Name: **Lab Section:**

CprE 288 Fall 2018 – Homework 6

Due Sunday. October 21 (on Canvas 11:59pm)

Notes:

- Homework must be typed and submitted as a PDF or Word Document (i.e. .doc or .docx) only.
- If collaborating with others, you must document who you collaborate with, and specify in what way you collaborated (see last page of homework assignment), review the homework policy section of the syllabus: http://class.ece.iastate.edu/cpre288/syllabus.asp for further details.
- Review University policy relating to the integrity of scholarship. See ("Academic Dishonesty"): http://catalog.iastate.edu/academic_conduct/#academicdishonestytext
- Late homework is accepted within two days from the due date. *Late penalty is 10% per day.* ***Except on Exam weeks***, *homework only accepted 1 day late.*
- ***Note:*** *Code that will not compile is a typo. Answering a question as "will not compile"* ***will be marked incorrect****. Contact the Professor if you think you have found a typo.*
- **Note: You are not allowed to use any MACROs in your code, except for register names.**
  - Example: You will lose points for: GPIO_PORTA_DEN_R = GPIO_PORTA_DEN_R | PIN1
  - Must use: GPIO_PORTA_DEN_R = GPIO_PORTA_DEN_R | 0b0000_0010; // or 0x02

Note: Unless otherwise specified, all problems assume the TM4C123 is being used

Question 1: ADC Successive Approximation (5 pts)

A 4-bit ADC (of 16 steps in the analog range) uses the Successive Approximation implementation. The input voltage range of the ADC is 0V-32V. If the input is 20V, how does the ADC work out each bit of the digital encoding? Fill the following table to show the steps (as done in class). The first step is given.

Step	Range	DN_Mid	AV_Mid(V)	Input >= AV_Mid
0	xxxx	1000	16	1
1	1xxx	1100	24	0
2	10xx	1010	20	1
3	101x	1011	22	0
4	1010			

The digital value is ___1010_______ (binary) and _____10_____ (decimal).

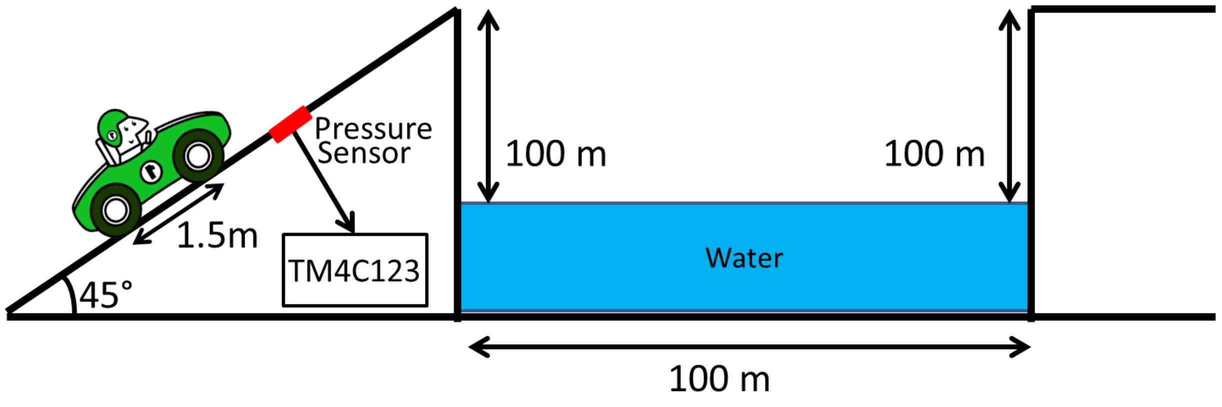
Name: **Lab Section:**

Question 2: Shutdown system (15 pts)

A pressure sensor is embedded in the ramp below. When pressure is applied to the sensor, a logic 1 is driven on the Timer 1 Input Capture input wire of the TM4C123, else a logic 0 is driven. You are to write a C-program that will apply the car's breaks to stop it if its velocity is not large enough to jump the gap shown below. This program is required to use the Timer 1 Input Capture interrupt.

