Exercise 6: The ADC

Lab IoT

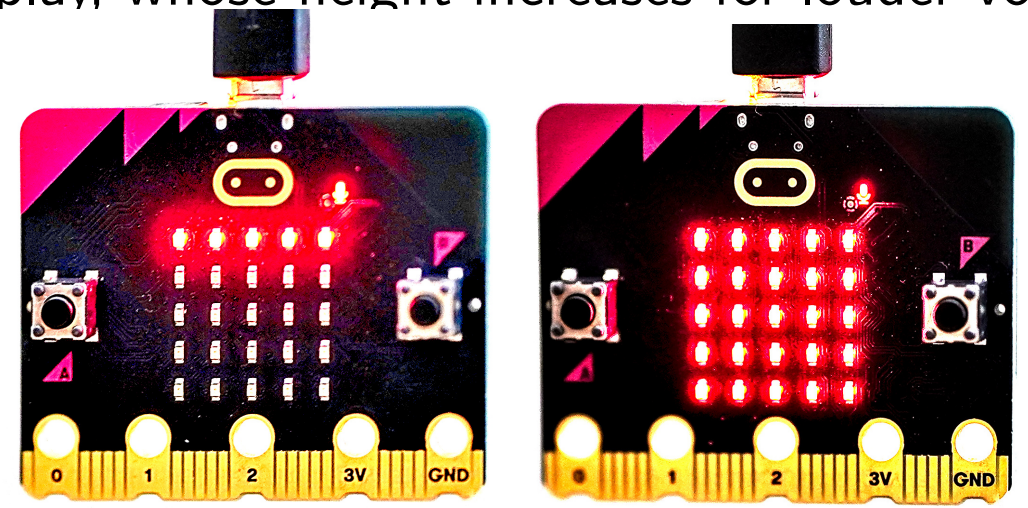
Philipp H. Kindt

Assistant Professorship for Pervasive Computing Systems (PCS)
TU Chemnitz

November 11, 2021

Goals

- In this exercise, we convert an analog signal into a digital signal using the ADC.
- In particular, we will convert the signal of a microphone on the Micro:bit board.
- The goal is to develop a volume meter to measure how loud the ambient sound is.
- The sound volume should be visualized using a bar on the LED display, whose height increases for louder volumes.



The ADC

- The ADC converts an analog signal into a digital one.
- In other words, a certain input voltage will be translated into a corresponding integer value.
- The input voltage is measured against a certain reference voltage V_{ref}. If the input is 0, then the converted data is 0. If the input it V_{ref}, it is 2^N, where N is the resolution of the ADC (in our case, 12 bit).
- We use a microphone as an analog source. More specific, we consider an integrated microphone, which combines a MEMs microphone with an amplifier.
- A low volume corresponds to a low voltage and hence digitized value, a very loud sound corresponds to a high voltage and hence ADC value.

