CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

SILICON UPDATE

by Tom Cantrell

More Than a Core

While examining 32-bit microcontrollers last month, Tom decided that the STMicroelectronics STM32 was worth a second look. With the new ARM Cortex M3 core, good peripherals, integration, and energy efficiency, this could be just the MCU for your next project.

Having covered the territory last month ("More Bits, Less Filling," *Circuit Cellar* 212, 2008), it's not my intention to get stuck on the topic of 32-bit MCUs. Believe me, there's plenty of other neat stuff going on with FPGAs, wireless, sensors, and other wonders of the silicon age. Nevertheless, if you have anything to do with embedded systems, you need to stay up to speed with the latest hot rod chips or you'll get left behind.

In some ways these fast and furious MCUs remind me of the brand new Tesla Motors high-performance electric vehicle just now hitting the streets. It's got the efficiency and green aspects of a golf cart, but can smoke the tires when you punch it. The big difference is that the 32-bit MCUs don't cost an arm and a leg, but in fact are a luxury any designer can afford.

So this month, you're invited to look over my shoulder as I pop the hood on the STMicroelectronics STM32 (see Figure 1). You'll recall from last time that its main claim to fame is the use of the new ARM Cortex M3 core. Sure, that's newsworthy, but there's more to the STM32 than that.

WORLD BEYOND CORE

Indeed, over the years, I've come to the conclusion that "core wars" debates have become less relevant, especially for blue-collar embedded apps. Maybe it's just battle fatigue, having seen so many architectures march off to war. Remember way back in the mainframe years (1960–1970s) when companies like Univac, Burroughs, and Honeywell challenged IBM with "better" architectures? All dead and gone.

Then there were the fabulous minicomputers such as the Data General Nova and the Digital Equipment VAX. Like teenagers, they seemed invincible. "Nova" indeed. Who would have thought these shining stars would burn out so fast?

The microprocessor was barely born before it headed into battle. Early 8-bit skirmishes foreshadowed the epic struggle between the Intel 'x86 and the then Motorola 68K, a battle that counted a myriad of upstart architectures as collateral casualties. May the 88K, i860, Clipper, 29K, and all of the others rest in peace.

True believers are entitled to pitch their favorite architecture and poo-poo the others. Taking nothing away from Cortex M3, the fact is that all of the

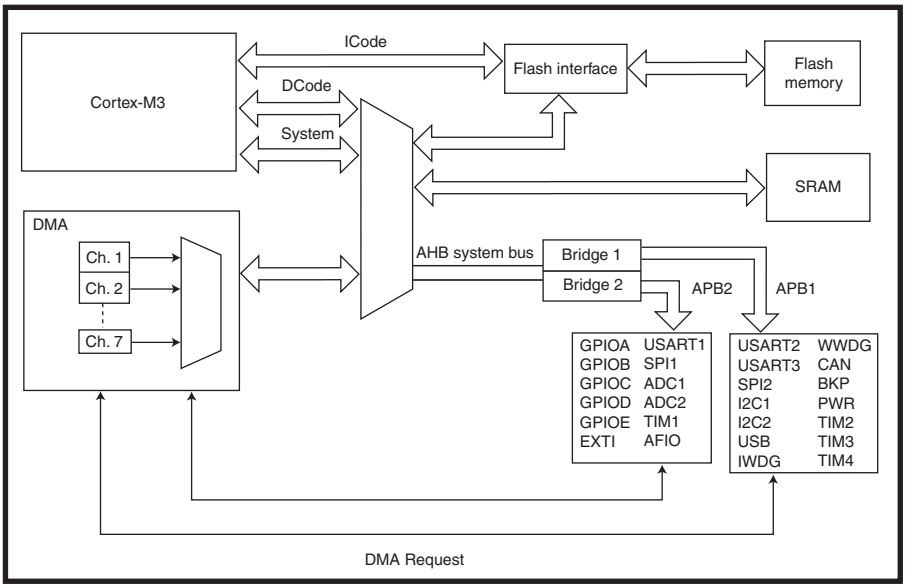

Figure 1—The ARM Cortex M3 core is the attention-getter in the new STM32 MCU from STMicroelectronics. But there's more to an MCU than a processor core, including lots of flash memory, fast SRAM, and a bunch of I/O.

