EFM32®

... the world's most energy friendly microcontrollers

Using EFM32 in Energy Harvesting Applications

AN0061 - Application Note

This application note presents considerations on hardware and software design when powering EFM32 microcontrollers from an energy harvesting source.

This application note includes:

- This PDF document
- Software examples (zip):
  - Example C-code
  - Multiple IDE projects

ARM ZERO TINY GECKO LEOPARD GIANT WONDER

SILICON LABS

1 Introduction

As energy consumption of low-energy electronics have continuously decreased, a new generation of energy sources has emerged. These new energy sources generate electrical current by harvesting the ambient energy that is always present in the environment.

Generating electrical current from energy in the environment is not a new invention. Wind and water turbines have been used for this purpose for a long time, but very small and efficient harvesters that can be used to power stand-alone embedded devices is a relatively new concept.

Energy harvesting is particularly suited for applications where battery power is impractical. A common example is devices that are inaccessible after installation. When powered by batteries, these devices end their lifetime when the battery runs out. Another example is large wireless sensor networks, where battery replacement is a significant maintenance task.

Due to low energy consumption of EFM32 microcontrollers, they are ideal for use in applications powered by harvested energy. But designers still need to take certain precautions when designing a system that utilizes energy harvesting. This application note addresses these particular challenges.

1.1 Energy Harvesting Sources

Energy Harvesting is a technology in evolution, and new energy sources for energy harvesting are still appearing. Some of the most common energy harvesting sources are presented briefly in this section.

1.1.1 Solar Energy

Solar cells are a well-established energy harvesting source for small devices. Small solar cells have been used for years to power everyday consumer appliances such as calculators and wristwatches.

Solar cells use the photovoltaic effect to convert solar radiation into electrical current. A modern solar converts around 20% of the solar radiation energy into electrical energy. Solar radiation energy will vary with the longitude, the seasons, the time of day, and the weather conditions. Maximum power density at equator is said to be around 100 mW per cm^2 while 24 hour average over the year typically vary from 30 mW per cm^2 near equator to 5 mW per cm^2 in the higher latitudes. Indoors, typical power output from a last generation solar cell can be 15 µW per cm^2

1.1.2 Thermal Energy

Harvesting thermal energy is done with a Thermoelectric Generator (TEG). A TEG generates electric current from a heat flow through a thermoelectric material. The output power of a TEG depends on its area and the temperature difference from one side of the element to the other.

1.1.3 Vibration Energy

A piezoelectric material is a solid material that will deform slightly when voltage is applied. The other way around, it will also set up a small voltage when it is compressed or stretched. The properties of these materials make them useful in many applications, and lately also in energy harvesting systems where the piezoelectric effect is used to convert vibration into electrical current. A recent evolution is to use this effect to generate electrical current from button clicks, generating as much as 2 mJ per click.

1.1.4 Current Loops

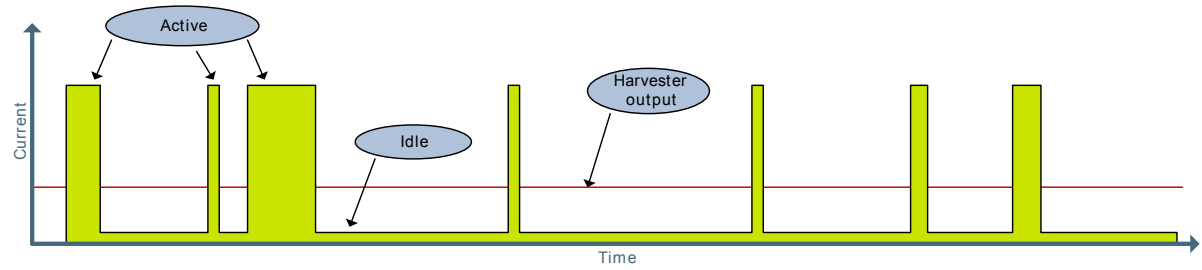
Signalling over a 4-20 mA current loop is a well-established technology in industrial applications. As current is always flowing through the loop, low-energy devices can harvest energy directly from the signal wires, without having to use a separate pair of wires for power.

