ECSE426 - Microprocessor Systems

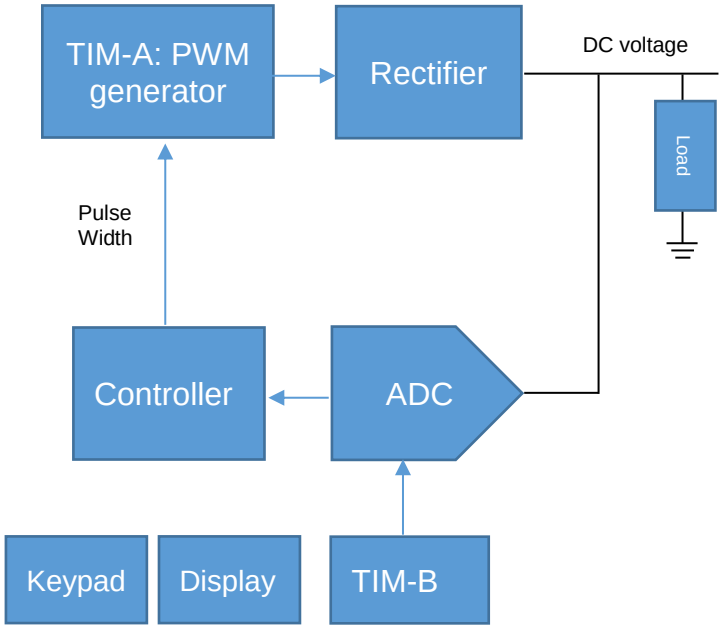
Winter 2018

Lab 3: Timers and Interrupts

Lab Summary and Objectives

This exercise will introduce you to use timers in the STM32F4 microcontroller. It is aimed at designing a system that generates a desired DC voltage based on RMS of PWM pulses. You will be using the ADC converter to read the voltage as a feedback to control the voltage level.

The board will be interfaced to a 4 by 4 (or 4 by 3) keypad. The user enters the desired voltage using the keypad and the 7-segment display the current voltage value once the user press enter. You should use **#** to simulate an ***Enter*** button and ***** to ***Delete*** the last digit was entered. The operation has two phases, first entering a desired voltage level, then by pressing "Enter" you get into phase two which is showing the measured voltage value as you had in assignment-2. To **restart** the operation of each scenario at any time, the user has to press *** for longer time** (around 1-2 sec) and to put the system into sleep, **hold *** for more than **3 seconds**. In sleep mode, the display should be OFF and the output voltage 0. The system can back to the operating mode by holding # for more than 3 second. In either of phases and button presses, the display should be stable and ON.



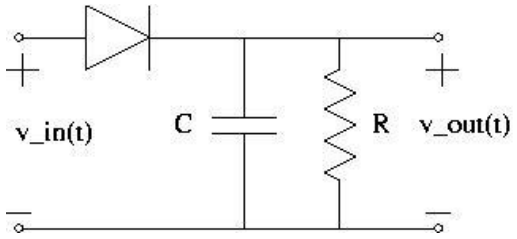
Lab requirements

(1) PWM pulse generation

You are required to design a system that generates PWM pulses and apply this voltage to the following circuit. This circuit is a very simple rectifier that holds the charge in the capacitor while the resistive load discharges the capacitor. The RMS voltage of the charge and discharge cycles results a voltage less than the peak value of input pulse. They duty cycle of this pulse controls the voltage level.

You can use 1N914 diode, capacitor and resistors in the LAB. Capacitor and resistor value selection is based on the frequency of PWM pulses. You are required to provide reasons for your selection on PWM pulse frequency, the resistor and capacitor values of the following circuit. You are free to modify the circuit and add components such as a BJT, opamp, etc. where needed.