Note: You are not allowed to use any MACROs in your code, except for register names.
- Will lose points for: GPIO_PORTA_DEN_R = GPIO_PORTA_DEN_R | PIN1
- Must use: GPIO_PORTA_DEN_R = GPIO_PORTA_DEN_R | 0b0000_0010; // or 0x02

Assumptions:
1) The car is moving at a constant velocity while on the ramp
2) The car is treated as a "point mass" for computing the physics of the problem



a) Complete `Config_Timer1` to program the Timer 1 configuration registers as follows: (5 pts)

- Input Capture interrupt enabled
- 24-bit timer
- Detect positive edge events

Name: **Lab Section:**

```
Config_Timer1()
{
   // 1. Setup GPIO
  //A. Configure GPIO module associated with Timer 1 A
       // Since A or B was not specified either is fine. I used A

    // i. Turn on clock for GPIO Port B
    // Note: Timer 1A can use Port B or F, this solution uses Port B
    // Note: Port F would use different pins of its port.
    SYSCTL_RCGCGPIO_R = SYSCTL_RCGCGPIO_R | 0b000010; // 0x02

    // ii.Enable Alternate function and set Peripheral functionality
    GPIO_PORTB_AFSEL_R |= 0b0001_0000; // Timer1 A Input is on bit 4
    GPIO_PORTB_PCTL_R |= 0x0007_0000; // use Timer input for wire 4

    // iii. set digital or analog mode, and pin directions
    GPIO_PORTB_DEN_R |= 0b0001_0000; //enable pin 4 digital mode
    GPIO_PORTB_DIR_R &= 0b1110_1111; // set pin 4 to input

  // 2. Setup Timer 1 A
  // A) Configure Timer 1 mode

    SYSCTL_RCGCTIMER_R |= 0b0000_0010; // Enable Timer 1's clock

    //Disable Timer 1 device while we set it up
    TIMER1_CTL_R &= 0xFFFE;

    // Set desired Timer 1 functionality
    TIMER1_CFG_R = 0x4; // Set to 16-bit mode
    TIMER1_TAMR_R = 0b0001_0111;//Count up,Edge-Time Mode,Capture Mode
    TIMER1_CTL_R = 0b0000_0000; // Detect positive edges

    TIMER1_TAPR_R = FF; //set timer prescaler(extended timer) and load
    TIMER1_TAILR_R = 0xFFFF // init value to max 24-bit Timeout value

  // B) Setup Timer 1 Interrupts
    TIMER1_ICR_R |= 0b0000_0100;//Clear Event Capture interupt status
    TIMER1_IM_R  |= 0b0000_0100; // Enable Event Capture interrupts

  // 3. NVIC setup
  // A) Configure NVIC to allow Timer 1A interrupts
     //Note: Priority was not specified so any priority is fine.
  NVIC_PRI5_R |= 0x0000_2000;//Timer1A IRQ Pri to 1 Grp5 bits 15-13
  NVIC_EN0_R |=0x0020_0000; //Enable Timer 1A IRQ(ie.IRQ21):set bit 21

  // B) Bind Timer 1A interrupt requests to User's Interrupt Handler
     IntRegister(INT_TIMER1A, Timer_1_Handler);

  //re-enable Timer 1A
  TIMER1_CTL_R |= 0x0001;
}
```

Name: **Lab Section:**

b) Complete the ISR (i.e. handler) called `Timer_1_Handler` to store `first_wheel_hit` and `second_wheel_hit` (5 pts)

c) Complete `Stop_car()`. It should return 1 if the car is not moving fast enough to jump the gap (5 pts)

```
// Global variables (you may use additional variables)
volatile unsigned int first_wheel_hit;  // When 1st wheel hits sensor
volatile unsigned int second_wheel_hit; // When 2nd wheel hits sensor
volatile int done_flag=0; // 1 after both first and second_wheel_hit
                          // have been stored


// Store first_wheel_hit and second_wheel_hit
void Timer_1_Handler(void)
{
  static int state = 0; // Using a global instead also fine

  // Check if an edge time capture interrupt occurred
  if(TIMER1_RIS_R & 0b0000_0100)
  {
    // Clear the interrupt
    TIMER1_ICR = 0b0000_0100;

    // Capture rising edge time for when 1st wheel hits sensor
    if(state == 0)
    {
      first_wheel_hit = TIMER1_TAR_R;
      state = 1;
    }
    else // Capture rising edge time for when 2nd wheel hits sensor
    {
      second_wheel_hit = TIMER1_TAR_R;
      done_flag = 1; //tell main done capturing 1st & 2nd wheel times
      state = 0;
    }
  }
}
```

