



ARMed and Dangerous

A lot has changed since Tom started covering ARM microcontrollers back in 2003. This month he brings you up to speed on the new parts and players in the ARM MCU world.

I first covered the ARM MCU trend back in 2003 when it was just getting started (“In ARM’s Way,” *Circuit Cellar* 159). Now, almost three years later, the bandwagon is really rolling. A popular catch phrase is that ARM is the 32-bit equivalent of the ‘51 of tomorrow, reflecting that venerable 8-bit MCU’s immortality. From a designer’s perspective, the fact that both ARM and the ‘51 are supported by multiple vendors differentiates them from other choices.

The multi-source advantage encourages a degree of competition and innovation that single-sourced architectures are challenged to match. That’s not to say competitors have any intention of throwing in the towel. The ‘51 may be popular, but by no means does it “own” the 8-bit market. The same goes for the 32-bit MCU space, where ARM-based parts face ongoing competition from both historic (e.g., 68K/Coldfire, PowerPC, and Renesas SH) and even new (e.g., Atmel AVR32) contenders.

But for this month, the proliferation of ARM-based MCUs easily fills an entire column. There are new parts, and new players, to catch up with, so let’s get started.

1 MEG OR BUST

The proliferation of ARM7 MCUs with 1 MB (and more) of memory represents kind of a milestone. After all, that’s more memory than the first generation of PCs had in the ‘80s, not to mention the “big iron” (and “big bucks”) mainframes of yore.

Of course, today’s “computer” demands gigs, not megs, of memory. But in the deeply embedded space, 1 MB

still goes a long way. That’s enough room for big “C” programs, an RTOS and network stack, and even some extras like HTML graphics, audio prompts, and so on. Some of the suppliers of big flash parts are using stacked-die, while others are monolithic. To tell the truth, I have trouble keeping them straight, and I’m not sure it matters.

Philips has staked out a strong position with mini-me (low pin count, small memory) LPC21xx ARM7 MCUs. Now they turn their attention to the high-end with the LPC2888, which includes 1-MB flash memory and 64-KB RAM. With built-in hi-

speed (480 Mbps) USB (including the PHY) and a variety of interfaces (external memory, MultiMedia Card, audio, and analog), the LPC2888 would seem ideal for all manner of USB gadgets.

A notably unique LPC2888 feature is the inclusion of both a linear regulator and a switching step-up DC/DC converter on-chip (see Figure 1). The former allows the chip to power itself from the USB bus (5 V), while the latter supports operation from a single AA(A) battery (0.9–1.6 V).

Texas Instruments is also hopping on the big-flash bandwagon with a version of their ARM7-based TMS470 MCU that integrates 1 MB of flash

