Lab Session 3 and Assignment 2: Interrupts for real time control implementation and Coulomb and static friction identification (Due 12th Sept)

Objectives:

1. To learn programming philosophy of periodic interrupt timer (PIT) interface and why it is necessary
2. Read through datasheet of XEP 100 and find out registers to be set and values in them to program interrupt to be generated every sampling time
3. Learn to use this interrupt to run a subroutine called interrupt service routine.
4. Learn why it is important to use this interface rather than looping continuously without bothering about exact sampling time.
5. Open loop program to identify Coulomb and static levels of friction in motor and attached gearbox combined as seen on motor side.

Background knowledge required:

Towards a goal of running motor in closed control loop we need to continuously keep doing the following tasks:

1 Read sensors for feedback (in this case potentiometer as position sensor)

2 Compute control law (say proportional control that you simulated for example)

3 implement PWM duty cycle corresponding to this control law. (in similar way you used PWM to drive motor)

In the normal way of programming we will typically tend to do 1-3 activities in a loop forever. However, in this way we would not have control over at what sampling time these activities are done. This will get decided based on what is the time taken for execution of the program you have written in for loop. There are a lot of disadvantages in this way of implementation one of them being the sampling time would change based on modifications in program and so on.

Interrupt philosophy (example of you receiving a phone call while reading a book, you will bookmark and come back to reading once call is over) would be helpful to overcome this problem.

A periodic timer based interrupts are provided in microcontrollers for this purpose (each pulse corresponds to phone call). Interrupts are programmed such that an "Interrupt service routine (ISR)" is run ONCE every time the interrupt pin sees a pulse on it (like your conversation on call, same every time though ☺). If now your tasks 1-3 are written in the interrupt service routine, your job 1-3 would be executed every sampling time (when the interrupt pin gets a pulse). This way irrespective of your program length in ISR the ISR is executed only at a fixed time interval called "sampling time".

In PIT interface in XEP 100 microcontroller, a flag is set in a flag register every time the down-counter counts value to zero before loading the count again. The setting of the flag (every time) triggers running of a specific function designated by a vector number (interrupt 66 in this case for channel 0). The flag is to be reset again to enable the next execution of ISR upon the next. In other microcontrollers the hardware generates a pulse every fixed time period

corresponding to sampling time. This pulse is given to an interrupt pin. The setting of this sampling time and other initiation of different registers can be done in the main program.

Things to do?

1. PIT: See the information related to the following registers: PITCE_PCE0, PITMUX, PITMTLD0, PITLD0, PITINTE_PINTE0, PITCFLMT_PITE, PITTF
   in the datasheet of XEP 100.
   Use given program to be more systematic, this is required because the complexity of your program is going to increase as we start putting together everything towards motor control operation. You may change things according to desired goal.

Recording the real time data in file:
After compiling the program and running the motor, write the command "cf log_command.txt" in the command window. It is assumed that the file log_command.txt (written below) is placed in project directory (same place where *.mcp of the project sits).
Wait for some time till the log file is written. Now open the log file and try to analyze the results.
Log file to be separately created:

Create a text file named 'log_command.txt' in the operating directory. Type the following in 'log_command.txt' :

```
DEFINE loop = 0
FOR loop = 1..10000,1
fprintf(log2.txt,"%d %d\n", count, En_Read)
ENDFOR
```

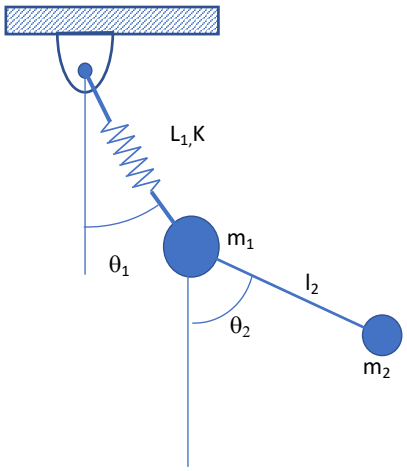
In log_command.txt, number 10000 indicates the number of times the logging has to be done, counter and En_Read are variable names (which are to be logged). Each reading is printed 16 times in this file for some reason. Use matlab to import and remove multiple readings and to observe the data carefully. Counter indicates the number of cycles passed before the next data is printed. Develop program to process data to finally get smoothly varying motor angle in rad as a function of time. You may need to use calibration in part 1. Show the data to TA and verify.

Assignment 2: (its not expected to be complete in this session but you will do again sometime in this week and submit on 12th Sept)
PART A *(This is already done without PIT or interrupts in lab class)*: Setup PIT to set sampling time approx. 1 ms. Generate triangular waveform of suitable amplitude, and suitable very low frequency in ISR (read complete problem to get what is "suitable"). So as the ISR is executed every sampling instant the waveform update will happen. Now set up program to run motor and assign the output value of this triangular waveform to duty cycle of PWM used to run motor. What is expected to happen at low and gradually increasing duty? Motor will start, run and stop again repeat the same in one direction. Use encoder to monitor at which PWM duty cycle motor starts and stops. Use these values to get Coulomb and static friction torque in the motor. Obtain torque in units of Nmm using datasheet of the motor and model of the motor.

PART B:
Consider double pendulum system shown in the figure. String of the upper pendulum is flexible while another string is not flexible. Identify degrees of freedom and the generalized coordinates you would like to use. Using Lagrange formulation, derive equations of dynamics of the system. Parameters of pendulum are mentioned below. Further develop matlab code to simulate and animate the system for

1. Roll nos ending with odd digit: Given initial conditions: $\theta_1 = 20\,\text{deg}\,\theta_2 = 40\,\text{deg}$ And spring into its unstretched position
2. Roll nos ending with even digit (including 0): Given initial conditions: $\theta_1 = 40\,\text{deg}\,\theta_2 = 10\,\text{deg}$ and spring into its unstretched position

While creating animation draw the size of the pendulum proportional to the mass of pendulum.



Parameters:
m_1 = (last digit of your roll no + 1) * 20 gm
m_2 = 30 gm
L_1 = unstretched length of the spring connected to m_1 = 0.5 m
L_2 = 0.3 m
Stiffness of spring = 1 mm/gm