WHITE PAPER

Jacob Borgeson
Stefan Schauer
Horst Diewald
Texas Instruments

TEXAS INSTRUMENTS

Benchmarking MCU power consumption for ultra-low-power applications

Introduction

Developers of battery-powered devices often have the challenge to offer high levels of functionality and performance while simultaneously maximizing battery life. Applications like water and gas flow meters, medical monitoring devices and remote sensors typically demand months or even years of battery life from a single battery. In some cases, developers are also challenged to develop next generation products with no battery at all, requiring energy harvesting from environmental sources such as heat, vibration and light. Furthermore, as the demand for longer battery life and smaller batteries increases in more applications, users continue to demand more functionality and higher performance in their products.

To maximize functionality and battery life, developers of these battery-powered applications must consider many factors in their system architecture and design. In these applications, the microcontroller is a primary power consumer and developers must carefully consider the way energy is used. This article focuses on breaking down the primary modes in which microcontrollers consume energy by describing the critical parameters that must be considered in each of these modes, and by providing a holistic framework for developers to evaluate and compare microcontrollers in the context of specific applications. By understanding the many ways microcontrollers consume energy, developers can make system architecture decisions, choose optimal components and provide microcontroller users with optimized functionality and longer battery life.

Power is more than just one number

To maximize battery life, developers must minimize power consumption over the life of the product. Total power and energy are defined as:

$$\text{Power} = I \times V$$

$$\text{Energy} = I \times \text{Time} \times V$$

To minimize power or energy consumption from the system microcontroller, a developer can simply examine product datasheets to determine the current consumed at the CPU frequency needed for the application. Multiply this current by the battery voltage, and use the resulting data to choose the lowest power microcontroller. This seems simple; however, consider a few hypothetical questions about typical applications to determine if this view of power consumption is comprehensive:

- Does the system go into a standby mode when the microcontroller is not running?
- Is the system required to automatically wake itself up at specific time intervals?
- Does the system take any real-world analog signal measurements?
- Does the system need to record any data for analysis or transmission at a later time?

While these are just a few of the questions developers must consider when optimizing power consumption, a majority of battery-powered applications answer "yes" to several of the questions above. The simple method of looking at microcontroller current and voltage does not result in an accurate representation of microcontroller power consumption.

For developers to gain a comprehensive view of microcontroller power consumption, they must consider four primary power categories:

- **Standby power** – Typical microcontroller applications spend a majority of their product life in a low-power standby mode waiting for an internal or external event to wake-up the CPU to process data, make decisions and communicate with other system components. In many battery-powered applications, standby power consumes the largest amount of energy and battery life.
- **Peripheral power** – Modern microcontrollers integrate many intelligent peripherals allowing communication with other system components and measuring real-world signals. In systems measuring analog signals, this can have a significant impact on battery life.
- **Data logging power** – Most microcontroller applications log data for analysis or transmission at a later time. This data is logged using non-volatile memory, either inside or outside of the micro controller. Depending on the frequency and amount of data that has to be recorded, data logging can greatly impact battery life.
- **Active power** – Understanding power when the CPU is actively processing is critical to maximizing battery life.

$$\text{Total Energy} = \text{Energy}_{\text{Active}} + \text{Energy}_{\text{Standby}}$$

$$\text{Time x I x V} = \text{Time}_{\text{Active}} \text{ x } I_{\text{Active}} \text{ x V} + \text{Time}_{\text{Standby}} \text{ x } I_{\text{Standby}} \text{ x V}$$

For battery powered applications, 3V is the typical nominal battery voltage. It is also the voltage at which many power consumption numbers are specified in datasheets. To narrow the elements in the power calculations to those under the influence of the microcontroller, we excluded voltage from further calculations so that average current is the focus. Obviously, power and energy consumed change dramatically if a different voltage level is used. Additionally, we removed the time aspects so each component of power is viewed as part of the overall average current. For information on the ratio of time spent in each component of the power calculation, developers should look at the particular system. In this case, we chose a generic remote sensing application as it is widely applicable, and we used the MSP430FR59xx as the example microcontroller with embedded ferroelectric random access memory (FRAM) for data logging.

