

10 tips for maximizing battery life

Jacob Beningo - May 7, 2013


Portable, battery powered devices are sweeping through society like wild fire. Mobile computing and sensor devices are springing up everywhere providing engineers with not only a plethora of data but also applications. Requirements often dictate constraints on size and weight that limit how much capacity the battery can carry. The number of features on devices in addition to the time between charges makes it very challenging to near impossible to meet the requirements. Selecting a low power microcontroller is an obvious first step but there is a number of software and hardware tips that can be followed to ensure that every last milli-Amp-hour of charge is put to good use.

Tip #1 - Create a battery budget

Early in the design cycle it is highly recommended that a battery budget be put together. Current requirements for each device on the board can be tallied together to get a rough idea of how much battery current is going to be needed and whether the selected battery is up to the job. Device datasheets have gotten pretty good at providing minimum, typical and maximum current data. Taking a very conservative approach, a battery budget could be based solely on the maximum current values for the devices; However, an excel worksheet is easy to duplicate and creating a budget for both typical and maximum will give a good ball park range. If more battery is needed than is available, please don't just move forward on the project! Make the necessary changes up front to spare weeks or months of heartache down the road! Figure 1 shows an example battery budget template that can be downloaded from the authors' website under Design Articles\Design Cycle.

Battery Budget Worksheet (Typical)

Device - Example Device



Battery Capacity (mAh)

Desired Battery Life (hours)

1200

24

Calculated Values

Component	On (mA)	Idle (mA)	Standby (mA)	On Duty Cycle (% per hour)	Idle Duty Cycle (% per hour)	Standby Duty Cycle (% per hour)	mAh	Comments
Microcontroller	24	5.6	0.1	20	80	0	9.28	Wake-up every 10ms to handle maintenance tasks
EEPROM(Write)	7	0.001	0.001	5	0	95	0.35	Assume writing 100 data points per hour
Radio Receive	28	1	1	40	0	60	11.80	On once every minute for 100 ms.
Radio Transmit	33	1	1	40	0	60	13.80	Transmit data once per hour
Sensor	27	0	0	50	0	50	13.50	Always on. Current from datasheet ~100mW
Accelerometer	0.2	0.2	0.01	0	0	100	0.01	Always on
Add Next Device	0	0	0	0	0	100	0.00	Enter Comment

Results

Average mA/h	48.74
Estimated Battery Life (hours)	24.62
Desired Life Remaining (hours)	0.62

Figure 1 – An example battery budget

Tip #2 – Set unused MCU I/O to lowest power state

It is easy to overlook what should be done with an input/output pin that is not being used. This oversight however can be the difference between having a marketable product and an expensive paper weight. Each microcontroller has different recommendations on what to do with unused pins and close examination of the datasheet will reveal what should be done. For example, an unnamed silicon vendor datasheet recommends that any unused I/O be set as an output and driven low. The purpose of this is to minimize leakage and quiescent currents in an effort to minimize power usage. While these currents are tiny, each unused pin

adds to this loss and over a period of a day can be a substantial amount of battery life.

Tip #3 – Turn-off unused MCU peripherals

Just like in any home, if you aren't in a room then the light should be turned off to conserve energy. It is the same thing with a microcontroller. If there is an unused peripheral like an analog-to-digital converter or a pulse-width-modulator, turn it off in order to save power! Peripherals can be quite a power hog! For fun pick out a favorite microcontroller and scroll through the datasheets power section and see how much current is being drawn by each peripheral. Some providers don't include this information and it is up to the engineer to setup some hardware on the bench and then using test software turn peripherals on and off one at a time to get an understanding of the current draw. Analog-to-digital converters and USB peripherals tend to be near the top of the biggest users list.

Title-1

Tip #4 – Turn off unused MCU clocks

Now that all the unused peripherals have been turned off, there is not much point in running a clock signal to them. Running clock signals to different peripherals within a microcontroller requires the use of energy! There are internal clock gates that need to be powered up in order to propagate the clock. These gates use voltage and a small amount of current. To help minimize the power profile of the MCU turn off any unused clocks. It may only be a small amount, but once again the laws of addition can be staggering!

Tip #5 – Use power savings modes

Every modern microcontroller has some type of power savings mode. The idea behind this is that the processor and peripherals can be put into a near shutdown or stopped state that minimizes power usage but still allows them to return to normal operation very quickly. Most microcontrollers will have at least three power modes but more sophisticated processors can have in excess of seven! The common modes are run, idle and standby. Examining a datasheet from one particular vendor revealed that run mode consisted of a current draw of 24 mA, idle mode 5.6 mA and standby 0.1 mA! What a difference! Proper use of the power savings mode can account for a very big increase in battery life.