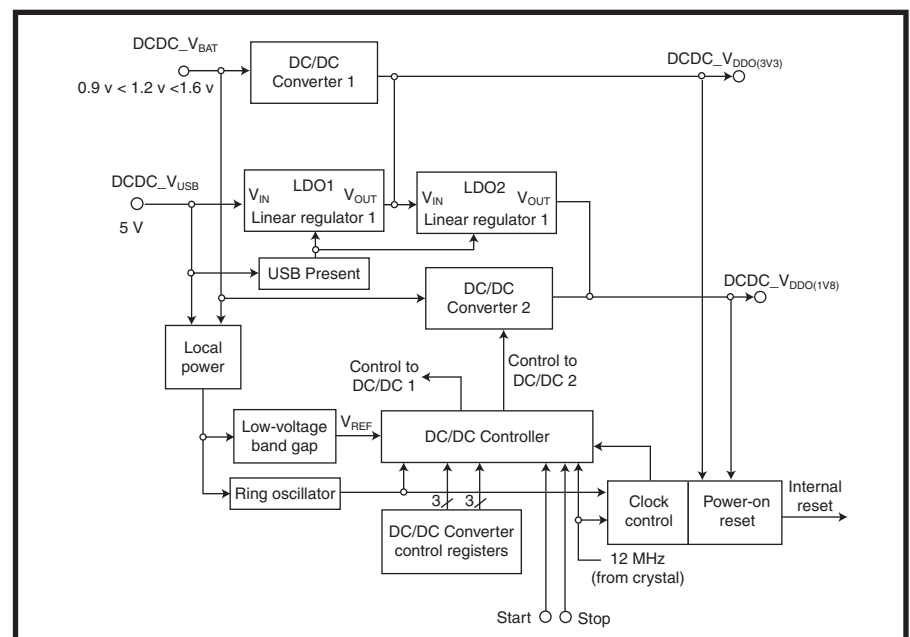


Figure 1—Analog features are where the action is as evidenced by the power management subsystem on the Philips LPC2888 ARM7-based USB MCU. Along with a linear regulator, it includes a DC/DC step-up converter that allows operation from a single AA/AAA battery (1.2 V). The subsystem is designed to seamlessly switch between battery and USB power.

memory and 64 KB of SRAM. Perhaps more important than the new part itself is a seeming shift in TI's marketing intentions for the TMS470. Originally, the focus for this line was automotive applications. While the parts themselves still reflect that heritage (e.g., sophisticated timer subsystem), it appears TI is interested in branching out into broader-base applications. To that end, they're now offering mainstream tools that position the TMS470 squarely in the mass-market fray (see Photo 1).

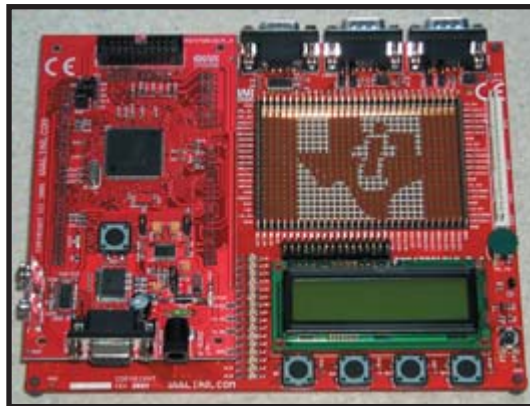


Photo 1—The TMS470 MCU with 1-MB flash memory and 64-KB RAM provides the means and a new Kickstart kit (\$399 including board, USB JTAG-debugger, and evaluation software) shows that Texas Instruments has the will to compete in the general-purpose ARM7-based MCU market.

CHIP WITH NINE LIVES

Until now, most of the single-chip MCU action has revolved around ARM7-based parts. As the entry level to the architectural lineup, they had the

best chance of meeting stringent price and power consumption requirements.

Meanwhile, the higher-end ARM9 chips have historically targeted per-

formance-oriented designs running sophisticated software in megs of off-chip memory. ARM9 processors may work well in cell phones and PDAs, but the multi-chip (external memory) form factor and “computer-in-drag” architectural aspirations (e.g., MMU) aren't a great fit with deeply embedded blue-collar applications.

But now I'm starting to see some single-chip ARM9 MCUs that actually make sense. For instance, consider the STR9X lineup from STMicroelectronics based on the ARM9E version of the core (see Figure 2).

The only question is: Where's the kitchen sink? These parts start with a bunch of memory, up to 512-KB flash memory, and a standout 96 KB of SRAM. Throw in a healthy comple-

