Strategies for Choosing the Appropriate Microcontroller when Developing Ultra Low Power Systems

By **Monica Redon**

Share on

The Internet of Things (IoT) is driving a huge demand for a wide assortment of battery-operated devices. This in turn is driving the requirement for ever-increasing energy efficiency of microcontrollers and other system-level components. As a result, ultra low power (ULP) has become an over-used marketing phrase, especially when used to describe microcontrollers. As a first step in understanding the true meaning behind ULP, consider the variety of its implications. In some cases, the lowest active current is required when the power source is severely limited (for example, energy harvesting). Alternatively, the lowest sleep-mode current is required when the system spends most of its time in standby or sleep mode, waking up infrequently (periodically or asynchronously) to process tasks. Furthermore, ULP can also imply energy efficiency, whereby most work is performed in a limited time period. Overall, a battery-powered device will utilize a combination of these requirements based on a set of trade-offs.

Of course, ULP is also a matter of opinion and function—for example, we would generally consider a microcontroller unit (MCU) to be ULP with an active mode in the range of 30 µA/MHz to 40 µA/MHz and a shutdown current of 50 nA to 70 nA. However, classifying a microcontroller as ultra low power is a complex combination of features, including architecture, SoC design, process technology, smart peripherals, and deep-sleep modes. In this article, we'll examine two microcontrollers from Analog Devices to help you understand how to interpret the true meaning of ultra low power in this context. We'll also examine the certification mechanism from the EEMBC consortium, which ensures the score's veracity in order to help system developers to choose the most appropriate microcontrollers for their solution.

Measuring and Optimizing Ultra Low Power

As a starting point in understanding ULP, we first explain how to measure it. Developers typically would look in a data sheet, where they would find current values per MHz, as well as current for the different sleep modes. The first problem is that, when looking at active current consumption, the data sheet usually fails to explain the conditions to obtain this value, such as code, voltage, and wait states on the flash. Some vendors use an active mode reference, such as the EEMBC CoreMark, while others will use something as simple as running a "while 1" statement. If there are wait states on the flash, the microcontroller unit's performance is reduced, increasing the execution time and therefore increasing the energy consumption to execute a task. Some vendors provide numbers at typical voltage, others at minimum voltage, whereas others don't specify any voltage. Perhaps these are subtle differences, but without a standard, comparisons are only approximate.

Typically, deep-sleep modes are fairly well explained in data sheets, but again, the conditions to obtain the current consumption on these modes varies from one vendor to another (for example, the amount of memory retained or voltage). Furthermore, in a real application, a user must also account for energy consumed when entering and exiting these modes. This could be either an insignificant value or really relevant depending on if the device spends most of the time in sleep mode or wakes up frequently. Which leads to the next point—just how much time does a device spend in sleep? The balance between active and sleep modes are important in determining the ULP measurement. To simplify the process, EEMBC used a 1 second period for its ULPMark-CoreProfile (ULPMark-CP), a benchmark that is used as a data sheet standard by many microcontroller vendors. Note: The decision of using 1 second was taken as a consensus within the EEMBC working group. Taking into account the active time of the ULPMark-CoreProfile workload, the duty cycle will be around 2%. In this benchmark, the device wakes up once per second, performs a small amount of work (the active cycle), then goes back to sleep.

Typically, in active mode there is an offset in the current consumption due to the analog circuitry and, therefore, minimizing active current and effectively using deep-sleep modes make sense in optimizing overall system energy use. Note that by reducing the frequency, the active current will be reduced, but the time will be increased, and the offset due to the analog circuitry mentioned previously stays constant while the microcontroller is active. However, what are the trade-offs between microcontroller choices, and what's the impact of the application's duty cycle and deep-sleep currents on that energy?

The energy per cycle, as a function of duty cycle D (given as percentage of time in active mode vs. total time), is defined by a simplified equation that assumes the energy in the on and off transitions are small.

$$Energy = V \times t \times [(I_{ACTIVE} \times D_{ACTIVE}) + (I_{SLEEP} \times D_{INACTIVE})]$$

where the slope is defined by I_{ON} since I_{SLEEP} is much smaller than I_{ON} and the y intercept is just I_{SLEEP}. This equation can help comprehend the duty cycle in which the active current is more important than the sleep current.

