Lab Session 4: Closed loop control of motor P and PD control

Objectives:

1. Learn to use ISR to setup closed loop control of motor using encoder feedback
2. Understand differences between theoretical and practical control implementation
3. Challenge goal: see if you can achieve error within 5 counts of encoder and reason out why not or how?

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 for example for todays lab)

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

We will do these tasks in with control law as proportional control and see. A few points for practical implementation, if you are getting stuck somewhere refer to these points and see if they can give you some hint.

a. The data type in microcontroller is integer either signed or unsigned. Some cases float is supported. However with float type memory required increases. Its best to know what are different limits of numbers that typically you will have and use integers values.
b. From perspective of a. if there are divisions resulting in fractions say 0.45 which is further multiplied by 100, the actual number that might go to resulting variable could be zero. So we need to ensure in this case that the multiplication takes place first and then division.
c. You need to be aware about bit accuracy and ranges of number possible for different variables and registers. For example EnRead is 16 bit number (defined by HCTL interfacing) although the representation may be in long int. It cannot have value more than 2^16 in unsigned definition and more than 2^15 in signed definition.
   For example, PWMDTYx register cannot have value more than 255 (since PWMPER register is holding 0xFF)
d. Control computation when error is multiplied by gains (or some cases divided by gains) may result in + or –ve number to be given to motor. In case of +ve number we need to set direction bits of motor driven in one way say InA 0 and InB 1 and in case of –ve number we need to set the other way. Absolute of the control computation needs to be given as duty cycle to PWM controlling motor (PWMDTY register).
e. The sense of mathematical +ve error needs to move the motor in direction such that error reduces. We need to make sure this happens else we are actually destabilizing the system and it would not control well (think what would happen in such case and actually observe). How will you take care of such possibility (Yes by reversing polarity of InA and InB inputs controlling motor direction).

Things to do?

1. Use given program and make connections and check if ISR set by PIT initiation is running properly and further encoder reading is coming out properly. If not debug using the Easyscope and correct.
2. Setup a reference position square wave with 1 revolution of gear as amplitude and 10 sec as the time period. This reference is to be followed by your motor actual position.
3. Compute proportional control using Error = Desired position - current position and assign resulting computation to PWMDTYx register to control motor with P control. (see that the newduty is the variable to which you assign this computation). Observe what is happening and see effect of proportional gain on control.
4. Next compute derivative by difference formula Error_dot = Error – Old_Error OR using difference of position to get velocity and implement only derivative action computation to PWMDTYx register. Now you should feel resistance only when you try to move the motor. 'Feel' this derivative action for different gains.
5. Finally use combination of P and D actions to get control where error is settled within 100 ms and steady state error is withing 10 counts of EnRead.
6. Think why there is steady state error in the first place. How can you take care of this error.

You may use the following if you would like to record and see through the data more closely.

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.

Feel free to ask me any doubts queries you have (in email if not in person) and I will sort out.