Analog to Digital Conversion

Measuring Analog Signals with a Digital System.

Preparations:

Read and try to understand the datasheet for the MCP3208, a 12-bit serial readout "Successive Approximation Register" (SAR) Analog to Digital Converter (ADC). This data sheet is *not* trivial! Specifically**, read the description**, which tells you what the device does, and **from the electrical specifications note the maximum (and minimum) voltages** that can be applied to the power input (VDD) and the analog inputs (IN). Next read the **pin descriptions**, **device operation, and serial communications**. Some of the details of this chip are also explained in the "Exploring Raspberry Pi" book, chapter 9.

Next read the datasheet for the LM385, a precision voltage reference. You will be using the LM385Z-2.5G, which is the 2.5V version of this device. This is an easier to understand device. It acts as a very precise zener diode. You can read more about zener diodes in "Practical Electronics" chapter 4.2.6. You also used a zener diode in Lab1, and you should compare the results you get here to that lab. You may need to repeat some lab 1 measurements to do so.

Finally, read the datasheet for the MCP9700, or TMP36, an analog temperature sensor, depending on which version you have. These two versions of this device are by different manufacturers, but otherwise they are very similar.

All datasheets are available from the course website on the Digital Resources page in the advanced components section.

In Lab Preparation:

Make sure you have the most recent updates in the course libraries on your Raspberry Pi. Log in and go to the "Phys605" directory (this is a "git clone" of the GitHub site https://github.com/mholtrop/Phys605. You can recreate the directory with the "git clone https://github.com/mholtrop/Phys605" command. On your own computer, you could also download this as zip archive). Once there, execute the command "git pull". This will update the libraries. Specifically, it will get the drivers in the Python/DevLib library. Check that the MCP320x.py driver is there. If not, get help.

READ:

- Lecture Notes.
- Data sheets: MCP3208, LM385Z-2.5G, MCP9700 (TMP36)
  - These data sheets are *not* trivial, see above!
- The DevLib code on GitHub for the MCP320x ADC and the MAX7219 display: https://github.com/mholtrop/Phys605/tree/master/Python/DevLib
  - You need to make sure you are familiar with this code so you can use it in the lab without spending a lot of time trying to understand what it is supposed to do. Specifically, make sure you understand the pin configurations that the code expects and how you can modify them to you own needs.
  - You will need to write a program that reads out an ADC channel and then display the measured *voltage* on the display.
  - You will need to write another program that reads out an ADC channel which is connected to the MCP9700, possibly using your instrumentation amplifier, and read the *temperature.*
  - Example code can be found in the Phys605/Python directory on your RPi.
- **Practical Electronics:** Chapter 12.9.5 and 12.9.6
- **Exploring Raspberry Pi:** Chapter 9 at least up to page373

Goals:

- Learn about Analog to Digital conversion.
- Create an accurate reference voltage.
- Learn more about the SPI bus for communication between chips, sensors and microprocessors.

Expectations:

- Keep good and accurate notes in your lab book.
- Carefully follow these instructions so you end up with the right measurements.
- Compare measurements with expected values.

Important

This is a multi-part lab where you will be putting together and investigating a data acquisition system. You are more likely to be successful if you plan ahead, and lay your circuit out carefully on your bread-board. Do not make assumptions, check the datasheet! Try to keep the circuitry compact in one area of the workspace, so that you have other parts free to make other circuits. You can use the circuits you are building for future experiments.

Stick with 3.3V:

The CMOS chips you are using work excellently well at 3.3 V[1]. Since at some point you will be connecting the circuit to the Raspberry Pi, please stay at 3.3 V throughout. Before connecting any circuit to the RPi, please double check your power supply. Only use the power supply from the Analog Discovery or the Raspberry Pi. Remember that when you use more than one device (i.e. RPi + AD), you need to make sure that the *ground* of these devices are connected to each other.

Introduction:

Nearly all measurements of properties of the physical world involve an analog signal, which we will need to convert to a digital signal if we want to make use of our computers to analyze the data. The simplest, but also most cumbersome, method of conversion is for a you to read the analog dial and enter the number into the computer. If you want the computer to acquire the information on its own, you need to perform an analog to digital conversion. Most such conversions are done with an ADC chip. We will be exploring an ADC chip, the MCP3208, which implements a successive approximation register, a method of obtaining the digital value of an analog voltage in several steps, each step increasing the precision. This is a natural choice for ADCs that are read out using a serial interface. The MCP3208 uses the serial peripheral interface (SPI) protocol to communicate with the microprocessor.

