ME218b
Smart
Product
Design

Laboratory Assignment Number 7 for ME 218b

Pre-Lab Due by 5:00 pm on January 30, 2018
Due by 5:00 pm on February 02, 2018

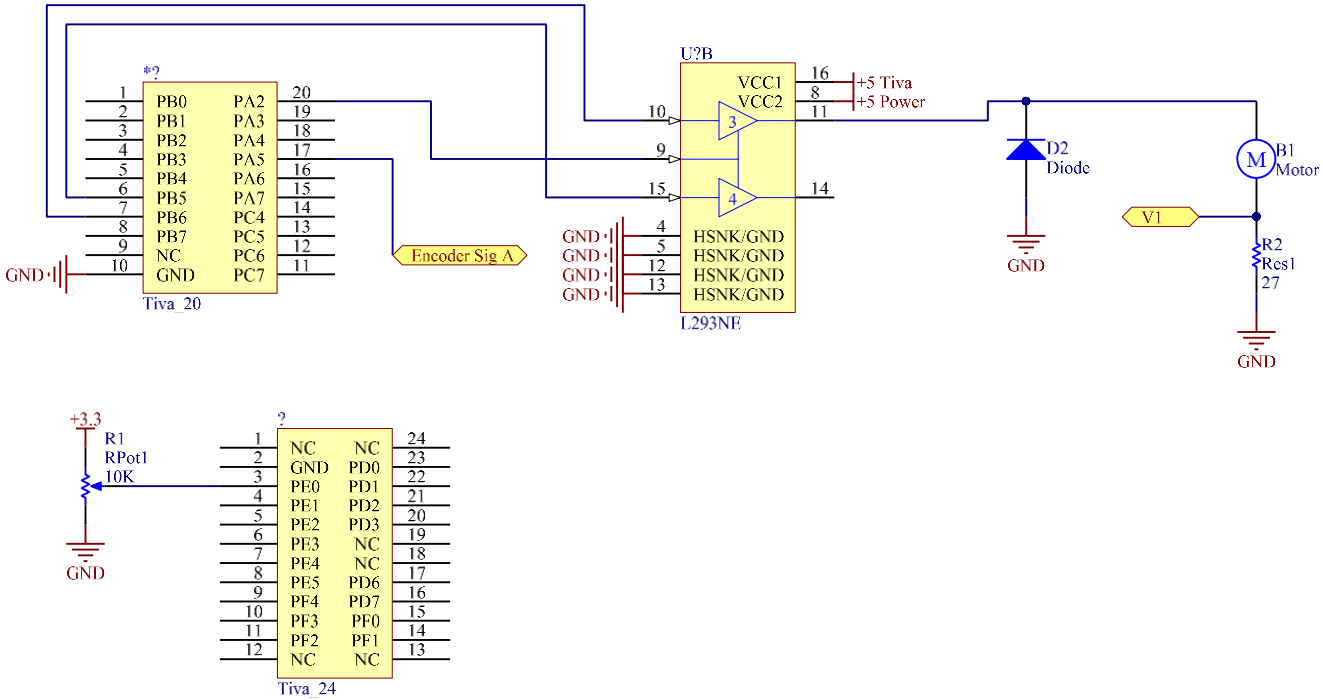
Goals:

In this lab you will explore the behavior of a DC motor being driven by Pulse Width Modulation (PWM) using a Drive-Brake mode. You will also gain experience implementing closed loop speed control of the DC motor.

Part 0: Pre-Lab

Assignment:

☐ 0.1) Design a circuit to drive the DC motor using Drive-Brake mode.



☐ 0.2) Figure out how you will measure the electrical time constant of the motor as requested in Part 1.1.

The electrical time constant of the motor can be measured by probing V1 and seeing the amount of time it takes for the voltage to get to 63% percent of the steady-state value. Using the equation tau = L/R, the inductance of the motor can then be calculated with R representing the resistance of the motor coil plus R2. The time constant of the motor itself can then be calculated by diving L by the resistance of jus the motor coils.

☐ 0.3) Design the software that you will use for Part 1. Include pseudo code for program(s).

☐ 0.4) Design the software that you will use for Part 2. Include pseudo code for program(s).

In the report:

Include your answers to all of the sections of the pre-lab in your final report document.

0.3)

```
ES_Event_t RunDCMotorService(ES_Event_t ThisEvent)
{
    ES_Event_t ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

    //read value from analog potentiometer service
```

```
    //convert from analog range to pwm range
    //NewValue = (((OldValue - OldMin) * (NewMax - NewMin)) / (OldMax - OldMin)) +
NewMin



    //set duty cycle of DC motor PWM dir pin to variable: speed
    //if close to within 5 of each extreme use GENA register to completely turn motor
on or off
    if (speed >= max)
    {
        //make 0 power
    }
    else if (speed <= 5)
    {
        //make full power
    }
    else
    {
        //set to normal action first
        //set comparator value to speed
    }
    return ReturnEvent;
}

Uint32_t timerValue = 0;
ES_Event_t RunADService(ES_Event_t ThisEvent)
{
    ES_Event_t ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

    if (ThisEvent.EventType == ES_TIMEOUT)
    {
        uint32_t analogData[1] = {0};
        ADC_MultiRead(analogData);
        timerValue = analogData[0];
        PostDCMotorService(ReturnEvent);
    }

    ES_Timer_InitTimer(AD_TIMER, AD_TIME);

    return ReturnEvent;
}
```

0.4)

```
void Feedback_ISR(void)
{
    uint32_t ThisCapture;

    //start by clearing source of interrupt, the input capture event
    HWREG(WTIMER0_BASE+TIMER_O_ICR) = TIMER_ICR_CAECINT;

    //Calculate duty cycle
    //error = abs(desiredRPM - currRPM);
    //errorSum += error;
    //if errorSum is greater than max values allowed
         //Make errorSum the max value
    //if errorSum is lower than the min value allowed
         //Make errorSum the min value

    //duty = error * Kp + (errorSum * Ki)
}
```

```
void ICR_Encoder_Input_Capture(void)
{
    uint32_t ThisCapture;

    //start by clearing source of interrupt, the input capture event

    //Grab captured value and calculate period

    //update LastCapture to prepare for next edge

    //restart one-shot timer and set flag as false
    oneShotFlag = false;
}
```