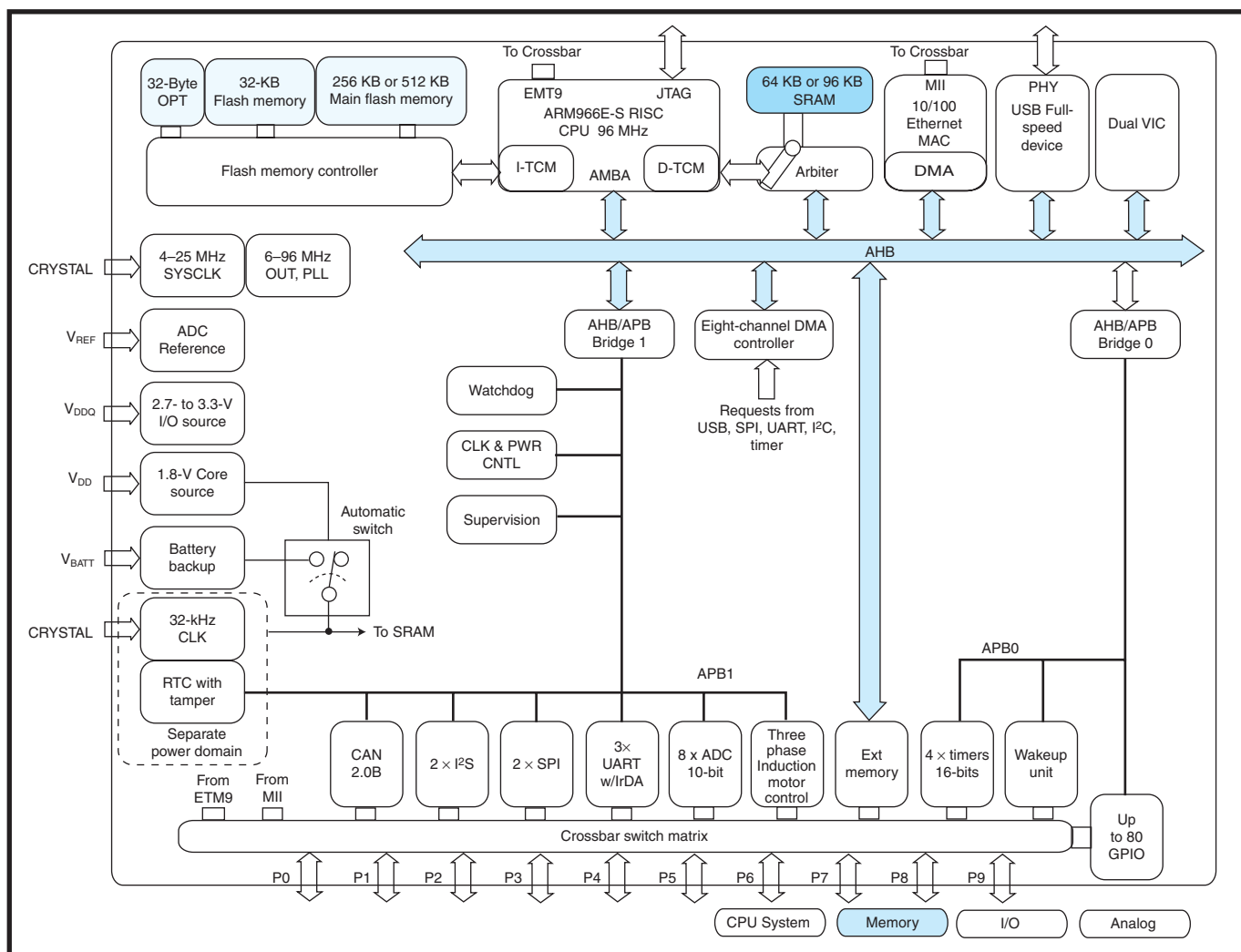


Figure 2—The new STR9X lineup is notable for the amount of peripheral integration, but it also represents a new direction for the ARM9 core. Adding on-chip memory and stripping out “computer” features like MMU and cache positions the core as a viable upgrade path for ARM7-based MCUs.



Photo 2—The Stellaris development kit comes with a lot of stuff (a board, JTAG debugger, and software tools) and a price to match. If you're on a budget, one alternative is to just use the \$190 part-specific daughter-board. Or, for that matter, rolling your own prototype is an option thanks to the chip's relatively spacious 0.05" pin pitch.

ment of I/O including all the usual suspects (serial ports, timers, etc.) and some premium upgrades (a fast 8×10 -bit ADC, an eight-channel DMA controller, and CAN). Top it off with your network connection of choice: Ethernet, USB, or, for that matter, both.

That's impressive, but perhaps what's most notable about the STR9X is what isn't there. For instance, there's no MMU because the intention is to leave your big-iron OS (e.g., Windows CE and Linux) at home.

There's also no cache, which I'm not a big fan of for real-time applications in any case. Instead, the chip optimizes the ARM9 Harvard architecture (separate instruction and data bus, which ARM7 lacks) with burst-flash prefetch queue and branch cache that deliver much of the performance of cache (approaching 100 MIPS peak) without all the jitter and power-wasting fills and spills.

Speaking of power, at 1.3 mA/MHz active and 55 μ A sleeping, the STR9X parts are truly viable for battery-driven applications. Yes, it isn't exactly the "nanoamps" of an 8-bit chip, but it's lower than many ARM7 chips and well within the capabilities of commodity batteries. Better yet, there are specific provisions for ultra-low-power operation, including running the processor off the

32-kHz RTC crystal (less than 1 mA typical) and shutting off the CPU and just preserving the SRAM contents and RTC (5 μ A at room temperature).

