CPE 323
MODULE 14
ENERGY, POWER, AND LOW-POWER MODES

Aleksandar Milenković

Email: milenka@uah.edu

Web: http://www.ece.uah.edu/~milenka

Overview

This module discusses energy, power, and low-power operating modes in the MSP430 family of microcontrollers.

Objectives

Upon completion of this module learners will be able to:

- *Describe factors impacting energy and power of a running program on a microcontroller;*
- *Utilize low-power modes in their programs to extend operating time of battery-powered embedded systems;*
- *Utilize EnergyTrace Technology in Code Composer Studio to perform power profiling of their programs.*

Contents

1 Introduction

We traditionally care deeply about several aspects of a computer system we design: (a) functionality – what services or tasks it can carry out; (b) performance - how much time does it take to complete a task; and (c) cost - how much does it cost to acquire and operate the computer system. However, energy-efficiency have emerged as a critical aspect of computer systems with proliferation of wearable and mobile computing platforms as well as wireless sensor network nodes that rely on batteries as a source of energy. The energy available is batteries is limited. A need to frequently replace or recharge batteries reduces convenience and availability and increases operating costs. The amount of energy stored in a battery (a.k.a., battery capacity) is directly proportional to its size and weight. On the other side, we want computer systems to be lightweight and have a small form factor. No one would be interested in a smartphone weighing 1 kg. These factors bring the total energy needed to complete a task and the average power consumed into the focus of computer engineers. We want computing platforms not only to give us excellent functionality and performance at minimal cost, but also to be very energy-efficient – meaning they can operate for a long period of time on a single battery charge and we want the battery to be lightweight and small.

We care about energy-efficiency of computing platforms for reasons that go beyond extending their operating time on a single battery charge. For example, in wearable electronics and wireless sensor networks we would like to be able to harvest energy from the environment (a.k.a. energy scavenging). The existing energy harvesting systems provide a relatively small amount of energy for low-power and low-energy electronic systems. So, designing low-power systems is imperative if we want to rely on energy harvested from the environment. Next, in high-performance and cloud computing, electricity bills for powering and cooling computing platforms and networking equipment dominate the operating costs. High-end processors have tens of billions of transistors, each switching billions of times in a second resulting in a significant amount of heat that needs to be taken off of the chip to prevent it from melting or malfunctioning. Thus, power impacts cooling costs, packaging costs, and reliability of computing platforms. Last but not least, the energy demands of modern computing infrastructures continue to increase with an increase in the number of computing platforms and their interconnectivity, contributing to growing emissions of greenhouse gasses.

In this module we discuss energy and power in CMOS integrated circuits (CMOS ICs). Specifically we discuss how power supply voltage, complexity of the chip, the activity of individual components, and the clock frequency impact energy and power. Next, we briefly discuss operating modes of the MSP430 family of microcontrollers and how to transition into and out of these modes. Finally, we discuss power profiling using the EnergyTrace Technology component available in the TI's Code Composer Studio (CCStudio).

2 Energy and Power in CMOS Integrated Circuits

Power is drawn from a voltage source attached to the V_{DD} pins of a chip. We define the following terms:

- The instantaneous power consumed or supplied by a circuit element is the product of the current through the element and the voltage across the element: $P(t) = I(t) \cdot V(t)$.

- The energy consumed or suppled over an interval *T* is the integral of the instantaneous power: $E = \int_0^T P(t) \cdot dt$.
- The average power over this interval is determined as $P_{avg} = \frac{E}{T} = \frac{1}{T} \cdot \int_0^T P(t) \cdot dt$

Power is expressed in units of Watts (W). Energy is usually expressed in Joules (J), where 1 W = 1 J/1 s. Energy in batteries is often given in W-hr, where 1 W-hr = (1 J/s)(3600 s/hr)(1 hr) = 3600 J. Alternatively, the battery capacity is given in mA-hr (milliAmpere-hours). For example, let us consider a standard AA battery that provides V=1.5 V and has the capacity of 2500 mA-hr. The energy of such a battery is E = 2.5 A-hr * 1.5 V * 3600 s/hr = 13,500 J.

To understand energy consumed by CMOS logic gates we consider the simplest logic gate, an inverter shown in Figure 1. It consists of a PMOS transistor in the pull-up portion of the circuit (between the V_{DD} and the output V_{out}) and an NMOS transistor in the pull-down portion of the circuit (between the V_{out} and the GND). The energy is taken from the source when the output of the inverter transitions from a logic 0 to a logic 1 (0→1). For this transition to happen, the input of the inverter transitions from a logic 1 to a logic 0 (1→0). When V_{in}=0 V (logic 0 at the input), the NMOS transistor is OFF, and the PMOS transistor is ON. Thus, we can model the equivalent circuit as shown in Figure 2. The PMOS transistor goes through saturation and liner modes of operation during this transition, but we can simplify its behavior and model it as a resistor with an equivalent resistance *R*. The rest of the circuit is modelled by a load capacitor with the capacitance *C*. The current flows from the source through the PMOS and charges the output capacitor. The transition is completed when the output voltage reaches the power supply voltage, i.e., V_{out}=V_{DD}, and the current stops flowing. The product of the resistance *R* and capacitance *C* is often referred to as the time constant, i.e., $\tau = RC$. Please note that at time time $t = 4\tau$ from the start of the transition, $v_{out} \approx 0.98V_{DD}$.

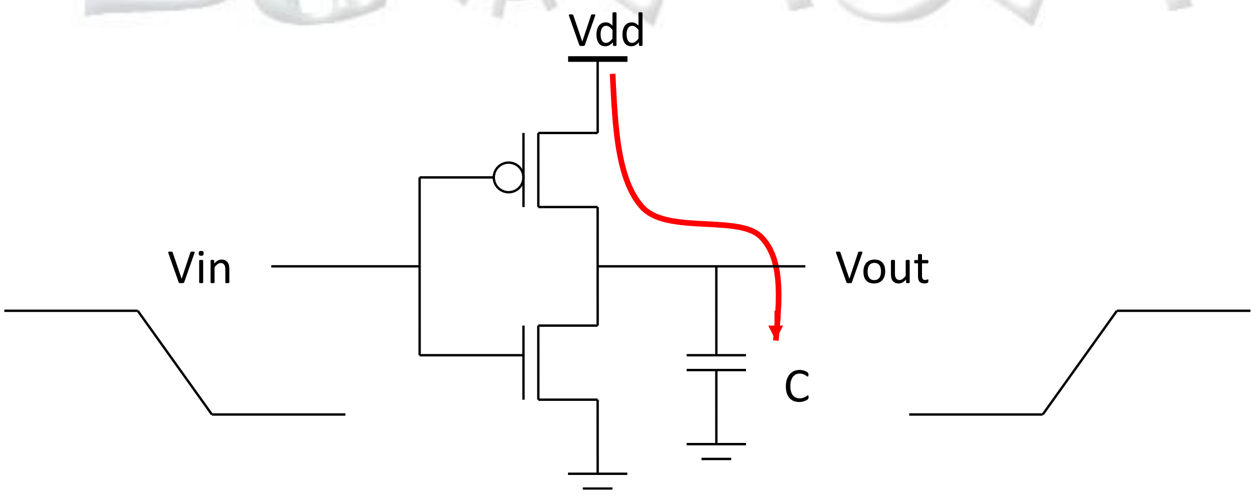
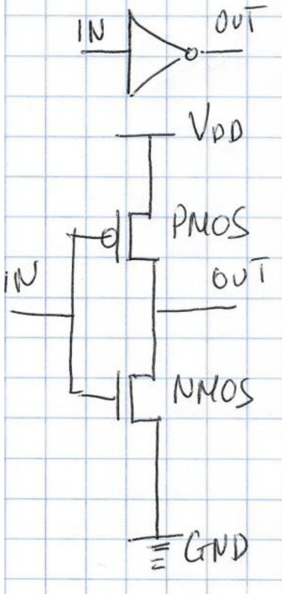


Figure 1. CMOS Inverter.

(1)

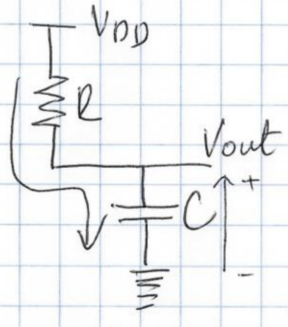


Energy taken from source
when OUT switches from 0 ⇒ 1
⇒ IN switches from 1 → 0

$V_{in} = 0\ V \Rightarrow$ PMOS is on
NMOS is off

Equivalent circuit:

$t=0 \quad V_{out}(0)=0$



$$V_{DD} = Ri + v_{out} \quad (1)$$

$$i = C_L \frac{dv_{out}}{dt} \quad (2)$$

$$V_{DD} = RC \frac{dv_{out}}{dt} + v_{out} \quad | : RC$$

$$\frac{dv_{out}}{dt} + \frac{1}{RC} V_{out} = \frac{V_{DD}}{RC} \quad (RC = \tau)$$

$$V_{out} = V_{DD} \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$$

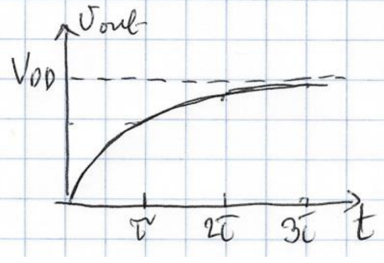


Figure 2. CMOS Inverter, dynamic response for (0→1) transition on the output.