2 Powering EFM32 with Harvested Energy

When powering the EFM32 with an energy harvester that is always able to generate more current than the application uses, no special precautions need to be taken. However, the normal case for the small harvesters used with microcontrollers is that the harvester output current is less than the peak current consumption.

In a typical application, a microcontroller spends only small parts of the time in active mode performing tasks like computation, communication or sensor evaluation. The rest of the time it is waiting in idle mode for user input, data acquisition or communication from another device. For EFM32 microcontrollers, active mode and the idle modes are called Energy Modes. Energy Mode 0 (EM0) is the active mode and EM1-EM4 are the different idle modes. For more detailed information about Energy Modes, please refer to application note AN0007 on Energy Modes and the device family reference manual. A typical current consumption profile for a low-energy microcontroller powered by an energy harvester is shown in Figure 2.1 (p. 3). For an EFM32 microcontroller the current consumption can vary by several orders of magnitude between the active periods and the idle periods, and the harvester output lies somewhere in between.

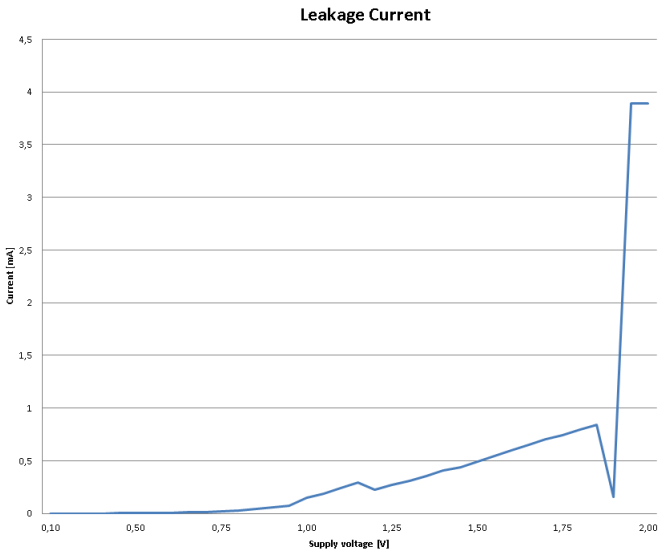
Figure 2.1. Typical current consumption



As active mode consumption is higher than the harvesting output, it is necessary to store the surplus energy that is harvested during the idle periods. Storing energy is done by adding an energy bank between the energy harvester and the microcontroller. When current consumption is high, the device is powered by the energy bank. The energy bank can be a rechargeable battery, a super-capacitor or a bank of capacitors. Which type is preferred depends on how long the bank must store energy. Electrolytic capacitors is the cheapest alternative, but they also have the highest leakage currents.

The energy bank will cause a significant slow-down of the power-up voltage rise time. With a limited current output, charging the energy bank will take some time. But as soon as the supply voltage starts rising above 0, the EFM32 will start consuming current and draining the energy bank. A measurement of typical current consumption for low input voltages is shown in Figure 2.2 (p. 4). The figure shows a slowly increasing current consumption until the supply voltage reaches around 1.85 V. Between 1.85 V and 1.98 V the current consumption is quite low until reset is released when the power-on threshold of 1.98 V is passed.

