ECSE-426 Microprocessor Systems Report 2

Group 8

Sean Stappas
260639512

Eric Vuong
260630559

March XX, 2018

1 Abstract

2 Problem Statement

The objective of lab 3 and 4 was to implement the concepts of timers, PWM, multi-threading, and keypad scanner into a system using the F4 Discovery board. . The final deliverable was a voltage generator based on RMS. The system should have the following requirements:

review after lab 4

- The discovery board should provide inputs with a numerical keypad, thus allowing the user to generate a desired voltage no more than 2.8 V with a good stability.
- Similar to lab 1 & 2, the board should be interfaced with a 7-segment display, showing 4 different outputs: RMS Voltage, Min, Max, and user voltage input in the form of X.Y.
- The system should have 3 modes, controlled by the numerical keypad: sleep mode, input mode (restart) , and display mode. To get into sleep mode, the user will hold the '*' button for more than 3 seconds. During this time, the display should be off with an output voltage 0. Moreover, power consumption of the board should be taken into consideration, such as turning off any unused components. For the restart mode, the user will have to hold '*' for around 1-2 seconds. Finally, the '#' button was used to simulate an *Enter* button and '*' was used for the *Delete* button, deleting the last digit entered. Pressing the *Enter* button would get into display phase.
- Timers should be used generate the PWM and ADC readings. Unlike lab 1-2, a timer would be used to trigger the ADC sampling time.
- The voltage generator would be controlled using PWM pulse generation, a simple rectifier circuit and a simple control system to adjust the output voltage based on load variations. The reaction time should not take longer than 0.5 seconds. Finally, depending on the pulse frequency chosen, a FIR filter and an analysis on the parameters of the filter would have to be performed to obtain a decent filtering.

graphs, visual analysis

- In lab 4, multi-threading would have to be designed with CMSIS-RTOS while ensuring the same functionalities from lab 3.

3 Theory and Hypothesis

3.1 Circuit

In order to output the desired voltage of a signal, a simple rectifier that would hold the charge in the capacitor, and discharge it with resistive loads. In order to charge and discharge the capacitor, a PWM pulse generator would have to be applied. Finally, the RMS voltage of the charge and discharge cycles should lead to a fairly stable output based on the parameters of the FIR filter.

The charge and discharge of a capacitor from Figure 1 was determined by the time constant:

$$\tau = R \times C \tag{1}$$

and the charge (2) and discharge (3) equation are the following:

$$V_c = V_s(1 - e^{\frac{-t}{\tau}}) \tag{2}$$

$$V_c = V_s \times e^{\frac{-t}{\tau}} \tag{3}$$

Thus, the generated output voltage is dependent on the width of each pulse and the values of the capacitor and resistor, the pulse frequency would have to vary accordingly.

A time triggered ADC would have to be used to obtain the output readings. This part was crucial, because the data would then have to be filtered and fed back into the control system to readjust the duty cycle of the PWM input depending on the load variations. The ADC will also be used with DMA, similar to lab 1-2.