We can determine energy taken from the source during one 0→1 transition at the output of the CMOS inverter as shown in Figure 3. Please note that we transition from the time domain to integrate over the output voltage that changes from 0 V at $t = 0$ to V$_{DD}$ at $t = \infty$. The energy depends only on the capacitance *C* and power supply voltage V$_{DD}$. This component is known as switching energy.

$$E_{source} = \int_0^{+\infty} V_{DD} \cdot i \, dt = V_{DD} \int_0^{+\infty} i(t) \cdot dt = V_{DD} \int_0^{+\infty} C \frac{dv_{out}}{dt} dt \quad \textcircled{2}$$

$$= C V_{DD} \int_0^{V_{DD}} dV_{out} = C V_{DD} \, V_{out} \Big|_0^{V_{DD}} = C V_{DD} (V_{DD} - 0) = C V_{DD}^2$$

$$E_{source} = C V_{DD}^2$$

Figure 3. Energy taken from the source during one 0→1 transition on the output of the CMOS inverter.

What happens with this energy? We can apply energy equations for each component, the PMOS transitor (modelled as a resistor) and the capacitive load (modelled as a capacitor). Figure 4 shows the energy equations for these components. One half of the energy taken from the source is dissipated into heat on the PMOS transistor and the other half is stored in form of charges on the capacitor. Once we have a transistion 1→0 on the output of the inverter, the energy stored in the capacitor will turn into heat on the NMOS transistor. Thus, all energy taken from the source eventually dissipates into heat that needs to be taken off of the chip. If you look into modern desktops, you will see that processors sit beneath very sophisticated heat sinks designed in such a way to maximize heat dissipation. Often a fan will be mounted on top of the heat sink to create airflow that is, with the help of additional fans, taken out of the computer case.

$$E_C = \int_0^{+\infty} V_{out}\, i\, dt = \int_0^{+\infty} V_{out}\, C \frac{dV_{out}}{dt} \cdot dt$$

$$= C \int_0^{V_{DD}} V_{out} \cdot dV_{out} = C \cdot \frac{1}{2} V_{out}^2 \Big|_0^{V_{DD}} = \frac{1}{2} C V_{DD}^2$$

$$E_{source} = E_C + E_{PMOS}$$

$$E_{PMOS} = \int_0^{+\infty} (V_{DD} - V_{out})\, i \cdot dt = \int_0^{+\infty} C (V_{DD} - V_{out}) \frac{dV_{out}}{dt} dt$$

$$= \int_0^{V_{DD}} C (V_{DD} - V_{out})\, dV_{out} = C V_{DD}^2 - \frac{1}{2} C V_{DD}^2 = \frac{1}{2} C V_{DD}^2$$

turned into heat at PMOS transistor

Figure 4. Energy taken from the source during one 0→1 transition on the output of the CMOS inverter is split between energy on the capacitor (one half) and energy dissipated into heat on the PMOS transistor.

The energy equations from above are somewhat simplified, but representative for any CMOS integrated circuit. No matter how complex an IC is, it is a collection of CMOS gates and the reasoning applied on the inverter applies universally. Let us now generalize – we have many gates on a chip and so far we considered only switching power (energy taken when we are having transitions at the output). Unfortunately, our input signals are not perfect step functions (0→1 or 1→0 transitions are not instantaneous), rather they have a slope. Thus for a short period of time, t_{SC}, both PMOS and NMOS transistors are ON. This results in a current flowing between the source and the ground that is known as the short circuit current, I_{SC}. This current also contributes to the total energy consumption. Next, modern transistors do not behave as perfect switches - as transistors are getting smaller and smaller, they become leaky. That means that some charges are leaking regardless of whether we have transitions or not. As long as the power supply is turned ON, some charges are lost and need to be compensated by the source. The lost charges are quantified by the leakage current I_{LEAK}. Please note that this component is static, i.e., it is present even though there may not be any transitions at the output of logic gates.

Figure 5 shows a general equation for the total energy consumed in CMOS ICs that includes the switching component, the short-circuit switching component, and the leakage current component. We assume that the program execution takes T seconds. Not every gate is going to switch every clock cycle. The total number of transitions 0→1 can be calculated as follows: $\alpha_{0\to1} = p_{0\to1} \cdot f \cdot T$, where $p_{0\to1}$ is activity or probability of having 0→1 transition and f is the system clock frequency.

$$E = \underbrace{CV_{DD}^{2} \cdot \alpha_{0\to1}}_{\text{switching energy}} + \underbrace{V_{DD} \cdot I_{SC} \cdot t_{SC} \cdot \alpha_{0\to1}}_{\text{short circuit current induced energy}} + \underbrace{V_{DD} \cdot I_{Leak} \cdot T}_{\text{leakage energy}}$$

$\alpha_{0\to1}$ = the number of transitions $0 \to 1$ at the output of gates

$$\alpha_{0\to1} = p_{0\to1} \cdot f \cdot T$$

↑ Probability that the output switches from $0 \to 1$

$$0 \le p_{0\to1} \le 1$$

Figure 5. Equation for the total energy that includes switching component, short-circuit switching component, and leakage current component.

The average power can be determined as P=E/T, i.e.,

$$P_{avg} = p_{0\to1} C V_{DD}^{2} f + p_{0\to1} V_{DD} I_{SC} t_{SC} f + V_{DD} I_{LEAK}$$

This equation gives us a framework to guide the design of energy-efficient circuits (design-time techniques) as well as to guide run-time optimizations to reduce power.

How do we minimize energy/power? The most promising knob we have is to reduce the power supply voltage, V_{DD}, because the switching energy is proportional to V_{DD}^2. The power supply of inner logic in modern ICs is below 0.5 V (this does not apply to the power supply used for external I/O pins that are in order 3-5 V). However, lowering power supply can impact how fast our gates are switching. Next, it is also important to minimize activity when possible – e.g., eliminating unnecessary glitches in logic design will also help reduce switching power. The total capacitance on a chip is a function of the number and type of logic gates, their fan-out, and wire lengths. Short-circuit current can be controlled by avoiding having small gates driving long wires or by avoiding large fan-outs. Finally, leakage current can be reduced by turning off unused portions of the chip. Modern high-end processors include a lot of hardware resources solely devoted to power management. They employ Dynamic Voltage and Frequency Scaling (DVFS) that allows for dynamic changes of the power supply voltage and frequency to adjust to the current workload – the processor can run at a lower frequency when you are reading an email or browsing the Internet and at a higher frequency when editing a video. Next, there are provisions to turn off unused components, so the leakage power can be reduced.

The MSP430 family of microcontrollers is the industry leader in its support for low-power modes. They allow us to progressively turn off clocks (MCLK, SMCLK, ACLK) or unused resources. In addition, sophisticated clock subsystems allow for software control of clock frequency on-the-fly (no need to stop program execution), so one can run at minimal clock frequency that meets application demands.

Please note that sometimes low-power does not necessarily means lower energy. Often times running computation at highest clock frequency and going into a low-power mode is better for the total energy rather than running at a lower clock frequency for longer time.

`Things to remember 2-1. Average power in CMOS ICs`

`The average power in CMOS integrated circuits is shown below:`

$$P_{avg} = p_{0\rightarrow1}CV_{DD}^2f + p_{0\rightarrow1}V_{DD}I_{SC}t_{SC}f + V_{DD}I_{LEAK}$$

It includes three components:

(a) switching power – it is a function of activity (probability for having transitions $0\rightarrow1$), total capacitance, clock frequency, and power supply voltage squared;

(b) switching short-circuit power – it is a function of activity, power supply, frequency, short circuit current, and short-circuit period;

(c) leakage power – it is a function of the power supply and the leakage current.

3 MSP430 Operating Modes

The MSP430 family is designed for ultra-low-power applications and uses different operating modes as shown in Figure 6. The low-power modes LPM0 to LPM4 are configured using dedicated mode-control bits in the status register: CPUOFF, OSCOFF, SCG0, and SCG1. Table 1 shows encoding for different operating modes.

