Selecting a 32-bit Microcontroller That Makes the Developer's Job Easier

Traditionally, the key factor in selecting a 32-bit microcontroller (MCU) has been the choice of central processing units (i.e., the core CPU). Until recently, 32-bit MCUs were based on a variety of cores including, in some cases, proprietary architectures; as a result, embedded designers either had to stick with one core or significantly increase their development time to learn new hardware intricacies and port existing software code. Over the last few years, the emergence of ARM Cortex cores in MCU products has changed the embedded landscape. Developers are migrating from proprietary 32-bit cores to MCUs based on ARM Cortex processors, so they are no longer locked into a single provider for their MCUs. A large and growing ecosystem has developed around ARM processor-based MCUs including third-party tools for compilers, real-time operating systems, software stacks, LCD graphics and more. Most major MCU suppliers now have an ARM processor-based offering, making the ARM Cortex core the de-facto 32-bit MCU standard.

Selecting a 32-bit MCU based on a standard core opens the door to more options than ever before. Therefore, choosing the right MCU for a given application can be a difficult task with many variables to consider. First, designers must narrow the potential number of MCU choices based on a variety of "checkbox" items, such as memory size, number of input and output pins, and communication interfaces. It is very likely that multiple suppliers of ARM processor-based MCUs will meet the basic requirements list. Therefore, developers will need to refine their selection by focusing on other important factors, such as mixed-signal integration, configurability, power consumption and ease of development.

Selecting a 32-bit MCU that integrates commonly used components will help the developer reduce overall system cost, design complexity and development time. For example, Silicon Labs' Precision32™ mixed-signal MCUs are engineered to provide a number of integrated features not typically found in other MCUs, such as a USB oscillator, 5 V regulator, six programmable high-drive pins that can provide up to 300 mA of current and 16 capacitive sensing input channels for touch buttons or sliders. This high level of integration helps to eliminate the need for multiple discrete components and provides a more flexible power domain, thereby reducing the bill of materials (BOM) and simplifying the development process.

To illustrate the benefits of using a highly-integrated mixed-signal MCU, let's examine a typical barcode scanner application. To read a barcode, the scanner shines a laser at an oscillating mirror powered by a motor (see Figure 1). This light illuminates the barcode, and the image is then captured by a charge-coupled device (CCD) sensor. The CCD sensor is a camera that captures one line of pixels at a time, e.g., 1 x 1024 pixels. The analog light levels are shifted out and captured by an analog-to-digital converter (ADC). Having an MCU that can supply high current eliminates the need for power transistors used to drive the laser and motor. Choosing an MCU that is designed to provide an interface with the CCD sensor for clock synchronization can make a designer's job easier, as well.

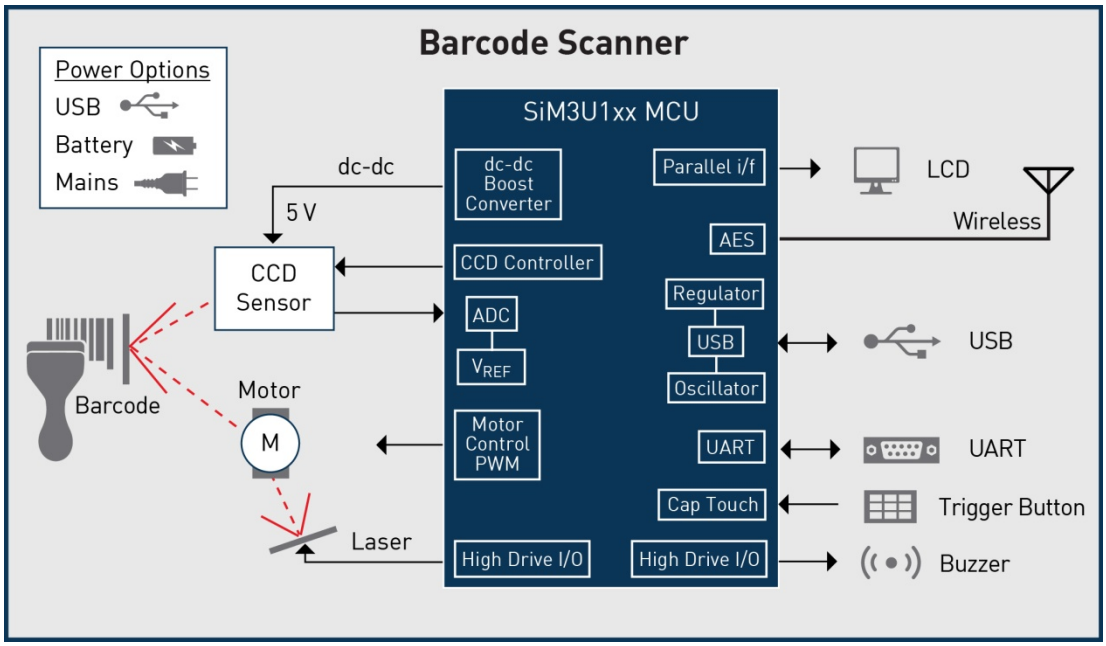

Figure 1. Barcode Scanner Design Benefits from Highly Integrated MCUs

For the optimal results, the MCU's ADC should be able to keep pace with the fast rate of the CCD camera (typically >1 MSPS). For 5 V CCD sensors, a power management IC is also necessary in most designs to provide the input voltage for the sensor, and a 3.3 V input voltage is needed for the MCU and other components.

