ARM Cortex-M3 core-based MCUs with ultra-low-power standby

By Jean-Michel Gril-maffre, STMicroelectronics

This article describes how the STM32 ARM Cortex-M3 core-based microcontrollers provide low power modes and features that mitigate the impacts of leakage on battery powered applications, where static current may be a major contributor to consumption.

■ Requirements for increased computing power and more integrated functions are driving a growing number of applications from 16-bit to 32-bit microcontrollers. This is true for battery-powered applications, which benefit from the lower voltage supply, as well as the high performance and small die size achieved by 32-bit devices based on advanced CMOS processes. However, deep submicron technologies also have an important drawback: their much higher leakage is a major issue, especially for the limited power resources of a battery-powered application. To enable migration, new 32-bit microcontrollers, including general-purpose devices, must provide very efficient ultra-low-power modes for long term standby. The leakage can be defined as the remaining continuous current in a CMOS gate in static state (no switching activity). It has several root causes, and increases with each new technology shrink. Its three main contributors are sub-threshold, gate, and junction tunneling leakage.

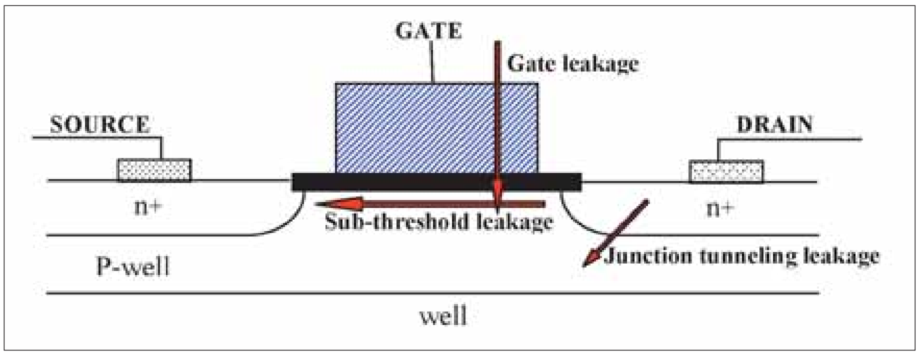
Sub-threshold leakage is linked to the threshold voltage diminution that is required with the decreasing voltages used in each new technology. Gate leakage is induced by the scaling of the gate oxide thickness that is needed to reduce the “short channel” effect. Junction tunneling leakage is induced by the electric field across a reverse biased p-n junction (tunneling of electrons). Leakage increases as temperature rises mainly due to the exponential increase of the sub-threshold leakage over temperature. Without any switching activity, the quiescent current of a 32-bit microcontroller using an advanced process can still be limited to a few µA at ambient temperature. However this leakage will rise with temperature and can even exceed 1 mA at 125°C. For this reason, it is very important to take into account the leakage at the maximum application temperature.

Even though several techniques exist to limit the leakage of a digital library (increase poly length above minimum allowed by the technology, increase gate oxide thickness on transistors), such modifications impact the propagation time in the digital cells. Using such a library in the entire core logic would prevent the device from achieving high performance in run mode. The added dilemma for 32-bit devices is that, from a structural point of view, the main contributors to leakage current in a microcontroller are digital logic and memories. In addition to the increasing leakage due to technology shrink, both the digital gate count and average memory size have increased dramatically in subsequent generations of 8-bit, 16-bit, and new 32-bit microcontrollers. As a result, leakage is a major problem for all general-purpose microcontrollers using the latest semiconductor technology, and is a particular consideration for battery-powered applications with their limited power resources. Static current consumption becomes the main contributor to average current as soon as the average runtime becomes very low compared to the standby time. Given the energy level provided by a battery, a quick estimate of the application lifetime (not including non-linearity of the battery capacitance described by the Peukert law) is:

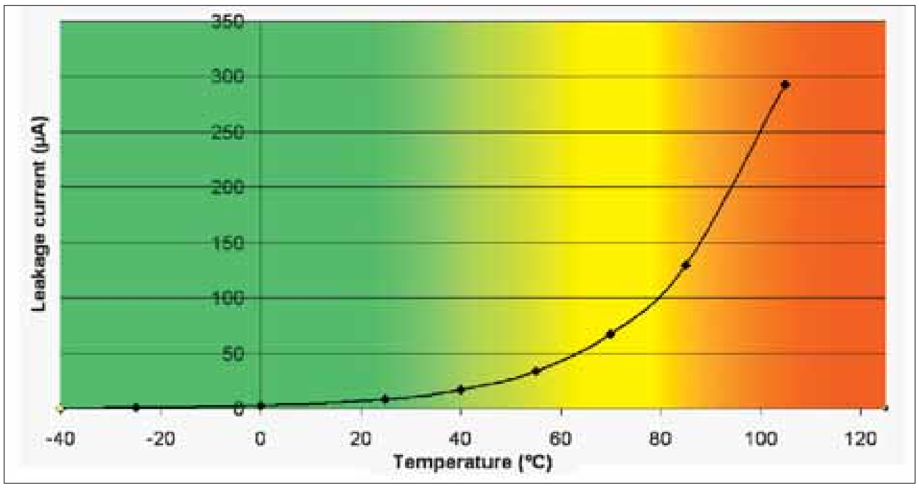
$$Tapp = \frac{Eb}{Irun - (Irun - Istdby) * Trs}$$