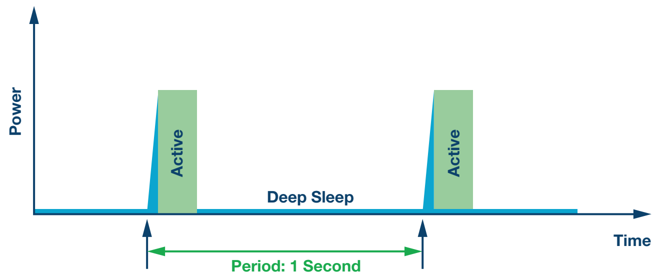


Figure 1. The ULPMark-CP has a period of 1 second. During this time, the device wakes up from deep-sleep mode, executes a fixed workload, and returns to deep-sleep mode.

The Ultra Low Power Test Platforms

As previously mentioned, we are going to compare the ultra low power (energy) of two microcontrollers from Analog Devices—namely the ADuCM4050 and ADuCM302x. In the ULPMark results table, the ADuCM4050 and ADuCM302x achieve scores of 203 and 245.5, respectively. Keep in mind that this benchmark is only exercising the microcontroller unit's core (hence the name CoreProfile). How does one account for the 18% difference?

The ADuCM4050 contains an ARM® Cortex®-M4F that implements the ARMv7E-M architecture. The ADuCM302x contains an ARM Cortex-M3 that implements the ARMv7-M architecture. While both cores have a 3-stage pipeline with branch speculation and both are similar on instruction-set architecture, only the Cortex-M4F supports DSP and floating-point instructions. As there are no DSP instructions on the ULPMark-CoreProfile, the Cortex-M4F part does not take advantage of FPU.

For the benchmark analysis, the ADuCM4050 and ADuCM302x were operated at 52 MHz and 26 MHz, respectively. With the ADuCM4050 requiring about 11,284 cycles to perform the ULPMark workload, and the ADuCM302x requiring 10,920, this means that the former completes the active mode portion in 217 µs of the 1 second period, whereas the latter is active for 420 µs. The reason why the ADuCM4050 uses a few more cycles than the ADuCM3029 is because of the frequency used (52 MHz and 26 MHz, respectively), the ADuCM4050 needs one wait state for the flash, whereas the ADuCM3029 has no wait states on the flash. As the ADuCM4050 has a cache memory, there is no major penalty by adding the wait state on the flash, as many instructions are executed from the cache memory, which may be accessed at full speed (52 MHz) without the need of an extra wait state. With respect to the execution time, as expected, the ADuCM4050 performs the workload faster than the ADuCM3029 as it runs at twice the frequency of the ADuM3029.

In order to obtain the EEMBC benchmarks code, you must be member or the working group. You can become a member here. Monica Redon is the representative for Analog Devices in the EEMBC board, so you may contact her for further information.

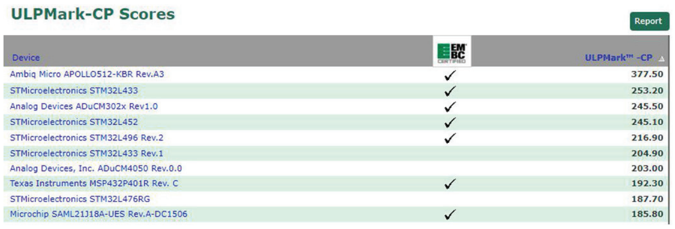
Table 1. Approximate cycles to complete the ULPMark-CoreProfile workload on popular ARM cores. The cycles are approximate because cycle count will have some compiler dependencies.

ARM Core	Approximate Cycles to Complete ULPMark Active Mode
Cortex-M0	15,174*
Cortex-M0+	14,253
Cortex-M3	10,920
Cortex-M4	11,852
Cortex-M4F	11,284

*This is an estimate based on the Cortex-M0+ and Cortex-M3 numbers.