In this barcode example, the Precision32 SiM3U1xx USB MCUs can drive a synchronized clock to the sensor, easily keep up with the fast CCD sampling rate and provide a 3.3 to 5 V dc-dc boost controller to power the sensor, further reducing the number of components in the system. Furthermore, in USB powered scanners, the Precision32 MCUs have a built-in voltage regulator to run the MCU directly off of USB power and an internal 48 MHz oscillator with an innovative clock recovery circuit that locks to the USB signal, providing better than 0.25 percent accuracy and allowing for crystal-less USB operation. Other integration in barcode scanners can include directly driving a buzzer to alert the user when a successful scan has been made, capacitive touch buttons to replace mechanical buttons and HW encryption for wireless scanners that need to send data securely.

Another important design consideration is the flexibility to accommodate change quickly and easily without driving up development cost. To speed development, designers often start with a previous project and modify the setting to fit the new requirements. However, to effectively repurpose the design, it is important to be able to choose and modify the MCU peripherals used and their placement. Most MCUs provide a preset location for peripherals with a fixed alternate selection. The preset pinout often leads to pin conflicts, forcing developers to change their design or move to a larger, more expensive package. An alternative dual-crossbar MCU architecture patented by Silicon Labs (shown in Figure 2) gives developers greater flexibility by enabling them to first select the peripherals needed and then choose the pin placement for them.

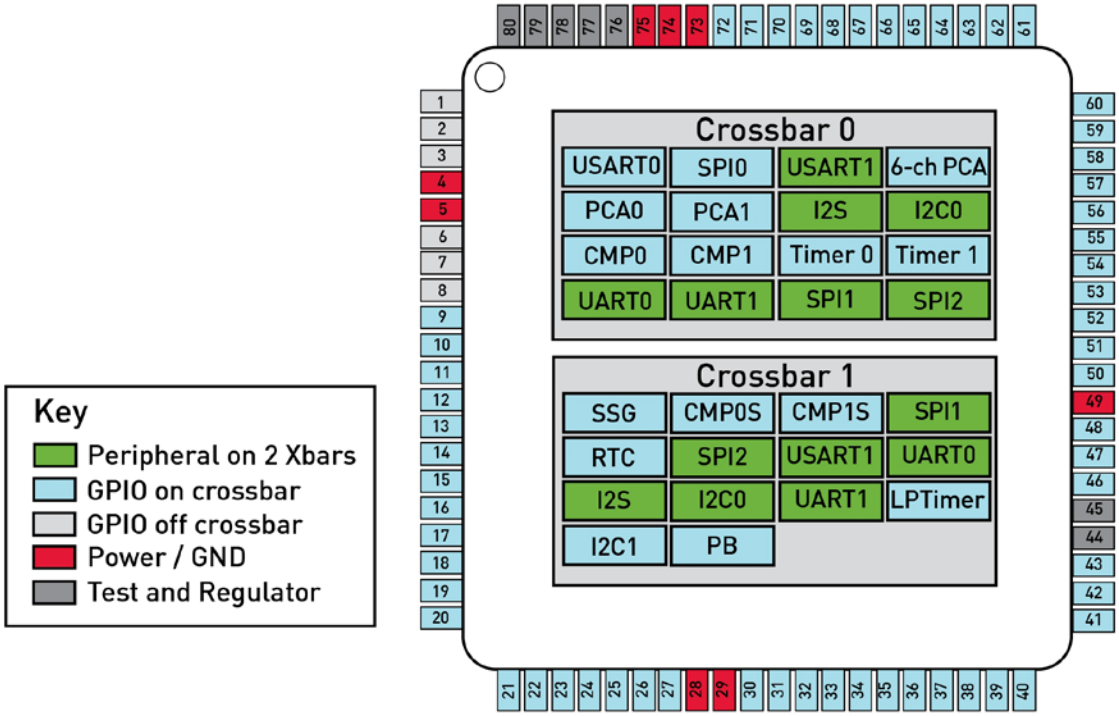

Figure 2: Flexible Dual-Crossbar Architecture Used in Precision32 MCUs

Having the ability to "cherry-pick" only the required peripherals often enables smaller, more cost-effective packaging options. For example, in a communications hub that required four UARTs with flow control (16 pins) and two SPIs (6 pins), a developer should be able to select an MCU with slightly more than 22 I/Os. However, using the standard fixed-architecture, four UARTs and three SPIs would necessitate a 64-pin or even a 100-pin package to achieve the precise peripheral mix. Using a flexible, configurable crossbar technology, the developer could easily fit this peripheral mix into a 40-pin package with several I/Os to spare. In addition, by optimizing the peripheral location, developers can place peripherals close to their connecting circuits, enabling shorter routes and potentially reducing the number of PCB layers needed for the design. Best of all, last-minute design changes can easily be accommodated in software. For example, if the communication hub required another IC with an SPI interface, that's no problem. A third SPI port can easily be added in the same footprint simply by modifying the software.

