Institute for Cognitive Systems
Technische Universität München
Prof. Gordon Cheng

Humanoid Sensors and Actuators - Tutorial 2 Part 1

Florian Bergner

Microcontrollers: Analog-Digital-Comparator (ADC) (50 points)

In the first part of tutorial 2 we will have a look at the analog-digital comparator (ADC) peripheral block of the microcontroller. The ADC allows us to sample analog signals, thus to embed the microcontroller in analog circuits.
In this tutorial we will learn:

- how to use the ADC peripheral block
- how to sample analog signals and send them to the computer
- how to measure the capacitance in an RC serial circuit

1 Using the ADC Peripheral Block (50 points)

1.1 Testing the AVR Potentiometer Simulator

This specialized simulator connects a simulated potentiometer to a desired pin (`ADC0` - `ADC7`) of the ADC peripheral block of the simulated microcontroller. This simulator also connects to the UART of the AVR and writes the received bytes into a file `adcLog.csv`. The project folder of the simulator also contains a Matlab/Octave script `plotCSV.m` to plot the content of this log file.
You can download the source code of the simulator for this tutorial by cloning its project as follows:

```
git -c http.sslVerify=false clone "https://gitlab.ics.ei.tum.de/flo/
    TumIcsAvrSimPotentiometerTemplate.git"
```

Then you change into the root folder of the project, create the build folder, and compile the project:

```
cd TumIcsAvrSimPotentiometerTemplate
mkdir build
cd build
cmake ..
make
cd ..
```

Please download the test program `adc_uart_template.elf` from Moodle and place it into the root folder `TumIcsAvrSimPotentiometerTemplate` of the simulator project. Now, you can run the simulator with this AVR program:

```
./avr_sim_poti_template.sh adc_uart_template.elf
```

The simulator should start. You should see the dial GUI of the potentiometer and current measured measured value in ADC ticks in the range 0 to 255. The value should change when you move the potentiometer.
You can also investigate the operation of the ADC and UART peripheral in the trace file `gtkwave_trace.vcd`. Furthermore you can use Octave or Matlab to run the script `plotCSV.m` and plot the voltage sampled by the ADC.

1.2 Sampling analog signals with the ADC (20 points)

Please use the template project `Atmega32Template` as basis for the tasks introduced in this section. Please submit the code you created as specified in the tasks. You can clone the template project as follows:

```
git -c http.sslVerify=false clone "https://gitlab.ics.ei.tum.de/flo/
   Atmega32Template.git"
```

T.1.1 **(8 points)** Consult the data sheet `atmega32.pdf` of the AVR Atmega32. Implement your own program in C to sample analog signals with the ADC considering the following instructions to set the proper bits in the registers:

- **(2 points)** The ADC uses the AVCC voltage (5 V) and assume that an external decoupling capacitor is connected to the AREF pin.
- **(2 points)** The ADC uses a prescaler of 2 such that the operating frequency of the ADC is half the CPU frequency, in our case 500 kHz.
- **(2 points)** The ADC operates in the free running mode.
- **(2 points)** The ADC operates in the 8 bit sampling mode, i.e. the correct 8 bits of a 8 bit sample are loaded into one sample register of the ADC.

T.1.2 **(2 points)** You use the AVR UART library introduced in the `Atmega32Uart512BitAdderTemplate` project to communicate with the computer. You will need to add the `find_package` command to the `CMakeLists.txt` cmake project file of your project. You can find this command in the `CMakeLists.txt` file of the `Atmega32Uart512BitAdderTemplate` project.

T.1.3 **(4 points)** Implement a function to sample and read the 8bit ADC value for a specified channel using the following function declaration:

```
void adc_readBlocking(uint8_t* b, uint8_t ch);
```

- **(2 points)** The function uses the `ch` argument and samples the correct channel.

– **(2 points)** The function samples and returns the correct 8 bits of the sample.

T.1.4 **(2 points)** Send the samples of pin `ADC2` in binary format to the computer.

T.1.5 **(2 points)** Connect the potentiometer to the `ADC2` pin, such that you can use the potentiometer to set voltages between 0V and 5V. Run your MCU program with the simulator and capture the data while turning the potentiometer. The computer program generates a CSV file `adcLog.csv`. Plot this file using Octave or Matlab and the script `plotCSV.m`.