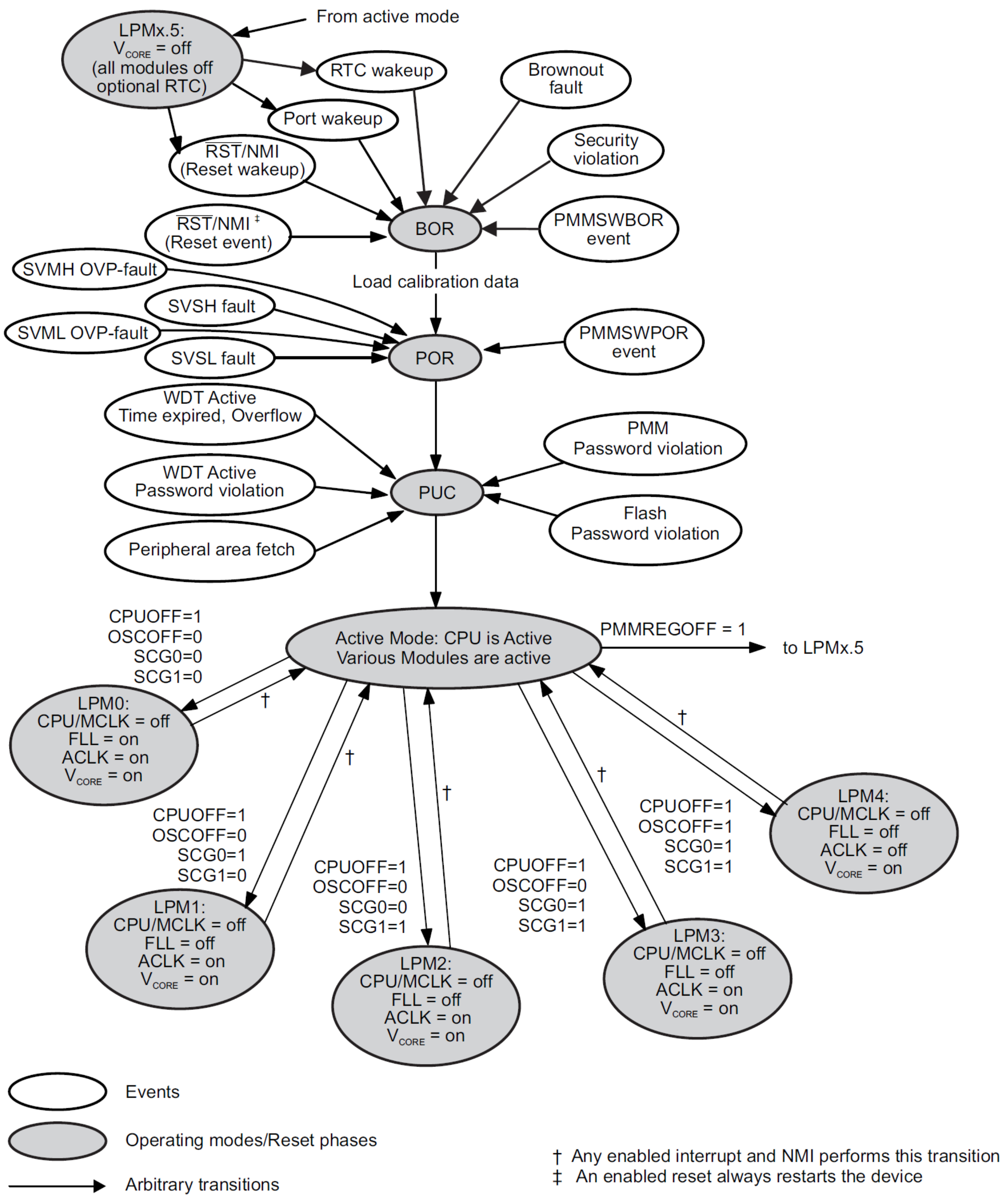

From active mode
LPMx.5: VCORE = off (all modules off optional RTC)
RTC wakeup
Brownout fault
Port wakeup
Security violation
RST/NMI (Reset wakeup)
RST/NMI ‡ (Reset event)
BOR
PMMSWBOR event
SVMH OVP-fault
Load calibration data
SVSH fault
SVML OVP-fault
SVSL fault
POR
PMMSWPOR event
WDT Active Time expired, Overflow
PMM Password violation
WDT Active Password violation
PUC
Flash Password violation
Peripheral area fetch
CPUOFF=1 OSCOFF=0 SCG0=0 SCG1=0
Active Mode: CPU is Active Various Modules are active
PMMREGOFF = 1
to LPMx.5
LPM0: CPU/MCLK = off FLL = on ACLK = on VCORE = on
†
CPUOFF=1 OSCOFF=0 SCG0=1 SCG1=0
CPUOFF=1 OSCOFF=0 SCG0=0 SCG1=1
CPUOFF=1 OSCOFF=0 SCG0=1 SCG1=1
CPUOFF=1 OSCOFF=1 SCG0=1 SCG1=1
LPM1: CPU/MCLK = off FLL = off ACLK = on VCORE = on
LPM2: CPU/MCLK = off FLL = off ACLK = on VCORE = on
LPM3: CPU/MCLK = off FLL = off ACLK = on VCORE = on
LPM4: CPU/MCLK = off FLL = off ACLK = off VCORE = on
Events
Operating modes/Reset phases
Arbitrary transitions
† Any enabled interrupt and NMI performs this transition
‡ An enabled reset always restarts the device

Figure 6. MSP430 Operating Modes.

Table 1. Operating Modes

Mode	SCG1	SCG0	OSCOFF	CPUOFF	CPU/Clocks Status
Active	0	0	0	0	CPU, MCLK active; ACLK active; SMCLK is optionally active (SMCLKOFF=0); DCO is enabled if it sources clocks; FLL is enabled if DCO is enabled;
LPM0	0	0	0	1	CPU, MCLK are disabled; ACLK active; SMCLK is optionally active (SMCLKOFF=0); DCO is enabled if it sources clocks; FLL is enabled if DCO is enabled;
LPM1	0	1	0	1	CPU, MCLK are disabled; ACLK active; SMCLK is optionally active (SMCLKOFF=0); DCO is enabled if it sources clocks; FLL is disabled;
LPM2	1	0	0	1	CPU, MCLK are disabled; ACLK active; SMCLK is disabled; DCO is enabled if it sources ACLK; FLL is disabled;
LPM3	1	1	0	1	CPU, MCLK are disabled; ACLK active; SMCLK is disabled; DCO is enabled if it sources ACLK; FLL is disabled;
LPM4	1	1	1	1	CPU and all clocks are disabled;
LPM3.5	1	1	1	1	PMMREGOFF=1, regulator is disabled; no memory retention; RTC can work;
LPM4.5	1	1	1	1	PMMREGOFF=0, regulator is disabled; no memory retention; RTC cannot work;

Entering a low-power mode is carried out by simply setting appropriate control bits in the status register. Exiting a low-power mode is done only through interrupt service routines. Recall that SR is cleared in the exception processing, ensuring that the MSP430 enters the active mode. By executing the RETI at the end of the ISR, the original SR is popped off the stack, ensuring a return into a low-power mode. Optionally, we can change these bits in the copy of the status register on the stack while inside an ISR, thus enabling a return into a different operating mode upon exiting the ISR. Code 1 shows assembly code snippets for entering low-power modes as well as code snippets that are executed within an ISR routine for exiting low-power modes upon return. Code 2 shows similar examples in C.

By changing any mode-control bits we enter the selected operating mode immediately. All peripherals operating on any of the disabled clocks become inactive. Thus, if entering an LPM that disables a clock that is powering a peripheral that should wake you up is clearly going to result in a deadlock. Thus, do not cook your rooster. While in an LPM all I/O port pins, registers, and RAM memory remain unchanged.

```
// Enter LMP0
  bis.w #GIE+CPUOFF, SR

// Exit LMP0 from an ISR
  . . .
  bic.w #CPUOFF, 0(SP)     // clear CPUOFF in the SR copy on the TOS (0+SP)
  reti

// Enter LMP3
  bis.w #GIE+CPUOFF+SCG1+SCG0, SR

// Exit LMP3 from an ISR
  . . .
  bic.w #CPUOFF+SCG1+SCG0, 0(SP) // clear bits in the SR copy on the TOS (0+SP)
  reti


```

Code 1. Assembly examples for entering and exiting LPMs.

```
// Enter LMP0
  __bis_SR_register(LMP0_bits + GIE);

// Exit LMP0 from an ISR
  __bic_SR_register_on_exit(LMP0_bits);

// Enter LMP1
  __bis_SR_register(LMP1_bits + GIE);

// Exit LMP1 from an ISR
  __bic_SR_register_on_exit(LMP1_bits);

. . .

// Enter LMP4
  __bis_SR_register(LMP4_bits + GIE);

// Exit LMP4 from an ISR
  __bic_SR_register_on_exit(LMP4_bits);

```

Code 2. C examples for entering and exiting LPMs.

```
Things to remember 3-1. MSP430 LPMs
The MSP430 family supports a number of low-power modes entered by setting
mode-control bits in the status register. Interrupt mechanism ensures
exiting the LPMs; inside ISRs the mode-control bits of the status register
copy on the stack can be cleared to ensure returning to a different
operating mode.
```

4 Estimating Operating Time

Figure 7 shows a typical current profile for MSP430F41x devices. The current drawn depends on operating mode, clock frequency, as well as power supply voltage. For example, in the active mode, the device running at 1 MHz clock frequency will draw ~300 μA if the power supply is 3 V or ~200 μA if the power supply is 2.2 V. The current drawn in LPM0 under the same conditions will be 55 μA. If we increase the clock frequency to 4 MHz, the current drawn in the active mode will be 4*300 μA or 1.2 mA. This information is typically available in device specific reference manuals. It allows us to quickly estimate the average current and operating time of our programs.

