Application Report
SLAA207 – June 2004

Choosing An Ultralow-Power MCU

Mike Mitchell
MSP430

ABSTRACT

This application report describes how to compare ultralow-power MCUs. It discusses the key differences between popular low-power MCUs and how to interpret features and specifications and apply them to application requirements.

Introduction

In today's world of battery-operated applications, power consumption of the embedded microcontroller (MCU) is becoming more and more important. Most MCU silicon vendors have low-power offerings, but choosing the best one for your application is usually more difficult than comparing bullet points on the first page of the data sheets. MCU features should be compared in detail to determine the lowest-power choice, including power-down modes, clocking systems, event-driven capability, on-chip peripherals, brownout detection and protection, pin leakage current, and processing efficiency.

Average Current Consumption

In low-power designs, the *average* current consumption determines battery life. For example, if an application uses a battery rated at 400mAh, the application must draw less than 400mAh/8760h = 45.7uA of average current to provide one year of battery life. Figure 1 shows that an application can draw much more current for short periods of time and still keep the average current consumption below the target.

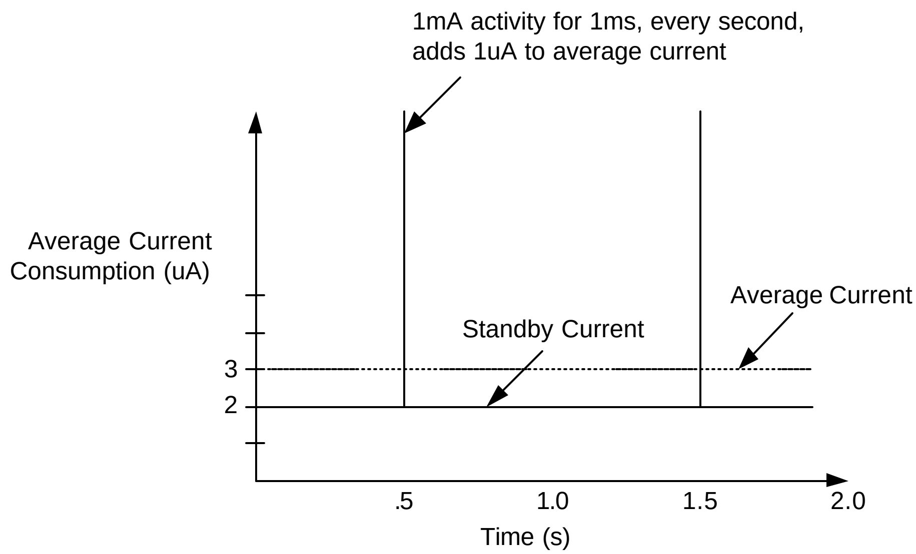

Figure 1. Average Current Consumption

Power-Down Modes

Power-down modes are among the most important features enabling the MCU to meet the current-budget. Low-power MCUs have power-down modes providing different levels of functionality. For example, the MSP430 provides five power-down modes. Low-power mode zero (LPM0) turns off the CPU and leaves everything else functional. Modes LPM1 and LPM2 add various clocking functions to the list of disabled functions. LPM3 is the most used low-power mode, leaving only a low-frequency clock-oscillator running, and any peripheral that uses that clock. LPM3 is often called the real-time clock mode because it allows a timer to operate from a low-power 32768Hz clock source, consuming < 1uA, and periodically wake the system for activity. Finally, LPM4 turns off all clocks on the device, thus turning off any peripheral that uses clocks automatically. Analog peripherals may still be active, but if none are, LPM4 current consumption is only 100 nano-amps, including RAM-retention.

Clocking Systems

The clocking system is critical to MCU power consumption. Applications may enter and exit various low-power modes several times or several hundred times a second. The ability to get into and out of the low-power modes, and process data quickly, is crucial because current is wasted by the CPU waiting for the clock to become stable. Most low-power MCUs have "instant-on" clocks that are ready for the CPU in less than 10-20us. But it is important to understand which clocks are instant-on and which are not. Some MCUs have a two-stage clock wake-up providing a low-frequency (usually 32768Hz) clock to the CPU while a high-frequency clock is being stabilized, which can take up to a millisecond or longer. On these devices, the CPU may be operational in about 15us, but running at a low, inefficient frequency or an incorrect high-speed frequency. If the CPU only needs to execute a few instructions, for example 25, it will take about 763us at 32768Hz. The CPU consumes less current at the low frequency than it otherwise would at a high-frequency, but not enough to make up for the difference in processing time with the low frequency. Alternatively, the CPU may be running at a high, but incorrect, frequency. In this case, the processing time is low, but the application cannot rely on the clock source for accurate timing. If accurate timing is required, the CPU must wait for it to become stable.