Name: **Lab Section:**

```
// Return 1 if the car is not fast enough to jump the gap
int Stop_car(void)
{
  float velocity_car;
  float time;

  // Compute car velocity based on captured times and distance
  // between wheels: d = v*t =>  v = d/t

  // Assuming at most one overflow, so not dealing with overflow
  // 16 MHz tick rate => .0625us tick period

  time=( second_wheel_hit - first_wheel_hit) * .0625 // In usec
  time= time/1000000; // Time in seconds
  velocity_car = 1.5/time;  // In m/s

  // Assume min velocity was computed offline, fine if computed
  // in program.  Min_velocity is about 31.2 m/s (see below for
  // one way to compute off-line)
  if(velocity_car < 31.2)
  {
    return 1;  // Not fast enough stop
  }
  else
  {
    return 0;  // Fast enough make jump
  }
}
```

Name: **Lab Section:**

```
// Stop Car if it is not fast enough to jump the gap.
main()
{
  int stop = 0;
  Config_Timer1();

  while(1)
  {
    // Wait for events to be captured
    while(!done_flag)
    {
    }

    done_flag = 0;  // Clear flag

    // Check if car needs to stop
    stop = Stop_car();

    if(stop)
    {
      lprintf("Stopping Car!";
    }
    else
    {
      lprintf("Car going to Jump!!");
    }

  } // end while
}
```

// Compute minimum required car velocity (**Vcar**)
1) **dx = V_car_x * t_air**, where t_air is time car is in air and V_car_x is the car's horizontal velocity, and dx is 100 m for this case

2) **dx = V_car_x * (2*t_up)**, where t_up is the time for the car to reach its max height after leaving ramp (t_air = 2*t_up)

3) **V_top = V_car_y_initial + a * t_up**, where V_top is the cars vertical velocity at its highest point (i.e. 0 m/s), V_car_y_initial is the car's initial vertical velocity after leaving the ramp.

4) V_car_x = cos(45) * Vcar ~ **.71*Vcar**, and V_car_y_initial = sin(45)*Vcar ~ **.71Vcar**, where Vcar is the speed of the car.

5) Solve for t_up, given V_top = 0, a = -9.81.
t_up = .71*Vcar / 9.81.

6) Subsitute t_up and .71*Vcar into 2) and solve for Vcar
100 = .71*Vcar * (2*.71*Vcar/9.81), **Vcar ~ 31.2 m/s**

Name: **Lab Section:**

Question 3: Software implemented Input Capture (10 pts)

a) Assume the TM4C123 does not have Input Capture hardware or Interrupts. Write a C program to save TIMER1's count value (i.e. TCNT1) when a positive edge event occurs on PortD, pin4. Note: you may want to do 3c first. (4 pts)

```
int main(void)
{
  unsigned int rising_edge_time;

  // Assume GPIO has already been configured properly
  // Assume the Timer has been placed in a Periodic Mode

  while(1)
  {
    // Check if the current value of wire 4 is 0
    if( (GPIO_PORTD_DATA_R & 0b00010000) == 0)
    {
      // wait for wire 4 to become a 1
      while( !( GPIO_PORTD_DATA_R & 0b00010000))
      {}
      rising_edge_time = TCNT1;  // Store counter value
    }                            // or alternatively GPTMTAV
  }

}
```

b) Describe two disadvantages of your software-implemented input capture program, as compared to using Input Capture hardware with Interrupts. (3 pts)

