Electronic Design

ALAN R. EARLS, Technology Editor

How to Choose the Right Microcontroller

Microcontrollers are everywhere and their use is growing. But a better understanding of their characteristics and potential use cases can help when it comes to selecting and deploying the technology.

First things first. Do you need a microcontroller or a microprocessor? These birds of a feather are distinct but inhabit overlapping use cases. Thus, understanding their comparative strengths is a necessary first step.

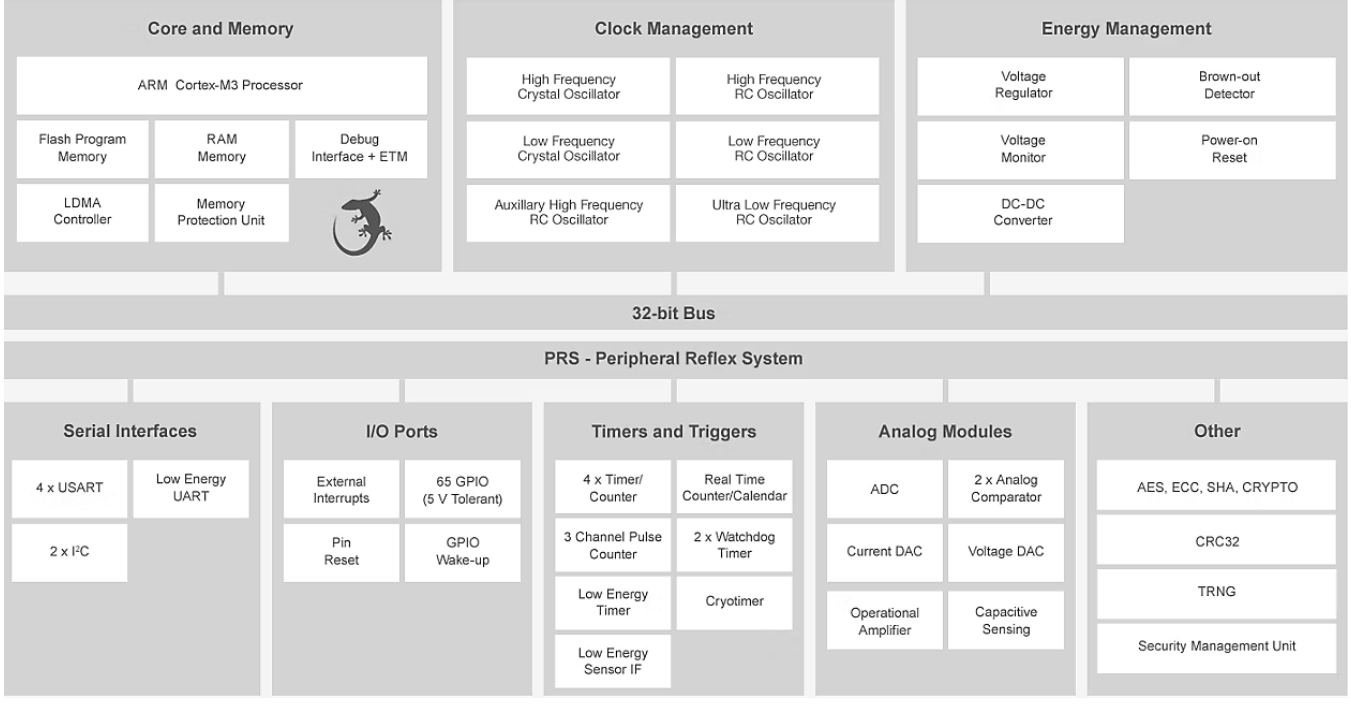
Microcontroller or Microprocessor: Making the Right Choice

Microprocessors have a single processing core (CPU), typically powerful enough to handle complex applications involving lots of memory. They're general-purpose oriented. On the other hand, microcontrollers include a CPU, memory, and I/O—enough of each to support single or only a few functions or applications; for example, a medical device or home automation system.

Microprocessors often work with external devices and systems such as memory, whereas microcontrollers are more likely to consist of a single package.