Tip #6 – Throttle the system clock

The clock frequency that the MCU runs at is one of the areas that have the potential to squeeze a lot of extra operating time out of the battery. There is a direct linear relationship between the frequency of the CPU clock and the amount of current that is drawn to operate the microprocessor. Take a look at Figure 2; the higher the frequency, the higher the current draw. Throttling the microcontroller clock up and down is a great way to save power. When a math intensive or fast operation needs to be performed, speed up the clock. When the task is done and the system operating at a lower frequency, clock it down. Throttling the system clock has the potential to add hours of operating time to the battery life. Please be aware that this can be a complicated endeavor. Any peripheral that is dependent on the clock may also have to have its clock dividers updated in order to maintain the same rate of operation.

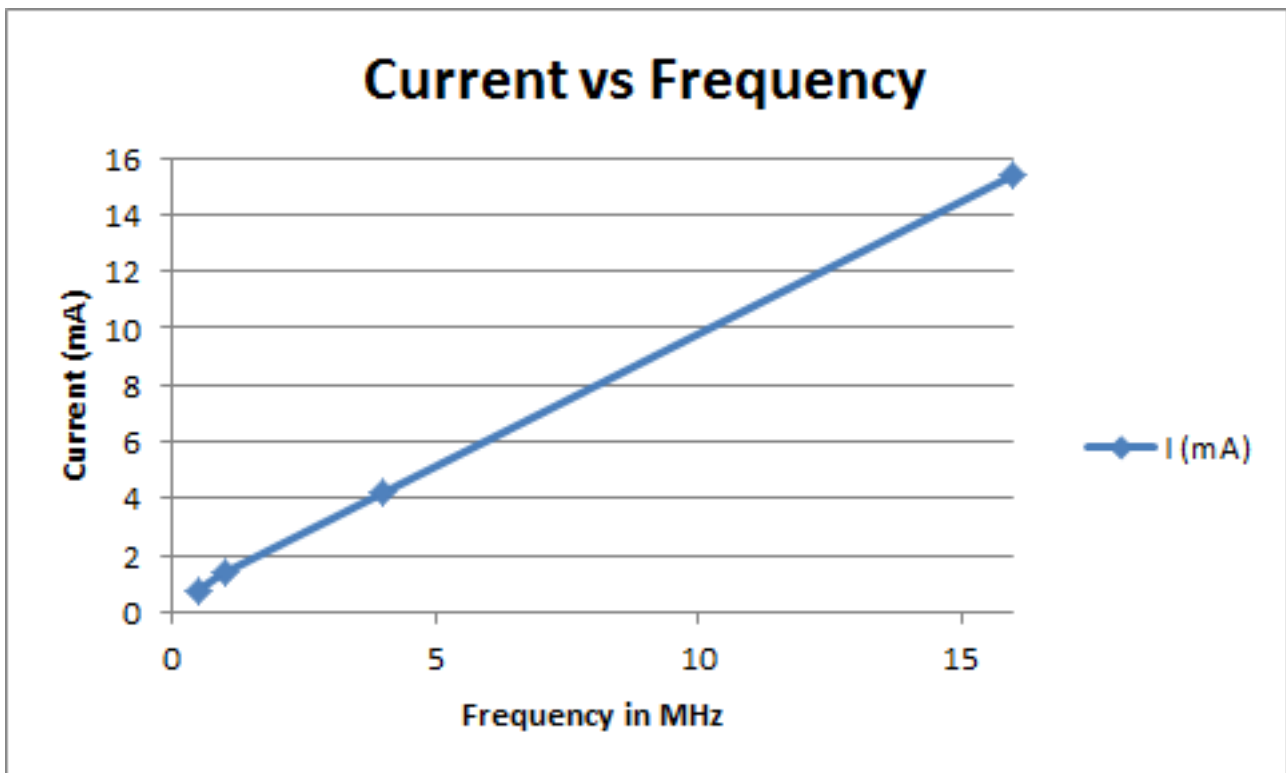


Figure 2 – MCU Current Draw as a Function of Clock Frequency

Title-1

Tip #7 – Use efficient algorithms

The idea of using efficient algorithms is to get at the fact that the more time that is spent in a low power mode and a throttled back frequency, the longer the battery is going to last. Using algorithms that are fast and efficient will result in the system spending more time in power savings modes. Power savings modes use only a fraction of the current that they use when in full out tilt mode. Try to design the software and the system to do what needs to be done and then get into a low power mode. The result will hopefully not only be longer battery life but hopefully even a smaller, lighter cheaper battery!

Tip #8 – Watch for devices with high leakage current

When circuits are being designed, make sure that the leakage and quiescent currents are well understood. If necessary, prototype out the circuit and verify what the current draw of the circuit is. Things to watch for are devices with high standby currents and low valued pull-ups or pull-downs. Make sure that this information gets put into the battery budget!