But why does the ADuCM4050 consume 10 µA/MHz more than the ADuCM3029? The reason behind this increase is because the former may operate at twice the frequency of the latter, requiring extra buffers to accomplish the timing constraints for a higher frequency. The ADuCM4050 also has some extra features compared to the ADuCM3029:

- Double the memory size (for both SRAM and Flash): 128 kB and 512 kB vs. 64 kB and 256 kB on the ADuCM3029. Depending on the application needs you might need the extra storage.
- Double frequency: 52 MHz vs. 26 MHz on the ADuCM3029, so the ADuCM4050 has a better performance.
- Added RGB timer.
- Added new security features: protected key storage with key wrap-unwrap and keyed HMAC with key unwrap.
- Added three additional SensorStrobe™ outputs.
- Added full SRAM retention: Up to 124 kB might be retained on the ADuCM4050 vs. up to 32 kB on the ADuCM3029.



Device	EEMBC CERTIFIED	ULPMark™ -CP
Ambiq Micro APOLLO512-KBR Rev.A3	✓	377.50
STMicroelectronics STM32L433	✓	253.20
Analog Devices ADuCM302x Rev1.0	✓	245.50
STMicroelectronics STM32L452	✓	245.10
STMicroelectronics STM32L496 Rev.2	✓	216.90
STMicroelectronics STM32L433 Rev.1		204.90
Analog Devices, Inc. ADuCM4050 Rev.0.0		203.00
Texas Instruments MSP432P401R Rev. C	✓	192.30
STMicroelectronics STM32L476RG		187.70
Microchip SAML21J18A-UES Rev.A-DC1506	✓	185.80

Figure 2. Top 10 results of ULPMark-CP, located in the EEMBC web site (18th of August 2017).[1]

Depending on application needs (power optimization, extra storage, active performance, retention capability …), you can decide to use the ADuCM4050 or the ADuCM302x product.

With respect to the deep-sleep mode, the ADuCM4050 achieves a lower hibernate current when retaining twice the memory of the ADuCM3029 does when running the ULPMark-CoreProfile (16 kB on the former vs. 8 kB on the latter). The reason for this improvement is an enhanced architecture on the newer ADuCM4050 product.

The Role of the Compiler

As described above, ULPMark is comprised of two operational states—an active state and a low power state where the device is in a sleep mode. These states are combined into a period of exactly 1 second. In the active state, each device performs the same workload. But as we saw, the efficiency of this work is influenced by the architecture. Additionally, it's also influenced by the compiler. Compilers may choose to change and optimize statements such that the instruction mix will change.

Depending on application needs, you might optimize for size, for speed, to balance size and speed, etc. Loop unrolling is a simple example where the ratio of branches executed to the code inside the loop changes. Compilers can still play a large role in finding a better way of computing results, but the work being done is equivalent. For example, the ULPMark-CP result for the ADuCM3029 might vary from 245.5, with high optimization for speed, to 232 for medium optimization or 209 for low optimization. Another example of the importance of the compiler is demonstrated by the ULPMark results for a Texas Instruments MSP430FR5969, which improve by 5% by applying a newer version of the IAR Embedded Workbench compiler—although it's not known what internal compiler changes were made to accomplish this improvement (eembc.org/ulpbench/). Similarly, without insight into proprietary compiler technology, it's not possible to determine why the STMicroelectronics STM32L476RG results improve by 16% in going from the ARMCC compiler to the IAR compiler.

Both results on the Analog Devices MCUs were generated using code compiled by the IAR compiler, but with different versions. The ADuCM4050 and ADuCM302x used the IAR EWARM 7.60.1.11216 and 7.50.2.10505, respectively. Again, it's not possible to know what internal changes were made. Both scores were submitted with the **no_size_constraints** option that corresponds to optimized speed.