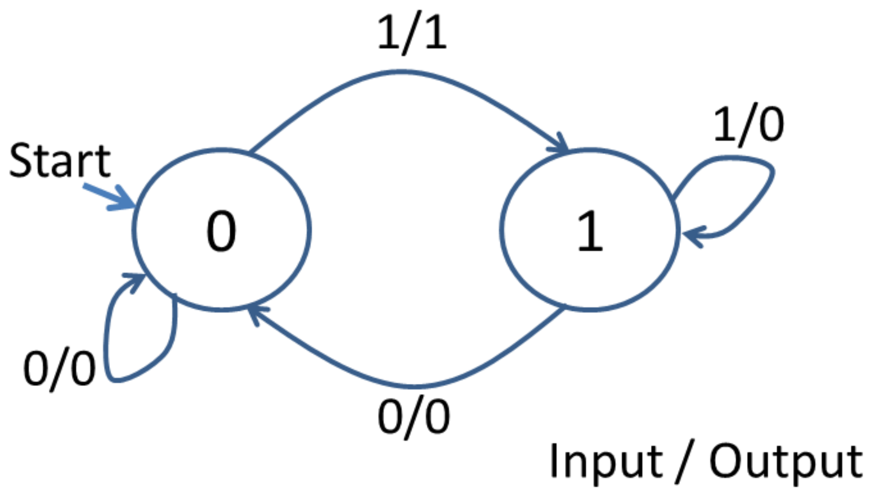
1. Does not allow other parts of the code to run, while it is running.

2. With the "software only" approach an interrupt could occur after detecting a positive edge, but before the TIMER value is read. Thus adding to the error of when the positive edge occurred.

3. The Input Capture hardware stores the value of TCNT as soon as the event occurs, while the software implementation will have error related to where in the polling loop the code is when the positive edge occurs.

Name: **Lab Section:**

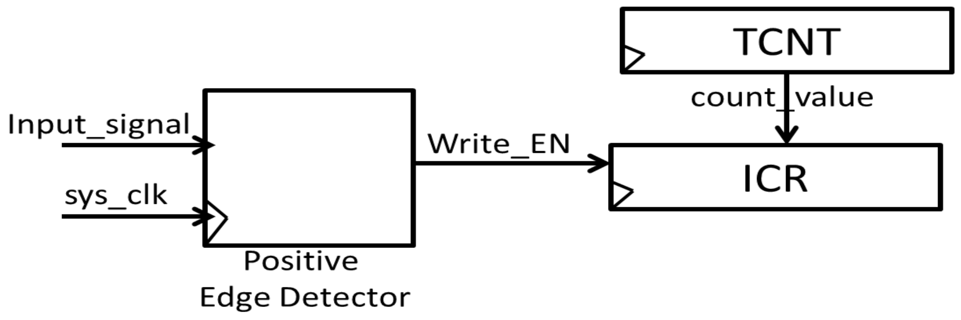
c) Complete the bubble diagram below to implement a Mealy State Machine that implements a positive edge detector (i.e. the state machine outputs 1 each time a positive edge occurs on the input). Assume at Start the input to the state machine is initially 0 (3 pts)



Name: **Lab Section:**

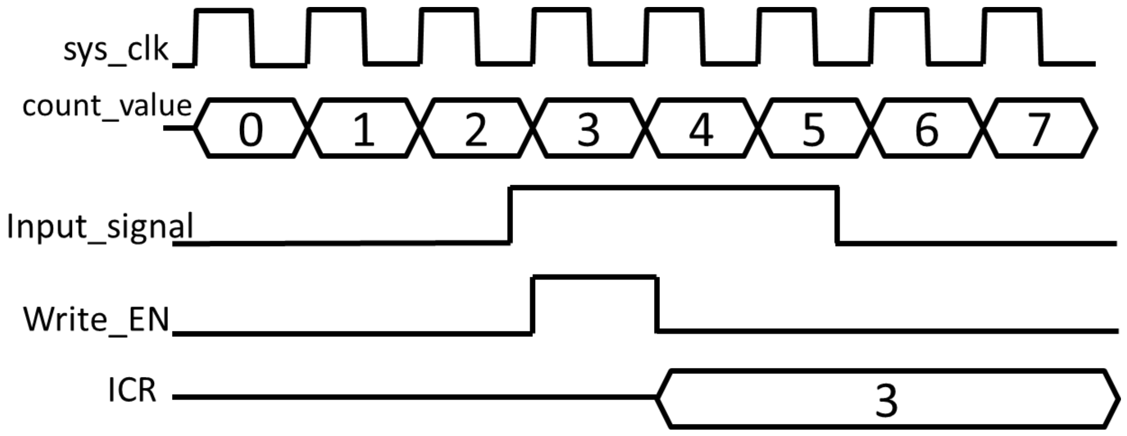
Question 4: Edge Detection (10 pts)