Tip #9 – Select External Devices that can be turned off

During the hardware design when components are being selected, it can be extremely useful to select sensors and external components that have low power modes themselves or that can be switched off. External parts like EEPROMs, flash and sensors usually support low power modes. When they don't, there are a couple of methods that can be used to disable them. One, is to design in a switch like a FET to turn power on and off for the device. One issue with this is that the engineer can't forget that there is a diode drop of at least 0.3 volts and up to 0.7 volts which can affect the operation of the device. The second option is to use a regulator that includes an enable/disable pin.

Tip # 10 – Add a voltage and current monitor circuit to the device

Engineers rely on data in order to make design decisions. In many cases battery life optimizations are the last thing done on a project. All other features are implemented first and then before the product rolls off the production line the team scrambles to improve battery life. One of the best ways to understand the battery performance of the system is to include two simple circuits to monitor battery voltage and current. This information can then be logged and used to determine discharge/charge cycles, determine steady state currents and really understand how the system is operating from a power usage standpoint. Figure 3 and Figure 4 show an example battery voltage and current over the course of a single discharge cycle. Armed with this tool, the savings of each tip can be determined as it is implemented!

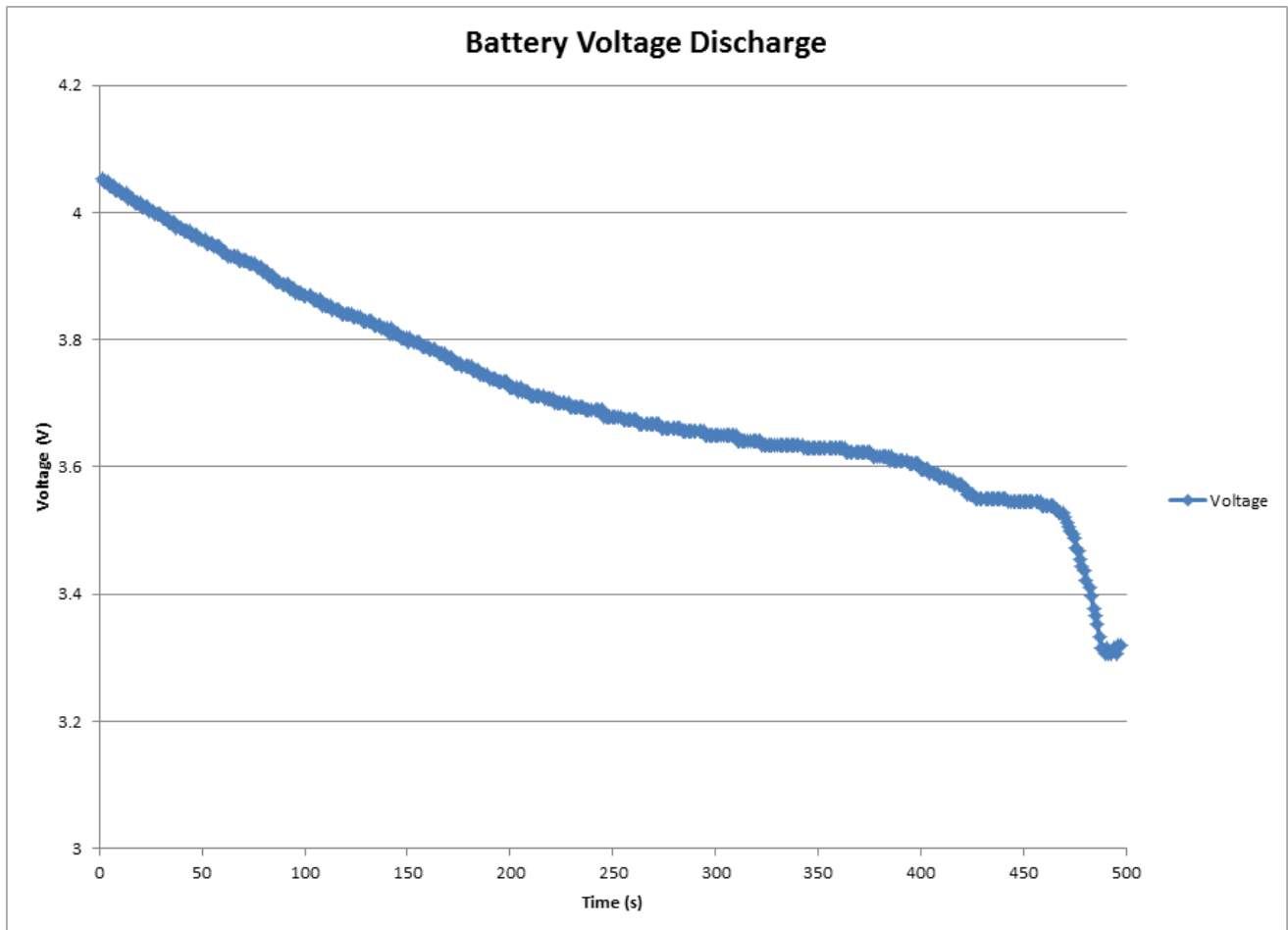


Figure 3 – Battery Voltage Discharge

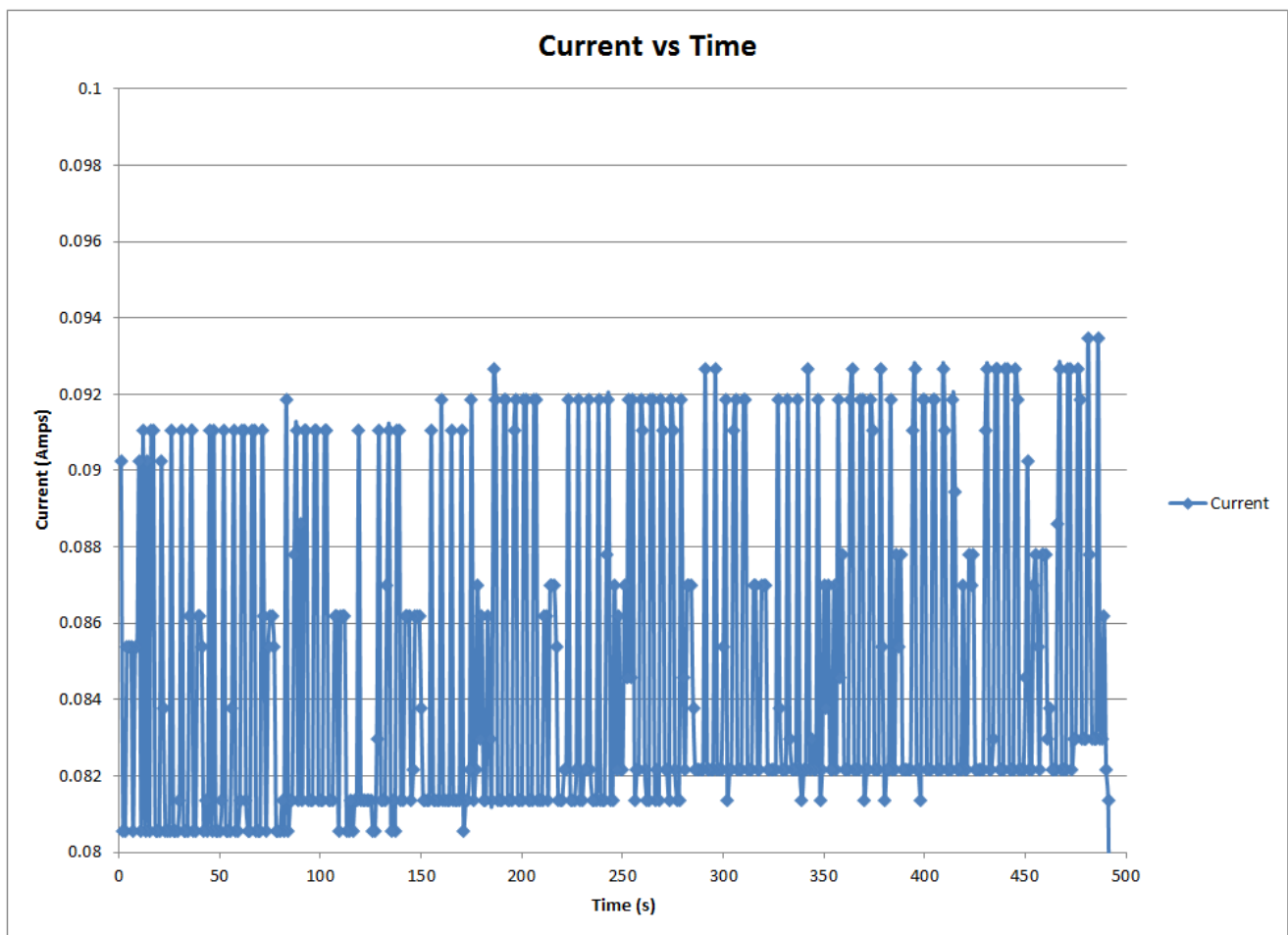


Figure 4 – Battery Current Use

Using these tips, it is possible to tame even the unwieldy of power hogging devices. It is important to keep in mind though that while these tips will decrease system power and improve operating efficiency, these tips need to be kept in mind during the design cycle and shouldn't be thought of as a last ditch effort to get a product out the door. With a little luck, that new sensor design will have enough battery life to last along time. Happy power savings!

Jacob Beningo is a lecturer and consultant on embedded system design. He works with companies to develop quality and robust products and overcome their embedded design challenges. Feel free to contact him at jacob@beningo.com, his website www.beningo.com or on twitter @Jacob_Beningo.