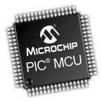


BME 580 Microcontrollers I

Lab 7: Micropower

Ross Rucho

In this lab you will learn about Micropower design techniques. You will use your temperature data logger from the previous lab, rethink the design, and incorporate features of micropower design.

You will only use the external temperature sensor for this lab, not the internal sensor.

Micropower design is essential for portable electronics, and is good practice for all electronic design. Most digital electronic devices will last a very long time if not abused. Usually the thing that finally kills an electronic gadget is (1) interface failure (switches and plugs) or (2) thermal cycling. Every time an electronic device is operated, the temperature of certain components can rise to surprisingly high levels. This results in mechanical stresses that eventually cause a failure of the component and thus the entire system. This is why most spacecraft and Mars rovers eventually fail: thermal cycling from the harsh environment of space induce failures. In outer space this is usually caused by the huge temperature variations between direct sun exposure and shade. For consumer electronics on Earth, the thermal cycling is caused by sloppy design: components cycle hot and cold simply because few designers spend time thinking about this aspect of their design. Battery size, weight, and operating time between recharges depends almost entirely upon the skillful execution of good micropower design. For portable medical devices, implantable electronics, and wearable electronics, good micropower design can be the difference between a great product and a total failure. And well-designed circuits that do not waste energy also tend to work more reliably and accurately. To truly master the art of electronic design, to transcend the mundane and achieve a Zen-like perfection of function and simplicity, your electronic designs should use precisely and only the amount of power required, and not one free electron more. Embedded systems are especially vulnerable to bad micropower design, and in general have the most opportunities for improvements in micropower design.

READING ASSIGNMENT (optional)

If you can find a copy of this book, this is probably the best single resource for an introduction to micropower design:

The Art of Electronics, 2nd Edition (Horowitz and Hill)
Chapter 14: Low Power Design

Note that you have to find a copy of the 2nd edition, because this vital information has been removed from the recently released 3rd edition of The Art of Electronics because the editors demanded to reduce the length of the book… in my opinion a shame. Since resources for this information are hard to find, I will summarize the Art of Electronics 2nd Edition chapter, as well as what I have learned over the past 3 decades, for you on the next few pages.

Guidelines for Micropower Design

Think about how energy flows through your circuit. Every time something changes or cycles, some energy is lost. In digital electronics this is usually linked to clock cycles and edges, when a logic state switches. In analog electronics this happens when energy flows back and forth across resistors. So, based upon a few simple equations the general guidelines for micropower design are:

Always use the highest possible resistor values: Energy dissipation, or Ohmic loss, across a resistor is calculated as

$P = I^2R$ (P is power, I is current, R is resistance)

Power dissipation increases as the *square* of current. So, to keep power dissipation in a resistor (or any conductor) to a minimum, you need to reduce the electrical current. For a circuit that operates at a specified voltage (E):

$E = I R$

Therefore, in order to reduce the current (I), you need to increase the resistance (R).

Turning now to capacitors, many analog circuits use RC (resistor-capacitor) pairs for timing and filters and many other functions.

The power needed to charge a capacitor (capacitance = C) over time (t) is calculated as:

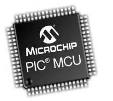
$P = C\,V(t)\,dV/dt$

For a circuit that operates over a specified voltage range V, the power required to cycle the capacitors is related proportionately to the capacitance.

So the first guideline is to use the largest practical R value and the corresponding smallest C value to achieve the desired RC constant required for your design. Since the energy from many timing circuits is dumped (lost) every cycle, the power loss of analog circuits can often be reduced by a factor of 10 or more by simply re-tuning your circuit with higher value resistors and lower value capacitors.

The above guidelines apply equally to analog and digital circuits. When you add a microcontroller into the mix, things get a lot more interesting because you can use both hardware configurations and firmware to optimize your micropower design.