The following simple block diagram illustrates how one may implement an Input Capture circuit.



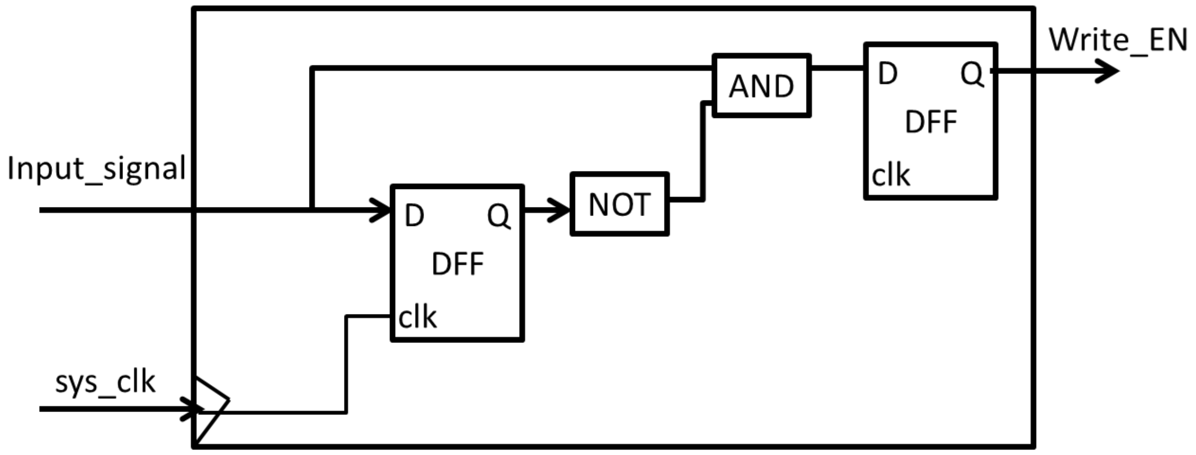
An example timing diagram for capturing the rising edge of `Input_signal` is given below. In summary, on the occurrence of a positive edge on `Input_signal`: 1) `Write_EN` pulses '1' for one `sys_clk` cycle, and 2) the value in the Timer/Counter Register (TCNT) is loaded into the Input Capture Register (ICR).

Assume: Any time `Input_signal` transitions is will hold its value for at least 1 `sys_clk` cycle.



