Prepare signals for microcontroller ADCs

Sean Newton and Alec Bath, STMicroelectronics - June 01, 2011

Many microcontrollers have built-in ADCs (analog-to-digital converters), which means you can use a single device to both digitize analog signals and manipulate the results. Once the ADC digitizes a signal, you can take advantage of the microcontroller’s I/O lines to analyze the data, sound an alarm, or control the measured quantity.

For most applications, you must condition the voltages to bring them within the ADC’s input-voltage range. Signal conditioners range from simple resistor-divider networks to amplifiers and filters. You’ll also have to plan for signal frequency and sampling rates when using an ADC. For AC signals above the audio range, you may need a microcontroller that contains an SAR (successive-approximation register) ADC, which can sample at rates of 1 Msample/s (see “Inside an SAR-type ADC”).

Preparing the signals

Before connecting an analog signal to a microcontroller, you must take into account the input-voltage range of the ADC. Consider an STM32 microcontroller as an example. The microcontroller has power pins specifically for the $V_{DDA}$ and $V_{SSA}$ of its ADC, where $V_{DDA}$ is the positive supply. The ADC’s input-voltage range is between 0 V and $V_{REF+}$, where $V_{REF+}$ can be the same as or lower than $V_{DDA}$ but not lower than 2.4 V. If $V_{REF+}$ is equal to $V_{DDA}$, then $V_{REF+}$ can go up to the maximum value of $V_{DDA}$, which is 3.6 V.

<table>
<thead>
<tr>
<th>$V_{REF+}$</th>
<th>μV/bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>586</td>
</tr>
<tr>
<td>2.6</td>
<td>635</td>
</tr>
<tr>
<td>2.8</td>
<td>684</td>
</tr>
<tr>
<td>3.0</td>
<td>732</td>
</tr>
<tr>
<td>3.3</td>
<td>806</td>
</tr>
<tr>
<td>3.6</td>
<td>879</td>
</tr>
</tbody>
</table>
The ADC has 12-bit resolution, which provides 4096 counts: 0 to 4095 decimal or 0x000 to 0xFFF hex. Because \( V_{\text{REF+}} \) ranges from 2.4 V to 3.6 V, it will govern the volts/bits ratio of the ADC. Table 1 shows the voltage-per-bit for several values of \( V_{\text{REF+}} \).

The microvolts/bits ratio is important because you can get finer resolution as you decrease the value of \( V_{\text{REF+}} \). Once the ADC input voltage is in digital form, you can use a software look-up table or an equation to convert the raw bit count to volts. Then, you can use another conversion routine to convert volts into your desired unit of measure, such as degrees Celsius.

Assume that you have a temperature sensor with a range 0 V to 2.4 V where the output is linearly proportional from -20°C to 200°C. In that case, an ADC count of 0 represents -20°C and an ADC count of 4095 represents 200°C if \( V_{\text{REF+}} \) equals 2.4 V. Once you know the volts/bits ratio, you can design signal-conditioning circuits that convert the analog-input signal to the appropriate input-voltage range for the ADC.

Two examples help illustrate how you can design a signal-conditioning circuit. For the first example, suppose you need to monitor a DC bus voltage using a microcontroller’s ADC input. During normal operation, the bus voltage is fixed at 24 VDC. For this example, the ADC will accept a maximum input voltage of 3.3 V. The signal-conditioning circuit need be only a simple voltage divider (Figure 1).

![Figure 1](image1.png)

**Figure 1.** A resistive voltage divider may be all you need to condition a voltage to within an ADC’s input range.

To protect against an overvoltage condition, the design requires you to set the maximum peak bus voltage threshold to 10 times the desired bus voltage, or 240 V. A resistor divider with a 520k resistor and a 7.2k resistor will give a maximum analog input voltage of 3.3 V when the DC bus voltage is 244 V. At 24 V, the analog input voltage will be 328 mV, which is 407 counts in this application. A conversion algorithm or look-up table can convert the digital count into volts for display or other action.

For the second example, imagine you have an AC-measurement application that requires you to sense the rotor position in a three-phase PMSM (permanent-magnet synchronous motor). The motor uses a permanent magnet in the rotor, and it uses digital commutation instead of brushes and copper-commutator segments.

You can find the rotor position using sense resistors in one or more of the three motor windings, from which you can calculate the commutation angle and motor speed. Figure 2 shows that you can
add the sense resistors between the low-side switch and ground.

![Figure 2. Shunt resistors in series with motor windings provide voltages that you can amplify and measure with an ADC.](image)

When the motor’s phase is inactive, you can measure the back-EMF (electromotive force) with an ADC. You need three of these circuits to make the three-phase measurement.

To make the PMSM measurement, you need a simultaneous-sampling ADC because you need to measure all three motor-winding voltages at once. The sense resistor typically has a value of less than 1 Ω. Thus, windings develop only small voltages, and you’ll likely need an op amp to boost the voltage to around 3.3 V (or perhaps slightly less).

**Setting the sample rate and input resistance**

Once you’ve conditioned the ADC’s input voltage to the proper levels, you must set the ADC’s sample rate. The ADC on an STM32 microcontroller, for example, can sample at rates up to 1 Msample/s. You must take into account the measured signal’s analog characteristics to find the best sample rate. Slow-changing signals such as temperature and pressure don’t need a high sample rate; sample rates of 1 s to several minutes may be sufficient. Fast-changing signals, however, need higher sample rates.

