EE212-Microprocessors Off-Lab Assignment 5

Due Date: 25.04.2022 - 13.30

1 Assignment Details

For this lab assignment, you will implement **Pulse Width Modulated (PWM)** signals using timers of the FRDM-KL25 board. Some pins of FRDM-KL25 board are capable of generating PWM signals. Indeed, PWM signals are nothing but periodic square waves with two different parameters: duty cycle (%) and periodic frequency (Hz). Following are to be implemented in this lab assignment:

1. *10 Points* Frequency (Hz) and duty cycle (%) values are to be read from the keypad, respectively for each signal; show input values on LCD screen. Note that **frequency interval is in between 5Hz and 300Hz and duty cycle interval is in between %0 and %100** (Duty cycle values %0 and %100 are not important). *Figure 1* shows how your LCD screen should look like (where *F1, F2* shows frequencies for each signal,*D1, D2* shows duty cycles for each signal).

Figure 1: Example of LCD screen

2. *60 Points* Using timers of FRDM-KL25 board (TPM timers), generate PWM signals on two-different pins of FRDM-KL25 (by using input values obtained in previous step). Generated signals will be measured from these pins by using oscilloscope probes. **Therefore, you will need two different probes for this assignment**. By using cursors, please check if generated signals have correct frequency and duty cycle values as your input values. Errors up to %3-5 percent are acceptable (for high frequencies, error limit can be a little bit higher). Please, read carefully guidelines for this part; these guidelines ease the process of your implementation.

   You use two different timers to control two different waveforms. *Figure 2* shows how your oscilloscope screen should look like when you take measurements with your probes.

3. *30 Points* Implement the following: **When '*' is pressed on the keypad, frequency of each signal(both two signals) is doubled and when '#' is pressed on the keypad, duty cycle of each signal(both two signals) is added by %10 percent** (If current

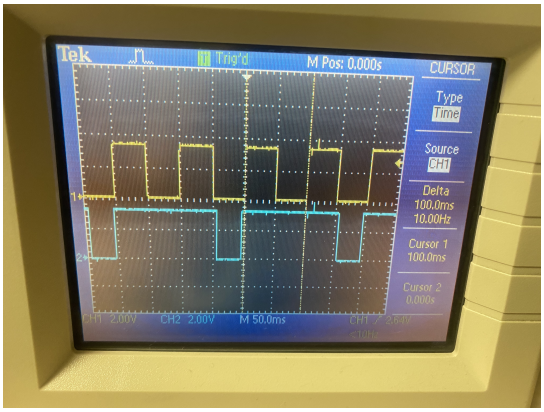

Figure 2: Example of Oscilloscope screen when values in Figure 1 is given to program.

duty cycle value is %50 percent, new duty cycle value becomes %60 if '#' pressed on the keypad). Indeed, one can press button '*' twice while program runs; in that case, frequency is multiplied with 4. Similarly, if you press '#' twice, duty cycle is increased by %20. (Note that if you press '#' multiple times, signal can be saturated which means that duty cycle becomes %100, be careful about that; you didn't have to generate such waveforms with %100 duty cycle). So multiple pressings of each button while running same program can be done. (In this step, you do not need to write new values of frequency and duty cycle on LCD screen).

Figure 3 shows an arbitrary waveform, when we press '#' then duty cycle value is updated. *Figure 4* shows updated waveforms with increased duty cycles (compare it with *Figure 3*). When we press '*' then frequency is doubled. *Figure 5* shows updated waveforms with increased frequency (compare it with *Figure 4*).

2 Guidelines

Follow the guidelines given below for implementation:

(a) Please refer to **Mazidi**'s book *Freescale ARM Cortex-M Embedded Programming*. Chapter 5 - Section 2 is dedicated for general configuration of TPM timers, Chapter 11- Section 2 is dedicated for generating PWM signals, Chapter 2 is dedicated to GPIO.

(b) If you are to use PWM signals generated by the board, you should choose pins which are capable of generating these signals (For example PORTE Pin 20 or PORTE Pin 31 are not used by any of the LCD and keypad connection and may be good option for generating PWM signals). Please refer to the manual in this case. This figure in manual shows which of these pins can be used as a PWM signal generating pin.

(c) TPM timers are used to generate PWM signals. So use TPM timers instead of Systick Timer; this will ease your process. Also note that, there is no restriction about how many timers you should use but one timer for each waveform would be easy and reasonable choice.

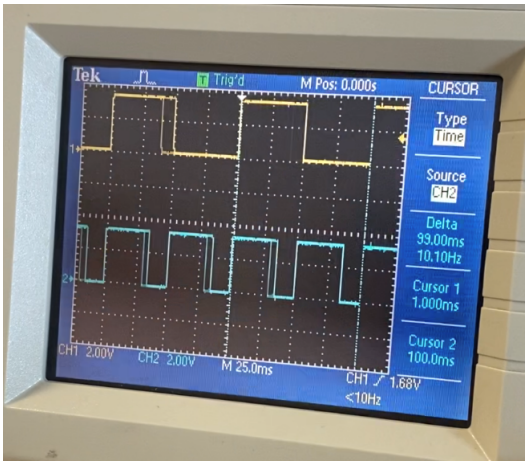

Figure 3: Arbitrary Waveforms

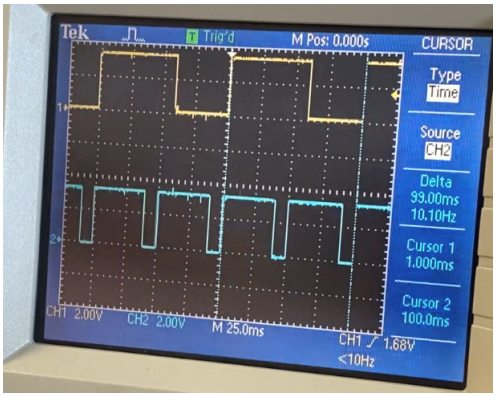

Figure 4: Updated waveforms with increased duty cycles (compare it with Figure 3).

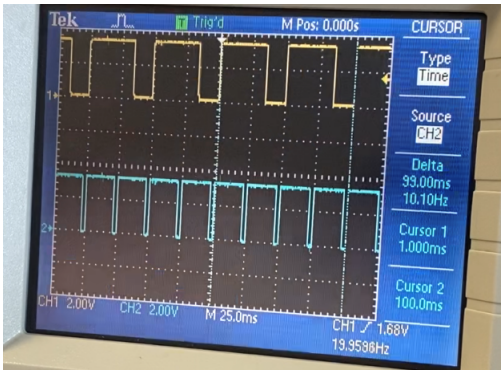

Figure 5: Updated waveforms with increased frequencies (compare it with Figure 4).

(d) Be careful about configuring your timers and ports; if you correctly configure your timers and ports, then most of the lab assignment is done. Try to thoroughly understand all steps of reference examples of PWM (Section 11-2 given in part (a)).

(e) You can use mathematical library for this lab. For example, rounding may be good option for calculating values for timers. So include "math.h " at the beginning of your code. Also, be careful about data types when you make mathematical operations. Although, you can see some warnings when you make mathematical operations with two different data types (for example, operations involving two different data types: float and int), it does not make such difference; it still works.

(f) You can prefer using interrupts when you change duty cycle and frequency instead of taking inputs from keypad. So if you use interrupts to change your values instead of taking input from keypad, it is acceptable and you get full credit for this part.