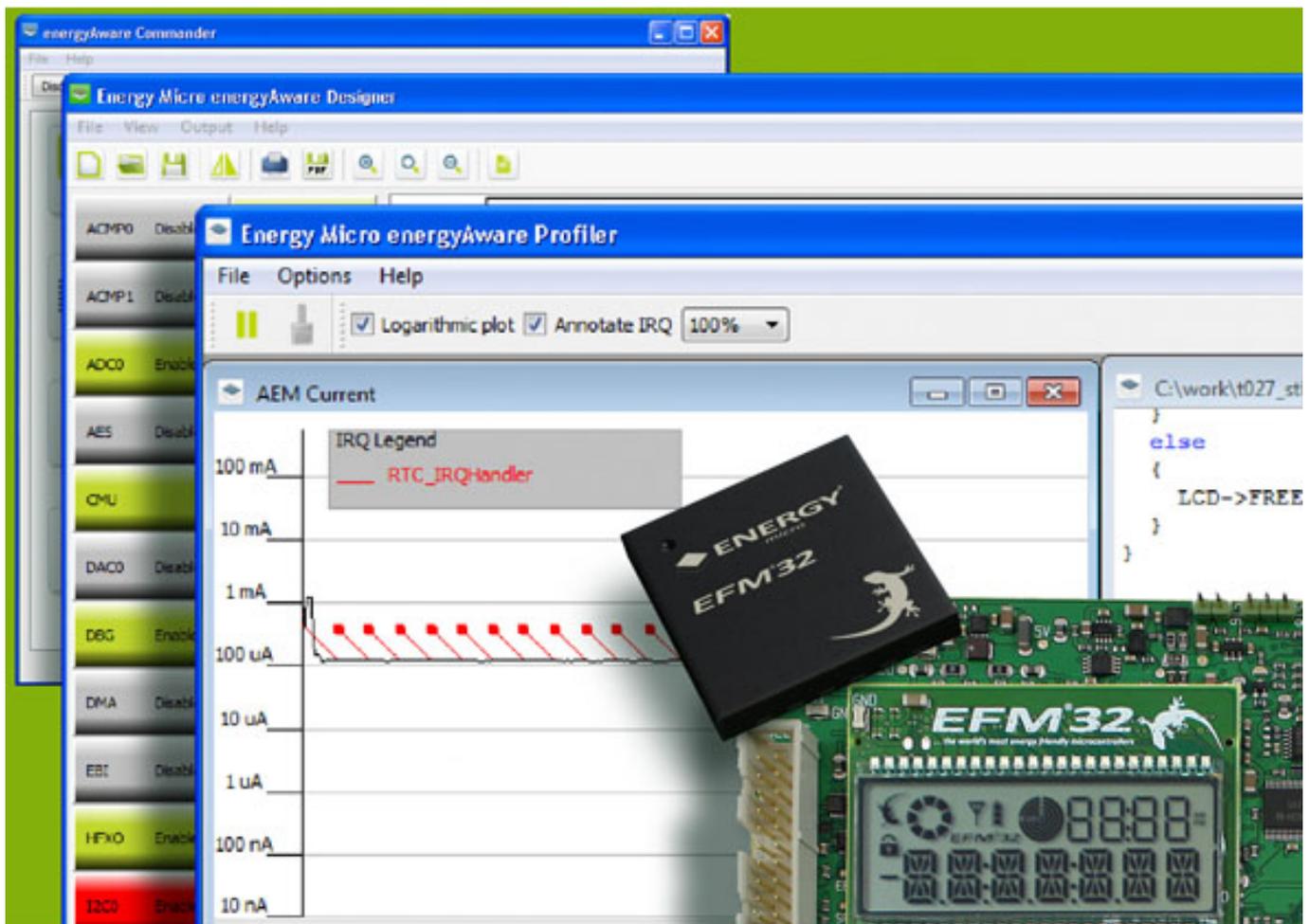


## Energy saving under new management

Andreas Koller, VP Marketing and Sales, Energy Micro, [www.energymicro.com](http://www.energymicro.com)

When a battery needs to support a product during many years or even decades of operation, incremental improvements in MCU integration and minor tweaks to basic processor architecture are insufficient to achieve the dramatic hikes in energy efficiency that are required. For a host of energy sensitive products in sectors such as metering, building automation, security and portable medical equipment, more wholesale advances in MCU design are needed if the often conflicting demands of energy efficiency and processing power are to be met.