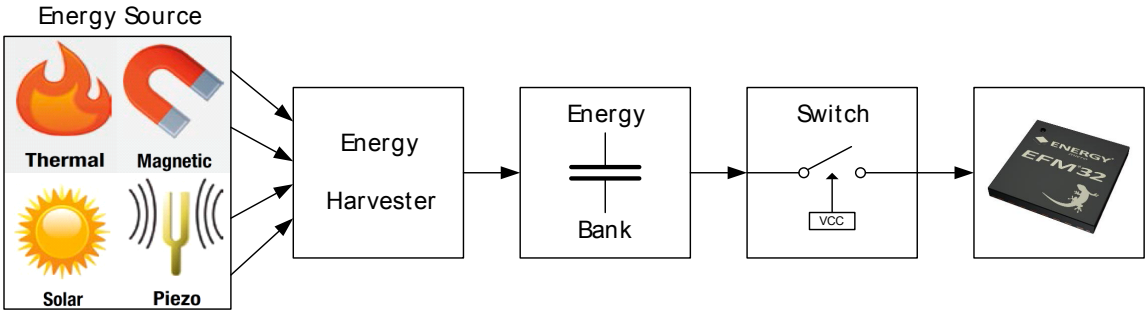
Figure 2.2. EFM32 Tiny Gecko leakage current for VDD below the operating range



The current consumption for voltages below the power-on threshold is caused by internal housekeeping circuitry, which is gradually powered up to ensure a controlled start-up of the microcontroller. If the supply voltage is switched relatively fast from off to on, this current consumption is hardly noticed. But when the supply voltage level is increased very slowly it quickly becomes significant. It may even drain the harvested energy source so fast that the output voltage will never reach the power-on threshold. The remedy for this is to isolate the energy bank from the microcontroller by a switch that turns on the micro controller supply voltage when the energy bank is charged, and disconnects the microcontroller when the energy bank is empty. This switch is further discussed in Section 2.2 (p. 6) .

A typical block diagram for an energy harvesting configuration is shown in Figure 2.3 (p. 4)

Figure 2.3. Energy harvesting block schematics



When planning a harvesting system, proper dimensioning of the harvester and the energy bank is important to ensure that the application always has enough available energy to perform its tasks.

2.1 Energy Storage Bank

When powering a device from an energy harvester, it is essential to be able to store the small amounts of energy that trickles in from the energy source. The most common way of storing energy is to use a bank of electrolytic capacitors, but batteries and super-capacitors may also be used depending on the nature of the energy source and the application requirements.

To dimension the storage bank, one must have a good knowledge about the energy source and the expected output of the harvester. It is equally important to know how much current the device consumes when performing its different tasks. In particular which tasks consumes the most energy, and how much

current they consume for how long. Typical energy intensive activities include MCU reset and power-up, communications on cabled or wireless interfaces and long computations. The Advanced Energy Monitor (AEM) on EFM32 Starter or Development kits is a very useful tool to characterize an application's current consumption. Using the AEM on energy harvesting devices is described in more detail in Chapter 4 (p. 10)

When dimensioning the energy harvester and the energy bank for start-up, it can be useful to split the current consumption into separate parts.

2.1.1 Charging the decoupling capacitors.

An EFM32 microcontroller can have very sudden increases in the current consumption. For instance when waking up from a sleep mode, switching GPIO pins on and off, and in many other situations. To ensure a stable supply voltage, a buffer of external decoupling capacitors must placed near the EFM32.

When unpowered for a long time, all the decoupling capacitors will be completely discharged. When the supply voltage switch is closed, there will be a sudden voltage increase over the decoupling capacitors. Initially the decoupling capacitors appear as a short circuit, and the current flow will only be limited by the resistance and inductance in the power lines.

Dimensioning the capacitor bank needed to charge the decoupling capacitors is done by considering the two capacitors on each side of the switch. Initially, the capacitor bank is fully charged to the voltage where the switch closes, while the decoupling capacitors are empty. When the switch is closed, charge will flow from the bank to the decoupling caps until a stable state is reached.

The stored charge in the energy bank before switching in the EFM32 is given as

$$Q_0 = C_{bank} \cdot V_{out} \tag{2.1}$$

where V_{out} is the output voltage of the harvesting regulator. To ensure that the stored charge is enough to avoid the voltage to drop below a minimum level when switching on the MCU, the minimum stored charge must be:

$$Q_{min} = (C_{bank} + C_{mcu}) \cdot V_{min} \tag{2.2}$$

where C_{mcu} is the total decoupling capacitance of the EFM32.