Package pins	36	36	48	48	48	64	64	64	100	100
Flash	32 KB	64 KB	32 KB	64 KB	128 KB	32 KB	64 KB	128 KB	64 KB	128 KB
SRAM	10 (6) KB	20 (10) KB	10 (6) KB	20 (10) KB	20 (16) KB	10 (6) KB	20 (10) KB	20 (16) KB	20 (10) KB	20 (16) KB
General-purpose timers	2	3	2	3	3	2	3	3	3	3
Advanced control timer	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)
SPI	1	1	1	2	2	1	2	2	2	2
I²C	1	1	1	2	2	1	2	2	2	2
USART	2	2	2	3	3	2	3	3	3	3
Full-speed USB 2.0	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)
CAN 2.0B	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)
12-bit 1-µs A/D	2 (1) × 10 ch	2 (1) × 10 ch	2 (1) × 10 ch	2 (1) × 10 ch	2 (1) × 10 ch	2 (1) × 16 ch	2 (1) × 16 ch	2 (1) × 16 ch	2 (1) × 16 ch	2 (1) × 16 ch
General-purpose I/Os	26	26	37	37	37	51	51	51	80	80
CPU Frequency	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz	72 (36) MHz

Table 1—STMicroelectronics blasts off the starting line with a full complement of 20 STM32 parts, divided equally between "Performance" and "Access" lines. In this table, the "Access" line features are shown in parenthesis where they differ from the "Performance" line. Another difference is that both lines come standard with a –40° to 85°C temperature range, but the "Performance" parts also have an extended temperature range (–40° to 105°C) option.

major 32-bit MCUs (including the ARM7 and ARM9 chips ST also offers) are fully capable of getting the job done in most applications.

Look at a die photo of any 32-bit flash MCU and what you'll find is a little processor core stuck in the corner, dwarfed by surrounding memory and I/O silicon. The fact is, while the architecture chosen for the core may be the sizzle, it's the implementation of an entire chip that's the steak.

FLASH FOR CASH

Sure architecture has an impact on performance, but so do a lot of other things starting with bus bandwidth. The differences (relatively minor actually) in the way competing architectures choose to deal with instructions and data don't matter nearly as much as how fast a particular chip can actually do it.

In the blue-collar space these chips target, we're generally talking about non-cache implementations. That means flash (i.e., instruction fetch) bandwidth is a critical limiting factor. The STM32 comes in two flavors, "Access" and "Performance," with a major difference being that the former runs up to 36 MHz and the latter to 72 MHz (see Table 1). Just keep in mind that higher clock rates require 0, 1, or 2 flash wait states for clock rates up to 24, 48, and 72 MHz, respectively.

If something isn't done, wait states can lead to the awkward situation where more "megahertz" means less performance. It's no surprise that most 32-bit MCUs devote silicon to the cause of getting around the flash bottleneck. The STM32 is no exception, using a 64-bit wide flash bus in conjunction with two 64-bit prefetch buffers to hide the flash latency. Even though this simple prefetch scheme is relatively unobtrusive, there may be times when you'd prefer to turn it off, which the STM32 allows you to do.

If you really need max MIPS, take advantage of the fact that the STM32 allows execution of code from the on-chip SRAM at full speed. You can use the SRAM as a "programmer directed cache," preloading it with performance-critical routines such as DSP inner loops and interrupt handlers.

Just remember that a MIPS rating is only half the story. You can crank through all of the instructions you want, but nothing useful happens until data makes its way to and from the pins. As a practical matter, the on-chip I/O devices are just as important as the processor core itself in achieving peak system performance.

I/O U

I/O throughput starts with the number and performance of the on-chip I/O devices themselves. The STM32 has a lot of fast I/O, but that can actually be a curse if the I/O traffic clogs available bus bandwidth and demands a lot of handholding by the processor. The STM32 avoids that pitfall with multiple on-chip I/O busses to boost bandwidth and a powerful seven-channel DMA controller that offloads the processor of I/O grunt work.

Another way to boost bus bandwidth is to demand less of it in the first place. As I went through the specs, I was impressed with the way the STM32 uses "smart" I/O devices that take care of their own dirty laundry rather than bugging the processor to do it for them.

Even the simple stuff such as serial and parallel I/O is pretty fancy these days. Every STM32 I/O line is individually programmable as input (pull-up and pull-down options) or output (push/pull or open collector with output drive strength options). I/O lines are also 5-V tolerant and can source/sink a whopping 25 mA, with the not unexpected caveat that total chip power is limited to 150 mA. A measure of port-remapping capability enables juggling peripheral pin assignments to best fit a particular application.