Most modern digital electronics are based on CMOS technology. In this architecture the logic gates dissipate very little (practically zero) power when they are quiescent, that is, when nothing changes. But CMOS gates incorporate capacitors by definition. These circuits also have capacitances on their loads and on other locations within the integrated circuit itself. Digital integrated circuit manufacturers usually lump all of the internal capacitance values that have an impact on power dissipation into one factor usually called C_{pd} (power-dissipation capacitance). Every time these capacitors go through a charge-discharge cycle, they dissipate energy. For digital electronics that are driven by a clock frequency (f), the consequence is that energy is lost with every clock cycle. This is energy wasted in the form of



heat. The rate of energy loss (power dissipation) for a digital circuit with switching frequency (f) operating at voltage V is:

$P = V^2 f C_{pd}$ Art of Electronics, 3rd Edition, page 754

This one equation is a gem for digital designers, and it tells you a great deal about why modern digital systems are designed the way they are. From the standpoint of micropower design, here are the take-away points:

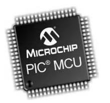
> Minimize the supply voltage. CMOS devices consume power proportionate to the square of the supply voltage. Thus, a microcontroller circuit operating at 5.2 V will consume three times as much power as an equivalent circuit operating at 3.0 V
>
> Minimize switching frequency. CMOS devices consume power proportionate to their switching frequency. If your microcontroller-based system can work at 32 kHz, you do not need to run it at 64 MHz.
>
> Reduce the values of load capacitors. The value of C_{pd} is not something you can change because it is a property of the integrated circuit itself, but you can think about which IC chip you choose to use based upon its micropower performance (low C_{pd}), and you can also minimize the external load capacitors in your circuit.

Frequency: It helps to run your microcontroller at the lowest practical clock speed, but it also helps to minimize the total number of other components operating at high frequencies.

Supply voltages can cause unexpected problems for other reasons. If your circuit has several different supply voltages you run the risk of current running through input protection diodes that act to limit input voltage to safe levels. This ends up being a 100% loss of energy, serving no useful purpose whatsoever. You would spot this using a thermal imager (see below), but careful design can prevent the problem.

Logic swings: Be sure to allow all logic swings to go all the way to the rails (+V and Ground, or equivalently, Vdd and Vss). Logic forced to operate at intermediate values tends to dissipate a lot more power than logic resting at a fully-defined state at the + or – rail.

Microcontroller and Logic Gate inputs: ALWAYS make sure there are no open inputs. Open CMOS input gates tend to drift, and as they drift they will cause input state switching (random oscillations) as well as Class-A switching currents. None of this is necessary or useful, and it can lead to unpredictable behavior of digital devices. Pull all digital inputs to either +V supply or ground. Which rail is best? That depends for each case. The logic state on an unused pin may be irrelevant, or it may actually be related to some internal or external function that draws power. So in each case you need to think it through, and the rule of thumb is to set the logic state so that the default condition of any device or function related to that pin is OFF (no power). If you do not know, then just tie the input directly to ground. If the digital input pin needs to be operable sometimes but might float at other times, then tie the pin to ground or +V using a 10K resistor (called respectively a “pull down” or “pull up” resistor). Conveniently, your microcontroller has internal pullup resistors that you can activate in firmware. So, to add pullup resistors to your circuit on unused pins, you simply add a line of code, thus no new external resistor components are necessary for many digital I/O pins on modern microcontrollers.



Avoid slow logic transitions: slow transitions on a logic input gate tends to cause a lot of class-A current during the transition, so keep the transition time to a minimum. One way to do this if you are using an analog signal to drive a digital input is to buffer the signal with a specific logic gate that has Schmidt trigger inputs to "square up" the analog signal to sharpen it and minimize noise and transition time. An example is a hex inverter with Schmidt trigger inputs: 74HC14 or CD40106. Some, but not all, of the logic input gates on modern microcontrollers also have Schmidt Trigger inputs when they are used as digital inputs. In the datasheet, the digital input may be designated as "ST", which means Schmidt Trigger.

Low current default state for loads: design your circuit so that the default (normal) state of loads such as resistors, LEDs, pull up and pull down resistors, etc., always operate in the lowest current condition in their default state.