Irun: *run current (mA)*
Istdby: *standby current (mA)*
Eb: *Battery capacity (mA.H)*
Trs: *relative time spend in standby mode from 0 to 1*

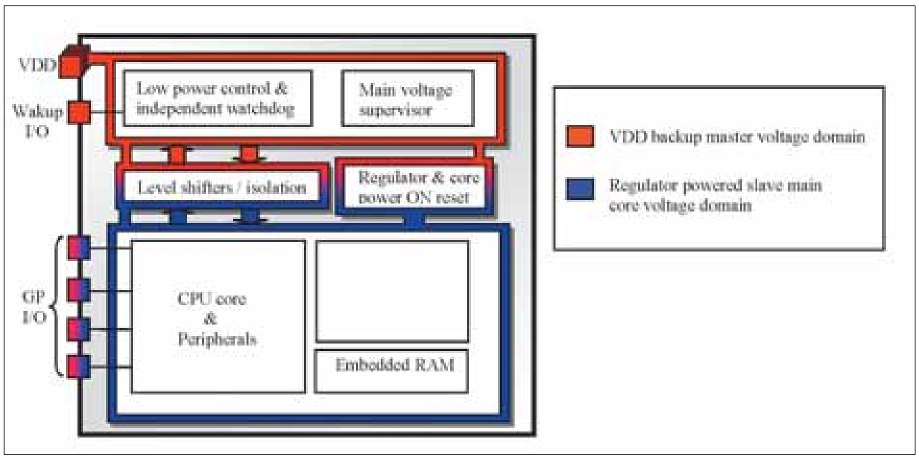
For example, if the specific ultra-low-power standby mode was not available on the STM32 128Kbyte flash microcontroller, the average current could be significantly impacted: typical run current at 72MHz with all peripherals enabled is only 36mA (0.5mA/MHz) thanks to the ARM Cortex-M3 architecture and low power design techniques. However, due to the use of advanced processes, the leakage current starts to be significant at 55°C. Thanks to a very low power voltage supervisor and regulator, static current is still limited to 50µA at 55°C. This is negligible compared to the run consumption. However if the application runs only one minute a day, the static current becomes the



Leakage currents in a CMOS transistor



Leakage increase with temperature


VDD
Wakup I/O
GP I/O
Low power control & independent watchdog
Main voltage supervisor
Level shifters / isolation
Regulator & core power ON reset
CPU core & Peripherals
Embedded RAM
VDD backup master voltage domain
Regulator powered slave main core voltage domain

Backup and core voltage implementation

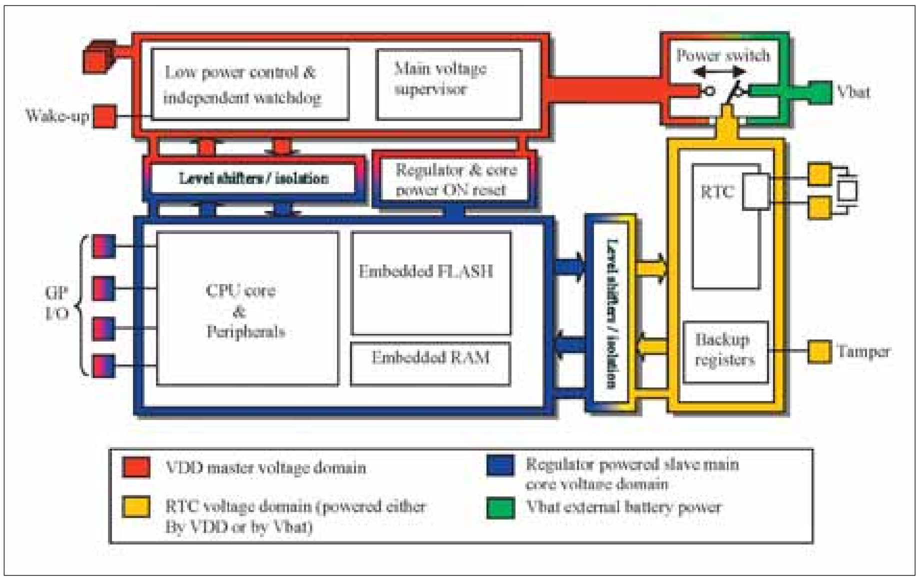
main contributor to consumption (64%). To address this problem, the designers of the STM32 went to great lengths to enable a true low-power standby by implementing an embedded regulator, independent voltage domains, and integrated power switches at the architecture level. This implementation enables low-power modes that can optimise battery life depending on the application.

