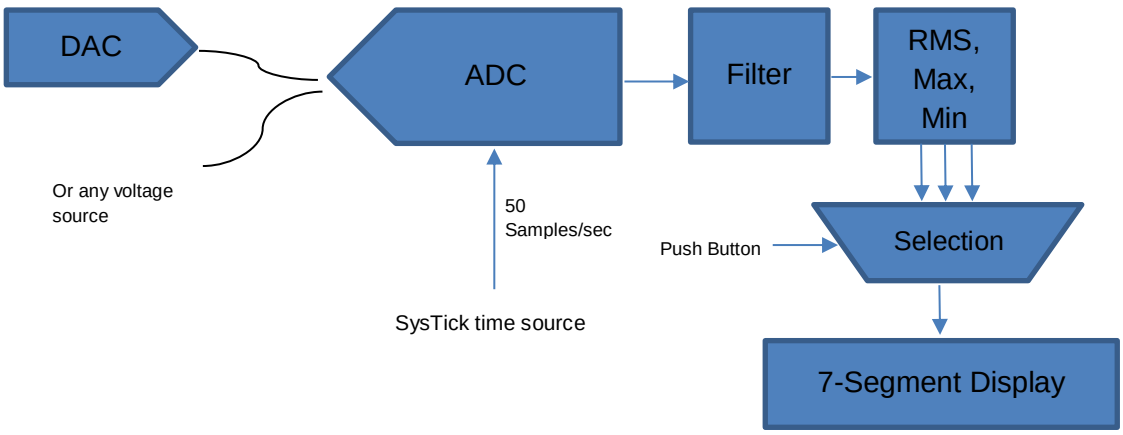
ECSE426 Microprocessor Systems

Winter 2018

Lab 2: Analog Data Acquisition, Filtering, and Digital I/O

Introduction

A common operation in embedded microprocessor systems is analog data acquisition and signal conditioning. If the signal is known to be noisy, it is beneficial to apply a filter to remove some of that noise. In this experiment you will construct such a system based on an external analog signal. You will perform a manual data acquisition and conditioning while providing a simple graphical output using 7-segment LEDs to show the RMS, Max and Min voltages updated within a period of time.



Preparation

To begin this experiment, you will first have to go to the ECE labs counter on Trottier's 4th floor to claim your team's development kit. You will be receiving two boxes: the grey box has the development board while the blue box has all necessary components for all labs. The kits will be ready for you to pick up on ***Monday, February 5th***. Try to get your kit as early as possible to start working on the experiment. You need to go in groups, present your IDs and sign forms to acquire your lab kit. You will be given a checklist for the items in the blue box. **Make sure all components are there before signing the forms**. You will be held responsible for any missing/damaged items at the end of the semester.

You will write your codes the same way you did with Lab 1 except for few changes in project setup. You need to switch from simulation mode to actual hardware mode by choosing the option of ST-LINKV2 SWD debugger. This should establish connectivity for kit programming and code debugging. Make sure you have **installed the ST-link** drivers beforehand (Check Tutorial 1 slides for link). Sometimes, if the kit does not get recognized, try changing the USB port or unplug/re-plug the board. If there are still problems, ask the TAs.

Tutorial II will cover in details how to use peripheral drivers based on the STM32F4 Cube HAL firmware. You will also learn about debugging, reading the board schematics and basics of interfacing (not covered in slides). To learn more on embedded C programming, some slides as well as an embedded C tutorial will be posted for your reference.

The Experiment - Functional Requirements

In this lab you are required to do the following:

1. *Setting up the blue button:* The discovery board has only user button. You should enable this pin as "Input" to read this button. You can enable the interrupt for this button instead of continuously reading.

2. *Setting up a DAC:* In order to test your system you should generate an analog voltage using DAC block. You can measure this voltage by the oscilloscope in the LAB. If you connect the output of DAC to the input of ADC by a wire you should expect to have a correct reading of the generated voltage.

3. *Setting up an ADC:* The STM32F4 microcontroller has three Analog to Digital converter blocks. You are required to write the code to set up the **ADC1 in the interrupt mode** to digitize the analog values by doing single conversion at a time. The analog input should be acquired at a frequency of 50Hz. You can use STM32CubeMX software to generate the configuration codes. You can refer to the ADC section in the STM32F4xx datasheet and the HAL_ADC driver to get started with the setup. You are required to know and explain in the report and demo time all the ADC settings.

4. *Setting up the sampling frequency:* To generate the required sampling frequency, you are required to use ARM's SysTick timer. **No software delays are allowed for the purpose of sampling**. Do **not** use any of TIMx peripherals yet. Only SysTick is allowed. Using SysTick timer entails two steps, setting up a required frequency and an associated interrupt handler code (see stm32f4xx_it.c). The SysTick interrupt body (as other interrupt bodies) should be lightweight as described in the tutorial. This interrupt body is initially provided when you use CubeMX. You only need to change its frequency in the Clock_Configuration function. In the report you must mention how you made the 50 Hz sampling frequency. You are encouraged to investigate the driver code of the function setting up the frequency. Note that this frequency is different than the sampling frequency of ADC part.

5. *Signal filtering:* This data acquisition is prone to noise, there are many sources of noise: electromagnetic interference, thermal noise and quantization noise due to ADC conversion to name a few. As such, one needs to filter these noise sources and minimize their effects. There are quite a number of filters used in all sorts of engineering problems. To keep things simple, you can use a simple FIR filter, the principle of which was covered in lab 1, to do the filtering. You are free to use your lab 1 implementation of the filter if you want, but you should set the desired frequencies of the filter. Make sure you have a correct functioning filter.