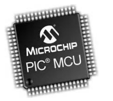
Turn OFF any unused peripherals. If you are not using a peripheral device such as a display or an external analog-to-digital converter or a sensor, turn it OFF or set it to a low power "sleep" or "disable" mode. One easy trick for this is to see whether or not some of your peripherals can be powered simply from one of your digital logic output pins. Basically if they draw less than about 15 mA and operate at the same voltage as your microcontroller, then you can probably power the peripheral directly from your microcontroller. For example, you can probably power the external temperature sensor you used in the last lab using a digital logic output pin on the microcontroller. The same is true for the OLED display, but keep in mind that once powered OFF, you need to re-initialize the OLED at power-up.

Advanced Techniques:

One option that many new microcontrollers have is firmware-controllable clock speed. The microcontroller we use in this class has that capability. Changing clock speed may introduce other problems that you would have to work out, but if an issue like maximum battery life is the most important design consideration, then you can consider changing the microcontroller clock speed during program execution. The idea is simple: crank up the clock speed only when you need it, such as during computationally-intensive bursts of activity. When the microcontroller is just standing by waiting for a signal or for time to go by, you can turn down the clock speed, but be sure to also adjust your timers and counters to compensate for the change in clock speed. This is analogous to an animal in hibernation: when food energy is scarce, some animals can dial down their metabolic clock.

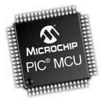
Some microcontrollers have a *sleep mode*. The microcontroller we use in this class has a sleep mode. You can use sleep mode, but be careful to make sure your device will wake up when it needs to. Some spacecraft have been lost in deep space because they simply did not wake up from sleep mode. I personally tend to avoid this mode and prefer to use the other approaches outlined above, but it is something you should consider when power use is a key design consideration.

Thermal imaging: Once you own a good multimeter and oscilloscope, one of the best advanced debugging tools you can invest in is a thermal imager. Until recently these were quite expensive. But recently the price has dropped and the performance has improved markedly. A thermal imager can be used to debug faulty and malfunctioning circuits, as well as to aid in micropower design. If you intend to make a serious commitment to electronic design in your career, consider buying a thermal imager. You might want to borrow a few different ones first to see if one is best suited to your needs.



You can use a thermal imager to detect hot spots in circuits. Simply run the circuit for a while and take a thermal image. Look for the hot spots, usually these will be individual components or traces on the PCB which get hot during operation. Once you have identified the components that are running hot, try re-tuning the circuit, change resistors and capacitors and use the other guidelines above. In some cases you will discover that certain components just tend to run hot. This is especially true for many voltage regulators. Some run cool and efficiently, and many run hot and are therefore inefficient. In my experience I find that the cheap old-fashioned linear voltage regulators with a heat sink tab on them tend to generate a lot of heat, but the modern surface mount regulators, especially "LDO" (low drop out) types tend to dissipate much less heat. In some cases you can detect this problem with the old-skool thermal imager: your index finger. In the lab you may sometimes notice that I go through your circuits by touching components one at a time… this is to see if they are running hot. Be careful, you can get a nasty burn. If you use a thermal imager you will discover these hot spots immediately, and it may prompt you to make a change in your design to use an entirely different component. The sooner you detect these problems the better. If you check for hot spots regularly, and you correct them through better design and better components, you will build up a list of efficient components that run cool and you will naturally begin to design more efficient circuits. Your circuits will use less battery power, require less additional junk like aluminum heat sinks and fans, they will last longer and run more accurately and reliably. This is the difference between a masterful circuit designer and, well, everybody else.

