

## Utilizing Microcontrollers' Low-Power Modes and Peripherals

Modern microcontrollers (MCUs) have a large variety of peripherals and features that can do a lot to help a design's power budget. It is important for an embedded designer to have complete knowledge of a device so that he can fully utilize the modes and peripherals available to reduce the power consumption of a design.

### Making the Most of Power Saving Modes and Features

Most microcontrollers have an abundance of modes that offer various power saving options. Though their names may vary, the typical modes available on nearly all MCUs are Run, Idle and Sleep. Figure 1 shows a summary of the modes available on many microcontrollers. In some cases, it is obvious when to use each mode; you have to use Run mode for the main processing work. Sleep mode is the obvious choice for the long periods of waiting for an external event, after processing completes. Idle mode is somewhat less clear cut, but is most frequently used when transmitting or receiving large blocks of data. It can also be used when waiting for short time delays to shut down the CPU while preserving other functions.

Operating Mode	Function	Typical Current	Typical Usage
Run	All Clocks Active	Frequency Dependent	Normal Operation
Doze	- Slower CPU Clock - Full-Speed Peripheral Clock	35-75% Active Current	Applications Using High-Speed Peripherals Requiring Little Processing
Idle	- No CPU Clock - Full-Speed Peripheral Clock	25% Active Current	App is Waiting for an External Event From Peripheral
Sleep	No CPU or Peripheral Clocks	50-100 nA	Main Low-Power Mode, Used in Most Applications

	32 kHz Crystal Enabled	
Deep Sleep	No CPU or Peripheral Clock 32 kHz Crystal Enabled	< 50 nA Apps With Low Run:Sleep Time Ratios or With Long Sleep times

Figure 1: Each mode is implemented to allow for a particular set of applications to reduce power consumption. Compare your application to what is available on your MCU, to make sure you fully utilize all available low-power modes.

Familiarity becomes important when considering the numerous other modes and features that are architecture specific. While a design can get to acceptable power levels using only the familiar modes above, to reach the absolute minimum current consumption in an application it is best to take advantage of the unique features on a device. One example is the new class of power-down modes on some MCUs that remove power from the core of an MCU in order to minimize power consumption, reducing it below the normal power-down mode.

Removing power from the core of the MCU has the cost of removing power from RAM as well, resulting in loss of information that would be retained in other low-power modes, such as Sleep. When would this high cost make using a Deep Sleep mode worthwhile? The benefit comes from the very low leakage current that is caused by powering down most of the device. This can result in substantially lower power-down currents—lower than 50 nA in some cases, as shown in Figure 1. Low leakage also makes Deep Sleep modes perform better in applications with high temperatures or voltages that would result in high Sleep-mode currents. The other major benefit of Deep Sleep modes is that they allow chip designs to move to smaller geometries with better performance, without giving up low power consumption. The best way to use such a mode is in applications with long power-down times, where the cost of re-initializing the application is far outweighed by the reduction in power-down current.

Another reason that device familiarity becomes important is that it is not only the low-power modes that can reduce power consumption. Many features that improve performance provide power benefits as well. For example, if the device has an internal oscillator, you can use it while your main crystal is starting to run initialization code. This reduces the total time the device is spending awake.

### Digital Peripheral Power Consumption

Integrated peripherals can help to significantly improve the performance of MCUs and allow the removal of external components, which both help to reduce power. However, if not used properly, the cost of running the peripheral can exceed the power savings. With a few simple techniques, the power cost of the peripherals themselves can be minimized to maintain a low-power application.

Generally, the most power hungry peripherals used in microcontrollers are the serial communication buses. I2C™ and SPI communications both use multiple high-speed

lines. The power cost of driving these lines is significant. SPI can consume many milliamps of current when run at high speeds, because it requires driving three high-speed I/Os. This causes significant current consumption from the switching losses of driving these busses. While I2C is slower, it can be worse because it uses pull-up resistors, which can draw significant current when low-resistance pull-ups are used to achieve high speeds.

The easiest solution for reducing the power of these serial-communication peripherals is to reduce speed; however, that isn't always an option. Since most of the cost of running serial communications comes from driving the bus, this is where the focus for reduction should be placed. For SPI it is important to have a clean board layout with short traces, to minimize impedance on the line. I2C requires the opposite—higher-value pull-up resistors on the bus will reduce the current consumption and can be used, in some cases, without reducing the maximum speed. In both cases, power can be minimized by reducing the number of devices on the bus or by powering off devices not in use, instead of using chip selects. In software, the power consumption of these peripherals can be reduced by making sure the CPU is disabled if the application is waiting on serial data. Additionally, grouping serial transmissions into clusters rather than constantly transmitting allows the application to spend more time powered down and less time waking up to send and receive.

### **Analog Peripheral Power Consumption**

The analog peripherals on MCUs can have a large impact on current consumption. Analog features such as BORs, comparators, and ADCs have to consume enough power to produce accurate results. So, they can't always be as power optimized as digital features, when running in low power modes. For this reason, it is important to make sure that the application only enables analog features during the times they are required, rather than leaving them always enabled. When using the ADC for slower sample rates, rather than lengthening the sample time or slowing down the ADC clock to get the sample rate required, use as fast a clock and sample time as possible. And, disable the ADC after the sampling is complete. In many MCUs, this will generate the same results with lower power consumption. Similarly, BOR features are more important when the device is running, to detect small voltage drops that can cause mis-execution than when the device is powered down, where it is only required to detect drops large enough to cause RAM corruption. So, it is useful to configure the BOR to use a lower power state in Sleep mode than is used in Run mode, taking advantage of the less power-intensive requirements and reaching as low as 50 nA or less current consumption when sleeping, while maintaining high performance when running.

### **Conclusion**

All MCUs implement an array of features that can be used to improve the power consumption of a design. However, only a small portion of these apply to all devices. It is important to be familiar with all the peripherals and unique features on the microcontroller in use, in order to truly have a power-optimized design.

## Utilizing Microcontrollers' Low-Power Modes and Peripherals

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

---

**Source URL (retrieved on 12/27/2014 - 3:08am):**

<http://www.ecnmag.com/product-releases/2009/06/utilizing-microcontrollers%E2%80%99-low-power-modes-and-peripherals>