$$I_{\text{Average}} = I_{\text{Standby}} \text{ x Ratio}_{\text{Standby}} + I_{\text{Active}} \text{ x Ratio}_{\text{Active}} + I_{\text{Peripheral}} \text{ x Ratio}_{\text{Peripheral}} + I_{\text{Data}} \text{ x Ratio}_{\text{Data}}$$

A remote sensor is one example of a typical low-power, battery-powered product, commonly used in industrial, seismology, agriculture, building automation and security applications. Typical processing flow in this type of application would be:

Component	Application activity
Peripheral	256 samples of sensor data; pressure, heat, vibration, chemical, etc
Active	Process analog inputs, compare to threshold, average inputs
Data	Store 16 bytes in non-volatile memory every cycle
Standby	Wakes once every 3 seconds to run the program; maintain RTC, capable of wake on IO interrupt

Table 1: Remote sensing application activity profile

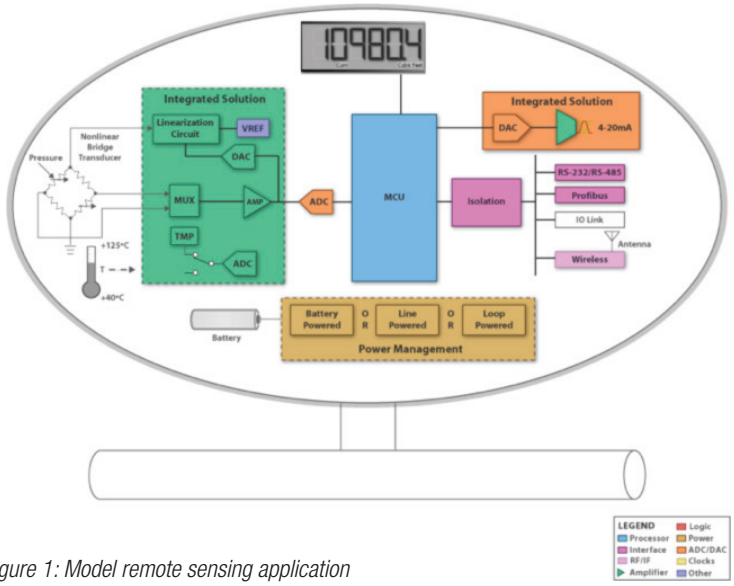

Figure 1: Model remote sensing application

Clearly, power is more than just one number, and the four categories previously mentioned provide a broad framework that must be considered when developing a battery-powered system. However, each category itself is complex and the specific application requirements require careful consideration.

Standby power

It is common for developers to start their processor power analysis by considering active processing power. Though it may seem counterintuitive, the power the microcontroller consumes when it is not operating is often more important than active processing power. Referring back to the remote sensing application, the system typically wakes up from standby mode once every three seconds, so the system is in standby mode greater than 99% of the time. The table below uses typical specs from Texas Instruments' (TI) "Wolverine"-based MSP-430FR59xx microcontroller family with a 500nA standby mode to illustrate the impact of standby power on total microcontroller power across varied active time to standby time ratios:

\Active to Standby Ratio	% Time in Standby	$Time_{Active}$ x I_{Active} (µAs)	$Time_{Standby}$ x $I_{Standby}$ (µAs)	Total Charge (µAs)	% Impact of $I_{Standby}$ to Total Power
1:10	90%	100	5	107	6.54%
1:100	99%	100	50	150	33%
1:1,000	99.9%	100	500	600	83.3%

Table 2: Impact of $I_{Standby}$ on total power based on ratio of active to standby time

As table 2 demonstrates, as the ratio of active to standby exceeds 1:100, understanding the true standby current becomes critical. At 1:1000, for example, standby current begins to account for nearly 85% of system power. Thus, a 10% error in standby power estimations corresponds to an 8.5% difference in battery operating life calculations – lowering battery life by months or even years.