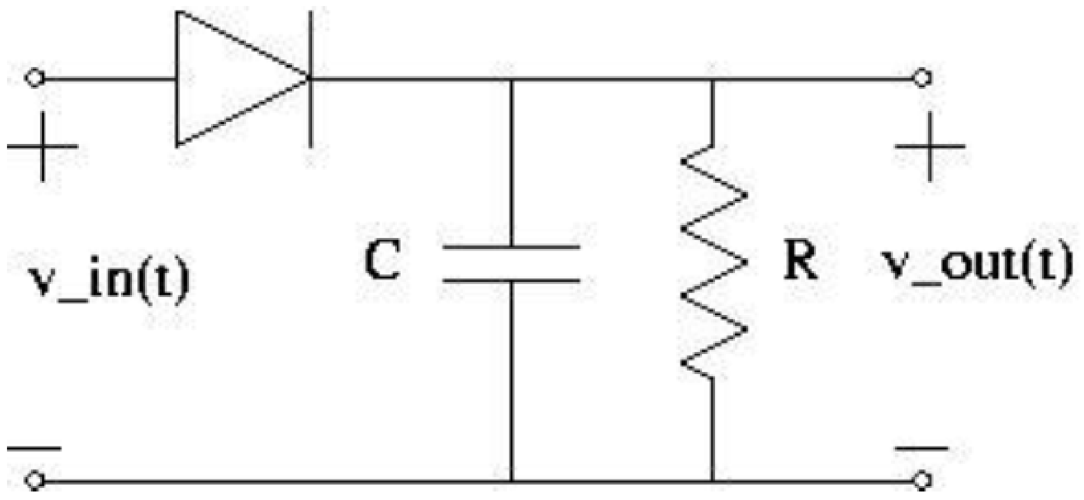


Figure 1: Simple rectifier circuit with capacitor and resistive load [1].

3.2 Filtering

The generated DC voltage is often noisy, a FIR filter would be recommended for this lab. Thus, depending on the desired cut-off frequency, filter delay and gain, the order and coefficients of the FIR filter, as explained in lab 1-2, would have to be calculated.

3.3 Keypad

For the 4x3 keypad, each pins corresponds to a column or row, for a total of 7 pins, excluding the ground. The buttons on the keypad are wired in a way, such that each button pressed corresponds to the connection between a row and a column on contact. Thus, a scanning algorithm could be implemented to determine the key presses by the user. The configurations of the GPIO pins would be necessary and debouncing measures similar to lab 1-2 would have to be considered.

3.4 Multi-threading

As part of lab 4, the aforementioned functionalities would have to be integrated and operating concurrently through the CMSIS-RTOS by multi-threading components. In other words, the PWM pulse generator, ADC sampling and controls, and the keypad would be functioning in parallel.

RTX-based?

4 Implementation

The peripheral configurations of the board were done using the STM32CubeMx software. This helped us speed up the set-up process for time controlled PWM, keypad GPIO pins, 7-segment display pins, and the time triggered ADC with DMA. Figure 2 shows an overview of the various implementations of the circuit. There were various design constraint that had to be taken into consideration during the implementation. Similar to lab 1-2, a blue button (push button in Figure 2) was used to switch the display settings between the processed RMS voltage, max and min values.

The pinout configuration used can be seen in , the details of which would be given in the next sections.