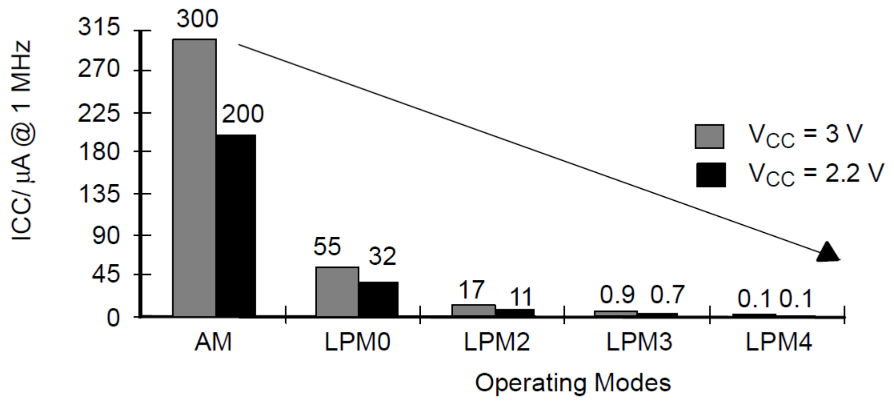


Figure 7. Typical Current Drawn by a MSP430F41x Device as a Function of Operating Mode, Power Supply per 1 MHz.

Let us consider the following example. Assume that your program reports the time with resolution of deciseconds (1/10th of a second). The main program loop is thus executed 10 times per second. The timer wakes up the CPU, a message is generated and sent over a UART link. Let us assume that the time needed for the CPU to update time, prepare message and send it over the UART link is ~2,000 processor clock cycles. Then, the main loop enters LPM0. Let us assume that we use two AA batteries to power the platform. Their capacity BC=2500 mA-hr. Based on these assumptions and information from Figure 7 we can estimate the average current drawn and operating time on a single battery charge as follows.

The first step is to determine the average current. One application period is 100 ms. Assuming the default clock cycle of ~1 MHz, one application period corresponds to ~100,000 processor clock cycles. The CPU active time is 2,000 clock cycles or 2% (2/100). The rest of the time (100,000 – 2,000 = 98,000 CPU clock cycles) or 98%, the CPU spends in LMP0. The average current is as follows:

$$I_{avg} = \frac{T_{active}}{T_{period}} I_{\text{active}} + \frac{T_{period} - T_{active}}{T_{period}} I_{\text{LPM0}} = 0.02 \cdot 300\mu A + 0.98 \cdot 55\mu A = 58.25\mu A$$

Once we know the average current drawn by our platform, we can estimate operating time. Operating time, OT, is determined by dividing the battery capacity with the average current drawn, i.e.,

$$OT = \frac{BC}{I_{avg}} = \frac{2500\ mAhr}{0.05825\ mA} \approx 42{,}918\ hr \approx 1{,}777\ days$$

Please note that the above example considers energy drawn only by the MSP430. Other chips present on a board will also contribute to the power consumed.

```
Things to remember 4-1. Estimating operating time.
The operating time of a battery-powered system can be estimated as follows:
```

$$OT = \frac{BC}{I_{avg}}$$

where BC is the battery capacity expressed in A-hr and I_{avg} is the average current drawn expressed in A.

5 Power Profiling Using EnergyTrace Technology in CCStudio

EnergyTrace technology enables analog energy measurements to determine the energy consumption of an application. This feature is integrated into CCStudio and works on MSP430 and MSP432 microcontrollers with selected debuggers. EnergyTrace tool gives you the information needed to help you achieve the lowest power design. This feedback makes it easier to implement the various techniques offered by the MSP architecture, as well as the many tools provided by TI, such as ULP advisor. More information about EnergyTrace is available on the following web site: https://www.ti.com/tool/ENERGYTRACE.

EnergyTrace implements a new method for measuring MCU current consumption. Power is traditionally measured by amplifying the signal of interest and measuring the current and voltage drop over a shunt resistor at discrete times. In debuggers that support EnergyTrace sofware, a software-controlled DC-DC converter generates the target power supply. The time density of the DC/DC converter charge pulses equals the energy consumption of the target microcontroller. A built-in calibration circuit in the debug tool defines the energy equivalent for a single charge pulse. Because the width of each charge pulse remains constant, the debugger can just count each charge pulse and then sum them over time to calculate an average current which leads to very accurate measurements. Using this approach, even the shortest device activity that consumes energy contributes to the overall recorded energy.

The TI MSP-FET (Flash Emulation Tool) supports EnergyTrace and can be used with the EXP430F5529LP Launchpad board. For this, we will need to bypass the built-in ez-FET lite emulator on the Launchpad by removing all jumpers between the emulator and the rest of the board. The MSP430 FET should be connected to the Launchpad as follows (see Figure 8):

- TDI/TDO (pin 1 of FET) goes to SBWTDIO of Launchpad;
- VCC_TOOL (pin 2 of FET) goes to 3V3 of Luanchpad;
- TCK (pin 7 of FET) goes to SWBWTCK of Launcpad;
- GND (pin 9 of FET) goes to GND of Launchapd.

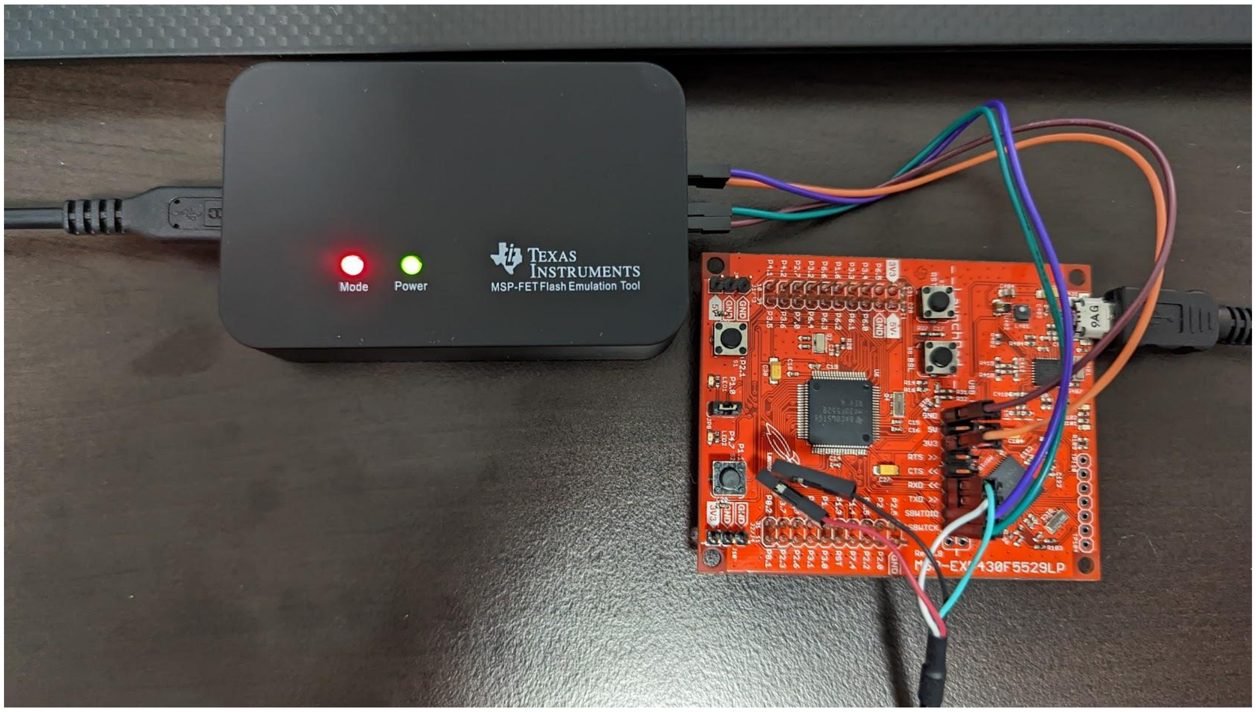
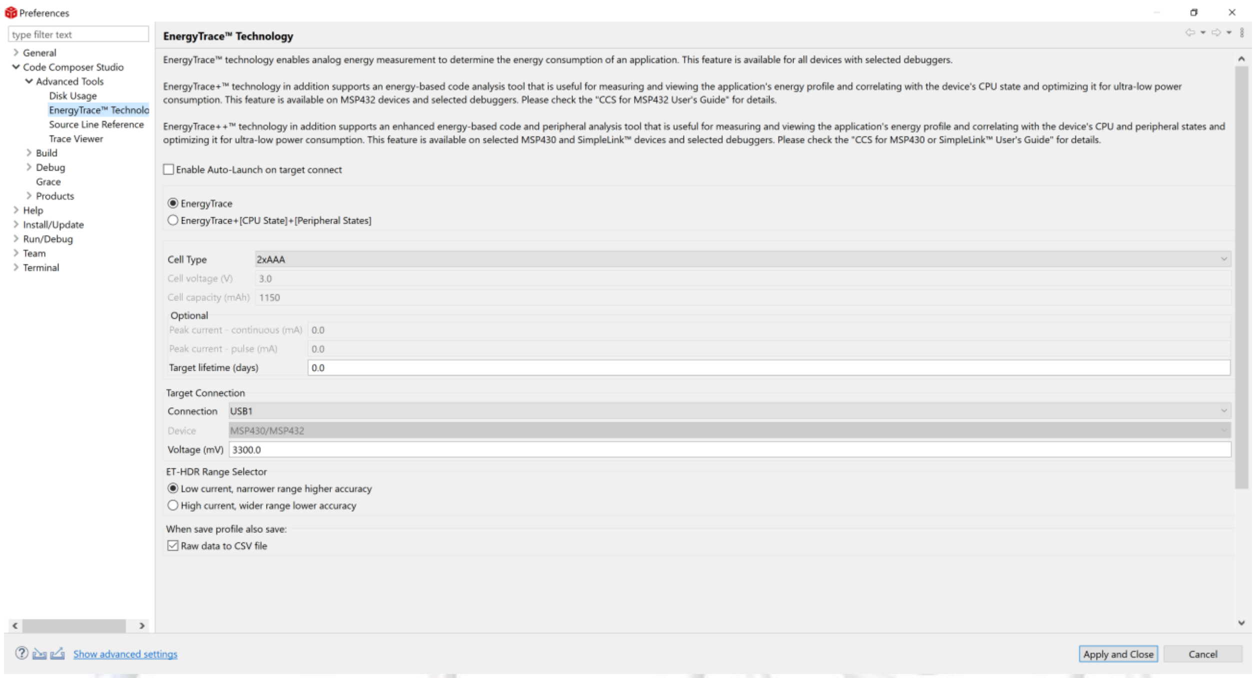