Another thing missing from an STR9X board are the outside glue chips typically required for a 32-bit "computer" chip. Besides an RTC, it's got all the "supervisor" functions (e.g., power-on reset, brownout detect, and watchdog timer) you expect to find on an MCU these days. And the clock generation is a blessing with a single 25-MHz crystal-driven PLL generating a myriad of clocks for the CPU, USB, Ethernet, and all those peripherals.

MORE BITS, LESS BUCKS

Atmel, Oki, Philips, TI, STMicroelectronics, and Analog Devices—did I forget anyone? By now, you know who the major ARM MCU players are, and quite a lineup it is. But now, there's a new kid on the block, start-up Luminary Micro. Their Stellaris MCUs are a unique take on the subject that's interesting from two different perspectives.

First, and fundamental to the concept of mass-market 32-bit MCUs, is

the question: How low (price) can they go? Luminary has an attention-getting answer headlined in their press release: "Luminary Micro Announces 32-bit Microcontrollers for \$1" (quantity 10,000 units). No doubt there are many could-be customers (and competitors) reasonably questioning the premise that 32-bit MCUs should move that far and fast down the price/performance curve. So, right off the bat, Luminary's announcement is newsworthy for reinforcing the "more bits, less bucks" trend.

Beyond the price story, Luminary's chips are technically notable for being first to market with the new ARM Cortex-M3 core. Announced about a year ago, Cortex-M3 freshens the admittedly long-in-tooth ARM architecture (which first appeared in the '80s) with features designed for the embedded market. In turn, Luminary takes the Cortex-M3 core and packages it in an 8-bit like form factor with 28-pin packages that accommodate a decent selection of peripherals. The smallish package is interesting, but the particular feature that stands out is the memory, or rather lack of same, with only 8-KB flash memory and 2-KB RAM. And the maximum clock rate for the initial parts is 20 MHz. Put it all together with the \$1 price tag, and you get the picture that the Stellaris

parts mean business, as in high-volume, cost-sensitive applications.

The Cortex-M3 core differs from the familiar ARM7 in a number of ways, some more apparent than others. Here's the scoop on the most noticeable changes.

The original ARM architecture featured a minimalist approach to interrupts that did little more than stack the PC and processor status and vector to a fixed location. That left it up to software to handle embellishments such as nesting and priority. On the other hand, a separate register bank reduced state save

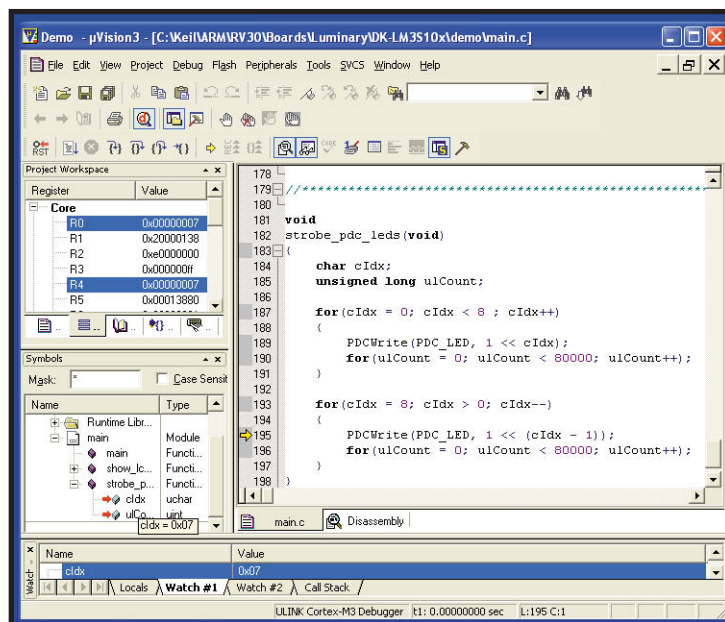


Photo 3—Lack of software support is the Achilles heel for most new architectures. But right out of the box, the Cortex-M3 walks tall with Keil RealView (shown here), IAR Embedded Workbench, and GNU.