As I've noted before, the traditional RISC load/store architecture is problematic for "atomic" bit operations because an interrupt might occur between the load and the store. The

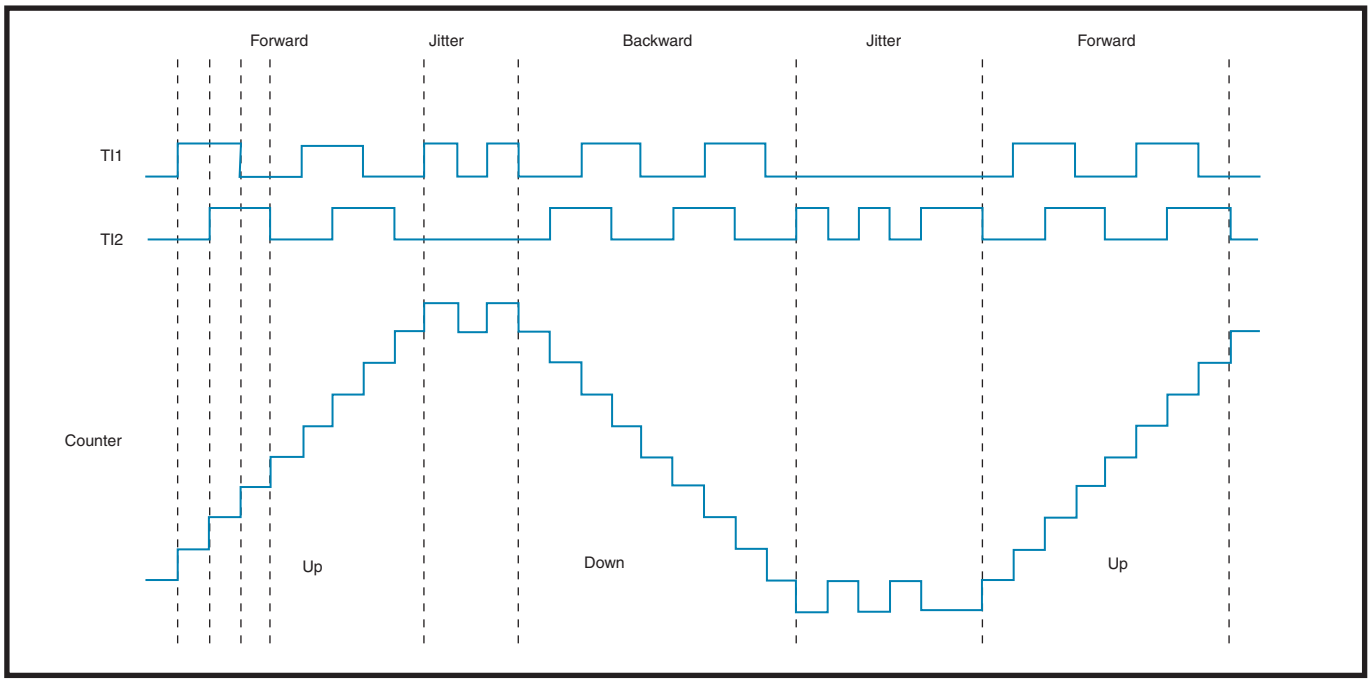

Figure 2—Smart timers are needed to enable real-time applications to handle tasks in hardware that would otherwise bog down the processor core. The Encoder mode of the Advanced Control Timer (ACT) included in STM32 "Performance" parts is a good example. It automatically monitors the phase relationship of two inputs and keeps track of the cumulative count.

Cortex M3 architecture takes a crack at the problem with a "bit-banding" capability that provides atomic access to single bits. In addition, the STM32 also incorporates "set/reset" shadow registers for I/O, a solution that has the advantage of being able to deal with multiple bits at a time.

In safety-critical applications (e.g., transportation, medical, and industrial), a single lowly I/O line can have life and death riding on its shoulders. The STM32 has a unique capability to "lock" the configuration of an I/O line against unintended reprogramming to help keep a software crash from leading to a real one.

Moving on to serial I/O, every STM32 includes a SPI port, an I²C port, and two USARTs while the larger parts add an extra one of each. That's a total of up to seven fast and full-featured serial ports, quite impressive for an entry-level part.