update the pin-out

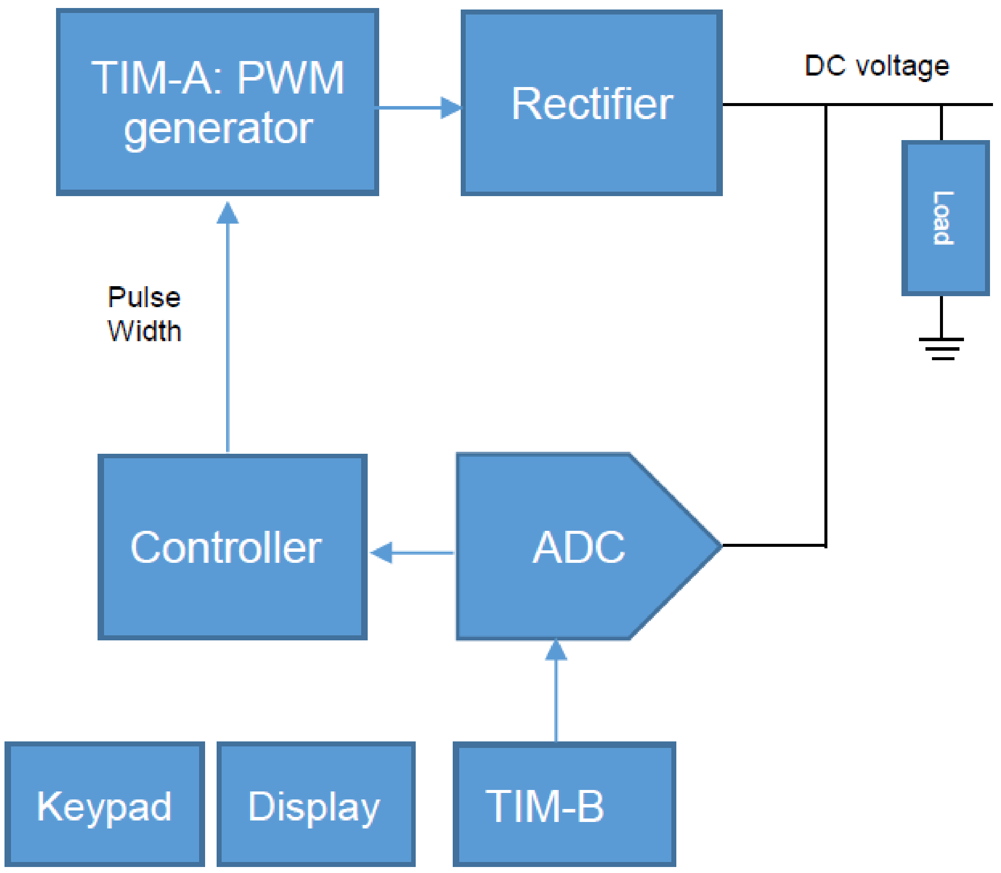


Figure 2: High level circuit description. [1].

4.1 Keypad

As aforementioned, the keypad provided for the lab was a 4x3 keypad, with digits ranging from 0-9 with 2 special characters: '*' and '#'. Since the keypad pins are wired in a way such that each key pressed would connect a row and a column together.

In this way, a scanning algorithm could be used to determine with key was pressed. To do so, we would have to write on the rows and read from the column pins. Moreover, we would be writing to the rows in a specific way and reading from the columns, which are initially set to '0' with the pull-down resistors being used to fix the initial states to be logic '0'. At any instant, only one pin of the 4 columns was set to '1' and the rest were set to '0'. When a button was pressed, only one of the column would turn '1', thus allowing us to determine which row and column has been pressed. This method is effective when the columns and rows were scanned at a fast enough frequency, such that duration of a button pressed lasts an entire scan cycle. Similar to lab 1-2, a debouncer has been implemented whenever a new button is pressed.

4.2 Timer Triggered ADC with DMA

A TIM peripheral timer could be used to trigger ADC readings. Some configurations were needed to setup the ADC to be triggered with timers. The ADC would need an external source, such as a timer to trigger the ADC for a new conversion. However, doing so would require the user to wait for the end of conversion before reading the data. Note that timers also has channels to use in PWM mode, generating pulse on a GPIO pin, which will be discussed later. When reading from the ADC, it was important to set the correct timer frequency. The timer would have to be triggering the ADC at a rate faster than