Translating ULPMark to an Energy Value

The ULPMark-CoreProfile uses a formula that takes the reciprocal of the energy values (median of 5× the average energy per second for 10 cycles).

$$Energy\ (\mu J) = \frac{1000}{EEMarkCP}$$

The energy is obtained as the sum of the energy consumed while the device is executing the workload (in active mode) and while the device is in hibernate.

$$Energy = Active\ Energy + Sleep\ Energy$$

According to the ADuCM3029 data sheet, the typical value for an active current is 980 µA when running prime numbers code. This code fits into the cache and takes advantage of its lower power consumption. For the ULPMark-CoreProfile code, as it is a mainly linear code, there is no big benefit of having the cache enabled, so the current consumption is similar to the one shown in the data sheet for the cache disabled, 1.28 mA. For the hibernate current, the ULPMark-CoreProfile requires having LFXTAL and RTC enabled, so the current consumption in sleep mode is 830 nA (according to the data sheet). As mentioned previously the active time duration is 420 µs.

$$Energy = Voltage \times Current \times Time$$
$$Active\ Energy = 3\ \text{V} \times 1280\ \mu\text{A} \times 0.42\ \text{ms} = 1.61\ \mu\text{J}$$
$$Sleep\ Energy = 3\ \text{V} \times 0.83\ \mu\text{A} \times 999.58\ \text{ms} = 2.49\ \mu\text{J}$$

According to the data sheet numbers and the execution time, the energy for the active current is 1.61 µJ, and the energy consumed during the sleep time is 2.49 µJ. The score according to those values matches the ones measured with the EEMBC EnergyMonitor software.

$$Energy\ (\mu J) = 1.61 + 2.49 = 4.10\ \mu\text{J} \cong \frac{1000}{245.5} = 4.07\ \mu\text{J}$$

One of the shortcomings of the first generation ULPMark is that the run rules restrict the operating voltage to 3 V (implemented this way by the working group to establish a common level for all devices). Most modern MCUs have much better energy efficiency running at lower voltages (although this could be affected by temperature and operating frequency). For example, the STMicroelectronics STM32L476RG's ULPMark result improves by 19% by utilizing a dc-to-dc converter to drop the voltage from 3 V to 1.8 V.

The STMicroelectronics STM32L476RG is not the only device whose published result is influenced by utilizing a dc-to-dc converter, although with some devices the converter is integrated into the device itself as in the ADuCM302x and ADuCM4050, where no external IC is necessary to improve the power performance of the device. Nevertheless, using a dc-to-dc converter helps level the playing field because it allows the device to operate at its optimal energy efficiency. For example, a device that only works at 3 V would not benefit from a dc-to-dc converter, as it is already at its optimal (or perhaps suboptimal) efficiency. On the other hand, a device that can work down to 1.8 V but that doesn't take advantage of a dc-to-dc converter is basically wasting 64% of the supplied energy. Furthermore, for a system designer whose priority is energy efficiency, the additional cost of an external dc-to-dc converter might not be important if the system is using a 3 V battery. Care must be taken in using a dc-to-dc converter to avoid measuring the energy efficiency of the converter and not the MCU. Nevertheless, one has to take in consideration that in real applications, dc-to-dc operation modes may have disadvantages such as extended transition times from/to active mode from/to sleep mode.