The SPI ports run at up to 18 MHz as master or slave in half- or full-duplex mode. Besides the usual options (clock rate, mode, 8- to 16-bit frame), there's hardware that takes care of the CRC for flash cards (e.g., SD Card). Likewise, the I²C port handles different modes (e.g., Slave, Multi-Master), speeds (standard and fast), and standards (e.g., SM Bus 2.0). No surprise that the USARTs are fast (up to 4.5 Mbps) and capable (e.g., LIN, IrDA) as well. Note that any or all of these serial I/Os work with the DMAC, taking advantage of its intelligence (e.g., 8-, 16-, and 32-bit bus matching, circular buffer manager), which leaves the processor free for more important tasks.

The "Performance" parts include USB 2.0 (full-speed, 12.0 Mbps) and CAN interfaces. This seems like a rather unlikely pairing and indeed the datasheet reveals that you can really only use one function at a time (they share the use of a 512-byte buffer). Once again, you'll find that these interfaces have the "smart" features that make life easier for the processor and programmer. For instance, the CAN controller has programmable message filters so it can screen message traffic by itself without bothering the processor.

If you want to do real-time, you need plenty of timers. General housekeeping is handled with an RTC, a free-running "SysTick" counter, and two separate watchdog timers, while three 16-bit units with input capture, output compare, and PWM do the heavy lifting. "Performance" parts go even further by throwing in an "Advanced Control Timer" that has even more bells and whistles (see Figure 2).

Analog capability is another difference between the two STM32 lines. The "Access" parts include one converter while the "Performance" line has two converters with the simultaneous sampling capability required for many applications (e.g., motor control

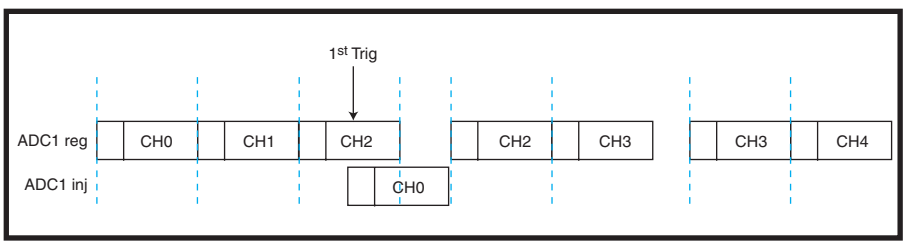

Figure 3—Automatic scanning of a group of analog inputs is a common feature in modern ADCs. The STM32 takes the concept a step further with the ability to interrupt one group scan by "injecting" another.

and power factor correction). While the basic converter specs (12 bits, 1 µs, up to 16 channels) are competitive, it's the sophisticated CPU cycle-savers that set this ADC apart from most.

Many ADCs include a "scan" capability to automatically convert a sequence of channels. The STM32 takes it to the next level by adding a second scan group that can be "injected" into (i.e., interrupt) the regular scan (see Figure 3). An "analog watchdog" capability provides independent threshold comparison for any/all pins in either the regular or injected scan groups, or both.

Above and beyond their individual capabilities, the timers, ADC(s), and DMAC can work together to handle high-speed timing critical tasks in hardware. Purists will argue that no MCU can match a DSP or specialized chip for applications like motor control, but I bet the STM32 might surprise them.

REALITY SHOW

There is no doubt that the processor and peripherals are the attention-getters for any MCU. But there are also a lot of nuts and bolts required to lash together a real-world design. Some particular little piece of "glue logic" may seem insignificant, until you need it and it's not there. Then all of a sudden it's a big deal with the potential to complicate the design or otherwise compromise the application.

Traditional RISCs, reflecting their "computer" (versus "controller") background, can be pretty lame when it comes to interrupts, but not so for the STM32. In addition to the Cortex M3 architectural improvements (e.g., built-in vectored interrupt controller and "tail-chaining" to minimize stack operations), the STM32 includes dedicated hardware to configure up to 19 I/O lines as external interrupt/event inputs.