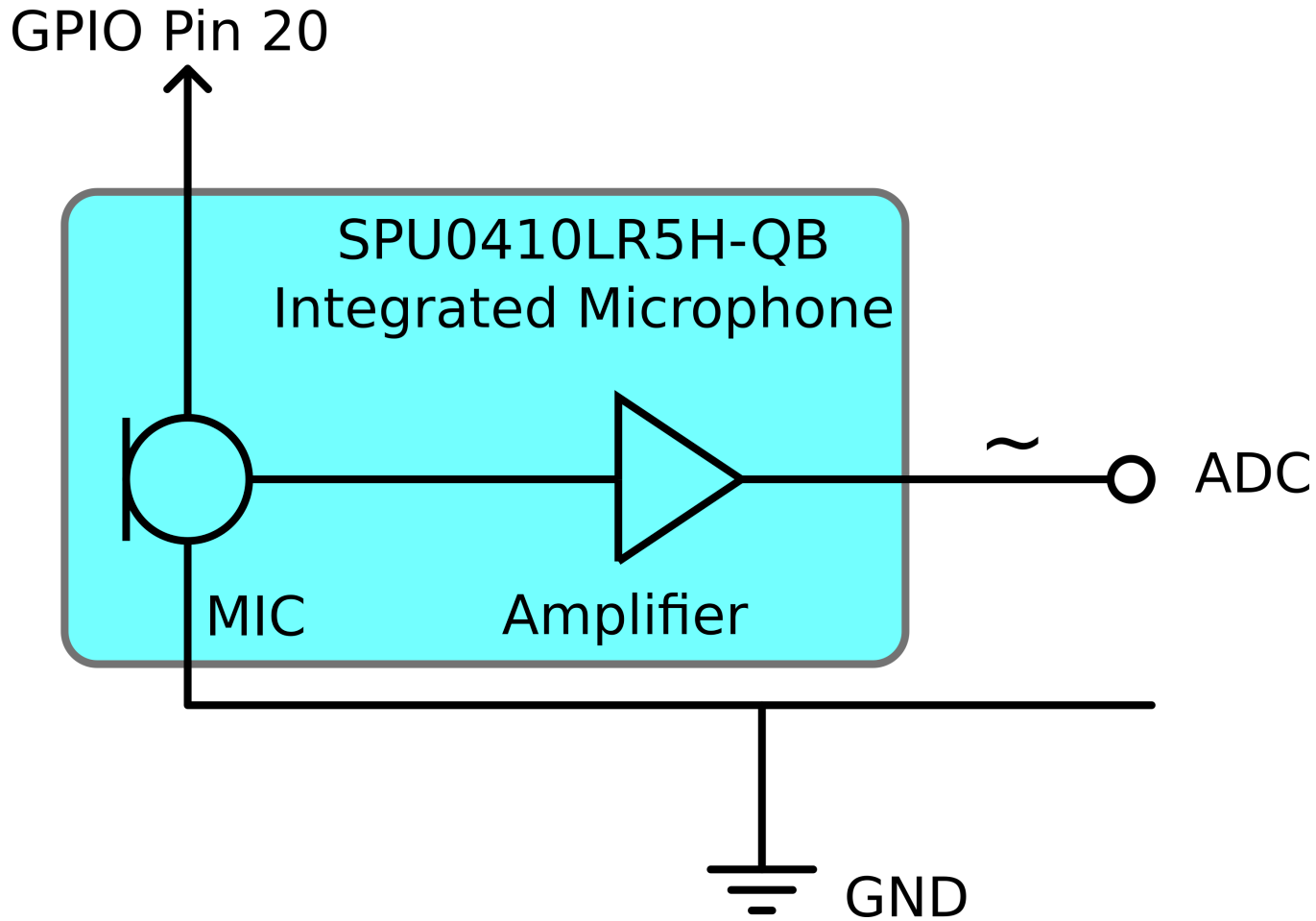
The ADC - Input/Output

According to the nRF52833 Product Specification, it is:

$$R = V_{ADC} \cdot \frac{GAIN}{V_{Ref}} \cdot 2^{Resolution} \tag{1}$$

- **R** is the digitized value.
- $\mathbf{V_{ADC}}$ is the input voltage of the ADC (which is to be converted).
- **GAIN** is a gain factor that can be configured using the ADC's registers. In our experiment, $GAIN = \frac{1}{4}$.
- $\mathbf{V_{Ref}}$ is the reference voltage to compare to. The ADC supports different reference voltages. In our experiment, we use $V_{Ref} = \frac{VDD}{4} \approx 0.75\,\mathrm{V}$.
- **Resolution** is the number of bits of the converted data. Our ADC supports up to 14 bit. We use 12 bit in this experiment, since no higher accuracy is needed for measuring the volume.

The Microphone Circuit on the Micro:bit



Operation of the ADC

Once started, the ADC will...

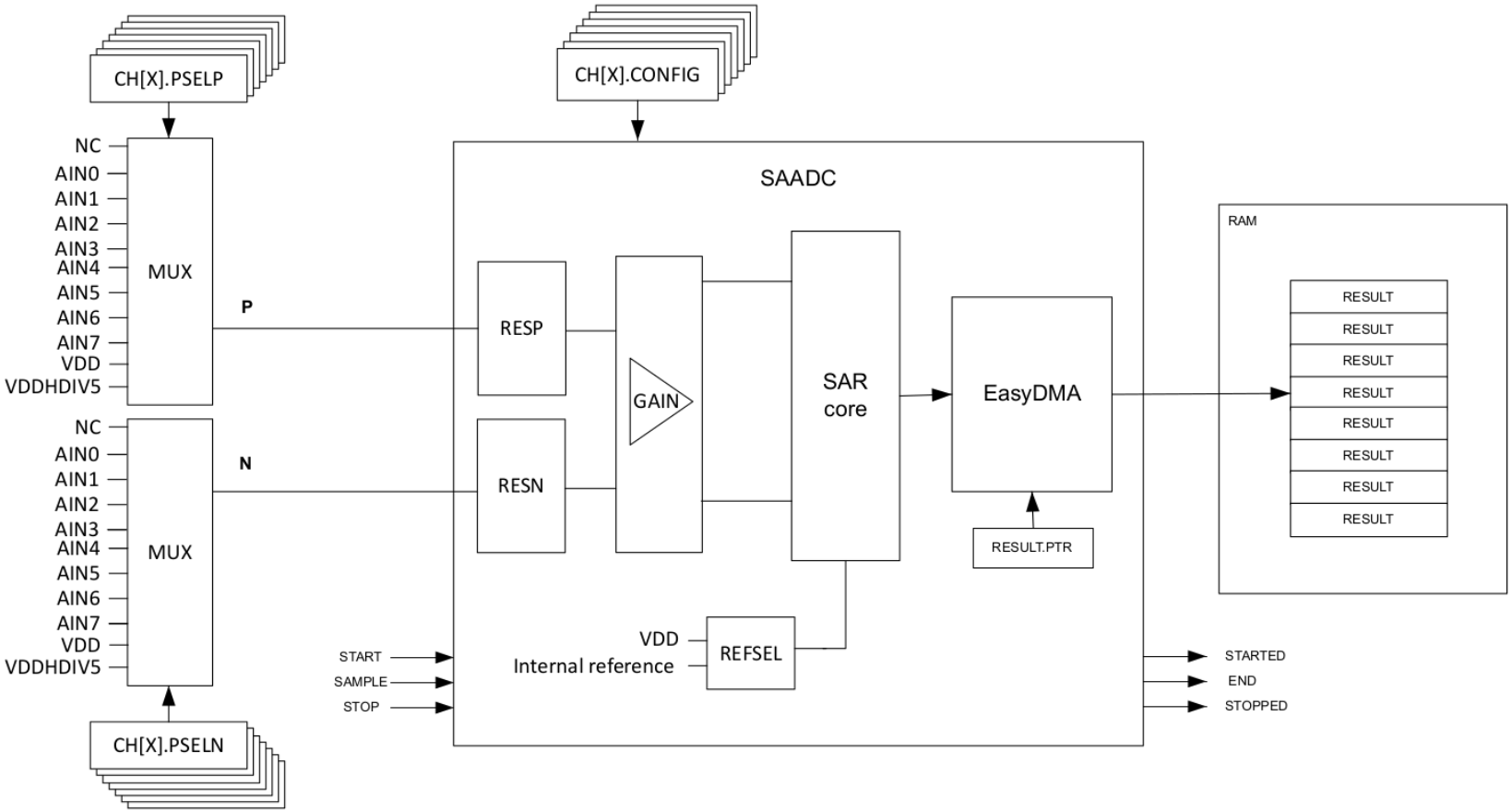
- convert a configurable number of samples
- write them into a preallocated memory buffer buffer using DMA
- generate an interrupt when the selected number of samples have been converted

