ECSE 444: Microprocessors
Lab 3: ADC

Abstract

In this lab you will (a) learn how to configure an analog to digital converter (ADC), (b) use it to sample the internal voltage and temperature of the STM32L475VG MCU, and then (c) calculate the operating temperature of your processor.

Deliverables for demonstration

- C implementation reading internal temperature sensor.
- C implementation reading internal reference voltage.
- C implementation scaling ADC outputs to calculate temperature in degrees C.
    - Using 12-bit resolution
    - Using some other resolution (6-, 8-, or 10-bit)

Grading

- Reading internal temperature sensor
    - 30%
- Reading internal reference voltage
    - 30%
- Calculating temperature
    - 10% Reading TS_CAL values
    - 10% Scaling TS_CAL values to account for VREFINT
    - 10% Scaling TS_DATA values to account for resolution
    - 10% Putting it all together

Changelog

- 10-Aug-2020 Initial revision.

Overview

In this lab, we'll take our first look at analog to digital conversion. Sensors, in general, provide a measurement of a physical quantity using a voltage; analog to digital converters (ADC) take that signal, continuous in time and in range, and discretize it, again, in time and range, for use in a processor. This involves sampling the voltage, and then representing it with a binary number. First, we'll configure ADCs to read the internal reference voltage used for analog to digital conversion, and the internal temperature sensor. Second, we'll use these values to determine the temperature of the processor in degrees Celsius.

Resources

HAL Driver User Manual
STM32L475VG Datasheet
STM32L47xxx Reference Manual

Analog to Digital Conversion

Before we configure the board, it's important to establish a bit of context. The code for this lab is straightforward; configuring the ADC properly so it produces valid output, and manipulating that output to recover the real value it represents, is not.

ADC Sample Timing

Analog to digital conversion takes a sample (in time) of a signal (usually a voltage), and converts it into a binary representation. The ADCs available on the STM32L475VG can produce 6- to 12-bit representations. More bits means more *precision* (a smaller voltage range is represented by a single digital value); in general, higher precision conversion requires more *time*. That's because the STM32L475VG ADCs are *successive approximation* ADCs, which refine their conversion over a series of steps. We'll cover that in greater detail in a lecture later; the key constraint for this lab is that the ADCs *must be given sufficient time to do their work*. It's up to us to figure out (a) how long the ADC needs, and (b) how to configure the board accordingly.

To get started, for an overview of how the internal temperature works, read Section 18.4.32 in the STM32L47xxx Reference Manual. Note (2) under *the temperature*, and find the appropriate sampling time when reading the temperature sensor in the STM32L475VG Datasheet.

Now read Section 18.4.16 in the STM32L47xxx Reference Manual. Note the assumptions about (a) sampling time, and (b) clock frequency. When configuring the board, we'll have to pick a clock frequency for the processor and for the ADC, and choose the sampling cycle for the ADC. For an example of how ADC sampling cycle relates to sampling time, see Table 67 in the STM32L475VG Datasheet.

Between the sampling time requirements for the sensor, and configuration options for the ADC, you should now have enough information to configure your board properly.

ADC Output Scaling

ADCs convert voltages into a digital representation by comparing the analog input with an internal reference voltage. Sensors have a range of analog output; under ideal circumstances, the sensor and ADC are calibrated such that this range maps onto the full range of the ADC's digital output. This calibration occurs under an assumption about the internal reference voltage

used by the ADC; understanding calibration conditions, and how they differ from operating conditions, is essential for writing software that correctly interprets sensor values (binary numbers scaled according to what is sensed relative to an internal reference voltage) as physical quantities (e.g., temperature).

For example, the processor temperature sensor in your development board has been factory calibrated at two temperatures. This means that the behavior of your sensor has been characterized under controlled conditions: if the reference voltage is X, and the temperature is Y, the ADC outputs Z. Given the ADC output at these temperatures, and the assumption that sensor output voltage varies linearly with temperature, we can interpolate to determine operating temperature. Note that these ADC outputs are taken assuming an ADC resolution of 12 bits.

In order to properly scale the ADC output for the temperature sensor, we need (a) these constants, and (b) an equation to relate them. The equation you've seen; it's in Section 18.4.32 of the STM32L47xxx Reference Manual, noted above. The equation also lists the various names of constants that you'll need. Note: the calibration output ADC values *are stored in the memory of your processor*. The STM32L475VG Datasheet has more information.

Additional notes:

- The constants in the reference manuals have different names in source code and headers used by STM32CubeIDE; you'll have to do some detective work to figure it out. (Hint: the names in the reference manuals do appear in the source and headers, even if they aren't used to identify the constants and addresses we're looking for.)
- You'll need to use the internal voltage reference sensor, too, which has also been calibrated as described above; repeat the above process for the internal reference voltage. Section 18.4.34 in the STM32L47xxx Reference Manual is a good place to start.

With information on calibration conditions, and where to find calibrated ADC outputs, you should now have the information you need to write code to scale the ADC output for the temperature sensor.

Configuring the Board

Initialization

As before, start a new project in STM32CubeMX and initialize the configuration, reviewing instructions in Lab 0 and Lab 1 if necessary.

ADC Configuration

You'll need to configure two ADCs: one to read the internal temperature sensor, and another to read the internal voltage reference. You'll find the ADCs on the left side of the *Pinout & Configuration* tab, under *Analog*; there are three, ADC1-ADC3. Each ADC has a number of channels that are available (under *Mode*); IN1-INX should be disabled in each case. Enable the temperature sensor channel in one ADC, and the Vrefint channel in another.

Under *Configuration*, and *Parameter Settings*, you'll see that an ADC is the most complex peripheral we've configured to date. Under *ADC_Settings*, make note of the options for *Resolution*. Under *ADC_Regular_ConversionMode* > *Rank* you'll find options for *Sampling Time*. You need to choose a sampling time for each ADC that satisfies sampling time guidelines for the sensors and takes the ADC clock frequency into account. The ADC clock frequency can be changed under the *Clock Configuration* tab.

Note: you can empirically test if your ADC has too short a sampling time. If you increase the sampling time, and the sampled value increases significantly, then you need to give the sensor's output more time to converge.
Now generate your code and load your project in STM32CubeIDE.

Reading ADC Output

The HAL Driver User Manual describes how to make use of the ADC API in Section 7.2. Note that STM32CubeMX code generation takes care of configuring the ADC; all we need to do is follow the instructions on how to perform ADC conversion using *polling*. *Polling* means waiting for the ADC to finish it's conversion; alternatively, interrupts can be used to notify user code when a conversion has been completed. This is the subject of Lab 4.

Write code to poll each ADC approximately every 100 ms. Verify your choice of ADC sampling time; observe the effect of changing ADC resolution.

Scaling ADC Output

Now extend your code to calculate temperature from the temperature sensor's ADC output.

Notes:

- You need to read the calibrated ADC outputs from memory; the calibration temperatures are defined as constants, but can also be found in documentation.
- You need to scale the calibrated ADC outputs based on how the internal reference voltage differs from calibration conditions. Temperature sensor output is linear with respect to reference voltage.
- You need to scale the ADC output for the temperature sensor whenever you use an output resolution other than 12-bit to account for the difference in output range.
- And be careful with operator order, data type casting, etc!

- You can check your implementation using the ADC calibration output values.

The scaled temperature value is expected to be close to room temperature.

For the demo, please ensure to have a way to heat up the board at the ready. We recommend a blow dryer, but any method that does not damage the board is also welcomed.