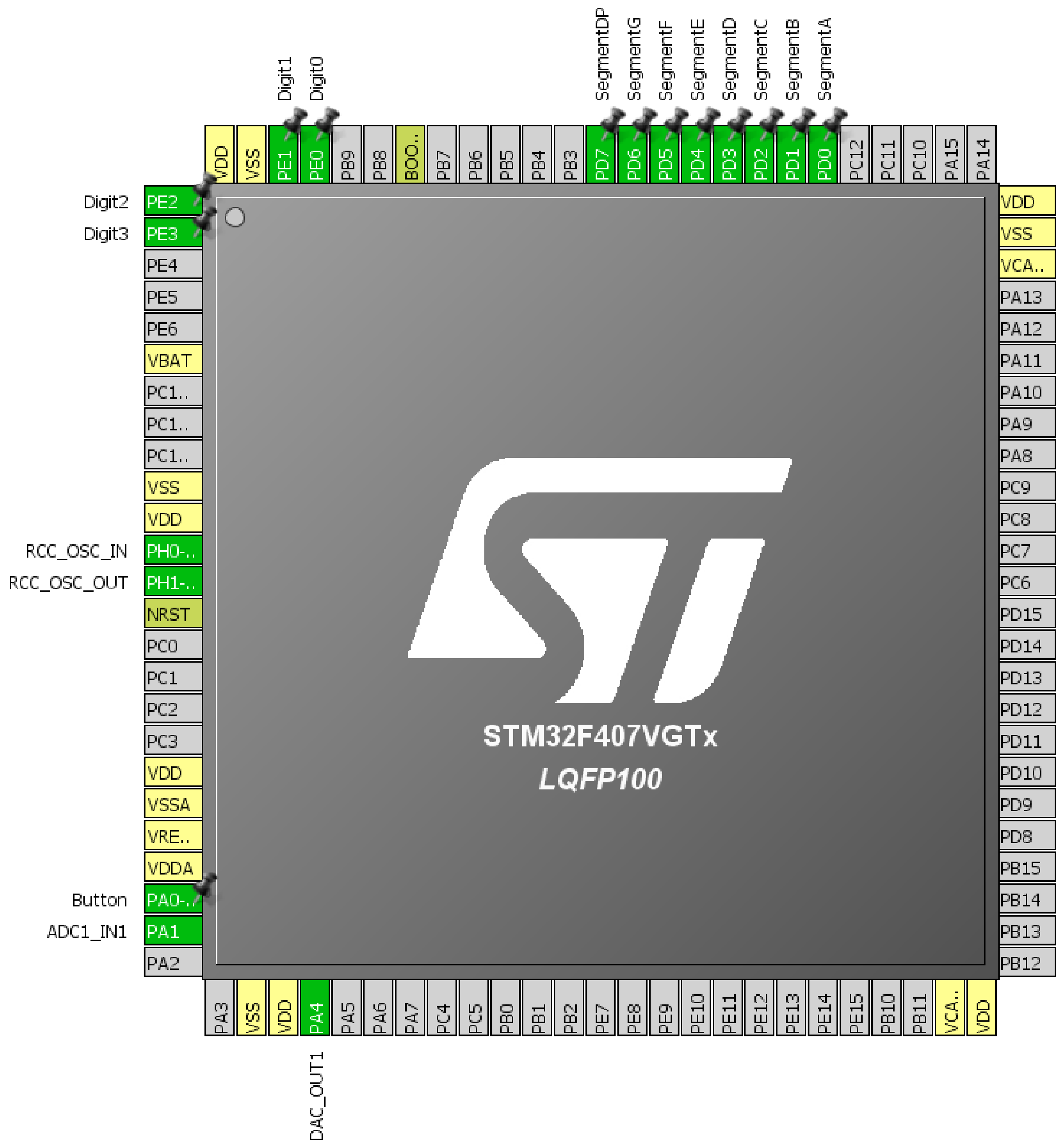

Digit1
Digit0
SegmentDP
SegmentG
SegmentF
SegmentE
SegmentD
SegmentC
SegmentB
SegmentA
Digit2
Digit3
RCC_OSC_IN
RCC_OSC_OUT
Button
ADC1_IN1
DAC_OUT1
STM32F407VGTx
LQFP100
VDD VSS PE1 PE0 PB9 PB8 BOO.. PB7 PB6 PB5 PB4 PB3 PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0 PC12 PC11 PC10 PA15 PA14
PE2 PE3 PE4 PE5 PE6 VBAT PC1.. PC1.. PC1.. VSS VDD PH0-.. PH1-.. NRST PC0 PC1 PC2 PC3 VDD VSSA VRE.. VDDA PA0-.. PA1 PA2
VDD VSS VCA.. PA13 PA12 PA11 PA10 PA9 PA8 PC9 PC8 PC7 PC6 PD15 PD14 PD13 PD12 PD11 PD10 PD9 PD8 PB15 PB14 PB13 PB12
PA3 VSS VDD PA4 PA5 PA6 PA7 PC4 PC5 PB0 PB1 PB2 PE7 PE8 PE9 PE10 PE11 PE12 PE13 PE14 PE15 PB10 PB11 VCA.. VDD

Figure 3: STM32F4 pinout diagram.

the PWM pulses. . The frequency of the timer was calculated with the specific bus clock value of the timer, prescaler and period value as follows:

$$DesiredTimerFrequency = \frac{BusClockFrequency}{Prescale \times Period} \tag{4}$$

hi

The ADC sampling rate was set at

did we have any issue, with the timer being too fast?

trying out vim lol

checklab

4.3 PWM Pulse Generation with Timer and Rectifier Circuit

Figure 1 shows the circuit implemented in the lab. The capacitor value chosen was 1 µF and the resistor value was 4.7 kΩ. This yields a time constant of 4.7 ms.