Finally, most integrated components such as voltage regulators and boost converters require the use of external components, usually resistors and capacitors, but also sometimes inductors and diodes. The manufacturer data sheet will usually recommend a range of optimal component values, but I find that from the standpoint of micropower design it is best to try different components and different values to get the best performance. This is something you can do by trial-and-error, changing one component at a time and measuring the results using a thermal imager and an oscilloscope. For example, for DC-to-DC boost converters, one thing that makes a huge difference is the switching speed of the diode. Slow diodes leak a lot of current before they shut off the reverse current flow. Changing the values of resistors, capacitors, diodes and inductors can have a huge impact on the tuning and performance of many integrated circuits. If a component is causing leakage problems, you can usually see this using a thermal imager. Slow diodes will get hotter than fast switching diodes in a DC-to-DC converter, for example. When these circuits ring or oscillate because of bad circuit tuning they are dissipating energy as heat while degrading overall performance. An oscilloscope is the tool you need to detect unwanted ringing or oscillation or instability. One thing you will see all the time is that when a circuit is carefully designed and tuned for micropower operation it will almost always just work better than a circuit that wastes energy. In the case of electronic circuits, efficiency and performance go hand-in-hand. Unless you are building a resistive heater, there is almost no advantage to a sloppy, inefficient, hot-running circuit.



BME 580 Microcontrollers I

LABORATORY EXERCISES:

1- Using a multimeter, measure the current that your temperature data logger circuit draws as it was designed for the previous lab. You can do this by:

- Set your multimeter to measure amps (or probably milliamps)
- You usually need to move the red ammeter wire to a different socket for current (Amp) measurements
- Disconnect the positive power supply wire to your circuit, then reconnect the power through the ammeter. This means that your power to the circuit now flows through the ammeter first, then into your circuit. This allows you to directly measure the current drawn by your circuit.
- List the measured operating voltage (your supply voltage) probably about 5V and current:

50 mA at 5V

2- Use the datasheets for each component on your circuit to estimate the power budget. In some cases, such as the potentiometer, you can simply calculate how much power it dissipates using Ohm's Law and the fact that power = I^2 R. Another concern is that some of the components draw different amounts of current depending upon whether they are active or standing by. One example is the SDCard module, which draws much more power when it is writing data to the card compared to when it is just in stand-by mode. You should check to see if other components draw more current in different states. If so, try to reflect that in the table below.

3- Fill in the table (based on data sheet values and theory):

COMPONENT	STATE	VOLTAGE (V)	CURRENT (I)	POWER (= V * I)
Microcontroller		5.0 V	1.4 mA	7 mW
OLED	active	5.0 V	10 mA	50 mW
OLED	stand-by	5.0 V	<1 mA	< 5 mW
SDCard Module	active	5.0 V	6 mA	30 mW
SDCard Module	stand-by	5.0 V	2 mA	10 mW
LED		5.0 V	25 mA	125 mW
POTENTIOMETER		5.0 V	1 mA	5 mW
Others (passive components)		5.0 V	<1 mA	< 5 mW

4- Now, take real measurements of the current used by each peripheral device using your ammeter: the OLED and the SD Card module. This is one reason why we used the green spring terminals to build the circuit: you can easily pull out the positive power wire and splice in your ammeter to measure current. If you used a pin header or soldered the SDCard module or OLED into the circuit, you can try splicing a wire in at the power supply to the module and running power through the ammeter. You will need to write a bit of code so that you can take measurements of the current when active versus when standing by. For the OLED, active = you are constantly writing new text to the display. For the SDCard module, active = constant data streaming at a high rate to the SDCard module (it is internally writing to FLASH memory)

Fill in the table using MEASURED values:

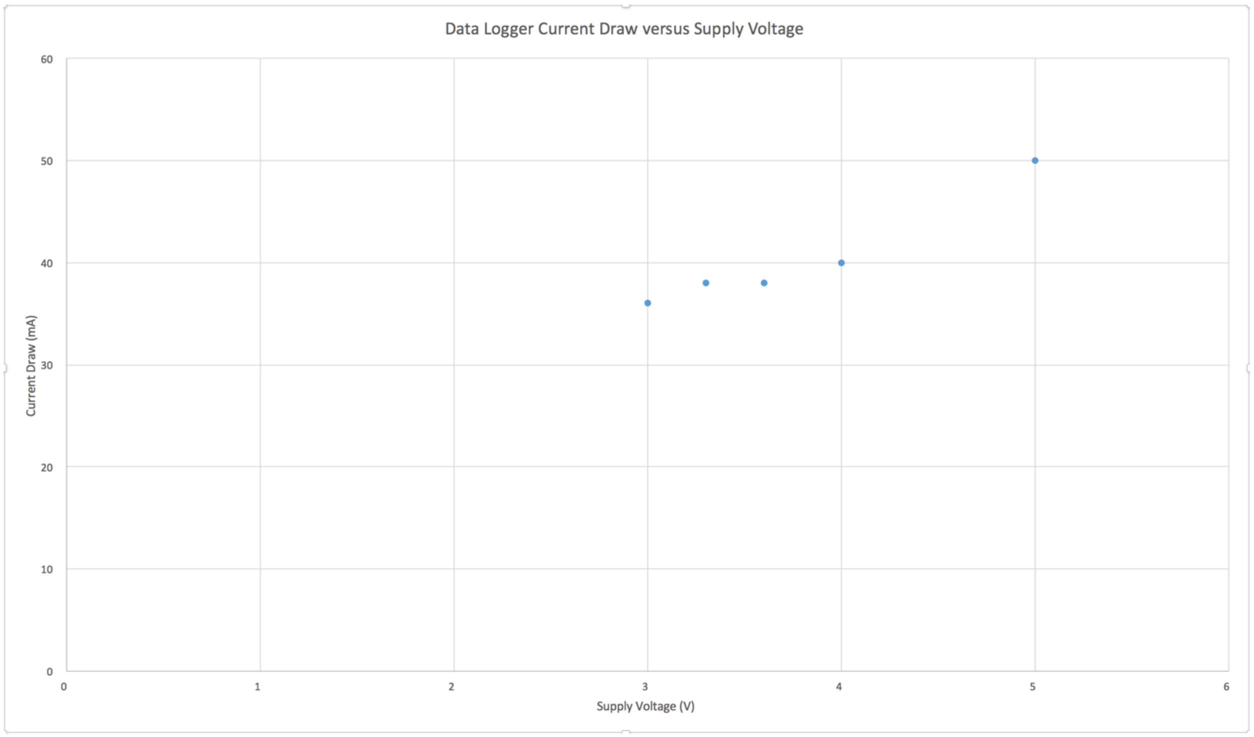
COMPONENT	STATE	VOLTAGE (V)	CURRENT (I)	POWER (= V * I)
OLED	active	5.0 V	7.6 mA	38 mW
OLED	stand-by	5.0 V	3.1 mA	15.5 mW
SDCard Module	active	5.0 V	3.9 mA	19.5 mW
SDCard Module	stand-by	5.0 V	2.2 mA	11 mW

Compare the theoretical or data sheet values (table under #3 above) with the measured values (table directly above). You might find that they are pretty close to the same. But sometimes with some components the values can differ by a factor of 10 or more

You should keep in mind that these measurements will change depending upon the supply voltage (most devices draw far less current at 3V than at 5V, but some devices can only operate at a narrow range of voltages, so be careful). Other factors, such as clock speed can also influence current draw. This is certainly true for microcontrollers, but it is also true for some peripherals. For example, you can command the OLED display to refresh the display more or less frequently, which would have an impact on current consumption.

Try running your data logger at the voltages listed below and record the current draw for the total circuit:

VOLTAGE (V)	CURRENT (I)	POWER (= V * I)
5.0	50 mA	250 mW
4.0	40 mA	160 mW
3.6	38 mA	136.8 mW
3.3	38 mA	125.4 mW
3.0	36 mA	108 mW



When you have all of this data then you can make an intelligent decision about many aspects of your design, for example:

-- Which components need to be changed or removed to get a reasonable power budget?

-- Can changes in the circuit layout improve the energy efficiency?

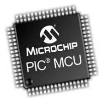
-- Can changes in the firmware improve energy efficiency?

-- All things considered, what is the optimal operating voltage?

-- What battery capacity is needed (mAh, voltage)?

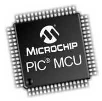
-- How long will the system be able to run between recharges or battery changes?

