ECSE 426 - Microprocessor Systems

**Lab 2 Report**

Group 15

Yan Ren 260580535

Fei Feng 260575129

***Abstract***

The goal of this experiment is to perform data acquisition and training by using STM32F407 microcontroller, then providing a graphical output on 7-segment LEDs to show the RMS, Max and Min voltages updated within a period of time. In this experiment, we use DAC (Digital to Analog Converter) to generate the analog voltage, and use ADC (Analog to Digital Converter) to digitize the analog values. Additionally, we activate the blue button on board for switching the 7-segment LEDs outputs between RMS, Max, and Min voltages values.

***Problem Statement***

Based on the main goal of this experiment, the objective is divided into three major parts:

First, properly acquiring voltage data.

* Initialize DAC and ADC ports properly.
* Obtain analog voltage by using DAC block.
* Convert the analog signal to digital, meanwhile making sure that analog inputs are acquired at required frequency.

Second, training the data to get RMS, Max, and Min voltage updated within a period of time.

* A filter is needed to eliminate the fluctuating data and keep the system stable.
* Calculate RMS, Max, and Min voltages without saving all data during the period.

Third, displaying data on 7-segment LEDs.

* Initialize a group of GPIO (General Purpose Input/Output) ports.
* To switch between RMS, Max, and Min voltage values, the pin of blue button needs to be an input.
* The on board LEDs show the mode when swiching between RMS, Max and Min.

***Theory and Hypothesis***

According to [1], the DAC(Digital to Analogue Converter) on the STM32F407 Discovery board converts digital binary values to analogue voltage outputs. The DAC block has several uses including audio generation, waveform generation, etc. In this experiment, DAC block is used as a voltage generator. The generated voltage will be used in later step as raw input data. The DAC block of STM32F4XX can be operated in either with 8-bit or 12-bit resolution. To operate the DAC, we turn on the desired output channel and write the digital data value. The outputs of the DAC block can be independently or simultaneously updated. The formula for voltage output from the DAC is as follows:

Vout = Digital Value \* Vref / (2^n - 1) where n is the number of bits. (1)

We can use this formula to compare with the value we capture in later steps.

We connect the output pin of DAC to and ADC which allows the analog sensor readings to be converted to digital values. STM32F407 has 3 ADC that can work independently. Every ADC have 18 channels. 16 channels are external and connected to GPIO pin. 2 channels are internal that connected to internal temperature sensor and ADC voltage reference. This reference voltage is equal to supply voltage to Vdd pin.

From the reference[1], there are two modes of operation primarily:

* Independent mode. It is just as the typical ADC use. Each ADC unit is operating on its own and without any mutual dependency.
* Dual mode.

A/D conversion can be:

* Single Conversion. One sample conversion at a given instant.
* Continuous Conversion. Non-stop sample collection and conversion.
* Discontinuous Conversion. Sequential conversion of some channels in a group.
* Scan Conversion. Sequential sampling and converting of an array of channels one after another.

The raw data from ADC is filtered by the FIR Filter implemented in lab1. It is a five items moving average filter of the form

y[n] = 0.2 \* (x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4]) (2)

The filtered data is then converted to voltage value by using the equation

Vin = Vref \* ADC data / 2^n -1 where n is the number of bits of ADC (3)

Then we implement the function in C to find out the RMS, also the Max, Min value over past 10 seconds. On board LEDs, blue button and seven segments display are used to display the RMS, Max, and Min. Blue button has two status set and reset. By reading the button status and counting the times, we are able to switch the value display on seven segment in three modes. The LTC-4727JS 7-segment display has 16 pins. Seven pins correspond to the seven LED segments, one pin corresponds to the decimal point, and four pins select which digit is being activated. All digits can be activated simultaneously, however they would not be able to display independent digits in such a manner. To achieve multi-digit display, the digits must alternate back and forth at a rate preferably greater than 45 Hz for a complete cycle of display (approximately the flicker fusion frequency for the human eye, allowing it to appear that all displayed digits are on continuously).

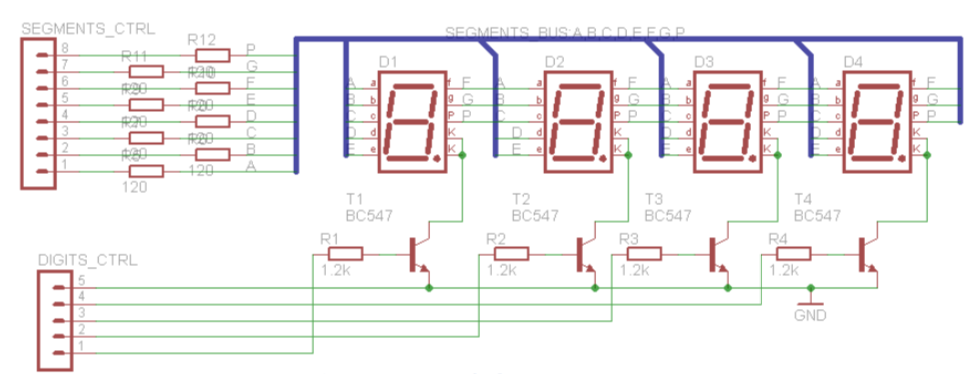
***Implementation***

The first step of the implementation is to enable all peripheral ports that we are going to use, which include ADC1, DAC, SysTick, and GPIO. They were initialized and configured by using STM32CubeMX. For configuring the ADC1, we set Clock Prescaler = PCLK2 divide by 2, Resolution = 8B, Data Alignment = right, and disabled ContinuousConvMode and ContinuousRequests. 8 bits resolution is chosen because the voltage is between 0 to 3.0, 8 bit is giving enough precision. To digitize the analog values by doing single conversion at a time, we also need to set up the ADC1 in interrupt mode. Hence, we set Number of Conversion = 1, and enabled ADC1 global interrupt.