Figure 8. MSP430-FET connected to Launchpad.

Figure 8 also shows a USB to TTL Serial cable from Adafruit and its connection to the TxD and RxD pins of the Launchpad. This connection is needed only if your application communicates with the workstation over the serial interface, which is the case in the example to follow. The cable is connected as follows:

- The green wire represents TxD of the serial cable and should be connected to the RxD pin on the Launchpad;
- The white wire represents RxD of the serial cable and should be connected to the TxD pin on the Launchpad;
- The black wire represents GND and should be connected to any ground pin on the Launchpad.

To enable the use of EnergyTrace, open the CCStudio, click on Windows→Preferences, and then select Code Composer Studio→Advanced tools→EnergyTrace Technology. A window shown in Figure 9 should open up. You can check the box in front of *Enable Auto-Launch on target connect*, if you want. You can specify the power supply voltage (3300 mV in this example).



EnergyTrace™ Technology

EnergyTrace™ technology enables analog energy measurement to determine the energy consumption of an application. This feature is available for all devices with selected debuggers.

EnergyTrace+™ technology in addition supports an energy-based code analysis tool that is useful for measuring and viewing the application's energy profile and correlating with the device's CPU state and optimizing it for ultra-low power consumption. This feature is available on MSP432 devices and selected debuggers. Please check the "CCS for MSP432 User's Guide" for details.

EnergyTrace++™ technology in addition supports an enhanced energy-based code and peripheral analysis tool that is useful for measuring and viewing the application's energy profile and correlating with the device's CPU and peripheral states and optimizing it for ultra-low power consumption. This feature is available on selected MSP430 and SimpleLink™ devices and selected debuggers. Please check the "CCS for MSP430 or SimpleLink™ User's Guide" for details.

☐ Enable Auto-Launch on target connect

[✓ EnergyTrace] EnergyTrace+[CPU State]+[Peripheral States]

Cell Type	2xAAA
Cell voltage (V)	3.0
Cell capacity (mAh)	1150
Optional	
Peak current - continuous (mA)	0.0
Peak current - pulse (mA)	0.0
Target lifetime (days)	0.0

Target Connection

Connection	USB1
Device	MSP430/MSP432
Voltage (mV)	3300.0

ET-HDR Range Selector

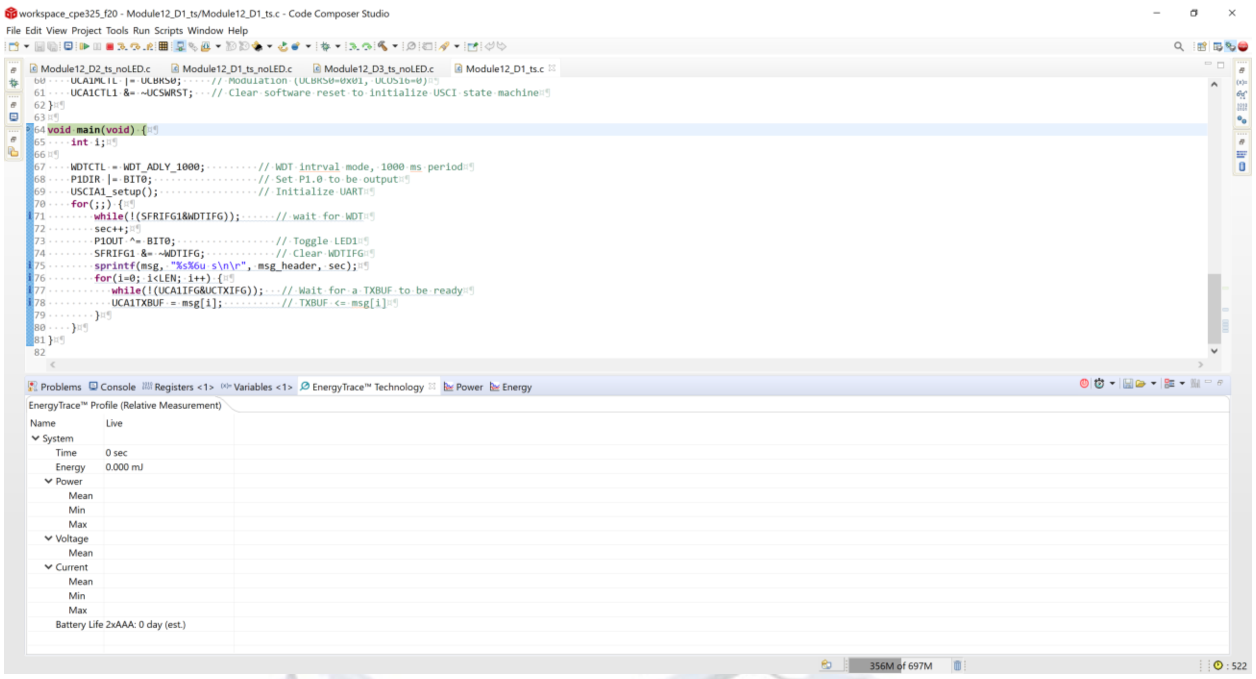
[✓ Low current, narrower range higher accuracy] High current, wider range lower accuracy

When save profile also save:

☑ Raw data to CSV file

Figure 9. Configuring EnergyTrace Technology in Code Composer.

To illustrate the use of EnergyTrace we consider three implementations of a program that sends timestamped messages to a workstation every second. The programs in Module 12, namely Module12_D1_ts.c, Module12_D2_ts.c, and Module12_D3_ts.c use polling, ISRs, and DMA, respectively. Use the project from Module12_D1_ts.c and click on Debug icon. A window like one in Figure 10 should open. In EnergyTrace Technology tab, click on stopwatch icon and set the measurement time to 1 minute. Enable Auto-launch on target connect in Preferences. Click run on the play icon to start a debug session. The measurement will take place for 60 s. Once the power tracing is completed, the total energy, power (mean, min, max), voltage, and current are reported as shown in Figure 11. Figure 12 shows the average power over the measurement period. If you recall in this program we periodically toggle LED1 (it is 1 s on and 1 s off). The current drawn by LED1 dominates the power. The power varies between two states: ~11.35 mV when LED1 is on and ~1.85 mW when LED1 is off. The average power is ~6.58 mW.