While it sometimes seems that all of the focus is on MIPS and megahertz, there is also the small matter of power consumption. "Small matter"

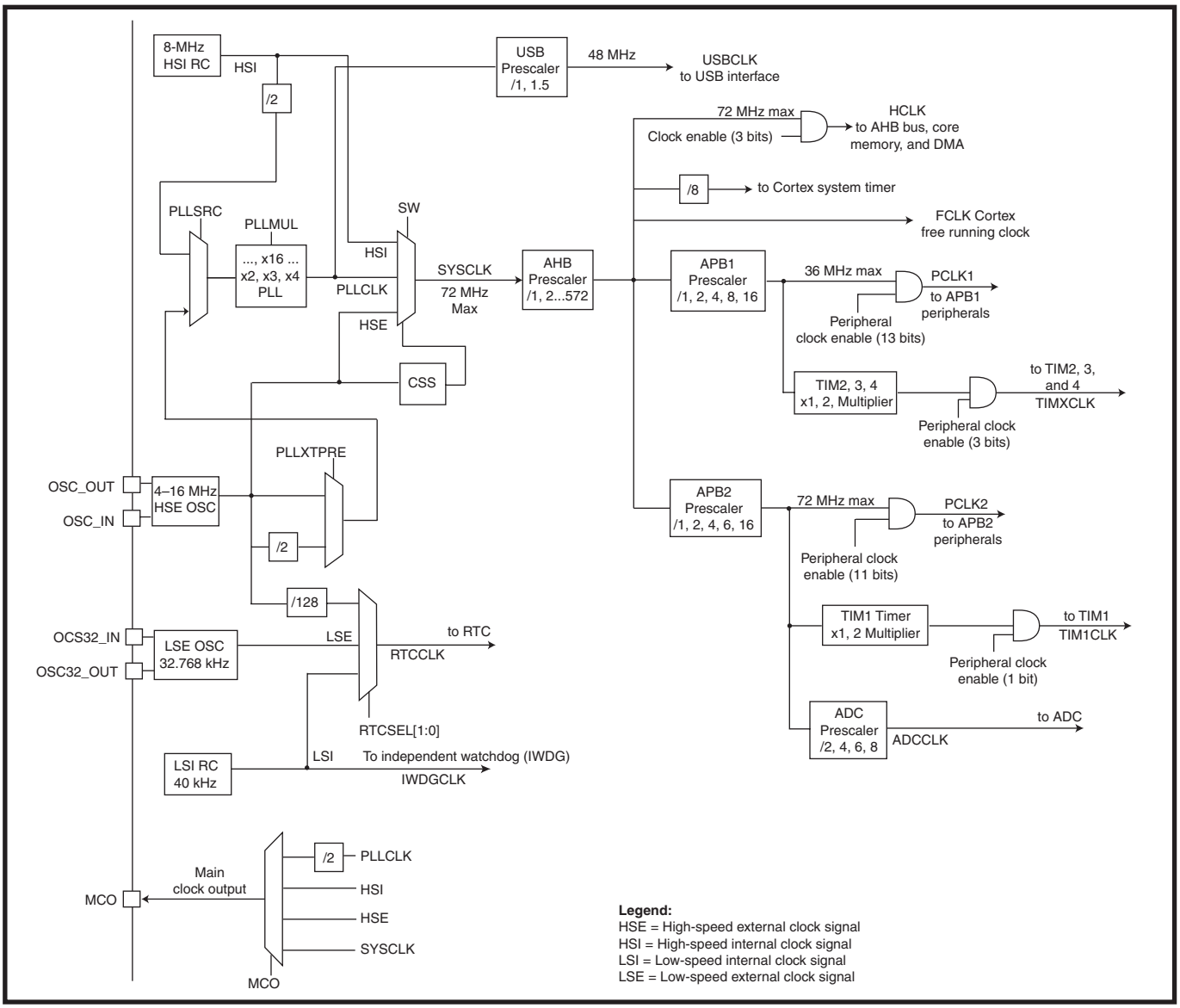

Figure 4—Some may consider it mere "glue logic," but the clock generator on a modern MCU such as the STM32 plays a critical role in achieving system price, power, and performance goals.

Photo 1—Drape this gadget around your neck and you'll be the life of the party! A good MCU needs a good starter kit and those provided by the likes of Raisonance (the STM32 primer shown here), Keil, IAR Systems, and Hitex Development Tools make it easy and inexpensive to check out the new STM32 MCU.

as in your design had better consume a small amount of power, or else. After all, a main claim to fame for all of the new-age 32-bit MCUs is that they can go head-to-head with 8-bit parts and that means battery-powered applications.

Powering the chip couldn't be simpler. Just hook it up to anything from 2 to 3.6 V and it springs to life. An on-chip regulator supplies 1.8 V internally while power-up/power-fail RESET and over- and under-voltage interrupts are built-in.