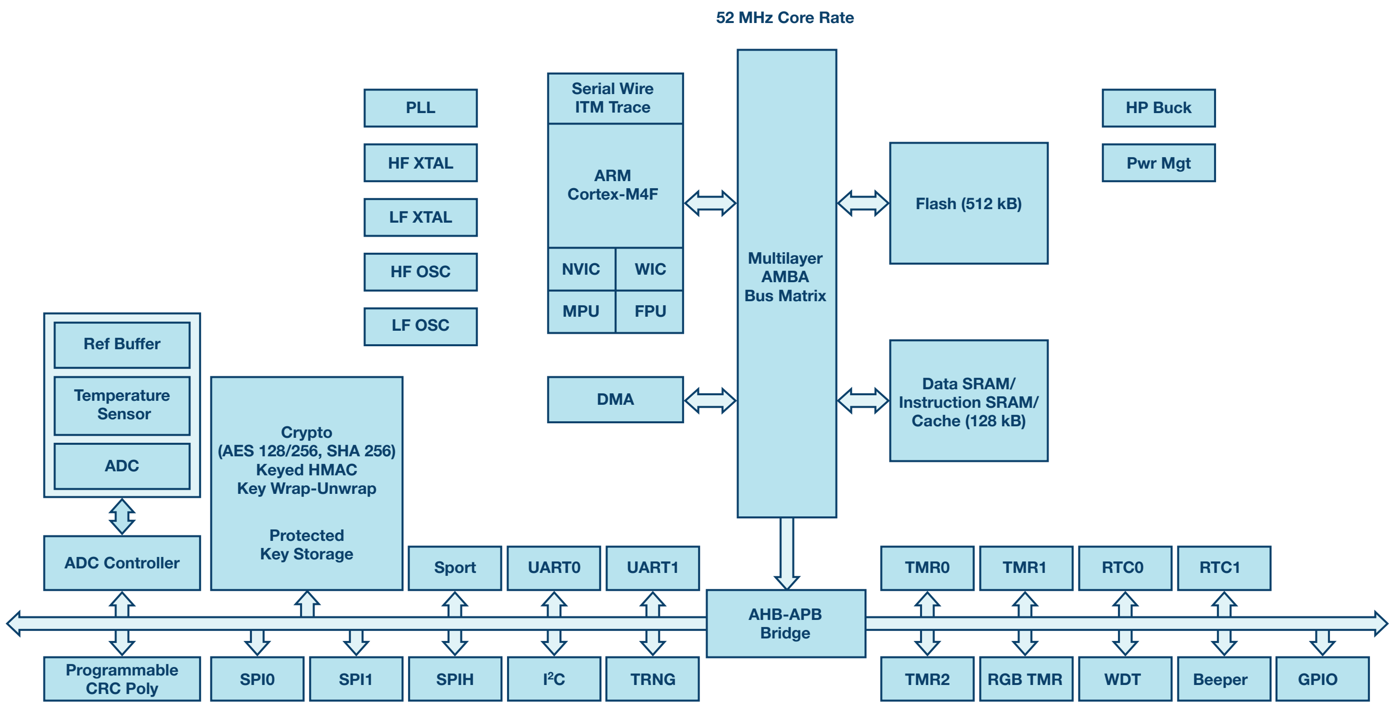


Figure 3. Block diagram of the ADuCM4050. It integrates a 1.2 V low dropout regulator (LDO) and an optional capacitive buck.

An additional consideration when utilizing a dc-to-dc converter is the type of the converter. Some converters are inductive-based, and they imply higher area, higher cost, and possible electromagnetic interference (EMI) problems. The ADuCM4050 and ADuCM302x devices use a capacitive-based converter to avoid these problems. For further information, refer to the user guide UG-1091 "How to Set Up and Use the ADuCM3027/ADuCM3029 Microcontroller."

When analyzing ULPMark-CP results or even data sheet values, it's important to acknowledge the subject of device variance. In other words, leakage current is a huge factor when it comes to measuring the energy efficiency of a device, especially in sleep mode. While traditional performance benchmarks are generally unaffected, various factors such as temperature and humidity have some degree of impact on a device's leakage current, which in turn will impact its ULPMark-CP result. In manufacturing, devices from a vendor will be different day to day or wafer to wafer. Even the energy consumption of the exact same device can vary (we've seen changes ranging from 5% to 15% depending on when and where the measurements are made). Fundamentally, this means that a given ULPMark-CP score should be used as a guideline of energy efficiency. For example, a device with a ULPMark result of 245 could range from 233 to 257 on the same device taken from a different wafer (assuming a 5% delta).

The Certification Mechanism—Establishing Credibility

In order to ensure the score's veracity, vendors willing to certify their devices send a board and tools to the EEMBC Technology Center (ETC), along with the platform specific configuration files. EEMBC integrates the platform configuration files onto their system files (that includes the workload) and measures the score multiple times in different boards. The score certified is the average of all the measurements.

In this way EEMBC ensures that the conditions are the same for all scores (same workload, same energy monitor board, similar temperature, etc.).

Figure 4 shows the connection setup to measure the ULPMark-CP on the ADuCM3029 EZ-Kit.