In taking a 'blue sky' approach to the design of its low power EFM32 Gecko microcontroller and the software and hardware tools that support it (Fig 1), Energy Micro has produced a device capable of consuming a quarter of the energy needed by incumbent 8-, 16- and 32-bit MCUs, meaning existing battery lifetimes can be dramatically extended. Looking at this in a different way, with such an energy friendly MCU, product designers are now able to significantly reduce the cost and/or size of the battery, while for certain products, such as energy meters and security equipment, the frequency, cost and carbon footprint of maintenance call-outs for battery replacements can be minimised.



**Figure 1: Energy Micro’s EFM32 Gecko microcontroller and Simplicity Studio, tailored to saving energy.**

Achieving such low power credentials in an MCU is not an easy task and years of development and some real innovation are needed. Take a peak at energy micro’s website and you’ll find its technology described under the grand title, ‘10 factors that make the 32-bit EFM32 the world’s most energy friendly microcontroller...’. In reality there are certainly even more factors than these.

Putting the ‘ultra low power’ specmanship to one side. When a finite amount of charge is available from a battery cell, it is how an MCU uses energy – power over time – that’s vital. Minimising the product of power and time is just as important during sleep periods as it is in active periods. The EFM32 MCU is based on the ARM Cortex-M3 processor core and has been designed to significantly reduce active mode power consumption. In benchmark tests, EFM32 at 32Mhz on a real world 3V supply, will run proper application code from Flash at 160µA/MHz.

So that’s fine, but how long it takes for an MCU to handle a task has a crucial bearing on energy efficiency. Hence the use of the 32-bit Cortex-M3. More processing efficient than 8- and 16- bit devices, it executes tasks in far fewer clock cycles thereby dramatically reducing the active period. By keeping the active cycle as short as possible the 32-bit MCU enables more time to be spent in deep sleep mode. Forget too about that old maxim about 32-bit processors being incapable of

# Energy saving under new management

Published on Electronic Component News (<http://www.ecnmag.com>)

delivering sub- $\mu\text{A}$  standby modes. By applying the right low power design techniques they most certainly can. The EFM32 will deliver all the baseline functionality like real time counter, RAM and CPU retention, brown-out detection and power-on reset in a deep sleep mode while only using  $0.9\mu\text{A}$ .

Often in the kind of target applications mentioned, MCU duty cycles can be really very low, with the MCU staying in a deep sleep state for as much of 99% of the time. So current consumption here is really very important to overall energy efficiency.

This advantage is lost though if the time it takes for the MCU to wake up from deep sleep and re-enter the active mode is long. Why? Because when an MCU goes from a deep sleep state to an active state, there is always a wake-up period, where the processor must wait for the oscillators and power supply system to stabilise before starting code execution. Since no processing can be done in this period, the energy spent while waking up is wasted energy, and so reducing the wake-up time is important to reduce overall energy consumption.

More than this, MCU applications impose real time demands, which often mean that the wake-up time must be kept to 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 many existing MCUs, the device is often inhibited from going into deep sleep at all – not a very good solution for energy sensitive applications.

In response, EFM32 uses a combination of design techniques to reduce the wake-up time from deep sleep to only  $2\mu\text{s}$ , ensuring as little energy is used before the CPU starts processing tasks.

If energy savings are to be put fully under control and truly optimised, system designers also need the flexibility of a range of well-architected energy modes to select from. As will be seen in Table 1, EFM32 provides a number of modes enabling designers to hone the resources used at any point in time in order to maximise energy efficiency.