In order to properly generate the desired voltage, the pulse period would have to be smaller than the time constant, in order to allow a better output voltage control by varying the pulse width (duty cycle). The pulse period chosen was then .

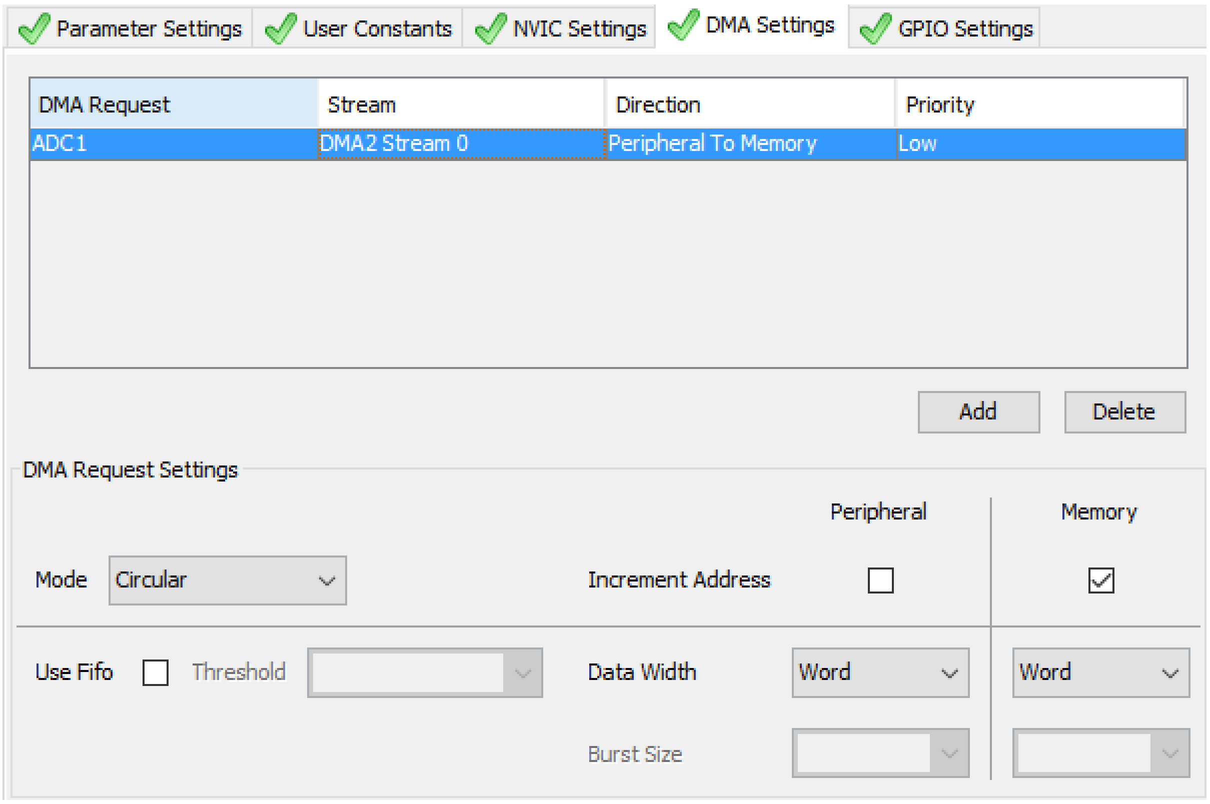
check the pulse period

The pulse width had 8000 steps, thus allowing a lot of resolution to the output voltage. A control has been applied so that any variations in the resistor value could be readjusted by changing the pulse width. To do so, at every sampling rate, the pulse width adjustment would be calculated based on the difference of the desired voltage and output ADC reading as follows:

verify the period, etc

$$VoltageDifference = DesiredVoltage - CurrentADCVoltageReading \tag{5}$$

Depending on the voltage difference from (5), an incremental or decremental pulse width would be applied to the current pulse width. In order to speed up the reaction time, the pulse width applied can be increased if the voltage difference is relatively high. In other words, a P-controller was applied.