The ADC features a precise on-chip 1.2-V reference voltage, but you can connect an external reference if you wish (noting that using the ADC boosts the minimum required chip voltage from 2 to 2.4 V). Finally, just hang a 1.8- to 3.6-V battery on the VBAT supply pins if you want to take advantage of the RTC and related backup features. Switchover between the primary and battery backup supplies is handled automatically on-chip.

Besides the RTC, VBAT also provides power for 10 16-bit "backup" registers (i.e., RAM). A unique protection option automatically clears the contents of these registers if "tampering" (i.e., unexpected activity on a pin) is detected.

More evidence that the STM32 takes the nuts and bolts seriously is the clock generator (see Figure 4). Make that clock(s) generator(s). This chip's got so many clocking options I thought I was in Switzerland.

The primary 8-MHz oscillator (factory trimmed for accuracy) drives a PLL to generate the myriad of high-frequency clocks required for the processor and peripherals. Alternatively, you can provide an external 4- to 16-MHz clock, in which case the internal clock serves as a monitor and backup should the external clock fail.

There's a separate low-speed (40-kHz) clock that's powered from the VBAT backup power supply. It's not accurate enough for real time, but it does fill the key role of providing an on-chip wakeup source when the MCU core (i.e., 1.8-V domain) is powered down.

And while better than nothing, a single watchdog timer always raises the question of who will watch the watchers? Taking advantage of the additional clock, the STM32 integrates two independent watchdog timers for a level of protection only true redundancy provides.

Together the power and clock systems give you a lot of power-saving options. Embellishments to a trio of low-power modes (Sleep, Stop, and Standby) include the ability to tweak various dials on the clock generator and the voltage regulator (run, power down, off). The lowest power mode (Standby) takes advantage of the separate backup supply domain to shut primary power off yet retain the ability to wake up from an RTC alarm or the independent watchdog.

And just how low power are we talking? According to the datasheet, even running full bore at 72 MHz with all peripherals enabled, you're looking at just 0.5 mA per 1 MHz typical (i.e., 36 mA at 72 MHz at room temperature). And here's another reason to put your most frequently executed routines in RAM: not only is it fast (zero wait states), but running code from RAM also consumes less than half the power (e.g., 14.4 mA at 72 MHz) of running code from flash memory. Another power-saving trick

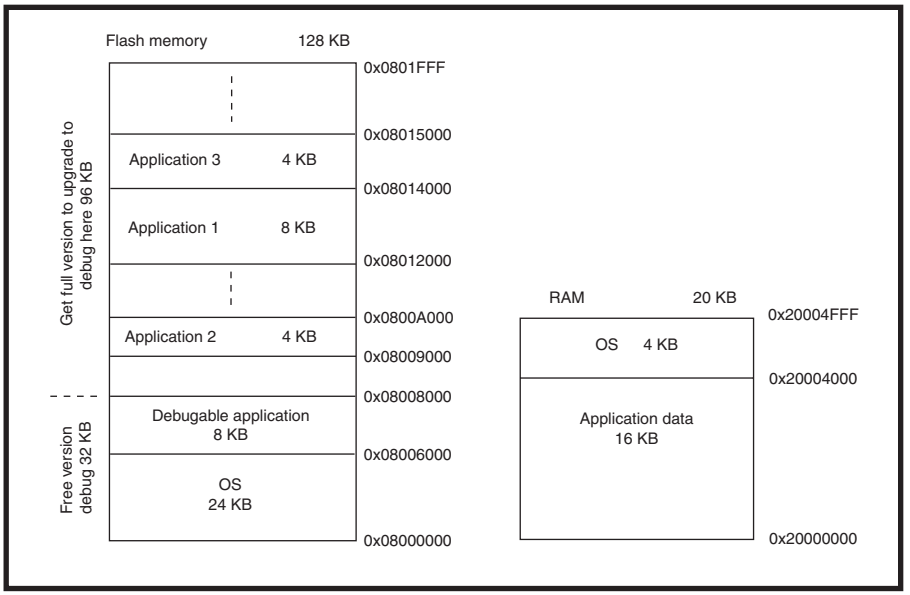

Figure 5—The STM32 primer may look like a toy, but under the hood is a "Circle OS" that supports application development and experimentation. There's plenty of room in the STM32 on-chip flash and SRAM for both "Circle OS" and application code and data.