EFM32 with 3V power supply. Real application from memory.	EM0 Run Mode	EM1 Sleep Mode	EM2 Deep Sleep	EM3 Stop Mode	EM4 Shutoff Mode
Current consumption	160 $\mu\text{A}/\text{MHz}$	45 $\mu\text{A}/\text{MHz}$	0.9 $\mu\text{A}$	0.6 $\mu\text{A}$	20 nA
Wake-up time	-	0	2 $\mu\text{s}$	2 $\mu\text{s}$	160 $\mu\text{s}$
Wake-up events	Any	Any	32 kHz peripherals	Async IRQ, I2C slave Analog Comparators Voltage Comparators	Reset, GPIO rising/ falling edge
CPU (Cortex-M3/M0)	On	-	-	-	-
High frequency peripherals	Available	Available	-	-	-
Low frequency peripherals	Available	Available	Available	-	-
Asynchronous peripherals	Available	Available	Available	Available	-
Full CPU and SRAM retention	On	On	On	On	-
Power-on Reset/Brown-out Detector	On	On	On	On	On

**Table 1: The EFM32's energy modes, helping to maximise energy efficiency.**

While even these energy modes may seem to some observers to be a coarse division, yet more fine-grained tuning of the resources within each mode can be achieved by enabling or disabling different peripherals. Either way, the EFM32's energy modes help eliminate any energy wastage.

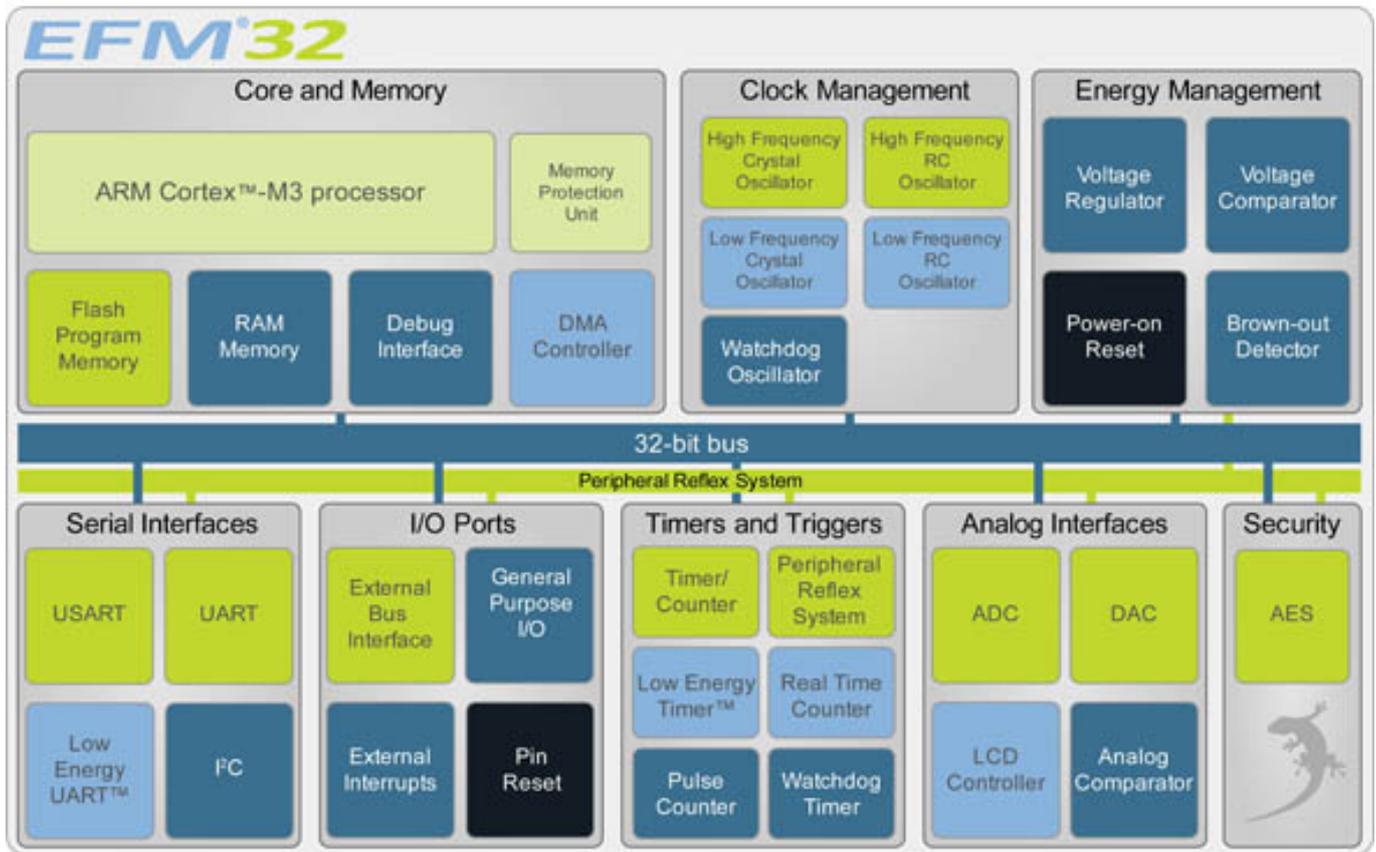
Naturally, the peripheral function blocks offered by a low power MCU need to be purpose designed for low power operation and the EFM32 is no exception. The MCU sports for example an 8-channel, 12-bit ADC using 350  $\mu$ A at full resolution and 1 Msamples/sec conversion rate; a 4x40 segment LCD controller using just 550nA sporting integrated voltage boost, contrast, animation and blink functions; and a special low energy UART, a full UART with 32kHz clock, consuming only 150nA at a data transmission speed of 9600 baud.