To help developers accurately evaluate the standby current the microcontroller will consume, the following system considerations and microcontroller parameters need to be considered and accounted for in the standby current estimate:

- **Automatic wake-up on time intervals** – Modern microcontrollers often offer real-time clocks (RTC) that can run in low-power standby modes enabling the microcontroller to wake-up automatically at specified time intervals. The remote sensing application uses this capability to wake up once every three seconds to measure analog data. It is important to understand the current required to run the RTC in standby mode as this can be a significant portion of standby current.
- **RAM retention during standby** – Maintaining RAM contents during standby allows microcontrollers to wake-up quickly without running startup code that consumes valuable energy. This saves energy and time for lower system latency. The current required to enable RAM retention modes can be significant and needs to be carefully considered.
- **Interrupt capabilities** – Microcontrollers can often leave certain peripherals active in standby modes, enabling the microcontroller to wake-up quickly with certain events such as a UART command or GPIO interrupt. The remote sensing example monitors two-to-three GPIO lines constantly to wake-up the processor for instant activity.
- **Power monitoring** – Brown out reset (BOR) and supply voltage supervisor (SVS) are important circuits that monitor the integrity of the microcontroller's power source. Faults and interruptions to the microcontroller power source can impact the reliability of operation. It is critical to include these currents in the standby current estimations. This can add as little as a few nanoamps to as much as 500microamps to standby numbers.
- **Temperature** – Temperature is often overlooked on low-power system designs, however modern semiconductor processes often drive much higher leakage currents at higher temperatures – in some cases as much as 10-15 times more standby current between 25°C and 85°C.

Peripheral power

Microcontrollers like TI's MSP430FR59xx devices have been designed from the ground-up to optimize for lowest-standby power employing advanced power and clock gating features, ultra-low-power analog circuit designs and advancements in silicon technology such as embedded FRAM. In the case of the remote sense application, the MSP430FR59xx microcontroller can enable RTC-standby mode with RAM retention, SVS/BOR and GPIO interrupt capability for as little as 500 nanoamps.

For many low-power battery-powered applications, standby power can be the primary impact to battery life, and it is critical that developers consider how the features required in the application impact the standby current. It can make a difference in years of battery life or save on system cost by using a lower capacity, less expensive battery. In the remote sensing example, the average standby current can be estimated as:

$$Avg I_{Standby} = I_{Standby} \times (Time_{Standby}/Time_{Total}) = (2.997s/3s) \times .5\mu A = .4995\mu A$$

(Assumptions: 500nA LPM3.5 w/ RTC, SVS, and BOR at 25C; SLAS704)

Low-power embedded systems often use analog, digital interfaces, references and other circuitry to achieve system functionality goals. Modern microcontrollers have increasingly integrated more of these functions to simplify design complexity, lower system cost, enable smaller devices and decrease power consumption. Developers of these systems must carefully consider both the capabilities of the microcontroller as well as the power consumption of peripherals to efficiently optimize designs for maximum functionality and battery life. In the case of the remote sensing application introduced earlier in the article, an analog-to-digital converter (ADC) is used to measure a real-world signal such as an infrared sensor, temp sensor, or some other sensor. To accurately estimate peripheral power, developers should consider the following system design characteristics and microcontroller parameters:

- **ADC** – Current consumed variable across sampling speeds and operating modes, so make sure to browse the datasheet for the mode most appropriate to each application.
- **Comparator** – Comparators are often low-power solutions to achieve the same basic analog measurements performed with an ADC.
- **Voltage reference** – To minimize external components, often internal voltage references are provided with microcontrollers. These references are used in ADCs, comparators and other analog circuits. In many cases, the current consumed by the voltage reference is not included in the current specifications for the peripheral in use. Developers need to consider this as the reference can often be larger than the comparator or ADC that is using it.
- **Digital interfaces** – UART, I2C, and SPI are among the many digital interfaces used in embedded systems. Each of these peripherals consume valuable energy impacting battery life. Data rate and drive strength on these digital interfaces need to be carefully considered to estimate the current consumption.