Photo 2—Small is beautiful, except when it comes to hand-wiring a tiny surface-mount chip. The STM32-H103 header board from Olimex makes it quick and easy to prototype your own STM32-based design.

is to take advantage of the fact that every peripheral has its own power switch (i.e., clock gate) and the datasheet helpfully itemizes the power consumption of each. The savings can add up considering the higher-power peripherals (e.g., timers and ADCs) consume a milliamp or two each.

Beyond active power consumption, low-power modes are where batteries live and die. The STM32 Sleep mode cuts power consumption roughly in half yet remains functional enough (i.e., many fast wakeup options) to use routinely. Taking a big step down the ladder, Stop mode specs at just 15 to 25 µA or so depending on the particulars (e.g., voltage regulator on/off, temperature). That's not bad considering the on-chip RAM is kept alive and it's relatively easy to wake up (e.g., via pin, interrupt, USB). If you don't need to preserve the contents of RAM, Standby mode slashes power to little more than 1 µA, yet still gives you some tools to work with besides just RESET (e.g., RTC wakeup alarm and the backup registers).

ONE LAST THING

I think you can see that the STM32 is firing on all cylinders (i.e., good core, good peripherals, good integration, and good energy efficiency). Guess what? A good chip is useless unless it's got some good tools to go with it. Fortunately, the STM32 gets to ride on the ARM bandwagon, which is standing room only with third-party tool suppliers including ARM and Keil (owned by ARM), Raisonance, IAR Systems, and Hitex Development Tools with no doubt more to come.

I got a chance to play around with the cute little "STM32 Primer" gadget courtesy of STMicroelectronics and Raisonance. Although the evaluation version of the Raisonance RIDE7 toolchain (GNU-based) that comes with the primer is limited to debugging 32 KB (a full-function upgrade is available from Raisonance) at just $32, the kit is still quite a bargain.

A close look reveals two MCUs (see Photo 1). At the top is the STM32 of interest, a 128-KB flash unit. On the left is an ARM7 MCU acting as a debug interface between your PC USB port and the STM32 software/JTAG debug pins. A benefit of the two-chip approach is that it leaves the STM32 USB port free for application use.

In the upper left is a part that raises the primer's fun quotient, a three-axis low-g MEMS accelerometer enabling a "tilt-o-whirl" user interface. Scrolling and menu selection is accomplished by tilting the gadget. The display automatically switches between Portrait and Landscape mode depending on orientation.

Taking advantage of the accelerometer, the Primer comes preprogrammed with some simple maze and breakout games. But it's more than a toy. Indeed, under the hood is a "Circle OS" that includes a simple task scheduler and a variety of I/O libraries for both the STM32 on-chip peripherals and the primer add-ons (MEMS accelerometer, graphics LCD, button, buzzer, and more) (see Figure 5). You can find the source code for Circle OS and example applications at www.stm32circle.com. The primer documentation walked me through the process of creating my own "Hello World" application in a matter of minutes, and everything worked without a hitch.

Rolling your own prototype is another option, but not always an easy one with fine-pitch surface-mount parts. Olimex provides a handy solution with a "header board" that includes the STM32 MCU, a USB connector, and easy access via standard headers to the chip's I/O lines (see Photo 2).

MOST SMARTEST MCU

In the reality show, that's the MCU business: the STM32 is more than a pretty face. Behind the looks of a flashy new core is a down-to-earth chip that's sophisticated, but not fragile or high maintenance.

And this is a supermodel that's accessible to mere mortals. Judging from all of the promotion commotion and third-party support, it is clear that STMicroelectronics is serious about going after the mass-MCU market, not just a few big-ticket focus customers. Wise move, because staying power in the MCU business is as much a matter of seats (i.e., number of designs) as sockets.

Is the STM32 the "best" 32-bit MCU? Who knows, and who cares? What matters is that it is a great MCU that leverages an entire ecosystem of chips, tools, and third-party support. Bottom line for designers shopping 32-bit MCUs? If you've got a short list of favorites, it just got a little longer.

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

SOURCES

Cortex M3 core
ARM
www.arm.com

STM32 Development tools
Hitex Development Tools
www.hitex.com

STM32 Development tools
IAR Systems
www.iar.com

STM32 Development tools
Keil
www.keil.com

STM32 Evaluation boards
Olimex
www.olimex.com

STM32 Development tools
Raisonance
www.raisonance.com

STM32 Cortex M3-based 32-bit flash microcontroller
STMicroelectronics
www.st.com