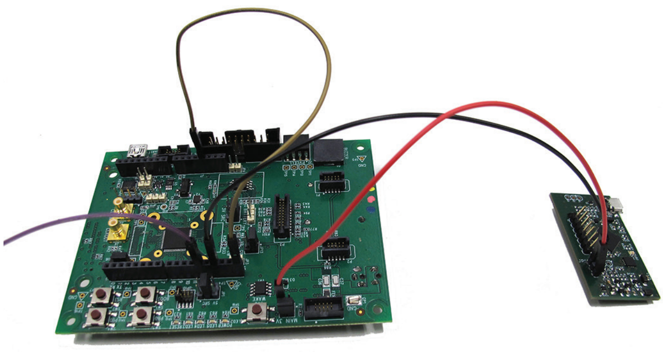
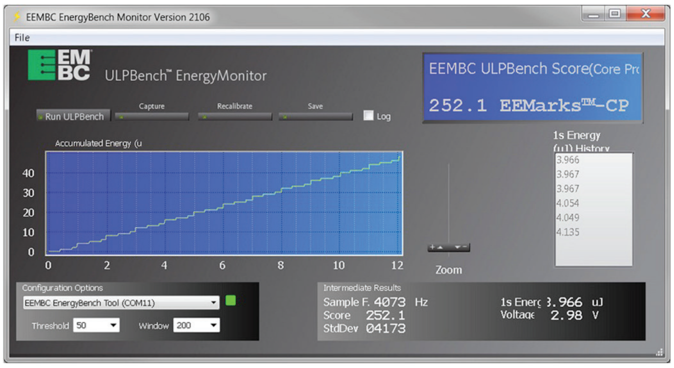


Figure 4. Board setup for measuring the score.

In order to measure the score, EEMBC provides EnergyMonitor software. By clicking the **Run ULPBench** button, the EnergyMonitor hardware powers the ADuCM3029 EZ-KIT® board and measures the energy consumption of the profile run. At the end of the execution, the software calculates the score and displays it on screen. The software also displays the average energy consumed for previous cycles in the history window.



EEMBC ULPBench Score(Core Pr
252.1 EEMarks™-CP

1s Energy (uJ) History
3.966
3.967
3.967
4.054
4.049
4.135

EEMBC EnergyBench Tool (COM11)
Threshold 50 Window 200

Sample F. 4073 Hz
Score 252.1
StdDev 04173
1s Energ 3.966 uJ
Voltage 2.98 V

Figure 5. EnergyMonitor software—GUI.

What's Next—MCU Efficiency Analysis

The ultimate EEMBC goal is to provide multiple suites of benchmarks that will allow a user to thoroughly evaluate an MCU. Beyond the ULPMark-CP, which focuses on the MCU's core efficiency, the newly released ULPMark-PeripheralProfile (ULPMark-PP) focuses on exercising various MCU peripherals, such as the ADC, PWM, SPI, and RTC. In ULPMark-PP, the active and light-sleep current consumptions are very important, as the device is executing several peripheral transactions in the workload. Results for ULPMark-PP are available from the EEMBC website; the combined ULPMark-CP and ULPMark-PP are available to EEMBC members or to license.

Next in development are the IoTMark-BLE and the SecureMark suites. The former focuses on measuring the efficiency of an MCU and radio to transmit and receive over Bluetooth®. The latter is a complex security suite that will measure energy and performance overhead of implementing various cryptographic elements for IoT devices. Both of these will be available to members and licensees before the end of the 2017.

Benchmarks are like cars—they both need people to run them. Therefore, we encourage you to encourage all MCU vendors to run and publish the results for their devices. We also need more vendors to include ULPMark results in their data sheets (similar to what vendors such as Ambiq Micro, Analog Devices, STMicroelectronics, and TI have done). This adds significantly more credibility and real-world comparability to the specifications in data sheets. If an MCU vendor doesn't publish these certified results, then you have to ask the question "Why not, are you hiding something?"

References

[1] Refer to the latest scores at www.eembc.org/ulpbench/index.php.

Monica Redon

Monica Redon [monica.redon@analog.com] joined ADI Spain in 2010. Currently she works as a systems engineer for Consumer Sensing and Processing Technology (CSPT). Previously she worked as applications lead for the IoT Platform Technology Group. Prior to ADI, she worked for 5 years in a power-line communication startup company, and for 5 years in the Wireless Network Team of the research institution Fraunhofer-Institut in Germany.