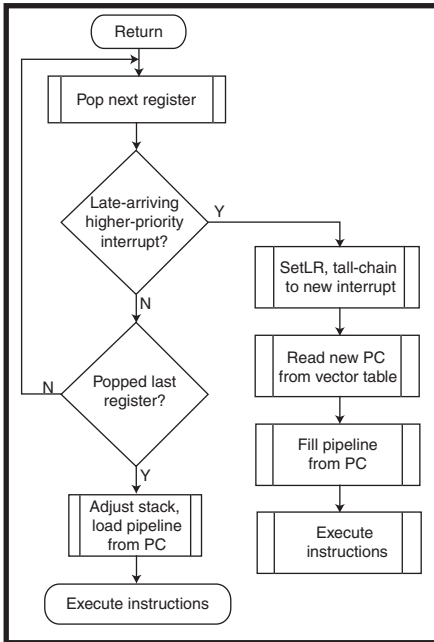


Figure 3—The Cortex-M3 interrupt response features the automatic stacking and unstacking of processor state. To minimize unnecessary overhead, a feature called tail-chaining eliminates unnecessary state save and restore when possible.

and restore overhead in simple (non-nested interrupts) applications.

By contrast, the Cortex-M3 does a lot more interrupt processing in hardware thanks to a tightly coupled Nested Vectored Interrupt Controller (NVIC). In addition to handling a potentially huge number of interrupt sources (hundreds) with dynamic priority, the most notable change is that state (i.e., register) save and restore is automatically handled by the processor, replacing the register bank scheme of yore (only the stack pointer is banked).

There are potential concerns with the switch from the do-it-yourself interrupt regime to the new paternalistic approach. Yes, it's nice to have hardware do the dirty work, but the danger is that performance for simple applications (i.e., few interrupts) and flexibility for complicated ones can both suffer. Cortex-M3 goes to some effort to “do no harm” in this regard with provision for “late-arriving” interrupts and so-called “tail-chaining.”

Late arrival refers to a situation where recognition of a low-priority interrupt is underway. The state is being saved, but control hasn't yet vectored to the handler. Now, imagine that a higher-priority

interrupt occurs. In a conventional scheme, the higher-priority interrupt would restart the interrupt response, notably including saving the state once again. However, that's kind of dumb since the state hasn't changed, so Cortex-M3 simply substitutes the higher-priority interrupts vector.

Similarly, what if there's an interrupt pending at the end of an interrupt handler (one example being the interrupt deferred by a late arrival as described above)? Because it would be superfluous to unstack the state only to restack it in response to the pending interrupt, Cortex-M3 shortcuts the process (from 12 to six cycles) by immediately vectoring to the pending interrupt's handler (see Figure 3).

Another embedded-friendly feature is improved support for bit handling. Cortex-M3 features an 8051-like “bit-banding” capability that allows addressing single bits in memory and I/O registers directly without the need for the usual read-mask-write sequence.

By far the most notable change for Cortex is the instruction set. Traditional ARM7 chips rely on a software-

directed scheme that switches between 32-bit “ARM” instructions for performance and 16-bit “Thumb” instructions for code density. By contrast, Cortex has a unified “Thumb-2” instruction set comprising 16- and 32-bit instructions. (CISC lives!) Other notable improvements include high-speed multiply and divide for math-intensive applications and the elimination of data alignment restrictions (i.e., 16- and 32-bit data can reside at any byte address).

With a three-stage pipeline, Cortex aspires to execute one instruction per clock and succeeds for the most simple register operations. However, as usual, things like conditional branches, memory bottlenecks, and the aforementioned unaligned accesses cause stalls. Eyeballing the instruction-timing chart, I figure you're looking at 1.5 to 2 clocks/instruction on average depending on the instruction mix and the compiler's ability to schedule around stalls.