The total consumption of a microcontroller is the addition of dynamic power (switching activity of CMOS gates) and static current (leakage and static analog consumption). Stopping the product clocks, thereby eliminating all dynamic consumption is obviously not a sufficient low-power standby for a battery-powered application where static current can be a main contributor to consumption. Even decreasing the core voltage when clocks are stopped is of little help in providing an efficient standby mode. To achieve ultra-low-power standby mode, most of the core logic (and memories) must be powered off. To do this, two voltage domains can be implemented at device level, which can be powered by the internal regulator: a small "always on" voltage domain for low-power control, and a "main core" voltage domain with all other functions powered through a switch in order to shut it down in standby mode. As a result, the main core voltage domain can be focussed on processing performance as the leakage (static current) design constraints are mainly important in the "always on" voltage domain.

However, in this implementation the internal regulator must be kept "on" in standby mode, implying a significant quiescent current. For this reason, it is better to stop the embedded regulator in order to reach an ultra-low standby supply current. The STM32 follows this type of dual-domain implementation with VDD backup master voltage domain and slave main core voltage domain. VDD backup master voltage domain is based on thick oxide high voltage transistors focusing on very low static current. With the high voltage transistors, it is directly powered by the main VDD. It includes low-power mode control and a very low-power watchdog, associated low-power RC oscillator and an optimised gate count. Slave main core voltage domain includes all other functions (CPU core, most peripherals, and memories) kept at lower voltage, focusing on high performance and low dynamic power. This implementation allows the STM32F103 to offer a safe, very low-power standby mode with only a 2µA typical current at 3.3V. The remaining 2µA current is the consumption of the accurate voltage supervisor that monitors the main supply voltage to ensure that the standby mode is as reliable as the run mode. Since leakage can be kept very low, increase of this standby current with temperature is very limited: 2.4µA at 85°C 3.3V for a typical device.

Dynamic functions can also be implemented in the master voltage domain. For example, the STM32 includes an independent ultra-low-power watchdog that is available in standby mode with a total added consumption (dedicated RC oscillator and watchdog digital consumption) of only 1µA at 3.3V for a typical device. This feature can prevent application failure in case of an unexpected entry in standby mode.

Separating the voltage domains inside the microcontroller silicon implies many specific design constraints. Full wake-up logic and analog must be implemented in the backup voltage domain making it difficult to offer a large number of possible wake-up sources. Isolation between voltage domains during power down must be implemented (all signals coming from core voltage are floating). The specific sequence for clock source stops and voltage power down/power on must be robust. The main core voltage logic needs a dedicated reset for example. Timing constraints between volt-


Wake-up
Low power control & independent watchdog
Main voltage supervisor
Power switch
Vbat
Level shifters / isolation
Regulator & core power ON reset
RTC
GP I/O
CPU core & Peripherals
Embedded FLASH
Embedded RAM
Level shifters / isolation
Backup registers
Tamper
VDD master voltage domain
Regulator powered slave main core voltage domain
RTC voltage domain (powered either By VDD or by Vbat)
Vbat external battery power

Simplified schematic of STM32 voltage domains implementation

age domains must be taken into account specifically because both domains are nearly independent for voltage and process, but not for temperature. This implies that more cases have to be checked during timing analysis (backup domain with worst voltage and process and main core voltage with best process best voltage for example).

Some security features like watchdog function must be implemented in the backup domain in order to protect the application from unexpected standby mode entry. Keeping the ratio of useful I/Os versus total number of I/Os is also required to offer the performance of a 32-bit product in small packages. On the STM32, the main core voltage regulator does not need external decoupling capacitors. This is why no extra power pin is lost on the package because of this dual power implementation. However, in exchange for this silicon design complexity, the STM32 gains a true ultra-low-power standby that will help application developers optimise battery consumption in their applications. As a result of the dual power domain implementation, STM32 provides two different low power modes: stop mode and standby mode. Both function with the voltage supervisor "on" to protect the application in case of a voltage drop. In stop mode, the low-power regulator is kept on but clocks are stopped. It provides very fast restart time on internal RC (<10µS) and retains the software context. Typical current at ambient temperature is 15µA (3.3V). However this mode does not mitigate the problem of leakage, which increases exponentially with temperature. In standby mode the regulator is off in order to provide a 2µA current at ambient temperature (3.3V) and very little increase with temperature increase (2.4µA at 85°C for a typical device). However, restart from standby implies that software content is lost: RAM, core and most peripheral register contents are lost. Restart from standby is nearly equivalent to a restart from reset for the software.