The ADC schedules its operation as follows.

- The ADC uses a 16 MHz clock for triggering the sampling.
- The frequency is reduced by prescaler between 80 and 2047.
- Every (prescaled) clock edge triggers one conversion.
- In addition, the ADC provides an oversampling option. Here, a configurable amount of samples are averaged and the result is written to the memory as one sample.
- Effective sampling rate:

$$f_{sample,eff} = 16\,\mathrm{MHz} \cdot \frac{1}{prescaler \cdot oversampling} \tag{2}$$

The ADC



Source: Nordic Semiconductor. nRF52833 Product Specification. v.1.4, 2021, `https://infocenter.nordicsemi.com/pdf/nRF52833_PS_v1.4.pdf`

Identifying the Right Sampling Rate

- What would happen if $f_{sample,eff}$ is too low?
- On the other hand, why shouldn't the sampling frequency become arbitrarily high?
- The Shannon-Nyquist theorem implies that we have to sample by at least $2 \times$ the highest frequency contained in a signal. How can we nevertheless account for frequencies over $1/2 \cdot f_{sample,eff}$?
- Based on these considerations, which sampling rate would you chose?
- In the code template, the ADC buffer has a size of 1000 samples. How many times per second will an interrupt be generated with your sampling rate? If the volume is estimated whenever the signal is triggered, will this be a reasonable update rate?

Task 6.1 - Configuring the ADC

- Goal of this task is to configure the ADC.
- Open *main.c* in the "*lab6*" directory.
- The given *configure_adc()*-function already does most of the configuration.
- Go through this code and look up the individual options in the nRF52833 Product specification. The goal is to understand every step.
- Some registers, which are essential for a proper operation, are not yet set. This is left to you.
- In particular, set the ADC into the *timer/continuous sampling mode* and configure the *prescaler* and the *oversampling* parameter, such that your desired sampling rate is realized.

Task 6.2 - Sound Volume Estimation

- The project template already has an interrupt service routine.
- In its current form, it does the following.
  1. There are two buffers into which the ADC can write. The ISR always swaps between them in an alternating manner, such that one buffer remains untouched for processing, while the ADC writes to the other buffer in parallel.
  2. The interrupt flag is cleared and the ADC is restarted.
- The goal of this task is to add the code needed for a volume meter.
- After estimating the sound volume, your code should create a LED bar on the Micro:bit screen, which becomes taller as the volume increases.

Hints

- For computing the volume, only look at one filled ADC buffer, without considering any previous values.
- The magnitude observed in this buffer corresponds to the difference between the largest and smallest sample value.
- You can put the estimated magnitude into a relation to the highest magnitude observed in all previous executions of the ISR. You can use a static variable in the ISR for this purpose.
- Compare the resulting percentage of the maximally observed magnitude against multiple thresholds (e.g., 20 %, 40 %, 60 %, 80 %, 100 %) and switch the lights of the bar accordingly using an if..then...else structure.
- Test your code on the Micro:bit board.