HANDS-ON

Putting aside the big issues sur-

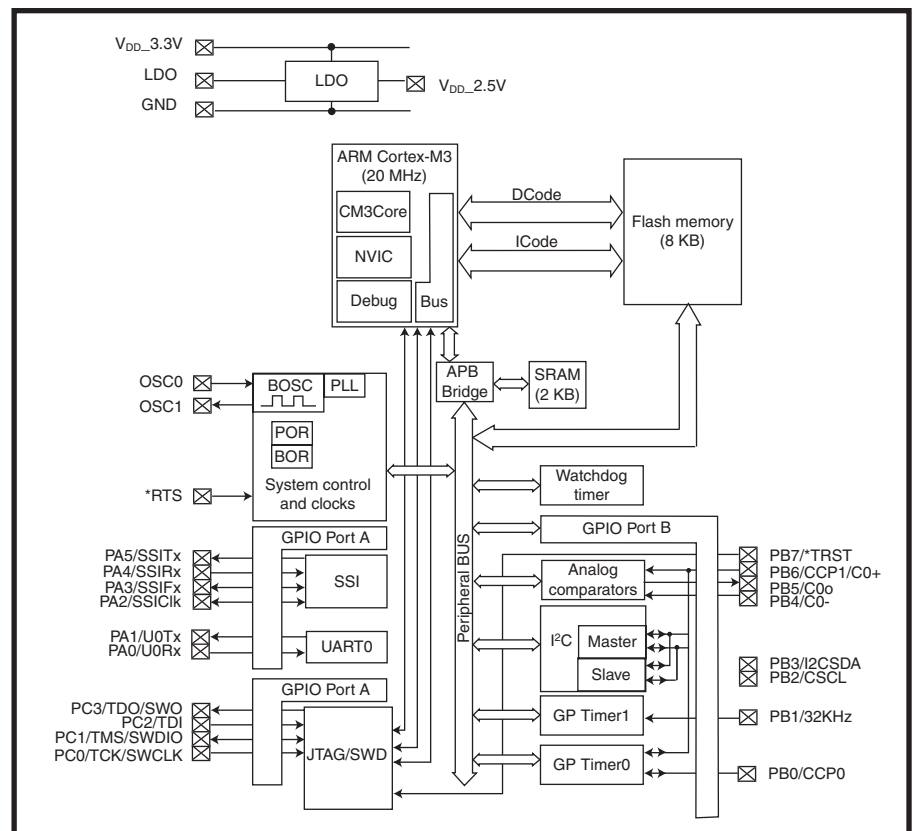


Figure 4—Under the hood, it may be 32 bits, but in terms of performance, packaging, and price, the new Cortex M3-based Stellaris chips from Luminary Micro are positioned to compete directly with 8/16-bit MCUs.

rounding Cortex for the moment, now's a good time to take a look at the Luminary chips themselves (see Figure 4). Although there are two part numbers, talking about "chips" plural is pushing it because the difference between the LM3S101 and '102 is minuscule. Essentially, the '102 has an I²C interface while the '101 doesn't. That minor differentiation seems a bit odd, perhaps a legal hangover from the historic I²C patents held by Philips.

Let's start with the basics, namely power, reset, and clock. Thanks to an on-chip regulator, the chips run off a single 3.3-V supply and power consumption is 35 mA at 20 MHz. As far as active power consumption goes, that's not bad. However, for the all-important low-power modes, there is a problem with the initial parts. Although the spec in the datasheet is to be determined, an errata notes that power consumption in Sleep modes exceeds 1 mA. Needless to say, at 10 to 100× the stand-by power of an 8-bit chip, that bug must be fixed if the parts are to make it in battery-powered applications.

RESET is addressed thoroughly with a full six ways of kicking things off, including an RST pin, Power-On-RESET (POR), Brown-Out RESET (BOR), software RESET, watchdog timer, and LDO RESET (i.e., the aforementioned voltage regulator falls out of regulation). There's a RESET CAUSE register with sticky bits so boot software can confirm the source of the RESET condition.

At start-up, clocking relies on an internal oscillator running at a nominal 15 MHz. In principle you can continue running off the internal clock, but the accuracy is so loose ($\pm 30\%$) as to require an external clock for most applications. External clock options are to use a crystal or square wave logic level input, subject to certain restrictions. In order to use the on-chip PLL, the crystal should be between 5 and 8 MHz. If bypassing the PLL, a 1- to 8-MHz crystal can be used. On the other hand, if the clock is provided as a digital input, a full range of DC – 20 MHz is possible.