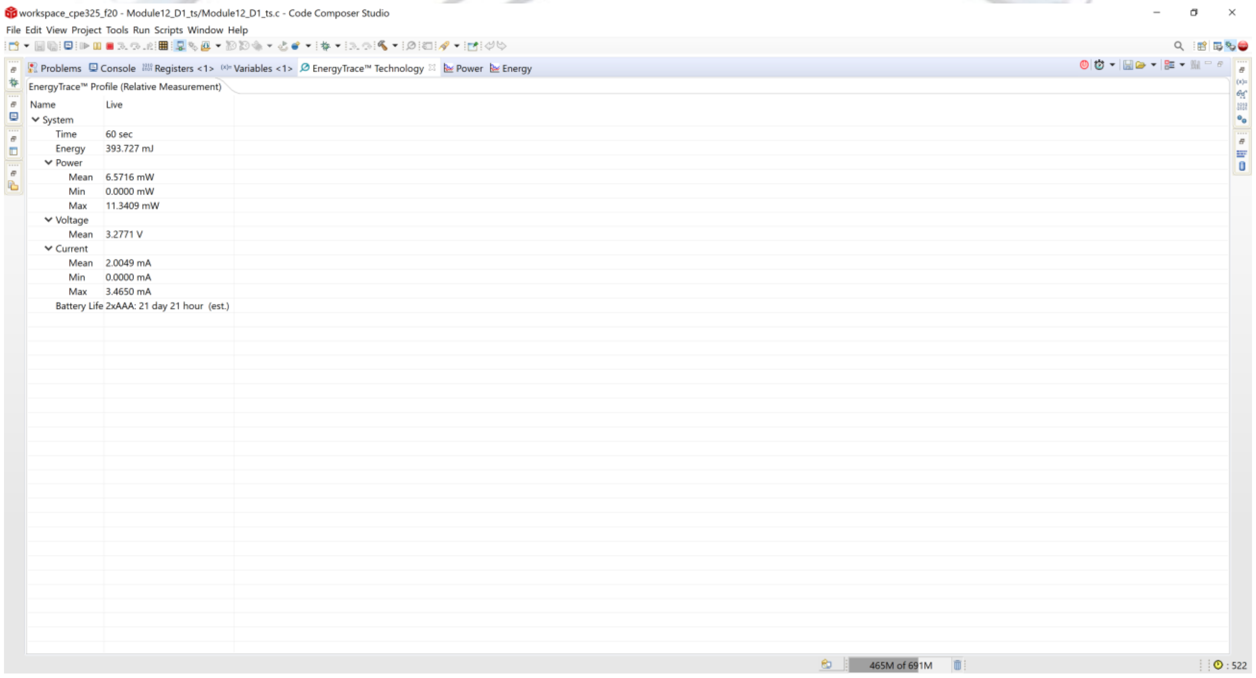
```
    UCA1MCTL |= UCBRS0;     // Modulation (UCBRS0=0x01, UCOS16=0)
    UCA1CTL1 &= ~UCSWRST;   // Clear software reset to initialize USCI state machine
}

void main(void) {
    int i;

    WDTCTL = WDT_ADLY_1000;          // WDT intrval mode, 1000 ms period
    P1DIR |= BIT0;                   // Set P1.0 to be output
    USCIA1_setup();                  // Initialize UART
    for(;;) {
        while(!(SFRIFG1&WDTIFG));      // wait for WDT
        sec++;
        P1OUT ^= BIT0;                 // Toggle LED1
        SFRIFG1 &= ~WDTIFG;            // Clear WDTIFG
        sprintf(msg, "%s%6u s\n\r", msg_header, sec);
        for(i=0; i<LEN; i++) {
           while(!(UCA1IFG&UCTXIFG));   // Wait for a TXBUF to be ready
           UCA1TXBUF = msg[i];          // TXBUF <= msg[i]
        }
    }
}
```

```
EnergyTrace™ Profile (Relative Measurement)
Name          Live
System
    Time      0 sec
    Energy    0.000 mJ
  Power
      Mean
      Min
      Max
  Voltage
      Mean
  Current
      Mean
      Min
      Max
    Battery Life 2xAAA: 0 day (est.)
```

Figure 10. EnergyTrace Technology Window in Code Composer before measurement is started.



```
EnergyTrace™ Profile (Relative Measurement)
Name          Live
System
    Time      60 sec
    Energy    393.727 mJ
  Power
      Mean    6.5716 mW
      Min     0.0000 mW
      Max     11.3409 mW
  Voltage
      Mean    3.2771 V
  Current
      Mean    2.0049 mA
      Min     0.0000 mA
      Max     3.4650 mA
    Battery Life 2xAAA: 21 day 21 hour  (est.)
```

Figure 11. EnergyTrace Technology Window in Code Compose after measurement is conducted.

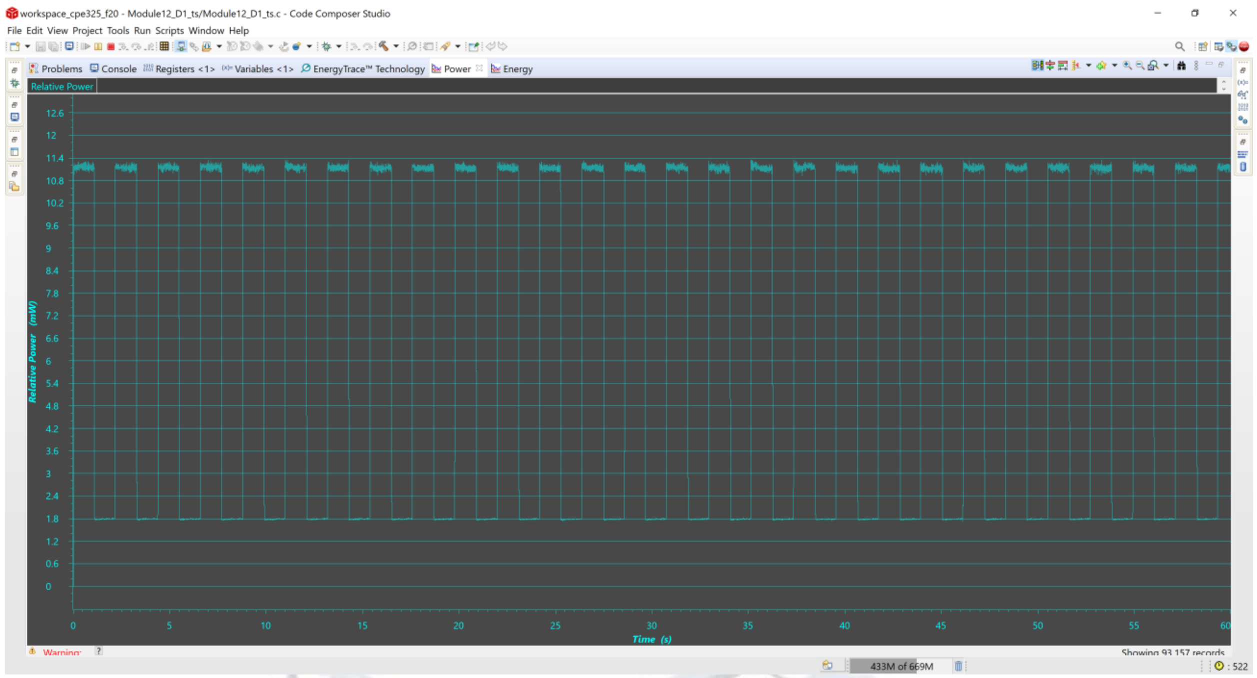
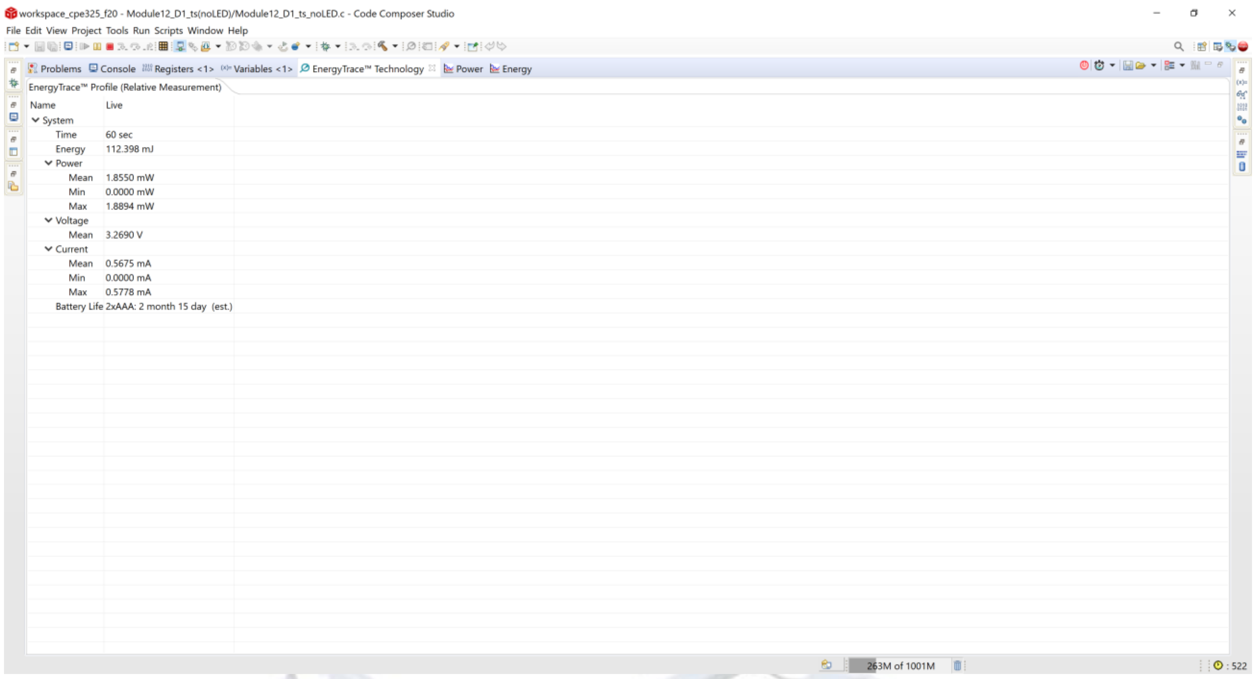


Figure 12. Power profile for 60 s of execution of Module12_D1_ts.c.