In any application that requires measurement of real-world analog signals or the use of digital interfaces, developers must consider these currents in addition to the active power or the standby power depending on what modes in which the peripherals are used.

$$Avg I_{Peripheral} = (Time_{Peripheral}/Time_{Total}) \times I_{Peripheral} = (0.00128s/3s) \times 75\mu A = .032\mu A$$

Assumptions: 256 samples at 200ksps at current of 75µA (SLAS704) is .00128s

Data logging power

Many microcontroller applications need to record measurements and data for use later in the application. For example, the most recent data could be compared to past data recorded to look for larger trends. Logging data on the sensor itself can also give users critical information from the moment of failure, as in the case of intelligent circuit breakers or automotive black boxes.

Data logging can be extremely difficult when using a small, low-cost battery. The current required to erase and program Flash can vary from 4-to-12 mA. This cannot be directly sourced by coin cells that have a maximum current of 3-to-4mA. Also, such operations take more than two dozen milliseconds just to set-up and erase the memory before programming, and then about the same time to write the required sector of Flash. In addition, if an external memory is used for storage, there is an extra cost for using the serial communication interface to transfer data. This adds peripheral and active current consumption to the system.

This is important because it means that either a larger, more expensive battery must be used, or a lower-power external memory must be added to the design. These are both costly options and sometimes difficult to implement due to size constraints.

Another key point is that logging data at slower rates in FRAM drastically lowers the current consumption. It can be as low as 9µA to log data at the relatively quick rate of 13 kilobytes per second. This is about 500 times lower than the same rate with Flash!

$$Avgl_{DataLog} = Time_{Data} \times (I_{Data}/Time_{Total}) = .0012s \times 9\mu A/3s = .0036\mu A$$

Assumptions: 16 bytes at 13kBps is .00128s at 9µA (Bench Measurements SLAA498)

Active power

Active CPU power is often viewed as the largest power consumer in battery-powered applications. As the previous sections have shown, there are many other sources of power consumption in a microcontroller system. That said, it is important that microcontroller software and hardware is optimized to minimize active power to take advantage of intelligent peripherals and utilize standby power modes that consume less energy.

To accurately estimate active power, developers should consider using the following microcontroller features, specifications and methods:

- **Software execution from non-volatile memory or RAM** – Developers must carefully consider whether software is executed from non-volatile memories or RAM in estimating the current consumption. Executing from RAM can offer lower active current specifications; however, many applications are not small enough to execute from RAM alone and require programs be executed from non-volatile memory.
- **Bus clocks enabled or disabled** – Most microcontroller applications require access to memories and peripherals during software execution. This requires bus clocks to be enabled and needs to be considered in active current estimates.
- **DMA** – Many microcontrollers have DMA features that allow peripherals to complete activities using DMA's while the CPU is sleeping – enabling substantially lower currents at the system level. It is important to consider whether this is possible with the particular peripheral or task as this can save significant current.
- **Acceleration** – Low-power microcontrollers typically employ accelerators that lower the number of cycles and energy required for specific tasks. For instance, AES256 encryption can take as much as 7,000 CPU cycles without acceleration. With hardware accelerators, the same encryption can take approximately 500 CPU cycles. In this case, hardware acceleration can reduce active processing time by more than 10x!
- **Use optimized code** – Microcontroller code can be optimized for shortest execution time using intelligent code structure and compiler optimizations.

Ensuring the proper conditions and optimization techniques are employed when estimating the number of cycles required for active processing tasks and the active current required is critical to accurately estimate peak currents and battery life impact. Referring back to the remote sensing application, the average current is calculated as:

$$AvgI_{Active} = Time_{Active} \times (I_{Active}/Time_{Total}) = 70\mu s \times 1230\mu A/3s = .029\mu A$$

Assumptions: 1000 cycles required to complete averaging of 256 samples at 16MHz = 70µs;

Active current @16MHz = 1.23mA

Bringing it all together: Remote sensing applications

To illustrate how standby, peripheral, data logging and active CPU power work together to determine total battery life, the remote sensing application calculations derived in each section can be easily combined to show the total projected battery life of this typical application. Starting with an average power calculation:

$AvgI_{Total}$

$= AvgI_{Standby} + AvgI_{Periperal} + AvgI_{Active} + AvgI_{DataLog}$

$= 0.4995 + 0.032 + 0.029 + 0.0036 = 0.5641\mu A$

As you can see, standby current has by far the largest impact on average current, followed by analog and active in the case of remote sensing applications. The impact of individual components is dependent on the duty cycle of the specific application. The average current can be easily translated into battery life by taking the charge in a common CR2032 battery that can be purchased for under $1 at 230mAh.