Creating an MCU architecture that enables the CPU to leave peripherals functioning autonomously is an important innovation in the quest for greater energy saving. So the EFM32 peripherals are designed to be able to look after themselves, leaving the CPU to either solve other high level tasks or simply fall asleep, saving energy either way.

Taking the autonomy ideal one step further, the EFM32's introduction of an additional programmable interconnect structure, called the peripheral reflex system, into an MCU architecture (Fig 2) enables peripherals to talk to peripherals without the intervention of the CPU, thereby reducing energy consumption even further.

# Energy saving under new management

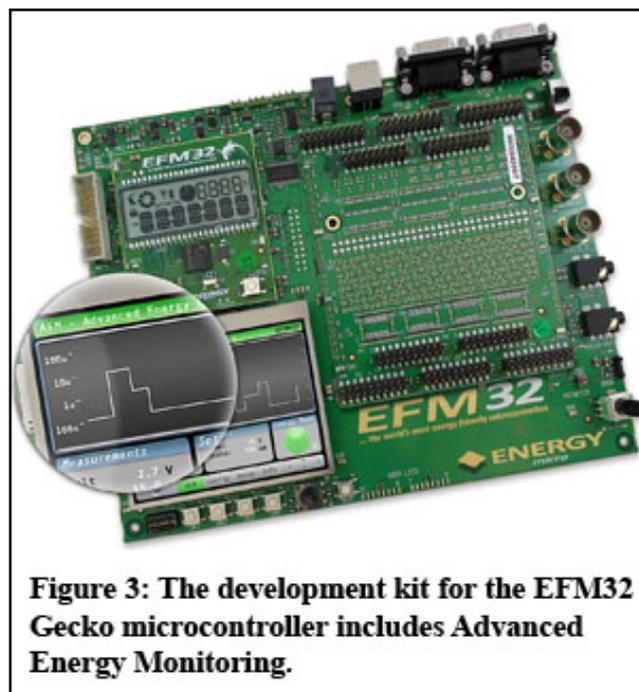
Published on Electronic Component News (<http://www.ecnmag.com>)



**Figure 2: The chip architecture for the EFM32 Gecko microcontroller showing peripheral reflex system.**

Having a super energy friendly MCU however will not by itself guarantee users the lowest possible energy usage. Having the right tools available to identify and remove energy drains at an early stage of prototype development for example can significantly reduce the overall energy consumption of the end product.

Energy Micro has recently announced Simplicity Studio, a complete graphical user interface development suite for the EFM32 microcontrollers. It will provide access to all the information, documentation and tools



required by hardware, firmware and software engineers to more quickly and effectively develop embedded systems. Many of its tools are already available.

The EFM32's development kits come with an Advanced Energy Monitoring (AEM) system, a facility that continuously measures current consumed. This measurement is integrated to depict accurately the power used over time, allowing real life use-cases to be optimised for low power operation. (Fig 3).

When used with the energyAware Profiler 'energy debugging' software tool, AEM enables the user to identify the actual source code being executed at a given moment in time as shown by an energy graph. This instantly gives the engineer a pointer to any part of the program that causes high energy consumption, allowing code to be optimised and energy savings to be managed more closely than ever before.

**Source URL (retrieved on 04/25/2015 - 1:11am):**

<http://www.ecnmag.com/articles/2011/04/energy-saving-under-new-management>