The minimum required bank capacitance that is needed to prevent voltage dropping below a minimum threshold is:

$$C_{bank} > C_{mcu} \cdot \frac{V_{min}}{V_{out} - V_{min}} \tag{2.3}$$

For an EFM32, the typical total decoupling capacitance is in the range 10-15 µF, and the minimum operating voltage is 1.85 V. Please refer to AN0002 on hardware design considerations for exact recommendations on decoupling, and the device specific data sheet for the exact operating voltage requirements.

2.1.2 Microcontroller start-up current.

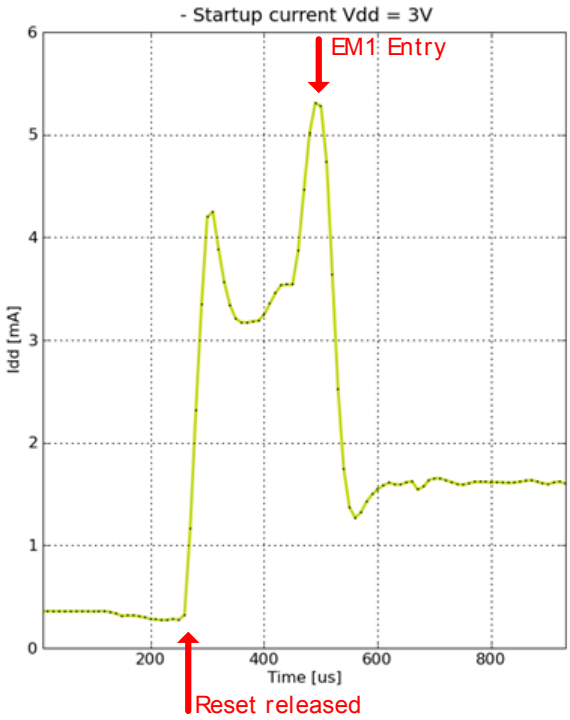
There is also need to store enough energy to get the EFM32 itself out of reset and up and running. Applying the same reasoning as when dimensioning for charging the decoupling capacitors, one can calculate the required capacitance for overcoming the power-on reset in the following way:

$$Q_{startup} = C_{bank} \cdot (V_{out} - V_{min})$$

$$C_{bank} > \frac{I_{startup} \cdot t_{startup}}{V_{out} - V_{min}} \quad (2.4)$$

Measurements show that starting up an EFM32 Tiny Gecko draws approximately 4 mA for 300 µs, as shown in Figure 2.4 (p. 6)

Figure 2.4. EFM32 Gecko start-up current consumption



In this measurement, reset is released around 250 µs, and the EFM32 enters Energy Mode 1 right at the start of the *main()* function around 300 µs later.

2.1.3 User application initialization.

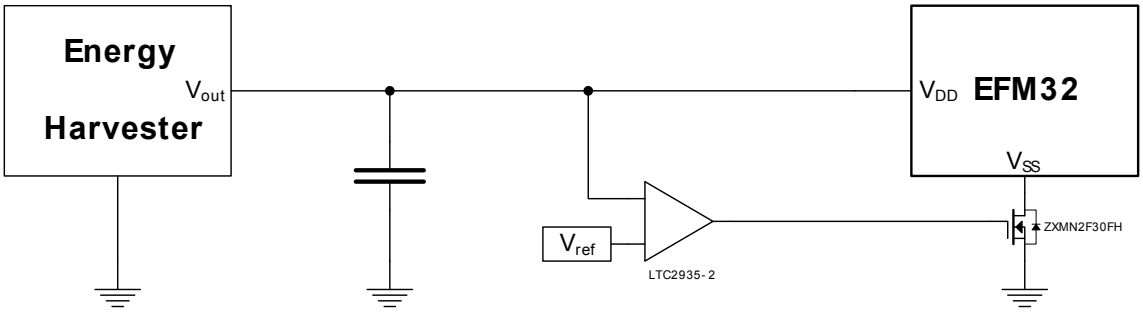
The start-up current and time that has been discussed so far is the absolute minimum start-up time. The energy calculations so far only covers what is consumed before the *main()* function starts. Normally, user software will need to do some initialization as well, so the capacitor bank must also be dimensioned for the current consumption during this phase. However, once user software is running, a sleep mode can be used to prevent the microcontroller from draining the energy bank.