The SPI protocol is a four wire serial protocol. It specifies a chip select that is active low, a clock that can be either active low or active high, and a Master Out/Slave In (MOSI) and Master In/Slave Out (MISO) for carrying the information. A good, detailed description of the SPI bus can be found online[2].

The ADC circuit

You will need to carefully setup the ADC chip for testing. Power the chip using the Raspberry Pi supply at 3.3 V. From the datasheet of the ADC you can learn that there are two ground pins, one for the digital section (DGND) of the chip and one for the analog section (AGND) of the chip. Connect each of these to your ground rail. There is a Vdd to power the chip, which needs to be connected to 3.3 V, and a Vref, which is the reference voltage the ADC uses to compare the input voltages to. To summarize the Vref: this is the voltage that is used in the SAR comparator to compare against, and thus this needs to be a well-known voltage so that you convert the codes from the ADC to actual

---

[1] The only exception perhaps is the 8-Segment Display, which is at full brightness when powered with 5V, but works fine a 3.3V.

[2] See for instance wikipedia : https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

voltages. It also means that this is the *maximum voltage* that the ADC inputs can register. You must have Vref <= Vcc. See the datasheet for details on Vref. Initially, you want to connect Vref to 3.3 V. Later you can build a dedicated reference voltage for extra precision.

For the digital communications, you want to connect CS-bar/SHDN, CLK, Din and Dout to the SPI GPIO pins of the Raspberry Pi. The driver in DevLib, MCP320x.py has a testing main function that expects the CE0 chip select. With minor modifications to the code, you can also use standard GPIO pins of the RPi. See the code for details.

Testing the ADC communications.

To test the ADC, you will want to get two potentiometers (variable resistors), one of 10kΩ and one of 100kΩ, which you use as a voltage divider to produce an input for the ADC. Connect one end of the potentiometer to ground, the other end to 3.3V. The middle pin, which is the slider, can then be dialed between 0 and 3.3V depending on the position of the shaft. Verify with your voltmeter that this is working as expected, and then connect the center pins of the two potentiometers to two separate ADC input channels. Connect the other input channels to ground or 3.3V or also to the potentiometers.

Setup the Analog Discovery logic section to look at the digital signals between the RPi and the ADC. You can do this by adding two SPI bus decoders, one for the MOSI and one for the MISO signal. The chip select and clock will be the same for these two. You will want to set the trigger to start on a low signal on the chip select. Remember to connect the ground from the AD to the ground of the RPi.

There is example code available on GitHub in the Python/ADC directory. You should already have this on your RPi in the "Phys605" directory. You would want to do a “git pull” (after you cd to the directory) to update the code, and then navigate to the Python/ADC directory.

TASKS:

- Wire up the ADC.
  - Copy down your *exact* circuit into your lab book.
  - Make sure to write down which GPIO pin is used for what function.
- Connect the AD and setup the logic analyzer to read the SPI signals.
- Use the “ReadADC.py” code to read the different channels of the ADC.
  - Verify the numbers that you obtain:
    - Does it go down to zero? Can you get a 1, or is that always skipped?
    - Use your voltmeter to measure the input voltage on one of the ADC channels. Does the value you read from the ADC correspond to that voltage? (yes you need to convert the integer to the voltage properly)
    - Trick question: page 17 of the data sheet shows the conversion formula. Do you agree with this formula, or should you be using 4095 instead of 4096 when you convert the digital code to a voltage?
      - Can you actually *measure* which of these two is the correct formula!

The Voltage Reference

An ADC compares the input voltage to a reference voltage. If the sensor that you are using provides an analog output that is *proportional* to the supply voltage (Vdd), then there is no problem in using Vdd for both that sensor and the reference voltage. Your potentiometer, which “senses” the position of the shaft, could be considered as such a sensor. Other sensors will have an analog output that is an absolute voltage value. In that case you will need an absolute voltage for your reference as well, so that your measurement does not depend on the exact value of Vdd, which would be a poor design, especially for battery operated equipment. In this section you will build a voltage reference.

The device you will use is the LM385Z-2.5G. This precision reference behaves like an ideal Zener diode, though it contains far more components inside. Your first step is to use the AD to create a current versus voltage curve, like figure 2 in the datasheet for the LM385. Note that that figure is for the LM385-1.2, the 1.2V version, while we are using the 2.5V one. Creating the I-V curve is similar to what you did for analog lab 2. You only need to make the curve for the reverse biased “diode”. It will be sufficient to go up to 5mA in current, but make sure you do not exceed

20mA in current.