One nice feature is clock verifica-

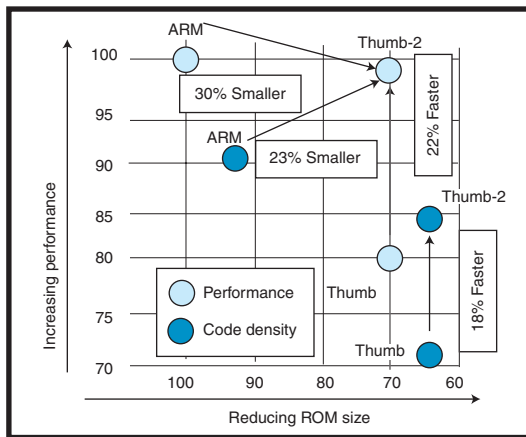


Figure 5—Cortex-M3 delivers better performance and code density than traditional ARM cores. However, that advantage alone, on the order of 20%, isn't necessarily compelling. Rather, the Cortex story is more about an upgrade path to future higher-performance versions rather than a short-term advantage.

tion in which the different clocks (i.e., internal, external, and PLL) monitor each other's operation. The external clock can check the PLL and internal oscillator while the internal oscillator checks the external clock. Should a failure be detected with the current clock source, the chip automatically switches to a working clock and generates an interrupt.

Peripherals start with two 32-bit timers, each of which can optionally be configured as dual 16-bit timers with input capture, edge counting and timing, and basic PWM capability. There's also a separate watchdog timer that can be configured to generate an interrupt or RESET and can be locked against disabling.

Serial communications are well served with both a UART and a synchronous serial interface (SSI). The former is similar to the familiar 16C550 UART, including separate 16 entry transmit and receive FIFOs and dedicated data rate generator (i.e., don't need to use one of the general-purpose timers). Meanwhile, the synchronous interface is fully programmable (i.e., polarity and phase) to accommodate the variety of popular sync serial flavors such as SPI, Microwire, etc. As I mentioned earlier, the LM3S102 also includes an I²C interface, which notably includes that protocol's fancier embellishments such as "Fast Mode" (400 Kbps) and multi-master arbitration.

Analog applications are served with one ('102) or two ('101) analog com-

parators, which compare the external input with an external or internal reference. The latter is programmable across a range of 0 to 2.37 V in roughly 100-mV steps. The result of a comparison can be driven on an output pin, generate an interrupt, or both.

Pins not otherwise used for the above peripheral functions are available for use as general-purpose I/Os. Programmable I/O pad features include pull-up and pull-down resistors, open collector output, variable drive (2, 4, or 8 mA), and slew-rate control. Any and all general-purpose I/O lines can also be configured as interrupt inputs with programmable edge sense, level sense, and polarity.

Finally, there's a full (five-pin) JTAG interface, three pins of which are shared with the ARM serial-wire debugger. After reset, the JTAG pins can be configured as general-purpose I/Os.

Although there are only 28 pins, belying its name, the "Small Outline IC" (SOIC) package isn't especially small at roughly 18 mm × 8 mm. On the other hand, the relatively spacious 1.27 mm (0.05") pin pitch makes for easy PCB layout and probing.

EASY EV

Getting up to speed with the Luminary parts is easy with their development kit. Although a bit pricey, for \$775 you do get a fairly elaborate board (see Photo 2, p. 80) and an official Keil (now owned by ARM) ULINK JTAG debugger and a bunch of evaluation software including two credible, although quite different, tool options.

First is the official Keil-now-ARM RealView toolchain (see Photo 3, p. 80). It combines the historically popular and familiar Keil μ Vision IDE with ARM's own finely tuned compilers and works with the aforementioned ULINK debugger. This evaluation version has some limitations, but the main one (16 KB code size limit) isn't an issue for the 8-KB Stellaris parts. Thanks to excellent step-by-step installation and demo instructions, I got the kit up and running quickly with absolutely no glitches or head scratching.

Second, representing a completely different approach is an evaluation version (30-day expiration) of the GNU tools from CodeSourcery. Instead of using the ULINK JTAG debugger, the CodeSourcery setup relies on the classic serial port debug approach (e.g., GDB). One nice touch is that the board includes an FTDI USB-to-serial chip. That means you can connect the board to your PC via USB for serial debugging while leaving the MCU's own serial port free for application use.