Note that the above equations describe ideal components, and capacitor self-discharge is not taken into account. So the capacitor bank must be larger than what is indicated above. The rate of self-discharge will depend on several external factors such as capacitor type, ambient temperature and more. Please refer to the capacitor manufacturers' data sheets for more details on this.

2.2 Voltage Controlled Power Switch

To prevent the MCU from draining the harvested energy before the energy bank has built up enough reserves to safely power-up the microcontroller, a switch can be inserted between the harvester and the microcontroller. Figure 2.5 (p. 7) shows a conceptual sketch of such a switch.

Figure 2.5. Power switch



Important considerations when designing a power switch circuit includes ensuring that the switch is well defined in the OFF position for input voltages down to 0 V. The switch should feature hysteresis between the ON and OFF thresholds to prevent the switch from shutting off the MCU before the energy bank voltage drops below the brown-out level. Also, the design should use an energy efficient voltage comparator.

The full schematics of the above switch can be found on Linear Technology's Energy Harvesting Multi-Source Demoboard. This demonstration board is paired with an EFM32GG-STK3700 and sold as an "Energy Harvesting Solution To Go" by Würth Elektronik.

2.3 Supply Voltage Level

As EFM32 microcontrollers feature internal voltage regulators, hardware designers do not have to spend PCB space on creating different voltages. When designing for energy harvesting, one should also be aware that the internal regulators are of the linear type. So the current consumption of the EFM32 is almost independent of the supply voltage. Therefore one might consider reducing the operating voltage to reduce the amount of energy that is burnt in the regulators.

3 Software Design Considerations

When writing software for applications powered by energy harvesting, the general recommendations given in AN0027 on Energy Optimization should be followed. Additionally, a few more advice specific to energy harvesting applications are presented in this section.

3.1 Fast Start-up

When power-on reset is released, it is important to go to a sleep mode fast to allow the energy bank to recharge before it is depleted. Global variables are normally initialized in the start-up code, which is run before *main()* is entered. If very large global variables are assigned, the start-up time will increase. If the start-up time is long, getting the microcontroller out of power-on reset requires a larger energy reserve.

It is possible to enter a sleep mode during the start-up code. This will give the energy source time to charge the energy bank for a while. When browsing all EFM32 example projects, one will notice the two device specific CMSIS files, `startup_efm32tg.s` and `system_efm32tg.c`. The latter file includes a function called *SystemInit()* which is called before the initialization of global variables. This function can be used to set up an RTC delay like explained in Section 3.1.1 (p. 8) or a voltage monitor that is described in Section 3.2.1 (p. 8) .

3.1.1 LFXO Start-up Software Example

Many applications require a 32.768 kHz precision oscillator for time keeping. However, starting a crystal oscillator can take many clock cycles. This application note includes a software example called `lfxo_startup` that shows how the LFXO can be started while the EFM32 waits in Deep Sleep.

In the example, the ULFRO is clocking the RTC which triggers an interrupt at a fixed interval. In the Interrupt Service Routine, the MCU wakes up and polls the CMU status register to check if the LFXO is started. Polling is used instead of an interrupt from the Clock Management Unit because CMU interrupts are not available in EM2.

3.2 Avoiding Brownout

Once the microcontroller is past the first power-on reset, further resets should be avoided. Depending on the application duty cycle, it is often more energy efficient to spend the idle time time in one of the sleep modes (EM1-EM3).