We are interested to evaluate energy-efficiency of 3 principal approaches to interfacing. To eliminate the dominating effects of the current drawn by LED1, we modify all demo programs to remove the LED1 toggling statement. The new versions of the programs are named Module12_D1_ts(noLED).c, Module12_D2_ts(noLED).c, and Module12_D3_ts(noLED).c. We repeat the process of power profiling for these three versions.

Figure 13, Figure 14, and Figure 15 show the summary and power profiles for Module12_D1_ts(noLED).c, Module12_D2_ts(noLED).c, Module12_D3_ts(noLED).c, respectively. The average power is 1.855 mW for implementation with polling, 0.8476 mW for implementation with ISRs, and 0.8509 mW for implementation with the DMA controller. We can further reduce power by experimenting with deeper low-power modes (e.g., LPM1 – LPM4) or using ACLK instead of SMCLK for TimerA or UART peripherals. Readers are encouraged to probe further and experiment with EnergyTrace and low-power optimizations.



workspace_cpe325_f20 - Module12_D1_ts(noLED)/Module12_D1_ts_noLED.c - Code Composer Studio

EnergyTrace™ Profile (Relative Measurement)

Name	Live
System	
Time	60 sec
Energy	112.398 mJ
Power	
Mean	1.8550 mW
Min	0.0000 mW
Max	1.8894 mW
Voltage	
Mean	3.2690 V
Current	
Mean	0.5675 mA
Min	0.0000 mA
Max	0.5778 mA
Battery Life	2xAAA: 2 month 15 day (est.)

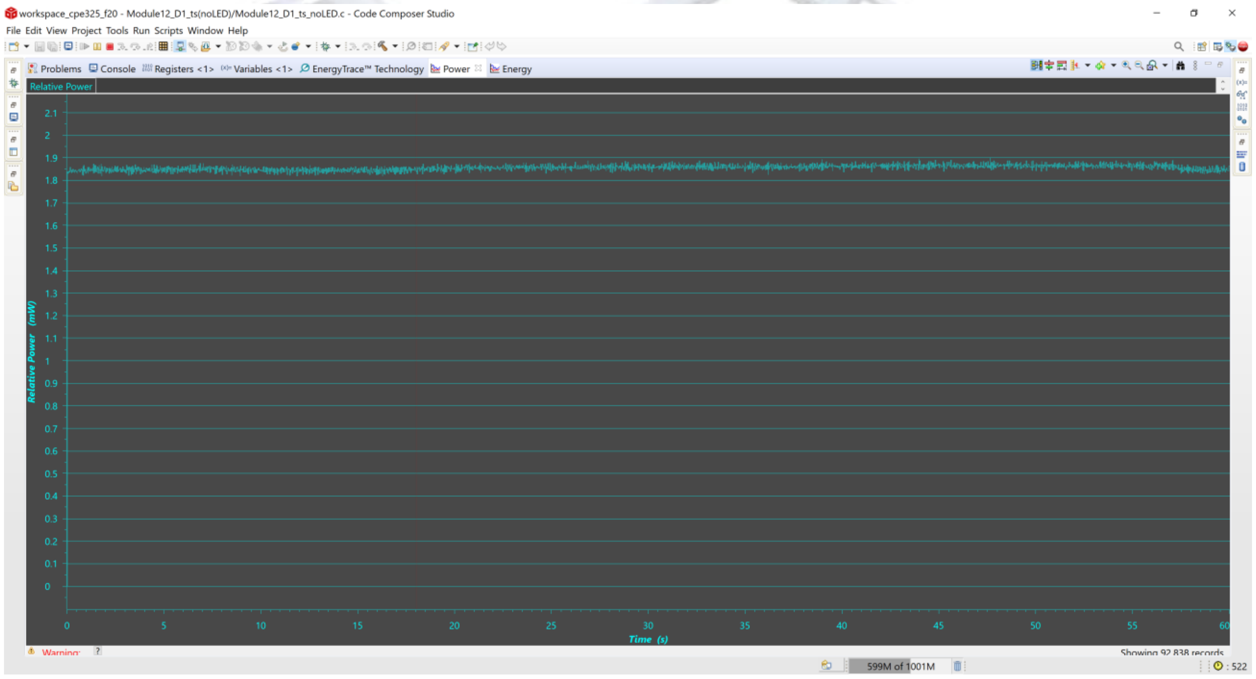
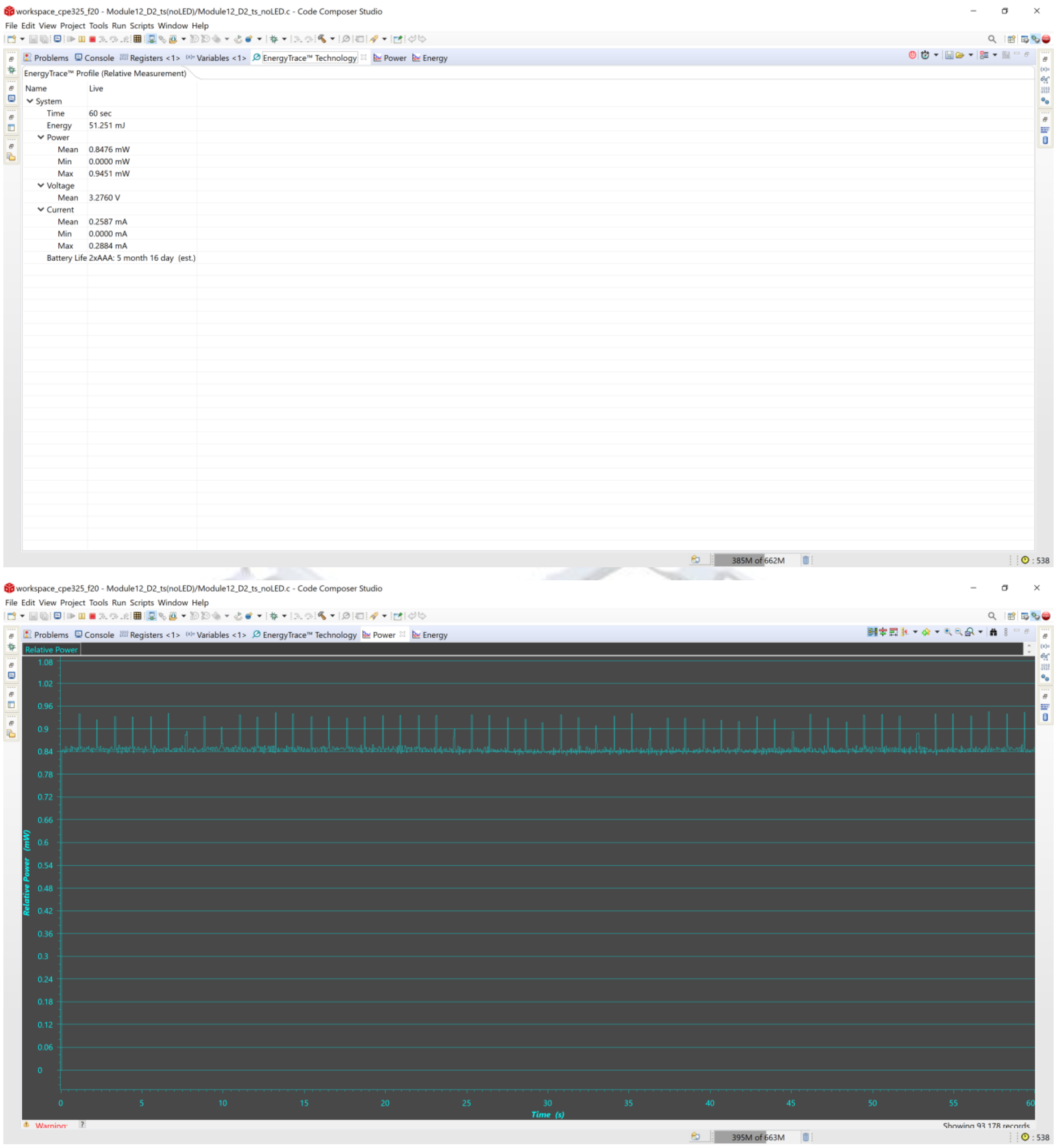


