8 Bit or 32 Bit? Choosing Your Next Design's MCU

Electronic Design

Ingar Fredriksen and Paal Kastnes

Fri, 2014-12-05 08:57

The rise in popularity of 32-bit microcontroller (MCU) devices across the embedded community comes as no surprise. These function-rich devices suit an array of different applications, which explains why many embedded developers select them for their next designs. Designers recognize that such complex devices offer everything they need in terms of raw compute power, a rich peripheral set, and easy access to a wide range of development tools and libraries.

Many of these 32-bit devices are based on the highly successful ARM cores. Thus, developers feel confident in having access to second source devices and a comprehensive set of development, test and validation tools being available in the market.

However, taking a closer look at recent MCU market trends reveals that 32-bit devices aren't the only ones experiencing strong growth *(Table 1)*. The surging 8-bit MCU market boasts a compound growth rate (6.4%) close to that of 32 bit (6.9%). Other industry analysts forecast identical growth rates for 8- and 32-bit microcontrollers.

The upswing in 8-bit devices clearly highlights that there must be some compelling reasons to use an 8-bit device in place of a 32-bit MCU. This article looks to shed some insight as to why 8-bit devices are retaining market share.

Essential Differences

The principle differences between 8- and 32-bit MCUs are cost and price structure, CPU performance, ease of use, efficiency in hardware near functions, and static power consumption. When embarking on a new design, developers need to carefully scope out the requirements for an MCU based on the amount of processing capability required, the degree of interfacing needed, and, for battery-powered designs, the all-important power consumption profiles. There's no doubt that a 32-bit MCU delivers higher performance than an 8-bit device, but the engineer faces the traditional decision of choosing between the best available device in the market versus an application's actual needs.

	2012	2013	2014	2015	2016	2017	CAGR
Total Semiconductor	325,367	339,666	361,612	385,052	395,974	413,602	4.9%
Microcontroller (MCU)	16,008	16,202	17,211	18,799	19,307	20,480	5.1%
4 bit MCU	154	159	161	157	145	133	-2.8%
8 bit MCU	6,057	6,565	6,936	7,532	7,768	8,259	6.4%
16 bit MCU	4,021	3,611	3,765	4,060	4,053	4,019	0.0%
32 bit MCU	5,776	5,868	6,349	7,050	7,341	8,069	6.9%

Of course, these decisions will greatly influence the likely bill of materials (BOM) cost. With a lower gate count, a less complex 8-bit device will certainly be cheaper than a 32-bit device. When comparing 8- and 32-bit MCUs from leading vendors, each with a similar amount of flash memory, pin-out etc., 8-bit devices typically cost about 20% less. But this is only the first of many considerations. Another aspect relates to the ease in setting up for a new development.

Ease of Development

MCU suppliers tend to add more features and functionality to their 32-bit devices as opposed to 8-bit products. Consequently, far more setup considerations emerge with a more complex device. While some 32-bit MCUs can run with a limited setup similar to that of an 8-bit device, you're unable to take advantage of the more powerful device's additional features.

For example, a typical 32-bit ARM device will have independent clock settings for the core itself, the AHB bus, the APBA bus, and the APBB bus. They all can be set to different frequencies. Typically, you will also have to switch to the clock you want to use because it's set in software, not in hardware like most 8-bit parts. Furthermore, changing the clock means you must set up the wait states for flash, possibly predicated on measured V_{CC} voltage.

Such a setup can be much simpler with an 8-bit MCU, though. For example, Atmel's tinyAVR and megaAVR products only require initialization of the stack pointer, which typically takes four lines of code, prior to coding the application. The choice of clock, brownout detector, reset pin function, etc., is all pre-programmed into the device.

The architecture is also much more straightforward than a 32-bit device with internal registers, peripherals, and SRAM all mapped on the same data bus. The peripherals and CPU would normally run at the same frequency, so no peripheral bus configuration is necessary. Moreover, designers can avoid being concerned about latency in synchronizing between different clock domains.

Performance

When it comes to desired CPU performance, the engineer should consider all use cases. The reality is that many embedded designs don't have high compute requirements. Often, very little manipulation of data is required, so balancing those needs against power-consumption and peripheral-interfacing requirements becomes crucial.

Related

Yes, You Can Easily Shift From 8-Bit To 32-Bit MCUs

Q&A: Microchip Discusses Recent 8-bit MCU Innovations

loading CPU Boosts Microcontroller Performance And Cuts Power

For instance, a simple thermostat application will spend most of its life in a sleep mode. Every so often, it will wake up and measure the temperature and then make a decision to turn a relay on/off or send an instruction to a host controller. Then it will resume sleep. The compute and interface requirements of this application are small, but many other applications such as fire detectors, power tools, flow meters, and appliance controls have a similar use profile, too.

Efficiency of Hardware Near Functions

Many modern microcontrollers incorporate some hardware functions that serve to help the CPU operate as efficiently as possible. In Atmel's case, both the 8-bit AVR and 32-bit ARM-based MCU families feature the Peripheral Event System. An event system is a set of hardware-based features that allows peripherals to interact without intervention from the CPU. It allows peripherals to send signals directly to other peripherals, ensuring a short and 100% predictable response time.

When fully using the capabilities of the event system, the chip can be configured to do complex operations with very little intervention from the CPU, saving both valuable program memory and execution time. In the case of detecting a hardware event, it's important to first detect the event and then switch control to the desired interrupt service routine (ISR).

In these situations, CPU speed isn't the single determining factor. It's a question of how long, in terms of cycles, does it take to respond to the interrupt, run the ISR, and return. As the following example will show, 8-bit devices can be more efficient in handling hardware near actions.