Next, we manually configured GPIO pins that we planned to use. For SysTick, we wrote HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq()/1000000) to let it generate an interrupt every one microsecond. To keep the sampling frequency at 50Hz, we also set a counter which counts until 20000 to start an ADC conversion and get data. It means that for every 20000\*10-6 = 0.02 second we can get one data, so that we get 50 data per second, which is 50Hz. After we get data, we filtered output values and converted them to voltage values. The filter we used in lab 1 was reused in this lab. Since we set the resolution equals to 8 bits, the original output is in range of 0 to 255. Therefore we calculate 3.0 \* filtered\_val / 255.0 to convert the original outputs to real voltage values. We verified the converted voltage values with the output voltage calculated from equation (1).

To find the RMS, Max and Min voltage values within the past 10 seconds, we updated those three values when new data came in. Since we got a new data, we compared it with current Max and Min value, if it is bigger than Max, or smaller than Min, the Max or Min values will be updated. Similarly, whenever a new data obtained, the RMS result will also be recalculated and updated. Since we can get 50 data for every 1 second, during this calculate process, we set a counter which counts to 500, so that it is able to get RMS, Max, and Min values in the past 10 seconds.

The next step is to implement seven segment display. Figure 1 is a schematic of 7-segment connection circuit we used. SEGMENTS\_CTRL part is used to control which segment is turned on, and DIGITS\_CTRL part used transistors to control which digit is turned on. According to [2], we connected LED display pins with GPIO pins properly. In our design, we let only one digit turn on at a time. To achieve multi-digit display, the digits must alternate back and forth at a rate preferably greater than 45 Hz for a complete cycle of display. We use two counter in SysTick. One is for updating the overall seven segment display and one is for update each display digit. This gives the illusion that all LEDs are turn on at the same time. By trying different displaying frequency, we finally set each digits flipping at 500 Hz to keep the digital display bright and stable.



*Figure 1 - Schematic of 7 segment connection*

To implement the switching between displaying RMS, Max, and Min voltage values, we enabled port A pin 0 as input to read the blue button. In C code, we use variable “displayMode = (displayMode + 1) % 3” to track the mode switching every time the blue button is pressed. To avoid push button is too sensitive with pushing. We use counter in SysTick as well, which lowers the push button frequency. To optimize the use of blue button, we changed its frequency to 5Hz. Sequentially when we push the button, it would not rapidly alternate between RMS, Max, and Min values. The on board LEDs LD3, LD4 and LD6 are used to indicate the value modes.

***Testing and Observations***

* *Functional testing*

To test whether our system works properly, we firstly used printf function to show RMS, Max, and Min voltage values in software. For testing purpose, we set DAC to alternate only between two values in a fixed frequency. Then we can manually calculate the Max, Min and RMS. We compare these value with the printf results to check function for RMS, Max and Min is working properly. This also shows the DAC and ADC are working properly, since voltage is generated by DAC and read through ADC.

For calculating RMS, Max, and Min voltage values, we have tried two methods. The first one is described in implementation section above. The other one is to use a vector with size equals to 20. In this method, when a new data obtained, the oldest data in the vector is discarded. Therefore, every time the vector updated, the RMS, Max, and Min values would be updated along. During the test, we found that seven segment display updated in a much lower frequency than the values updated, so we cannot correctly display the Max and Min during past 10 seconds. Hence, we decided to use the first method in our data conditioning code.

To test the digital displaying functionality, we began with displaying a single number in specific digit position. After all digit positions could display numbers properly, we connected the board and tried to display the real voltage values.

* *System testing*

After the functional testing, we perform the system testing still by setting DAC to alternate only between two values in a fixed frequency. The printf function is still used in code for printing RMS, Max and Min. The program was then running on the microcontroller in debug mode. It was observed that the on-board LEDs were turning on/off as expected with the pushing of blue button. The seven segments display results matched with the printf results in console window. Overall, after repeating the testing multiple times, it was observed that the operation of the system corresponded to the requirements.

***Conclusion***

In the course of this lab, using the STM32F407 Discovery board, a system that acquires voltage readings from the on-board DAC and provides visual feedback to the user was implemented. After acquiring voltage readings and converting them to a digital format via the ADC, the results were converted into a voltage format. In order to reduce the effects of noise, the raw data was filtered using a FIR filter. The function for calculating RMS, Max and Min is performed on filtered voltage value and the results are stored in variables. The user was provided with a visual feedback using the 3 on-board LEDs and seven segment display which were controlled via a GPIO. The use is able to switch the display mode by pushing on board blue button. The corresponding on-board LEDs will light up and seven segments display will update the value with the mode switching. Through a series of tests, it was concluded that the behavior of the system appeared to be in line with the requirements. The successful implementation of this experiment, provided our group with a solid introduction to the methods and techniques for interfacing with the various on-board hardware components.

***Reference***

*[1] “STM32F405xx/STM32F407xx Datasheet - production data”, st.com, 2018. [Online]. Available: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf. [Accessed: 19-Feb-2018].*

*[2] “LED Display Product Data Sheet LTC-4727JR”, optoelectronics.liteon.com, 2018. [Online]. Available: http://optoelectronics.liteon.com/upload/download/DS30-2000-193/C4727JR.pdf. [Accessed: 19-Feb-2018].*