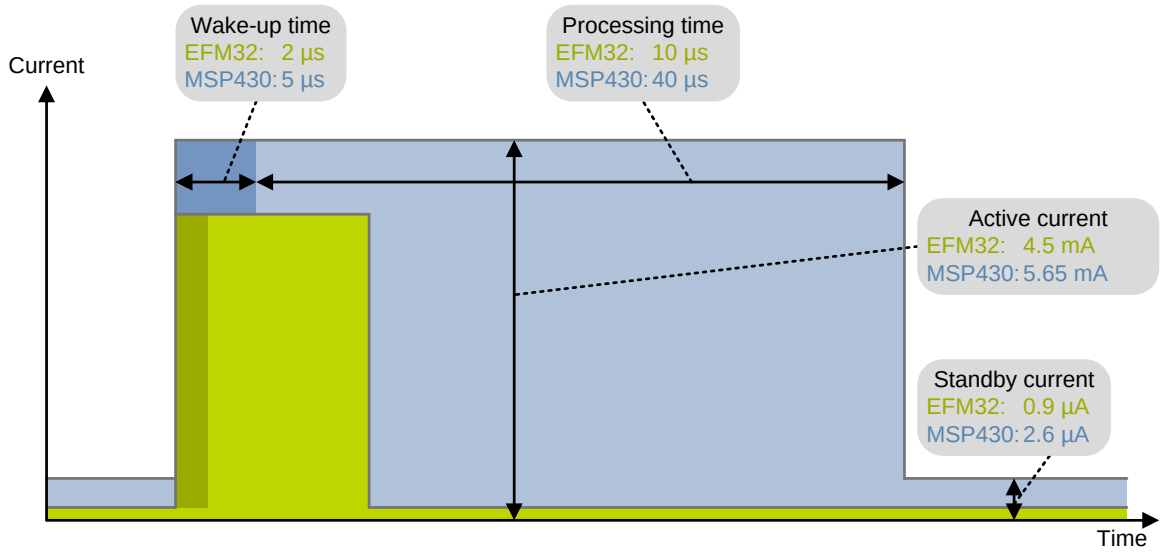

Figure 9: EFM32 vs. MSP430

For applications where the standby mode dominates the energy consumption the EFM32 will have a third of the energy consumption of an MSP430F51x. For applications where the active mode dominates the power consumption while processing is lower, but the major difference is caused by the significantly improved processing performance of the Cortex-M3 core. This ratio will depend on the application code, optimization etc. but is seen to be in the range of 4x.

Conclusion

The EFM32 devices have been designed with energy consumption in mind from day one. The result is significantly increased battery lifetime compared to existing solutions without sacrificing performance. Where longer battery lifetime is not needed, the extremely low energy footprint opens up for lower cost batteries in applications where higher priced batteries would normally be a necessity. The EFM32 development environment with the innovative Advanced Energy Monitoring system enables further energy optimization of the total application at the prototyping stage. In the end, Energy Micro delivers both the world's most energy friendly microcontrollers and the tools you need to extend the battery lifetime in your application beyond four times.