Then, when you have done this you should spend some time really thinking about how to modify your data logger design to get the best energy utilization. With some careful thought you can easily get good performance with a 90% reduction in power. With very careful analysis you can probably reduce the total power consumption by 99%. The interesting thing is that once you have done this design work carefully, it is usually *less expensive to build a better circuit*. The improved performance comes only at the cost of better design, not necessarily better components or processes. Although there is a cost associated with doing the engineering work, it may only take a few days of careful work to improve your design by a factor of 10 or more. So, the cost of

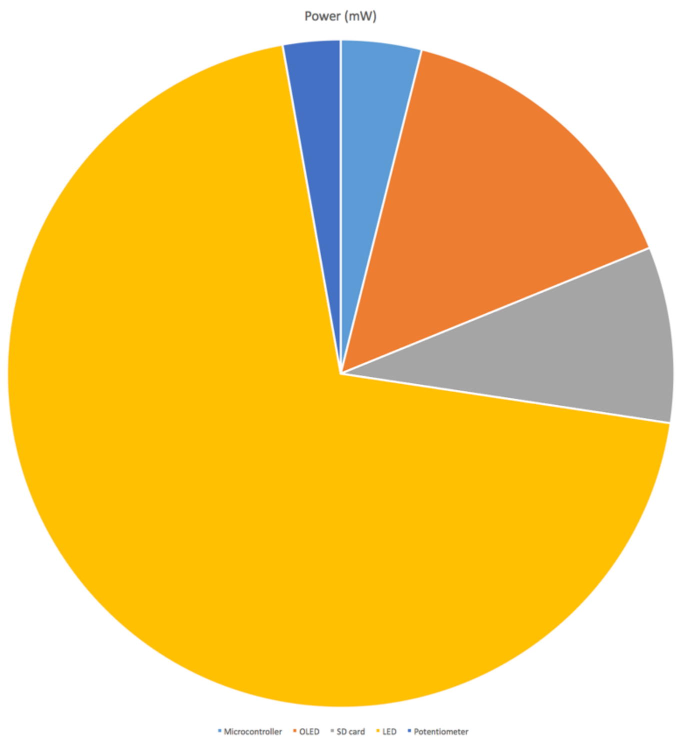


a few days of solid engineering work can be justified when your ultimate product is many times better than competing products for something as simple as battery life.

This kind of engineering skill is crucial for designing high-quality and high-value electronic products including medical devices. So please take the time this week and next to think through it carefully.



5- Insert an Excel pie chart showing how much power is consumed by each component:

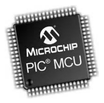


6- What component consumes the most power, and can that component be changed or eliminated, or can you change how you use that component to minimize power usage? There is a lot of room for creative design here.

The LED consumes the most power and this component can be removed from the data logger entirely because there is a small flashing blue LED on the SD card module that can signal to us that our data logger is functioning.

7- For the above component, how does the total theoretical (estimated) power consumption compare with the measured power consumption using your multimeter?   *The theoretical power consumption of the LED is approximately 125 mW while the actual power consumption of the component is 39 mW. In our firmware, the LED is only being operated for about 1/4 to 1/3 of the total time of the while loop which greatly decreases its power consumption with respect to the theoretical value.*

8- Review the micropower design guidelines at the beginning of this lab template to decide which approach you might be able to use to get the best reduction in power for your circuit. You can use more than one approach if you want to. Then try these approaches and measure the change in current that your data logger draws.



I modified the firmware to implement all of the changes specified in the micropower design guidelines for firmware. I implemented Schmidt Triggers on all of the digital I/O pins and activated the internal pull-up resistors to ensure that unused pins didn't float. Further, I removed all of the firmware pertaining to unnecessary peripherals, and slowed the clock rate to 32 kHz when the faster clock rate was not required. Finally, I lowered the supply voltage to 3.3 V.

Overall, the data logger current draw decreased from 50 mA to 28 mA.

9- Overall, how much power were you able to save by making these changes (in terms of % power reduction)? *In total, I was able to achieve a 63% reduction in power.*

$$\left(\frac{250mW - 92.4mW}{250mW} * 100 = 63\%\right).$$