Although the CodeSourcery brochure alludes to a modern Eclipse-based IDE, the demonstration instructions that came with the kit called for a command line setup and lot of manual installation, so I passed. Anyway, the point is well taken that GNU is a viable option for Cortex and Luminary.

Also, although not available at the time I received the kit, don't overlook the fact that major toolchain player IAR Systems has announced support for the Luminary Chips as well. As with the Keil/ARM RealView evaluation version, their Embedded Workbench KickStart version 8-KB code "limitation" isn't a limitation at all for the Luminary chips.

The first test any "new" architecture faces is tool support, and failure is not an option. But with Keil, IAR, and GNU on board, I think it's fair to say Luminary has achieved liftoff. The fact ARM is bringing their big balance sheet to bear on the success of Cortex is further insurance that tools will be available.

THUMBS UP, THUMBS DOWN?

This 32-bit MCU stuff is very interesting to me. Usually, I feel pretty confident in my instincts when studying the tea leaves in the marketplace. But I honestly have to say that this time a lot of different scenarios could play out. Here's my take on the various issues.

Will 32-bit MCUs "replace" 8- and 16-bit parts? No way. First, 8-bit MCUs still have technical advantages, including lower price and power consumption. The advantage has shrunk, but it remains. Second is sheer inertia in the embedded market where prod-

ucts can have decades-long life cycles. Finally, there's the basic reality that the developing economies around the globe will fuel demand for all the simple 8- and 16-bit gadgets that we take for granted.

On the other hand, are the lean-and-mean 32-bit MCUs contenders for a traditional 8/16-bit application? Absolutely. From a designer's perspective, the proper choice is probably driven by whether the application is an "upgrade" of an existing 8/16-bit design or a clean slate. In the latter case, the ability to cover a \$1 to 100-MIPs-and-beyond range with a 32-bit chip is a big plus.

Will all those 32-bit designs be ARM-based? It's true that ARM chips are the '51s of tomorrow. But while the multi-sourced '51 is extremely popular, it's by no means the only 8-bit player. Just as the '51 shares its market with sole-source PICs, 'HC11s, H8s and so on, ARM will be joined by other, albeit proprietary, 32-bit architectures.

And speaking of ARM, just which ARM are you talking about? Are you referring to the lineup of proven ARM7/9 chips already shipping from top-tier suppliers? Or are you talking about the "new and improved" hot-out-of-the-lab Cortex architecture? It's an odd situation in which ARM is seemingly competing with their licensee customers.

Make no mistake: the new Cortex/Thumb-2 architecture is technically superior to the old ARM/Thumb. The questions are how much and does it matter (see Figure 5)? As of now, the advantage is less than it might appear because existing ARM7 MCUs incorporate their own upgrades (e.g., interrupt controller and bit manipulation). However, that may change over time as more chips are released and ARM marches onward and upward with the Cortex architecture. Luminary has already announced plans for M3 chips with larger memory and more advanced peripherals, while ARM has laid out a roadmap for Cortex to higher-performance R4 and A8 versions.

But won't Cortex face the same New Coke fate that befell chips like the Philips 'XA and Intel '251 that dared to dabble with the recipe? Not

necessarily. The situation is notably different this time around because ARM is an architecture company, not a chip company. The 'XA and '251 were proprietary, but I fully expect to see multiple Cortex licensees emerging over time. In principle, both new and old ARM architectures can co-exist and probably will.

The only thing for sure is that Moore's law is still marching on, albeit with a less lively step. While there may come a time when designers don't have interesting decisions and trade-offs to make, we aren't there yet. The wave of 32-bit MCUs is a reminder to pay attention or you'll get left behind. ☒

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 architecture and tools

ARM
www.arm.com

GNU-based tools for Cortex-M3

CodeSourcery
www.codesourcery.com

Embedded Workbench

IAR Systems
www.iar.com

RealView Development tools and ULINK debugger

Keil
www.keil.com

Stellaris Cortex-M3-based MCUs and development kit

Luminary Micro, Inc.
www.luminarymicro.com

LPC2888 ARM7 MCU with USB

Philips Semiconductors
www.standardics.philips.com/microcontrollers/

STR9X ARM9E-based MCUs

STMicroelectronics
www.st.com

TMS470 Microcontroller

Texas Instruments, Inc.
www.ti.com