Given the benefits of a flexible crossbar architecture, are there any downsides to using an MCU with a highly-configurable crossbar? Some developers are concerned that a crossbar architecture may be complex to program. Silicon Labs has addressed this concern by creating AppBuilder, a free software development tool designed to simplify initialization and configuration. The GUI-based AppBuilder tool enables developers to quickly and graphically choose their peripheral mix, select peripheral properties, set up clocking modes and customize pinouts, all without having to read the data sheet. AppBuilder even generates source code that can be used with popular compilers, such as Keil, IAR and GCC.

A final important consideration in selecting a 32-bit MCU is power efficiency. In fact, ultra-low power has become a critical concern in a wide variety of embedded applications. With today's emphasis on "going green" and minimizing energy consumption, designers must pay close attention to their total power budget. There are various methods of significantly reducing energy. The most effective method depends on the end application. For example, with a blood glucose monitor, the patient only uses the device a handful of times throughout the day. The vast majority of the time, the device is in a deep sleep mode. Therefore, in this application, it is most important to minimize the power consumption in sleep modes.

A sensor node device, on the other hand, must constantly be monitoring for an event to happen. If it is constantly monitoring for an event, it must always be in active mode, right? Not really! The sensor node can be in sleep mode, wake up quickly, determine if an event, such as detecting smoke, is happening, and then go back to sleep. In this type of system, it is important to have a low-power sleep mode that includes a real-time clock (RTC) to wake up at a regulator internal, such as every 100 microseconds. It is also important to have a fast wake-up time and a processor to quickly run the fixed number of commands to determine if an event is happening.

Some applications, such as equipment on a factory line, never go to sleep. In these applications, having an MCU with low-power active current is very important. In addition, there are other tricks that can be employed to save power, such as reducing the frequency rate on-the-fly to use only the processing speed needed for a given task.

It can be challenging to find a 32-bit MCU that excels in ultra-low power in sleep mode, active mode, wake-up time and on-the-fly frequency changes. The Precision32 MCU family addresses these challenges by providing several low-power options, as shown in Figure 3. The MCUs can operate below 100 nA and include a brown out detector and 4 kB of RAM retention. The real-time clock can be enabled for an additional 250 nA of current. There is an analog comparator option for an additional 400 nA and even a low-power timer and pulse counter. The MCUs can wake up from low-power sleep mode in microseconds. In addition, the Precision32 MCUs have a very low active mode of 275 µA/MHz, and they feature a sophisticated PLL that can lock to any frequency from 1 - 80 MHz, enabling the developer to optimize power for the task at hand.

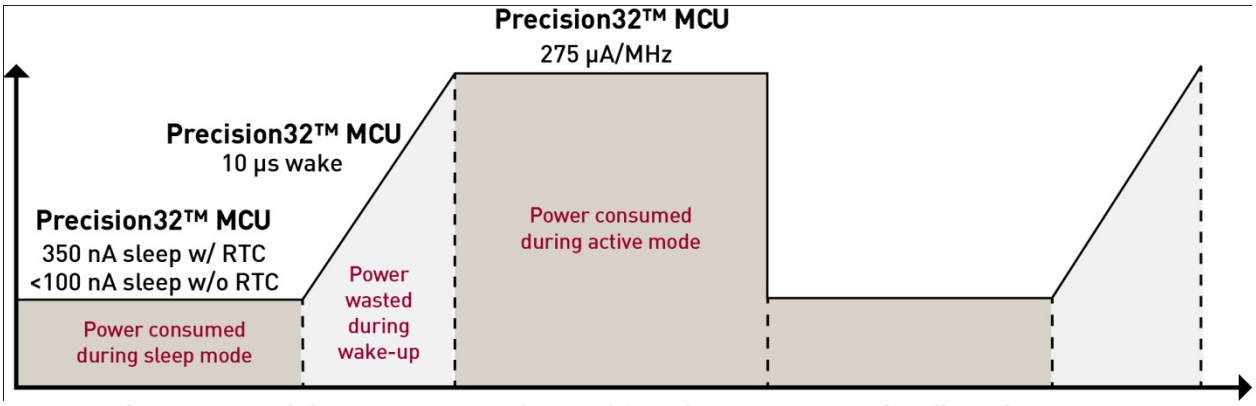

Figure 3: Precision32 MCUs Engineered for Ultra-Low Power in All Modes

At a time when many major MCU suppliers have families of 32-bit devices using the same core with similar memory sizes and large numbers of I/Os and serial peripherals, it is easy for a designer to think that the choice of MCUs in an embedded design doesn't really matter. However, by selecting the right MCU for a given design, developers can significantly reduce development time, power consumption and total system cost while gaining the flexibility to make last-minute changes without a major system redesign. In short, it makes sense to choose a 32-bit MCU with a flexible architecture engineered from the ground up to make the developer's job easier.

# # #