DMA Request	Stream	Direction	Priority
ADC1	DMA2 Stream 0	Peripheral To Memory	Low

DMA Request Settings

			Peripheral	Memory
Mode	Circular	Increment Address	☐	☑
Use Fifo ☐	Threshold	Data Width	Word	Word
		Burst Size		

Figure 4: ADC DMA configuration.

The ADC was also configured to use continuous DMA requests as seen in Figure 4. This allows for much simpler reading of the voltage values from a location in memory, with the trade-off of having unnecessary conversion take place.

4.4 Filtering & Processing

4.5 7-Segment Display

Since it was required to display the acquired data in RMS, as well as the maximum and minimum voltage updated for each interval of 10 seconds, a three digit 7-segment display will be used in the from of YX.XX, where X is the desired value in floating points and Y is the display mode. Figure 5 shows the meaning of each pin, which has been tested by applying a high voltage and grounding each pin through trial and error in the lab.

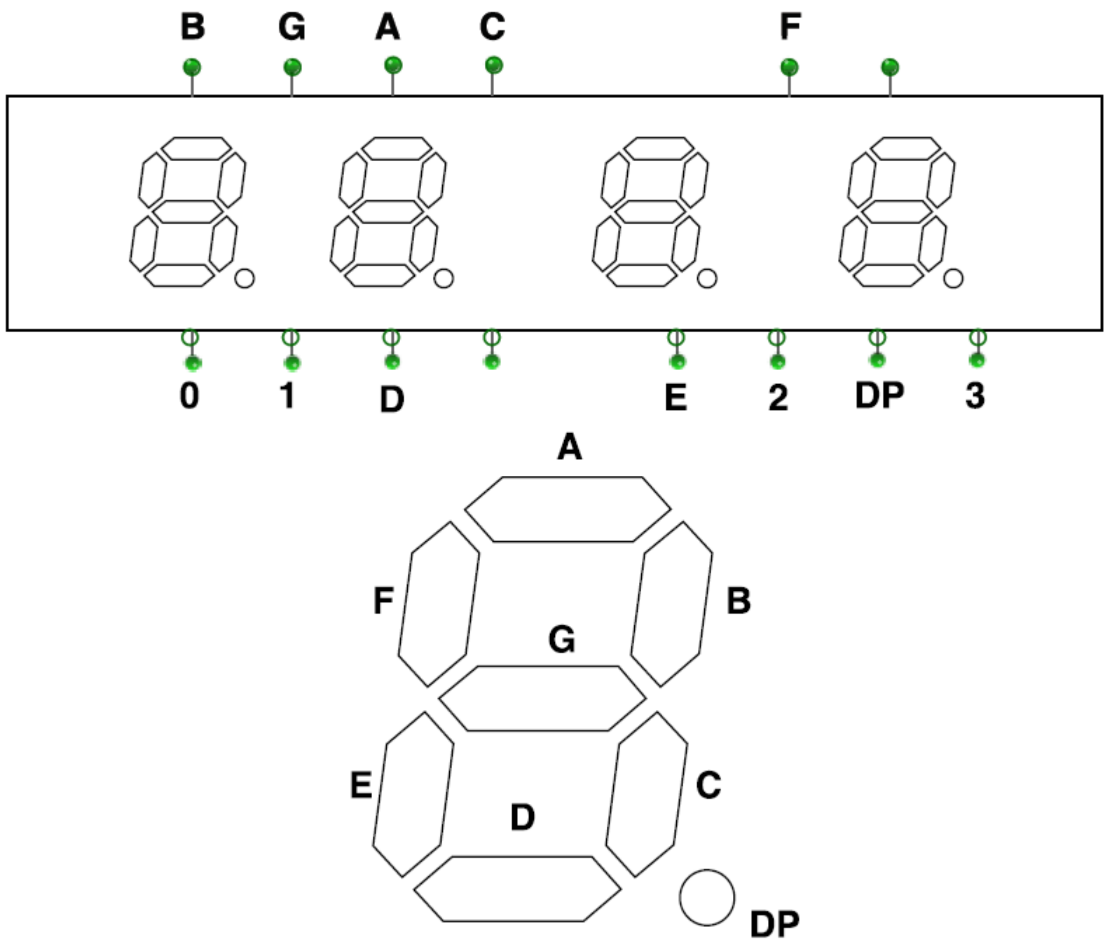


Figure 5: Meaning of pins on 7 segment display. Numbers 0 to 3 represent the 4 digits from left to right. The letters represent segments, as shown in the bottom of the figure. DP represents the decimal point.

Since the 7-segment display uses multiplexing technique, it is important to adjust the 7-segment switching rate to prevent flickering. To handle this, the SysTick timer can be used. As stated earlier, we chose a switching rate of 200 Hz (every 5 ms). Moreover, current limiting resistors will be used to prevent overheating and damaging the display.

As shown in Figure 5, 8 GPIO pins were connected to the 8 segments (SEGMENT_A-G,SEGMENT_DP to PD0-PD7) and 4 GPIO pins were connected to the 4 digits (DIGIT0-4 to PE0-PE3) on the board. As each segment pin is an anode and each digit pin is a cathode, both will need to be active in order to properly display a number. For instance, displaying the number 0 on the second digit will require all segments to be active except segment G (PD6) and digit 2 (PE1) to be active.

To display the current mode of operation (RMS, Max, Min), the first digit of the 7-segment display was used. The RMS, Max and Min modes are thus represented by digits 0, 1 and 2, respectively. In each mode, the remaining 3 digits represent the desired number, with two decimal places.

5 Testing and Observations