Tasks:

- Build a test circuit for the I-V curve, and record the schematic in your notebook.
  - Choose an appropriate resistor to limit the current through the device.
  - Use channel 1 of the AD scope to measure the voltage across the "diode", and channel 2 to measure the current, by measuring the voltage across the resistor. Since the AD has differential inputs, this is much easier than with the Rigol scope.
  - Use the WaveGen to create the voltage source. Set it to go from 0 to +5V.
  - You can now see the I-V curve in the x-y display of the AD. Record the actual data from the scope for a more precise offline analysis for your report.
- From the curve you created, choose an operating point (i.e. how much current through the diode) ***below*** 1mA, and choose the appropriate resistor to set the "diode" to that operating point if the supply voltage is 3.3V.
- Now, using this information, build your voltage reference and measure and record the exact voltage using your voltmeter.
  - Is your regular voltmeter precise enough to measure 2.5V with an equivalence of 12 bits?
  - You can get a high precision voltmeter if you would like.
- Connect the reference voltage you created, Ref-V, to an *input* of the ADC and measure the ADC values for >5 samples.
  - Is every sample identical, or do you get different values each time?
    - Comment on whether or not *oversampling* makes sense in this situation.
  - Any fluctuation of the value would be due to changes in Vdd, the supply voltage. You can use this information to make a correction to values you read from other channels.
- You can also use the Ref-V to calculate what Vdd is. For a battery powered device, this would be a way to check the level of the battery. Disconnect the Vdd from the RPi, and use the AD power supply to mimic a battery on Vdd. Use your ADC and Ref-V to measure the Vdd for several setting of Vdd between 3.5V and 2V.
  - Note that the ADC cannot measure a voltage larger than Vref, so you cannot measure Vdd directly. There are two ways to measure Vdd:
    - Use a voltage divider to measure Vdd on an ADC input, and put Ref-V to Vref (of the ADC). Make sure you *measured* the properties of the voltage divider.
    - Put Ref-V on an ADC input and Vref on Vdd, and calculate Vdd using this information.
  - Check that your measurement is correct by varying Vdd between 2 and 3.5 V and record the data.
  - Can you use each method down to 2V, or does one of them stop working at 2.5V?

Temperature Measurement

You may have used the MCP9700 (or TMP36) already at the previous analog lab, when you made an instrumentation amplifier for it. Using the datasheet, hook up a TMP36, powered at 3.3V.

Tasks:

- Measure and record the output voltage of the TMP36 with the voltmeter.
  - Warm up the the MCP9700 with your hand, and measure again. Are the values what you would expect?
- Connect the MCP9700 output directly to one channel of the ADC.
- Using the instrumentation amplifier from your previous analog lab, connect the MCP9700 output to your amplifier input, and connect the output of the amplifier to another channel of the ADC.
- Since the MCP9700 will output voltages up to 3.3V, use your Ref-V to another input channel on the ADC as a reference voltage. This will provide the virtual reference value, and works even if the input to other channels get larger than 2.5V.
  - Test the circuits by taking a few measurements, changing the temperature by warming up the temperature sensor.
- Write a Python code to take a temperature reading once every second, converting the ADC values to degrees centigrade.

- Make sure your code makes use of the Ref-V reading to correct for changes in Vdd.
- Display the output value, in centigrade, on the 7-segment display.
- Test everything by warming the sensor.
- Test you code and the robustness of the temperature measurement device by lowering the supply voltage in steps to 2.5V. Do the temperature readings change?

In your Progress Reports:

Progress report:

- Make sure you **answer all the questions** in this lab writeup.
- Fully describe the ADC circuit and your test results.
- **Describe the voltage reference** testing circuit and AD setup.
  - Make a nice graph (not a screen shot) of the data you obtained for the I-V curve.
- Describe the voltage reference circuit you build (Ref-V), and **show your calculation for R**.
  - Have a circuit diagram for the voltage reference part.
- Describe your tests of the temperature sensor.
- Describe your temperature measurement setup.
  - Include a detailed circuit diagram.
  - Include the diagram for the instrumentation amplifier.
- Show the results of your temperature measurements.
  - Make a table of the measurements
  - Compare the temperatures obtain from directly reading the sensors to those where you used the instrumentation amplifier.
- **Submit your code in an appendix** to the report.
- What you have learned and who helped you section.