At 1 Msample/s, a signal’s highest frequency component must be less than 500 kHz to meet the Nyquist criteria. Otherwise, aliasing will produce unwanted artifacts. For an accurate signal representation, you should use a sample rate that’s 10 times the measured signal’s highest frequency, which could limit your signal to 100 kHz. Audio applications fit nicely within this limitation. Ultrasonic applications, though, could soon overrun the ADC’s sampling ability, resulting in poor digital-signal definition.

The ADC’s input impedance relative to that of the signal source is also important because it can limit the ADC’s maximum sample rate. Capacitance associated with the internal ADC circuitry of the microcontroller, combined with the resistance to the incoming signal source, forms a low-pass filter. The filter’s RC time constant affects the time needed for the signal to stabilize before the ADC can sample it. If the signal changes rapidly over time, the low-pass filter can distort a signal and thus its digital equivalent.
You can compensate for high source resistance by adjusting the ADC’s sample rate. A slower sample rate provides more time for a signal to settle, if you don’t sample too slowly and produce aliasing. Table 2 shows the relationship between the sample time and the signal-source input resistance for an STM32.

You must also consider other ADC parameters such as accuracy, linearity, and resolution before connecting an analog circuit to a microcontroller. But a basic understanding of the ADC’s voltage input range, sampling frequency, and input-resistance requirements can get you started connecting analog circuits to a microcontroller’s ADC.

**Handling the data**

After a microcontroller’s ADC digitizes analog signals, what do you do with the data? Depending on the sample rate and processing speed of the microcontroller, you may be able to process and act upon the data in real time. For example, you can use software to process the data using digital filters, or you can operate control loops with the microcontroller’s digital I/O lines.

Most 32-bit microcontrollers include a DMA (direct-memory access) controller, which lets you move data directly into system memory. A DMA controller can relieve the microcontroller’s CPU core from transferring data into memory, which increases performance.

In its simplest form, a DMA controller consists of three parts: a memory source, a memory destination, and a trigger. The ADC’s data register is usually the memory source. An array of data in the microcontroller’s RAM is the destination, and the ADC’s conversion-complete flag is the trigger. Because you usually want to capture a large number of samples without overwriting any data in memory, you need to increment a data-array pointer for each new DMA transfer. You can configure either the source or destination memory locations for auto-increment with the STM32. Other DMA options may include the ability to move different data types, such as bytes, 16-bit half-words, or 32-bit words into memory.

Microcontrollers may have several communications peripherals. Traditional communications ports
include I2C, SPI, and USART (RS-232). Advanced communications ports include CAN, USB, and Ethernet. Communications ports let you move data to networked devices or to a local PC.

Microcontrollers also contain storage memory (typically 32 kbytes to 512 kbytes) of flash and EEPROM. If you need more storage, you can connect the microcontroller to an SD (secure digital) memory card that you can easily remove or replace. External memory controllers provide access to large amounts of flash memory. Through a USB port, you can connect to a flash drive or hard drive and store your data for analysis. T&MW

Sean Newton is a field applications engineer in STMicroelectronics’ marketing organization supporting microcontrollers. He has worked at the company since 2001 supporting microcontrollers, microprocessor units, and smart card products. Prior to ST, Newton worked in several manufacturing facilities with Electronic Data Systems. He graduated with a degree in electrical engineering and computer science from the University of Nevada, Reno.

Alec Bath is a field applications engineer in the microcontroller division of STMicroelectronics. He has worked at ST since 2000 supporting microcontroller and memory products. Prior to ST, Bath worked for Motorola. As an avid adventure racer and ultra marathoner, he has been tinkering with home-brew microcontroller-controlled LED headlamp systems.

Inside an SAR-type ADC

An SAR (successive-approximation register) ADC provides relatively high-speed sampling with enough resolution for many measurement applications. A typical SAR-type ADC consists of a successive-approximation register, a DAC (digital-to-analog converter), a comparator, and a sample-and-hold circuit (see figure). The comparator, sample-and-hold circuit, and the output of the DAC contain the analog parts of the ADC, while the SAR and the digital input of the DAC contain the digital parts of the circuit.

The sample-and-hold circuit samples the analog input signal for a defined period, called the sample time. At the end of the sample time, the analog input signal disconnects from the sample circuit and the capture voltage is held for conversion.

The SAR starts by setting the MSB (most-significant bit) of the DAC to 1, driving the DAC to $V_{REF}/2$. The output of the DAC drives the comparator, whose result is a digital 0 or 1 depending upon whether the analog input voltage is greater than or less than the DAC output voltage. If the analog input voltage is less than the DAC output voltage, the current bit is reset and stored in the ADC’s end-of-conversion register.

For each successive ADC clock pulse, the next bit of the SAR is set and compared against the analog input voltage. The comparator sets its output high or low depending on the results of the comparison. The process continues until the comparator samples the LSB (least-significant bit) and the complete digital value is stored in the end-of-conversion register. Upon completion, the ADC typically triggers an event to signal that the conversion is complete.

The ADC contains digital and analog portions that separate into two domains: the analog domain, which is powered by the voltage differential between $V_{DDA}$ and $V_{SSA}$, and the digital domain, which is powered by the voltage between $V_{DD}$ and $V_{SS}$. Separating the digital and analog power supplies provides isolation and helps reduce switching noise induced by the digital side of the circuit. Along with a separate analog power-supply voltage, an external analog-reference voltage ($V_{REF+}$ and $V_{REF-}$) is available that lets you adjust the input-voltage scale.—Sean Newton and Alec Bath