Configure a timer to generate the pulse at a frequency of your choice between 500 Hz to 2 MHz on a GPIO pin. To get the specific bus clock value of the timer you are using, refer to "**Doc_00 - STM32F4 Processor clock tree and configuration values".**



(2) ADC and Timers

In the second assignment, for a new conversion you used HAL_ADC_START_IT. This technique is called ADC_SOFTWARE_START, as you see this term in your ADC configuration. There is another way of triggering your ADC from other sources such as timers. You should use a timer to generate these triggers as a source of sampling time for your ADC. The choice of the sampling frequency is based on your frequency of PWM pulse and how you control the pulse width. Therefore, it is your decision and has to be reported.

(3) Filtering

You are advised to use the FIR filter for your design. The generated DC voltage is noisy. You have to *do an analysis on the parameters of the filter* so as to get decent filtering. The parameters are filter order and filter coefficients. You are required to find a best value for parameters based on your requirements, theories and observations such as your desired cut-off frequency, filter delay and gain. You have to present a solid convincing case to justify your choice. **We also expect some visual analysis (i.e. graphs). All the parameters and your criteria for choosing those parameters should be in your report as well.**

(4) The alphanumeric keypads.

In order to read a 4*4 or 4*3 keypad given to you, a polling technique can be considered to perform this task. You should write the code such a way to minimize the bouncing effect. A tutorial on how to use keypads is uploaded on myCourses. The entered voltage value is in the format of Y.X, such as 0.9 or 1.3 volts. It might be impossible to generate voltages less than 0.5v and more than 2.8v with a good stability. Therefore, we assume the entered number is within this range.

(6) Hint for the next lab

Next lab (Lab 4) will be based on this lab. So, make sure your codes are working well and independent of other blocks. Moreover, we might replace/extend certain functions.

Putting all requirements in one perspective

Here we will summarize the programming requirements you need to do:

1. Setting up a Timer in PWM mode
    a. Setting up a PWM channel at your desired frequency,
    b. Apply this pulse to the V_in of the above circuit,
    c. Tune the duty cycle of your PWM pulse to make the desired voltage at V_out.
2. Feedback operation:
    a. To ensure the correctness of generated voltage, use your ADC and take samples at V_out,
    b. Use another timer as a source of sampling time for your ADC,
    c. Filter the data through FIR filter and investigate filter parameters.
3. Controller block
    a. Design a controller block to tune the PWM pulse width based on the readings from ADC.
4. Setting up the alphanumeric keypad
    a. Look for free (no conflict/hardwired) GPIO pins and configure them as GPIO as needed, *(Avoid at all costs using PA13, PA14 and PB3. Reconfiguring these pins will damage the board as they are interfaced with the debugger/programmer. You will lose connectivity to the board and ST-Link will no longer detect the board)*
    b. Write the alphanumeric keypad scanning algorithm,
    c. Write the code to handle button press bounce-free and holds.
5. Maintaining the 7-Segment display
6. Write the high-level application which glues everything together toward the intended program
    a. Use state machine to switch between modes easily,
    b. Code documentation and organization is important. Use of labeled constants is highly recommended.

Hints for this lab

It is highly recommended to write your code based on state machines, from getting numbers by keypad to the end which is a continuous closed loop system. As an example, a counter that counts up at every SysTick interrupts can be a source of time for your state machine to switch between modes or tasks you assign.

Demonstration

You will need to demonstrate that your program is correct, and explain in detail how you guarantee the desired precision, and how you tested your solution. Your program should feature each and every requirement from 1 to 6 listed above. Your effort in reducing the power consumption in sleep mode matters. Report and present them in the report-2 and demo 3.

The final demonstration will be on Friday, March 3rd

Bonuses

1. **Early demo bonuses only on the week before the demo**: Thursday 0.2, Wednesday's 0.4, Tuesday's 0.6
2. Three best groups in controller block will get 0.5 bonus. Criteria is based on the lowest ripples on the DC voltage, highest tolerance and fastest reactions on load variations.

Report

The report for Lab 3 will be merged with the report for Lab 4. So, the delivery is after the demo-4.