Firstly, the 7-segment board's circuit has been tested in order to determine the corresponding pins to each segments and digits. Hence, a simple test was performed to validate the connections by toggling all the segments continuously and generating all 10 digits.

The switching frequency of the 7-segment display was also tuned such that it would trick the eye into thinking it is formed by one continuous light. With a frequency of 200 Hz, this was achieved.

The goal of implementing a 7-segment display was to display the RMS, max and min. Thus, printf() function was used to validate the accuracy of the display, such that the displayed value and the printed value was consistent. This involved checking that the correct mode is displayed in the first digit and that the displayed value is correct.

The voltmeter was tested to ensure that it is measuring accurately. To do so, simple tests were created by generating a voltage level of 1.5V with the DAC using the GND and 3V pins on the board. It was observed that the RMS voltage of the measured values were all within 5% of the expected values.

Moreover, the voltmeter was tested using a varying voltage source during the demo. The max and min values were updated once every 10 seconds, thus respecting the lab constraints. The RMS was continuously updating and responding to the input voltage changes.

6 Conclusion

In conclusion, a voltmeter was built using the STM32F407VGTx. The display board was able to properly display the desired values without any flickers or obvious transitions. The voltmeter was able to successfully read the data and an error of less than 5% is observed. Also, the user was able to switch between different modes (RMS, max and min), where the max and min would update only once every 10 seconds.

Many improvements could be made to our design and method. For example, to represent our input vector, we used a simple array structure with shifting of values at each new datum. However, since it is essentially a FIFO queue, a linked list structure would be more appropriate and efficient. An improvement to our testing method would be to generate more interesting voltage signals. Indeed, we simply generated DC voltages, but it would have been interesting to generate more complex AC signals, such as a sine wave, and validate that the correct RMS, max and min values are captured.

References

[1] McGill, “ECSE426 Lab 3 Handout.”