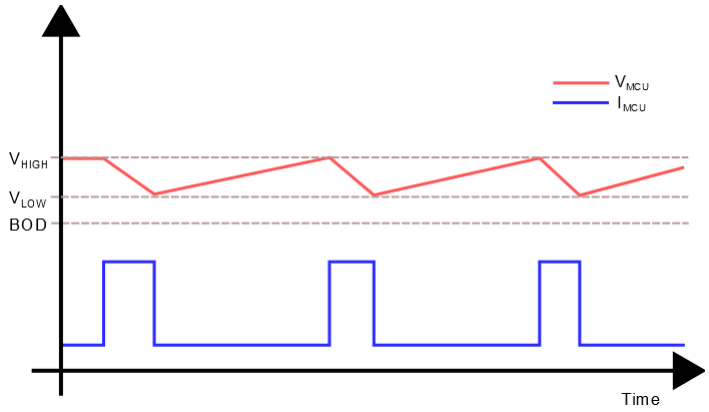
Wake-up from EM4 requires a reset, so in energy harvesting applications EM4 should be used with care. A rule of thumb is that using EM4 will result in a lower average energy consumption than EM3 only when the wake-up interval is above 1-2 seconds (see AN0027 for more details).

3.2.1 Voltage Monitoring

The voltage comparator (VCMP) on EFM32s can be used to monitor the supply voltage. shows how the VCMP can be used to put the MCU to a sleep mode if the supply voltage falls below a configurable threshold. This function is implemented by combining the voltage comparator (VCMP), the interrupt controller (NVIC) and the Cortex-M “sleep on exit” functionality. The "sleep on exit" functionality is explained in more detail in AN0039 on Interrupt Handling.

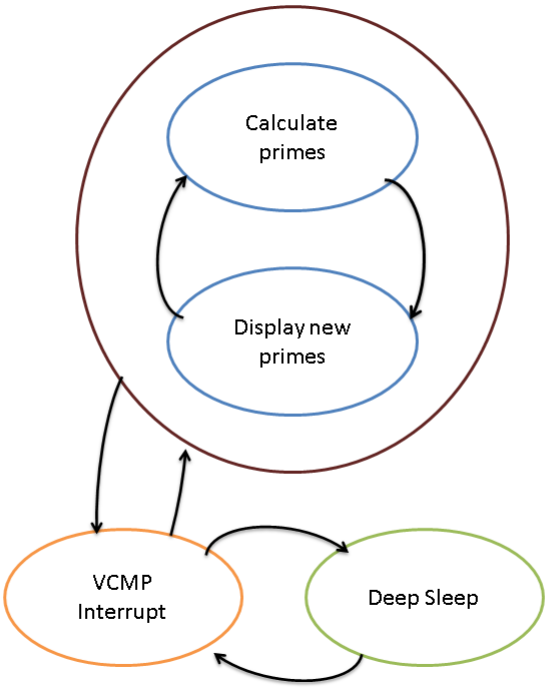
Once the supply voltage drops below a specified low voltage threshold, the VCMP interrupt handler enters a low-energy mode (EM1-EM3) until the voltage has been built up to the voltage high threshold. As long as the current output of the energy harvester is more than the sleep-mode current consumption, this will prevent a brown-out reset. Figure 3.1 (p. 9) shows how the VCMP makes the EFM32 enter a sleep mode when the supply voltage drops below a pre-defined threshold, which should be above the brown-out level.

Figure 3.1. Voltage Monitoring



The `vcmp_monitor` example that is included with this application note shows how this can be implemented. The example application is continuously calculating prime numbers. Every time a prime number is found, it is displayed on the LCD. When the supply voltage drops below the VCMP threshold, the VCMP interrupt triggers and sleep is entered. When rising supply voltage triggers the VCMP interrupt, the application resumes prime number calculation from the point it was before sleep was entered.