Microcontrollers are a good fit for lower-power and real-time applications and data gathering, as well as applications with limited available space or power.

Both will require programming and, thus, knowledge of a language such as C or assembly language. Working backward from what tasks must be performed or what tasks might need to be performed in the future is important in



The Silicon Labs Gecko is a typical microcontroller based on the Arm Cortex-M3 architecture. (Silicon Labs)

properly selecting either a microprocessor or a microcontroller. Then pick an appropriate vendor and product.

Start with a plan and begin to gather requirements. This information should be analyzed to scope out likely hardware and software choices, and initial coding might be roughly defined to ensure that the hardware chosen will be a good fit.

Essential Steps for Selecting a Microcontroller

Having access to a debugger can speed and simplify the process of fielding anything that includes a microcontroller. Debuggers are able to support code upload and real-time debugging such as setting breaking points. Manufacturers often provide their own debuggers, which can be worthwhile if you select their microcontroller product, but generic debuggers are also available.

Envisioning a Complete Product

What else is needed to create a successful microcontroller implementation? Quite a bit actually.

Microcontrollers will need a source of power, which is often 3.3 V. And that power needs to work through a supply rail. Pins also need a decoupling capacitor. To accommodate any analog peripherals, you may need to have a so-called Pi filter to keep noise from impact analog signals.

Uploading code can generally be accomplished through a debug probe or a bootloader.

An oscillator is also an important consideration. Most microcontrollers have one built in, but an external crystal oscillator may be needed as well, especially for higher rates or greater accuracy. This isn't simply a "plug-and-play" process. You may need to include load capacitors in the circuit, based on the datasheet of the crystal, plus a feed resistor to avoid overdriving the crystal. This can reduce or prevent the development of harmonics.

"Decoding" the microcontroller pins so you can create a complete device is an additional step needed to make a microcontroller implementation a reality. Again, datasheets are important. Better yet, some microcontroller vendors offer development environments and tools to simplify these activities.

Depending on the nature of your project, it can be helpful to assemble a list of the external interfaces that must be accommodated. A hardware block diagram is one way to approach this task, often starting with the communication interface. UART, USB, I2C, and SPI are among those likely to be used by a microcontroller. USB and Ethernet are less common. If they're part of the mix, recognize that they will tend to require more program space. There may also be a need for digital or analog-digital I/O.

And, speaking of software, getting clarity regarding how a given program will run on the hardware you're considering is a very important step. Do any processing steps intensively use resources? What are they? Might they occur simultaneously? Are there any examples of similar software running on the hardware you're considering to offer a rough proof of concept?

The Long and the Short of It

Choosing controllers isn't a one-and-done exercise. Odds are good that improvements in technology or a desire to upgrade a product will make it necessary or desirable to upgrade microcontrollers, perhaps multiple times over the lifetime of a product. So, long-term thinking early in the design process can pay off later. Some quick calculations should clarify whether an 8-, 16-, 32-, or even 64-bit microcontroller is appropriate and likely to serve the projected product for a reasonable time.

A Glimpse into the Microcontroller Future

And what's the likely evolution of the microcontroller market? According to the experts at Grandview Research, the 32-bit segment recently garnered 50.0% of the value of all microcontrollers and the market as a whole is expected to grow at a CAGR of 12.3% throughout the forecast through 2030. So, there are still plenty of 8- and 16-bit microcontrollers, but over time, more powerful and capable 32- and 64-bit devices will likely become dominant.

The researchers also noted an important emerging architecture trend. The Von Neumann architecture segment dominated in 2023, gaining a revenue share of over 42.0%. The affordability and familiarity of Von Neumann and its comparatively simple design and development processes are its strength.

However, so-called Harvard architecture microcontrollers, which feature separate storage and signal pathways for instructions and data, are also growing at a rate of 12% annually. Harvard architecture is especially useful for real-time applications and where high performance is needed. That's yet another permutation to consider.

References

Tutorial: How to Design Your Own Custom Microcontroller Board (video)
Developing Microcontroller Based Product – A Case Study
DEFINITION—microcontroller (MCU)