The MSP430 provides a stable, high-speed clock to the CPU in less than 6 microseconds (often faster), requiring only about 9us for the same 25 instructions (6us wake + 25 instructions at .125us instruction rate) and enabling instant-on high-speed serial communication and instant access to a timing-accurate clock source. An example of an instant-on 8Mhz clock start-up is shown in figure 2, requiring only 292 ns to be fully stable.

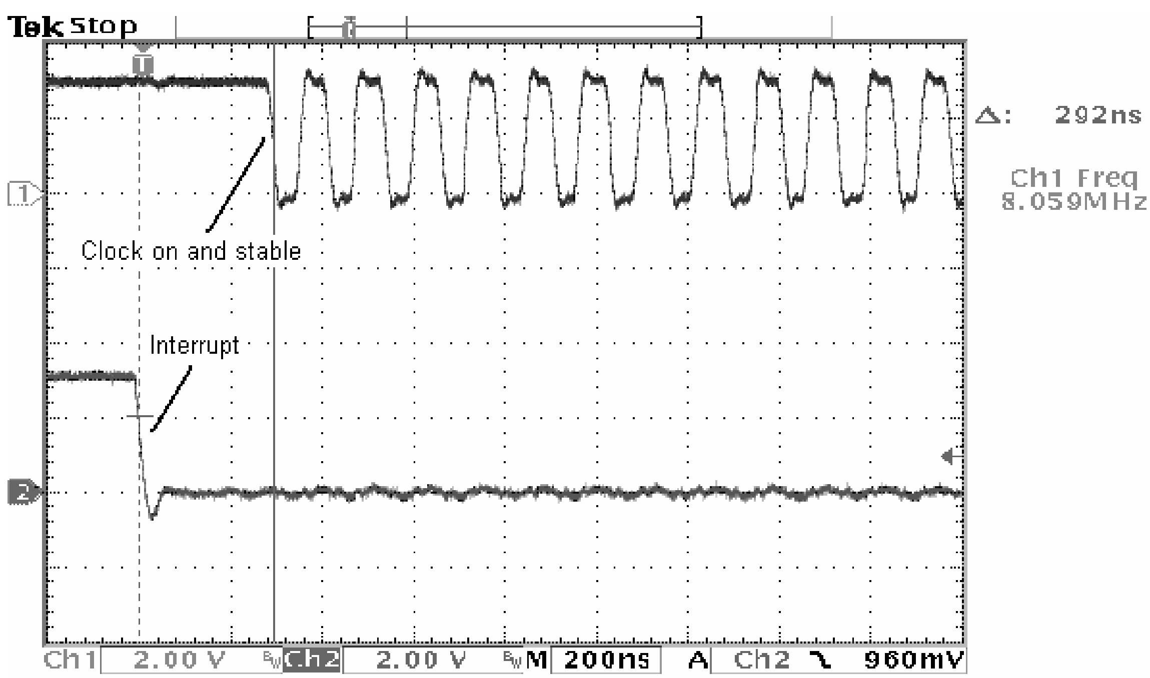

Figure 2. MSP430 Clock Startup

In addition, if the MCU clock system provides multiple clock sources for the peripherals, the peripherals can operate while the CPU is asleep. For example, an A/D conversion may require a high-speed clock. If the MCU clock system provides the high-speed clock independently of the CPU, the CPU can sleep while the A/D converter is operational, saving the CPU current consumption.

Interrupts

Event-driven capability goes hand-in-hand with clock system flexibility. Interrupts bring an MCU out of a low-power mode, so the more interrupts an MCU has, the more flexibility to prevent current-wasting CPU polling and reduce the power consumption. Polling can mean the difference between making the power budget and not, because it wastes CPU bandwidth and requires excess current while waiting for an event to occur. A good low-power MCU will have extensive interrupt capability, providing interrupts for all its peripherals and many external interrupts for external events. The MSP430 provides interrupts for 16 I/O pins and all peripherals. Some peripherals, like Timer_A, Timer_B and ADC12 have multiple interrupts for total flexibility.