Choosing the best mode for an application can have a large impact on battery life. Here are some basic tips to consider when selecting a mode: 1. Check if the microcontroller state in standby is compatible with application requirements (for example: I/Os standby state, wake-up sources). 2. Consider the impact on battery life of the "worst case" temperature conditions under which application functionality must be guaranteed. 3. Check what the restart from standby time is, and if it is fast enough for the application restart time requirements. 4. Check if there is a saving in energy consumption in standby compared to stop mode. Between two events, is the standby consumption plus the restart from standby consumption less than the consumption in stop mode.

These questions are application-dependent. Estimating the restart time from standby mode includes the time from wake up to reset vector fetch, which depends on hardware (regulator startup time, clock source startup time around 40µS in STM32) and the time needed by the software to restore the application context. Typically the software must check the wake-up source(s), recover context information from backup registers and re-configure the microcontroller functions used by the application. Because of this software- dependent restart from standby, the energy lost during this wake-up phase is also application-dependent. One practical way to estimate this energy loss is to produce a given amount of wake-ups in a time frame (software going back in standby mode just after the wake-up) and compare the average current consumption when no wake-up is generated.

In order to optimise the restart time from standby mode, the developer must not forget to optimise the initialisation phase added by the compiler and reduce it as much as possible (RAM initialisation should be removed for example). The real time clock feature is a common requirement for battery- powered applications. Moreover, core voltage shut off implies losing the complete program context and is nearly equivalent to a product restart from reset. Implementing a backup register bank for application restart allows the recovery of minimum context required for program execution.

Integrating these functions directly inside the microcontroller can be done in a backup domain. However, the RTC function is typically supposed to be available over an extended period of time (years) while the main application, even if battery-powered, is often based on a rechargeable battery. Creating a third power domain for the RTC and offering a dedicated pin for its power supply allows the use of a small coin cell dedicated only to this function, while the main application is supplied by another main supply source. This way the coin cell power is only used by the RTC and associated oscillator, and not by the other functions, such as the voltage supervisor which is still available in standby mode. However, this implementation is not optimal as the coin cell is always used to provide the power to the RTC and backup registers even when the main power supply is available. A smart alternative that is implemented in the STM32 is to extend RTC battery life by adding a power switch to provide current to the RTC and backup registers from the main supply when it is available and from the battery when the main power is not available. The switch command is provided by the main voltage supervisor with a specific latching mechanism. When the voltage drops below the VDD low threshold, the switch changes the RTC and backup registers power source to external VBAT power. If VDD rises above the VDD high threshold, the switch automatically selects VDD as the power source for this dedicated voltage domain.

One additional advantage with this implementation is that extra dynamic power consumption resulting from software read/write access to this specific voltage domain (through level shifters) never implies extra consumption on the coin cell. In run mode current is always taken from the main supply. Thus, coin cell minimum battery life can be directly calculated based on RTC consumption and the coin cell energy. On STM32 with a typical RTC current of 1.4µA (ambient temperature 3.3V) the minimum battery lifetime when using a CR2032 battery is close to 20 years. However, if the main power is present most of the time, the life can be much longer and a coin cell with smaller capacity can be used. The implemented RTC and backup registers are, of course, available in standby mode. Thus, the RTC can be the source of the wake-up from standby and some key values can be saved in the backup registers before entering standby mode. This implementation significantly increases the complexity of the microcontroller design. It requires more complex isolation between voltage domains; robust power switch design, correctly adjusted to expected consumption (internal RTC voltage domain is not present on I/Os to avoid reducing the number of general purpose I/Os available in small packages, so no decoupling capacitor can be added). It also requires consideration of different startup scenarios without added static consumption on Vbat. For example Vbat rising when VDD is not present must not lead to an unexpected state (there must be no consumption in this state because the coin cell may be soldered to the application during production phase and consumption would cause an unnecessary depletion of its energy level). The RTC voltage domain must be designed to tolerate a significant voltage drop below the VDD minimum threshold before switching on Vbat.

In spite of some new application considerations linked to context loss in a standby state, ultra-low-power standby and multi-voltage power architectures like that of the STM32 can be effective solutions that allow an application to function in a high-performance run mode, while mitigating the impact of static consumption in standby mode. In addition, thoughtful integration of standalone functions, like the RTC in the STM32, can enable fast and efficient development of battery-powered applications and optimum use of application power supplies. ■

For more information please go to www.st.com/stm32