Figure 3.2. Voltage Monitoring Example Program Flow



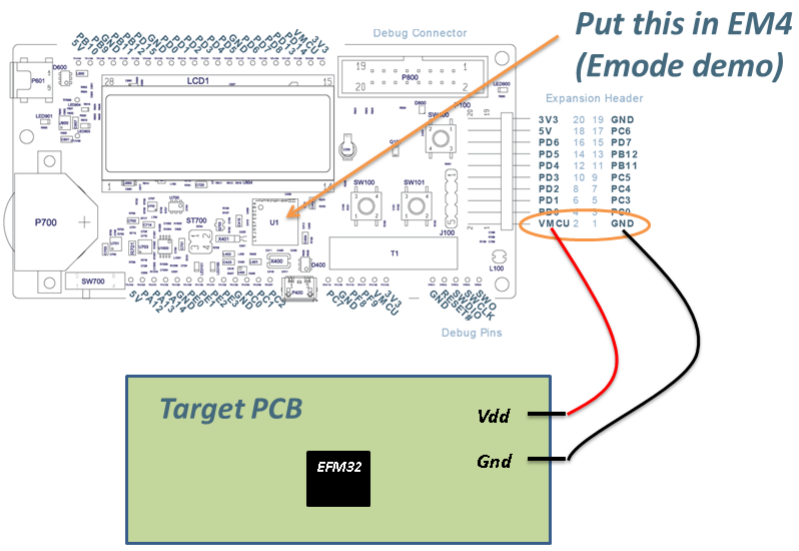
Any other interrupt can also wake the EFM32 even if the supply voltage is below the specified threshold. To avoid this, the VCMP interrupt handler can be modified to enable and disable the interrupts that can be ignored when the supply voltage is low.

4 Using The AEM for Energy Harvesting Applications

AEM, the Advanced Energy Monitor is a feature that is built into all Energy Micro Starter and Development Kits. The AEM is continuously sampling the Program Counter while simultaneously measuring the current and voltage consumption. The energyAware Profiler displays the AEM data on a computer screen. This makes the AEM a powerful tool when designing a device powered by energy harvesting.

The AEM is most often used to measure the energy consumption of the MCU on board the Starter or Development kit, but it can also be used to measure the current and voltage consumption on another PCB. To do this, connect the PCB as shown in Figure 4.1 (p. 10), and use the **emode** demo to put the kit MCU in Energy Mode 4 where its 20 nA current consumption is negligible.

Figure 4.1. Using AEM to measure an external PCB



When using the AEM to characterizing the energy consumption of a design, the first step is to measure its current consumption. To use the AEM to measure the current consumption of an external PCB, the target board must be supplied by the 3.3 V VMCU net on the Energy Micro kit, and the energy harvester must be disconnected from the target MCU. On a Starter Kit, the battery switch must be in the "DBG" position.

The AEM can also measure the supply voltage over time. In energy harvesting applications this makes it possible to monitor state of the energy bank. To measure the supply voltage on a target PCB, the energy harvester must be connected to the target MCU, and the VMCU rail on the kit must be unpowered. On a Starter Kit VMCU is turned off by removing the battery and setting the battery switch to the "BAT" position. On a Development Kit, VMCU can be shut off by using the user interface on the TFT display.

If the SWO output from the target board is connected to the AEM, the program counter on the target board is also sampled, making it possible to link both the current consumption and the voltage profile with the application code.

For a detailed introduction to the energyAware Profiler and the AEM, please refer to AN0027 on Energy Optimization, Chapter 4 and the Lizard Labs training module on the energyAware Profiler: www.energymicro.com/lizard-labs/energyaware-profiler.

5 Revision History

5.1 Revision 1.02

2013-09-03

New cover layout

5.2 Revision 1.01

2013-06-12

Ported **vcmp_monitor** example to EFM32GG-STK3700 so it runs with the Energy Harvesting To Go Design Kit

Minor corrections

5.3 Revision 1.00

2013-06-03

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories 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 Silicon Laboratories 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. Silicon Laboratories 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. Silicon Laboratories 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 Silicon Laboratories. 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. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories 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

Silicon Laboratories Inc., Silicon Laboratories, the Silicon Labs logo, Energy Micro, EFM, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

B Contact Information

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

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

List of Equations

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4