Action required	8-bit AVR (cycles)	32-bit CM0+ (cycles)	Comment
Detect interrupt and jump to interrupt vector	3	12	
Identify interrupt triggered	0	6	Not needed on AVR since every interrupt has its own interrupt vector. CM0+ core is limited to 32 vectors
Push register to stack	1	0	Not needed on CM0+ since interrupt detection will push 8 registers to stack
Read received byte from SPI to register	1	2	Assuming peripheral and CPU running at same speed. (Best case for CM0+)
Write received byte to SRAM	2	2	
Pop register from stack	1	0	Not needed for CM0+ since interrupt return will pop the 8 registers back
Return from interrupt	3	10	
Cycles in main ( ) until next interrupt can be serviced	1	1	
Minimum number of cycles to receive one byte from SPI using interrupt	12	33	

Consider receiving one byte on the SPI, using an interrupt to detect it, and then running a simple ISR routine to read the byte from the SPI peripheral and store it in SRAM. Using this scenario, Table 2 draws comparisons between an Atmel 8-bit AVR device and an Atmel ARM Cortex M0+based 32-bit MCU. Calculated with information available, the results are based on minimum implementations. However, engineers should check with their own applications since the interrupt detection and return from interrupt could take more cycles than

wn in the table. Requiring 12 cycles versus 33 cycles equates to having a theoretical maximum SPI dwidth of 1.67 MB/s for the 8-bit CPU and a 606 kB/s bandwidth for a 32-bit CPU when running at 20 MHz.

The degree of numeric processing can also have an impact on the stack and required memory. Applying the Fibonacci algorithm is one particularly good method for testing memory requirements. Since it only uses a local variable, everything needs to be pushed to the stack.

When making a comparison between an 8-bit AVR and an ARM 32-bit CM0+-based device, and using a recursive 15-stage Fibonacci algorithm, the AVR uses a total of 70 bytes of stack, including 30 for return stack (15 calls deep). The ARM-based device uses 192 bytes (60 should be return stack). This means the CSTACK is more than three times the size of the 8-bit solution. In typical C code, more of the variables on the stack often come in a packed format, so this is an extreme corner. However, saying 1.5 to 3 times more SRAM is needed for the same 8-bit-centric application on a 32-bit (versus a native 8-bit) device is a fair estimation.

Power Consumption

No MCU article would be complete without investigating static power consumption. This alone may be a key factor in choosing between an 8- or 32-bit device, especially for battery-powered applications. Table 3 illustrates power-consumption differences between 8- and 32-bit devices in both active and static modes.

	CPU	Flash	Pin Count	Active[1]	Static[2] Typ @ 25°C	Static[2] Max @ 85°C
Atmel SAM D20	32-bit ARM Cortex-M0+	16 – 256kB	32 – 64	140 µA/MHz	2 µA	85 µA
Freescale Kinetis K20	32-bit ARM Cortex-M4	32 – 160kB	32 – 64	280 µA/MHz	1.9 µA	30 µA
Atmel ATmegaX8PA	8-bit AVR	4 - 32kB	28 - 32	300 µA/MHz	100 nA	2 µA
ST STM8S00X	8-bit STM8	8 - 64kB	20 – 48	230 µA/MHz	4.5 µA	17 µA

[1] CPU active, running code from internal NVM memory at 3V under typical conditions.
[2] CPU off, microcontroller in sleep mode with full SRAM retention at 3V under typical

Aggressive manufacturing technologies increase transistor leakage current, which roughly doubles with each process generation, and is proportional to the number of gates. Leakage current increases exponentially at higher temperatures, which can be easily overlooked when designing a consumer design. Mobile phones and personal media players are transported everywhere, and as we have all found out, temperatures experienced during the summer inside a car can easily climb above 40°C.

The amount of time the microcontroller will spend in active mode versus static mode contributes significantly to the overall application power budget.

Naturally, the ratio between active and static modes will vary depending on the application requirements. Taking the previous SPI interrupt example *(Table 2, again)* and assuming a SPI data bandwidth of 80 kb/s, the 8-bit CPU will spend 1.2% of its time in active mode compared to that of the 32-bit, which will spend 3.3% in active mode *(Table 4)*.

Power budget	8-bit AVR	32-bit CM0+
Active mode at 10MHz	1.2% x 3000uA	3.3% x 1400uA
Sleep mode	98.8% x 0.1uA	96.7% x 2.0uA
Average consumption	36.1 uA	48.1 uA

Conclusion

Contemplating whether to use an 8- or 32-bit microcontroller for a future design may involve an Internet of things (IoT) application. How IoT actually takes shape provokes lots of debate, but it will certainly challenge engineers to make a detailed appraisal of the MCU requirement. Wireless connectivity, especially ZigBee, will also be an essential component, but that doesn't automatically mean that it will need a higher power device.

A number of available 8-bit microcontroller products satisfy the need for low levels of processing and wireless connectivity. One such example is the Atmel ATmegaRFR2 series, which provides an IEEE 802.15.4-compliant, single-chip, 2.4-GHz wireless microcontroller solution that suits battery-powered, low-cost IoT designs.

Ingar Fredriksen, Atmel's MCU marketing director for the EMEA region, is based in Trondheim, Normay. He received his Bachelor in Science degree in electrical engineering from Trodheim University College.

Paal Kastnes staff engineer in the MCU Applications team specializing in training Atmel staff and customers in using Atmel's microcontrollers and tools, is also based in Trondheim. He received his Bachelor in Science degree in electrical engineering from Trondheim University College.

Source URL: http://electronicdesign.com/microcontrollers/8-bit-or-32-bit-choosing-your-next-design-s-mcu