You submit the following files containing your solution for tasks **T.1.1** to **T.1.5**:

- `main_adc_poti.c`
- `adc_poti.elf`
- `adc_poti.csv` containing the sample of your potentiometer experiment
- a screenshot `adc_poti.png` of the plot that you created with the `adc_poti.csv`

1.3 Measuring the capacitance of a capacitor with the ADC (10 points)

Now, we use a simple RC series circuit to measure the capacitance of a capacitor with the ADC. The resistance is known, as is the sampling frequency.
To test the AVR program you will implement in the following tasks, we need a new AVR Simulator project that simulates the RC series circuits. The simulator project is similar to previous `TumIcsAvrSimPotentiometerTemplate` project, we just exchange the potentiometer with the RC series circuit. The simulated RC series circuit can be connected to a desired ADC pin (`ADC0` - `ADC7`) and a desired charge/discharge pin. This simulator also connects to the UART of the AVR and writes the received bytes into the file `adcLog.csv`.
You can download the source code of the simulator for this tutorial by cloning its project as follows:

```
git -c http.sslVerify=false clone "https://gitlab.ics.ei.tum.de/flo/
    TumIcsAvrSimRCSeriesCircuitTemplate.git"
```

For implementing the AVR program, please use the template project `Atmega32AdcTemplate` as basis for the tasks introduced in this section. Please submit the code you created as specified in the tasks. You can clone the template project as follows:

```
git -c http.sslVerify=false clone "https://gitlab.ics.ei.tum.de/flo/
    Atmega32AdcTemplate.git"
```

T.1.6 **(2 points)** Use the pin `PC1` to charge/discharge a 1 µF capacitor via a 1 kΩ resistor.

T.1.7 **(4 points)** Use the `ADC0` pin to sample the loading curve with maximum ADC sampling frequency. To minimize any delays between samples first store 1024 samples into the SRAM.

T.1.8 **(2 points)** Then send the 1024 samples you stored into the SRAM to the computer using the

```
void uart_writeBlocking(const uint8_t* d, uint8_t size);
```

function.

T.1.9 **(2 points)** Save the CSV file of the 1024 samples and plot it. Note: You will need this CSV file to answer some questions in your report.

You submit the following files containing your solution for tasks **T.1.6** to **T.1.9**:

- `main_adc_rcsc.c`
- `adc_rcsc.elf`
- `adc_rcsc.csv` containing the samples of your RC series circuit experiment
- a screenshot `adc_rcsc.png` of the plot that you created with the `adc_rcsc.csv`

1.4 Report (20 points)

R.1.1 **(4 points)** In **T.1.1** you use the ADC with specific settings and a CPU frequency of 1 MHz. How many CPU cycles pass between the acquisitions of two ADC samples? Elaborate and explain.

R.1.2 **(2 points)** What is then the sampling frequency of the ADC?

R.1.3 **(2 points)** Assuming that each iteration of your for-loop for the 1024 samples (**T.1.7**) takes 32 µs, what is limiting your actual sampling frequency:

- the iteration time of the for-loop (32 µs), or
- the sampling time of the ADC in free running mode?

R.1.4 **(2 points)** Use your results of **T.1.9** and measure the 3τ ($\tau = RC$) value. Remember the capacitance is not known and will be determined using the τ value.

R.1.5 **(2 points)** What ADC tick value (range: 0 to 255) did you use for measuring 3τ? Please explain your computations in detail.

R.1.6 **(2 points)** What voltage level does this ADC value represent? Please explain your computations in detail.

R.1.7 **(2 points)** Use the τ value you determined in **R.1.4** and the given resistance $R = 1\,\text{k}\Omega$ and calculate the capacitance C of the capacitor.

R.1.8 **(4 points)** We discussed the operation of the ADC in the lecture. Let's assume that the conversion logic uses a digital counter with a DAC to generate a voltage ramp. Considering the Sample&Hold time (see datasheet) and the total conversion time of the ADC (see datasheet, **R.1.1**) with the settings used in **T.1.1**, what would be the minimal frequency of the clock that drives the digital counter generating that voltage ramp?