6. *Making sense of data*: Convert the acquired readings from a digital value to the real voltage value from the filtered data.

7. *Voltage display:* You are required to display the RMS voltage on a three digit of 7-segment display in the form of *YX.XX*, as X indicates the voltage as a floating value. You need to store values and then calculate the RMS. **Provide a reason for choosing the length of this vector and report it**. You also have to show the Max and Min values of the measured voltage updated within the past 10 seconds**. You are not allowed to store the 10 second records and then find the Max and Min.** The maximum vector size should be less than 20.
   You should use the blue button to switch the display between the RMS voltage, Max and Min values. The digit Y can be a sign for the user to understand which mode is showing. Also, you can use the 4 LEDs on the board instead of digit Y. All 7-segment data lines must share the same processor port/pins to save I/O pins. As such, to avoid conflict, you are instructed to make use of 7-segment multiplexing technique. Check the Appendix at the end of this experiment.

8. *ADC with DMA:* To get the bonus, you are required to configure ADC with Direct Memory Access (DMA), instead of simple conversion mode to get up to 0.5 point extra. In this case, the ADC conversion runs in background and stores data into a location in the memory (DMA with circular mode). So, you only need to read the content of the location at the sampling frequency (mentioned in number 2).

Note the following:

- The 7 segment switching when done at a quite fast speed will fool the eyes to see the whole set of digits as being all on at the same time. You have to try different values of time delays to find which will give a consistent output. That is, no flickering or any obvious transitions should be visible.
- You can use SysTick timer for this purpose or software delays.
- You will use common cathode displays and others common anode ones that need logic high to light and uses NPN transistors as a switch.
- **Don't forget to use the current limiting resistors** (Rs resistors, ranges in Ohm) with the display (either during testing them or using them). Otherwise, you risk damaging the display.
- You need to write some code to translate in between actual numbers and their 7-segment representations (number 2.58 should be divided into 2, 5 and 8 digits).
- All of you got more resistors and transistors than you need in case you break some pins or lose some.

Testing

In the lab, there is an osciloscop which you can use to observe and measure the voltages you generate or read. Other than DAC, you can use the GND and 3.3V pins on the board as a source of voltage. **Before any modification on your wiring, please unplug the board from USB and then apply your changes.**

Reading and Reference Material

- Tutorial II by Amir Shahshahani
- C Tutorial by Ben Nahill
- Doc_03 - Technical and Electrical
- Doc_05 - STM32F4xxx Reference Manual
- Doc_10 - Discovery Kit F4 Rev.C
- Doc_16 - Debugging with Keil
- Doc_17 - Introduction to Keil 5.xx
- Doc_19 - Documentation of STM32F4xx Cube HAL drivers
- Doc_26 - General Report Grading Guideline for TAs Ver 1.3 (Report related)
- Doc_27 - Lab Report Guidelines ver 2.0 (Report related)
- Doc_31 - Clock Display Datasheet
- Of course you are free to refer to any other posted documents which you see useful.

Bonuses

1. **Early demo bonuses (13th – 15th February)**: Thursday demos have a bonus of 0.2, Wednesday's 0.3 and Tuesday's 0.4.

Demonstration

Demos will take place Friday, **February 16th**. We expect that your code be ready by your appointment time that day and that you will be ready to demo. No extensions allowed without penalties. You will lose 20% for the Monday (19th) demo (add 10% per day beyond Monday). We take deadlines seriously. The experiments have been designed such as a group of two could handle it in less than two weeks' time **provided that they attend the lectures and Tutorials.**

Final Report Submission

Reports are due by Monday, February 19th, at 11:50 P.M. Late submission will be penalized by 10% a day. **Submit the PDF report and the compressed project file, separately, in their relevant submission folders**. **Don't include your report in the archive. The ZIP file and PDF names should be LAB2_Gxx.**

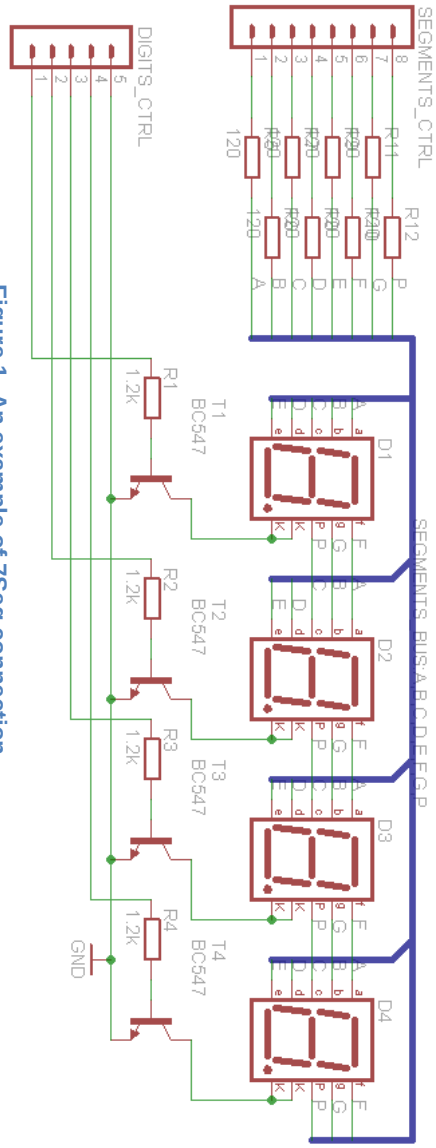


Figure 1- An example of 7Seg connection.