230000µAh/0.5641µA

= 407,729 hours

= 16,988 days

= 46 years of battery life (far longer than the shelf discharge of 8-10 years)

Because the microcontroller power consumption has been minimized, this gives the developer the opportunity to add other system features to their system and still offer 5-to-10 years of battery life from a single coin cell battery. In addition, this type of calculation can be easily extended by using the same use case to evaluate several different microcontrollers to determine which offers the lowest power capability for a specific application. In the table below, four different microcontrollers are evaluated using the same remote sensing application example:

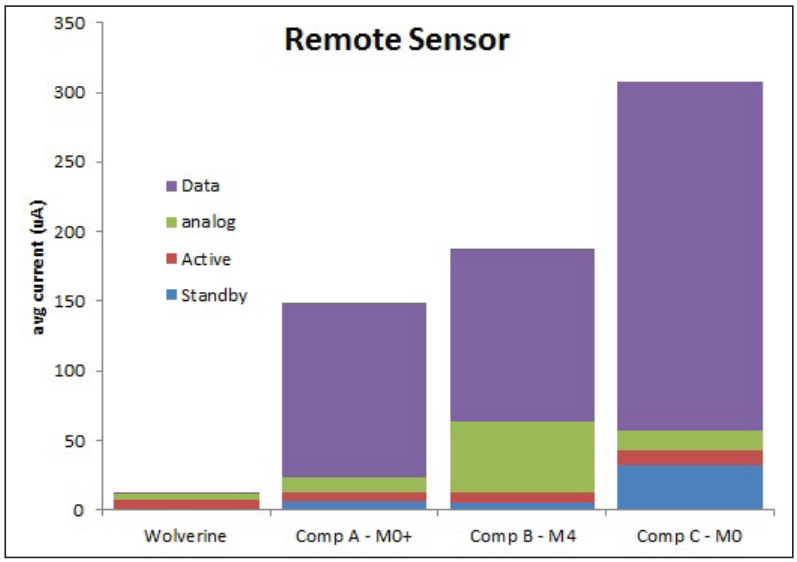

Figure 2: Average current comparison

While the remote sensing application was used as an example, it is typical of use cases for many low-power, battery-powered products that require long battery lifetimes. Water and gas flow meters, heart rate monitors, pedometers and industrial sensors all employ similar techniques to minimize costs, meet regulatory standards and meet consumer demands. To maximize battery life, all four components of microcontroller power consumption must be considered.

Summary

For developers, meeting the needs of increased system functionality and performance objectives while increasing the battery life of products is a significant challenge. To effectively develop products that deliver the longest possible battery life – or even operate with no battery at all – requires a deep understanding of the both the system requirements and the microcontroller's current specifications. This is much more complex than simply estimating how much current the CPU consumes when active. Depending on the application being developed, standby current, peripheral current, or data logging current may have a more significant impact on battery life than CPU power.

This article was created to build a simple framework to help developers understand how microcontrollers consume power, how power can be optimized and how power consumption can be fairly compared across available microcontroller solutions, to enable the developer to meet the specific needs of their application.

TI will continue to push the boundaries of low power through optimization of ALL key areas of power consumption. TI's ultra-low-power MSP430™ portfolio has been designed with low-power at the foundation – through silicon technology advancements, low-power design techniques, intelligent peripherals and accelerators and low-power design tools.

For more information on ultra-low-power design and TI's extensive portfolio of ultra-low-power products, visit **www.ti.com/ulp**.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components which meet ISO/TS16949 requirements, mainly for automotive use. Components which have not been so designated are neither designed nor intended for automotive use; and TI will not be responsible for any failure of such components to meet such requirements.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265