Applications with buttons or a keypad exemplify the benefits of external interrupts. Without interrupt capability the MCU must poll the keypad or buttons often to determine if they are being pushed. Not only does the polling itself use power, but controlling the polling interval requires a timer, which uses additional current. In contrast, with interrupts, the CPU can sleep the entire time and only awaken when a button is pushed.

Peripherals

Peripheral power-consumption and power-management should also be considered when choosing a low-power MCU. Some low-power MCUs are retrofitted versions of 20-30 year-old architectures that were not designed for low power. The MSP430 was designed for low power from the start and has low-power features built into the peripherals. One thing to look for is the ability to individually enable or disable the peripherals when needed, or more importantly the automatic enabling or disabling of the peripherals. The MSP430 ADC12 is an example of such intelligent peripherals. If it is not actively converting, it draws no current. It automatically disables some its internal oscillator and digital circuitry when it is not converting, effectively turning itself off automatically. The require circuitry is then re-enabled, or turned on, automatically if a conversion is triggered.

Another key benefit peripherals can provide is interoperability. For example, the MSP430 ADC12 module can be triggered by Timer_A or Timer_B. This provides direct, accurate control over ADC sampling, but it also allows the sampling to occur automatically – without the CPU. Again, the CPU does not have to wake up to initiate the ADC conversion. The timer controls the timing, and initiates the conversion. The ADC12 can then wake the CPU after the conversion or conversions have completed.

Some MSP430 devices include direct memory access (DMA) capability that automatically handles data without CPU intervention. Using the DMA controller can increase data-handling throughput, but more importantly, it lowers power consumption. Using the DMA to automatically move ADC data, for example, to RAM allows the CPU to remain sleeping while ADC conversion are taking place, only waking the CPU when all desired conversions are completed.

Peripheral clock control is another feature that reduces system current consumption. The MSP430 I^2C module has direct control over the system clock. This means that if the I^2C module needs a clock, it automatically enables it. This again allows the CPU to remain asleep longer, plus is provides the ability to catch the incoming I^2C address without having to wake the CPU to enable the I^2C module.

Brownout Protection

Many MCUs have integrated brownout protection, resetting the MCU when the power supply dips below the normal operating range. For most, the brownout reset circuit requires an additional 10 – 70 uA of current for operation. The ability to enable or disable the brownout protection to save power consumption is often given, but brownout protection must be enabled 100% of the time to be useful because brownout cannot be predicted. This adds the brownout current consumption directly to the bottom line of the power budget of the system.

The brownout protection of the MSP430 cannot be disabled and requires 0uA of additional current. It is included in the datasheet specifications for the power modes.

Pin Leakage

Leakage current is sometimes overlooked when choosing a low-power MCU, but it must be considered for the most demanding low-power applications. Most low-power MCUs have specified input leakage currents of 1uA. This could consume up to 20uA for a 20-input device! The MSP430 specifies pin leakage currents of 50nA – 1uA max for the same 20-input device.

Processing Efficiency

Finally, MCU processing efficiency is often misunderstood. It is often thought that 16-bit MCUs require twice the memory of 8-bit MCUs, but a 16-bit architecture can actually require less code than an 8-bit architecture and 16-bit MCUs usually execute tasks more quickly. For example 8-bit MCUs require CPU overhead to manage data in applications that have 10-bit A/D conversion data or applications that require 16-bit math. Further, many MCU offerings today have a single working file, or accumulator, that data must be moved through to be acted upon, requiring excess CPU overhead when compared to register based architectures. Table 1 shows the instructions required to move 10-bit A/D data on both the MSP430 and an 8-bit architecture. Operating with a 1Mhz clock, the MSP430 requires 6us for the move, whereas the 8-bit device requires 24us.

16-Bit MCU	**8-Bit MCU**
`mov.w   &ADC10MEM,&RAM`	`movf    ADRESH,W`
	`movwf   RAML`
	`bsf     0x20`
	`movlf   ADCHRESL,W`
	`bcf     0x20`
	`movwf   RAMH`

Summary

Choosing a low-power MCU requires investigation beyond datasheet bullet points. But the effort pays off when battery life is extended, or the current-budget is met.