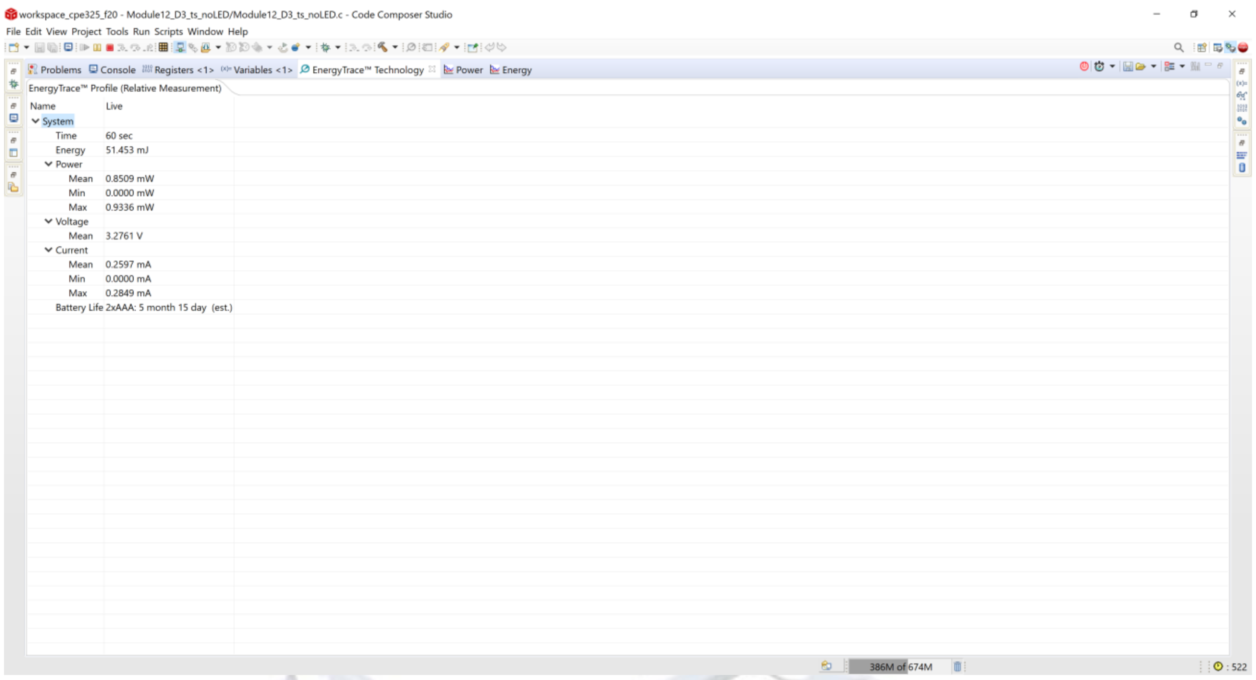
Figure 13. Power Profile of Module12_D1_ts(noLED).c.


workspace_cpe325_f20 - Module12_D2_ts(noLED)/Module12_D2_ts_noLED.c - Code Composer Studio

EnergyTrace™ Profile (Relative Measurement)

Name	Live
System	
Time	60 sec
Energy	51.251 mJ
Power	
Mean	0.8476 mW
Min	0.0000 mW
Max	0.9451 mW
Voltage	
Mean	3.2760 V
Current	
Mean	0.2587 mA
Min	0.0000 mA
Max	0.2884 mA
Battery Life 2xAAA:	5 month 16 day (est.)

Figure 14. Power Profile of Module12_D2_ts(noLED).c.


workspace_cpe325_f20 - Module12_D3_ts_noLED/Module12_D3_ts_noLED.c - Code Composer Studio

EnergyTrace™ Profile (Relative Measurement)

Name	Live
System	
Time	60 sec
Energy	51.453 mJ
Power	
Mean	0.8509 mW
Min	0.0000 mW
Max	0.9336 mW
Voltage	
Mean	3.2761 V
Current	
Mean	0.2597 mA
Min	0.0000 mA
Max	0.2849 mA
Battery Life	2xAAA: 5 month 15 day (est.)

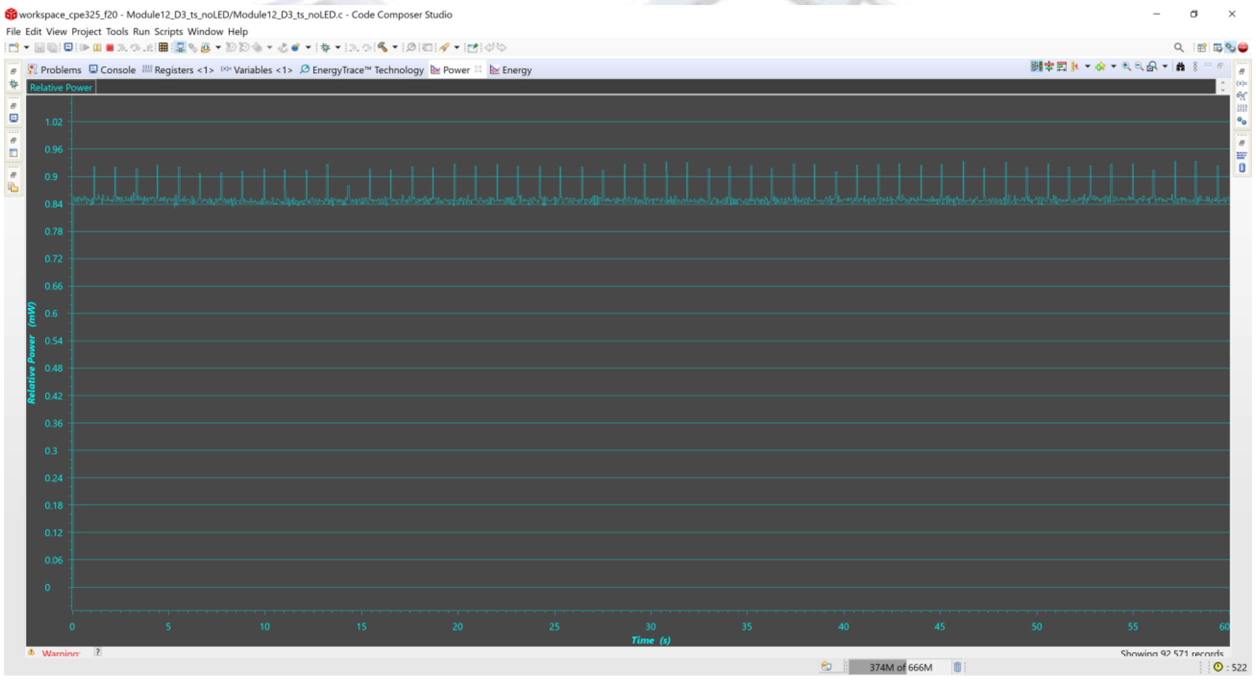


Figure 15. Power Profile of Module12_D3_ts(noLED).c.

6 Exercises

Q1.

We are considering processing requirements of an application with the main loop that repeats every 200ms. Each application cycle invokes three interrupt service routines: isrA is invoked 10 times per application cycle, isrB 20 times, and isrC 20 times. We profile the service routines and find that isrA takes 250 clock cycles, isrB takes 200 clock cycles, and isrC takes 400 clock cycles. The time spent in the main loop during one application cycle is 20,000 processor clock cycles. Answer the following questions.

Q1.A. How many clock cycles is required for the CPU to stay in the active mode during one application cycle?

Q1.B. If the processor clock is f_{MCLK}=4 MHz, what portion of the application cycle can be spent in a low-power mode?

Q1.C. Do we need to support nesting of interrupt routines for this application to function properly? Assume that interrupts isrA and isrB are generated periodically during an application cycle? Explain your answer. What is the maximum time one service routine may be pending before it gets serviced?

Q1.D. What is the operating time of the application if we know the following: the battery capacity is 660 mA-hr, current drawn by the microcontroller in active mode is 4 mA and 10 μA in a low-power mode?

Q1.E. What is the minimum processor clock cycle that we could have and still meet the timing requirements for the application?

Q2.

Let us consider a program that displays time on the LCD of the TI's Experimenter platform. The time is displayed with resolution of one tenth of a second (100 ms, deciseconds). The processor wakes up periodically, updates local time variables, performs necessary format conversions, and displays the time on an LCD in the following format (hh.mm.ss.d). These steps take N=2,000 processor clock cycles during which the processor is in the active mode; after that the processor goes back into a low-power mode. The MCLK in the active mode is 1 MHz. The platform draws 1 mA when the MSP430 is in the active mode and 0.1 mA when in the low-power mode.

Q2.A. What is the total active and sleep time during one application cycle? Calculate the average current drawn by the platform.

Q2.B. Calculate the average power P consumed (in milliWatts) if we power the platform by two AA batteries (V=3 V). Determine the system operating time in days if we know that the battery capacity is 2500 mAh.