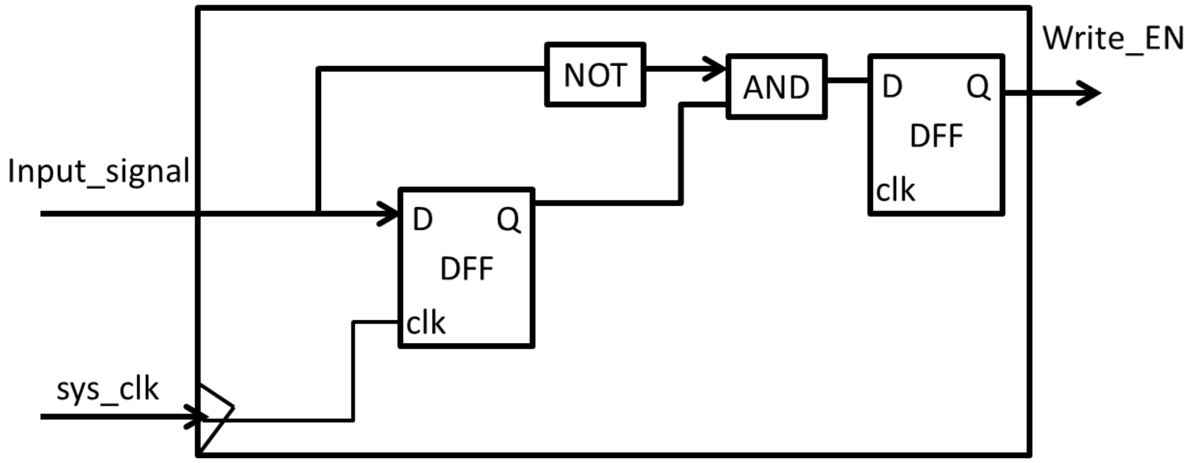
Name: **Lab Section:**

a) Draw out a digital circuit to implement an Edge Detector that creates a 1-sys_clk-wide pulse on Write_EN to load the TCNT register into the ICR register when a rising edge occurs on Input_signal. The only components you can use are D-Flip Flops, and AND, OR, NOT gates (5 pts)



Positive Edge Detector

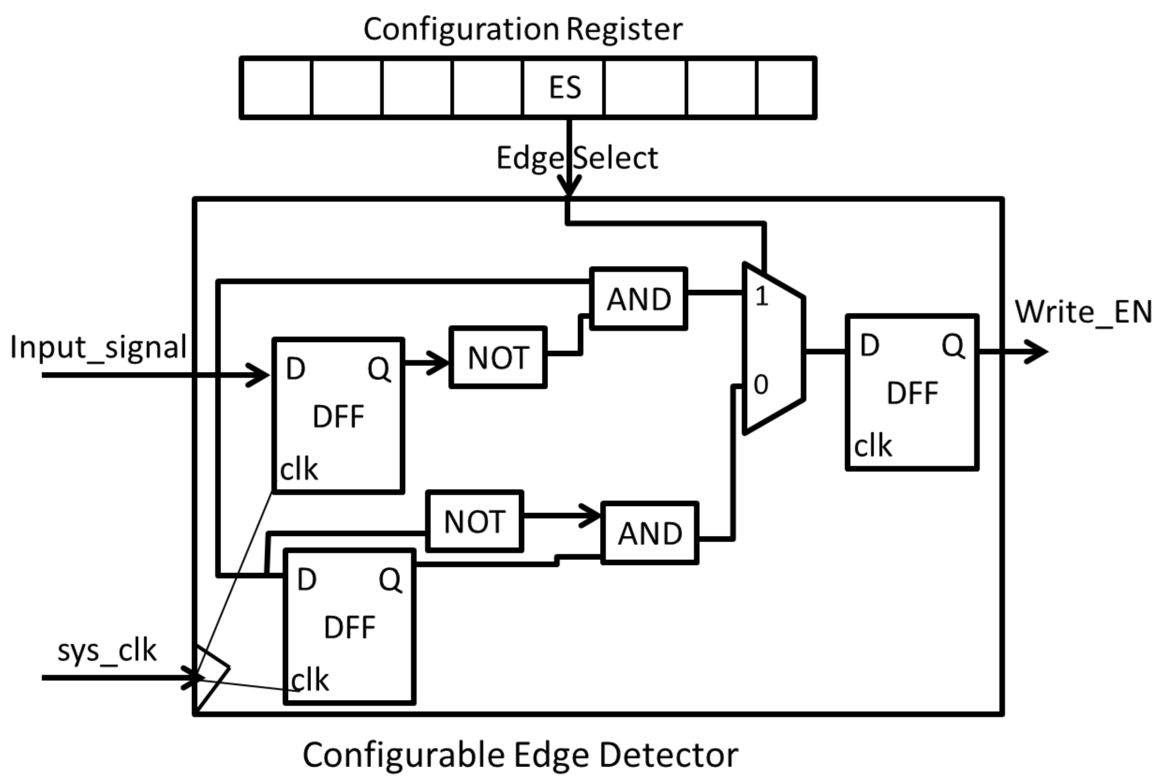
b) Repeat a) for an edge detector that creates a Write_EN pulse for detecting the falling edge of Input_signal. (3 pts)



Negative Edge Detector

Name: **Lab Section:**

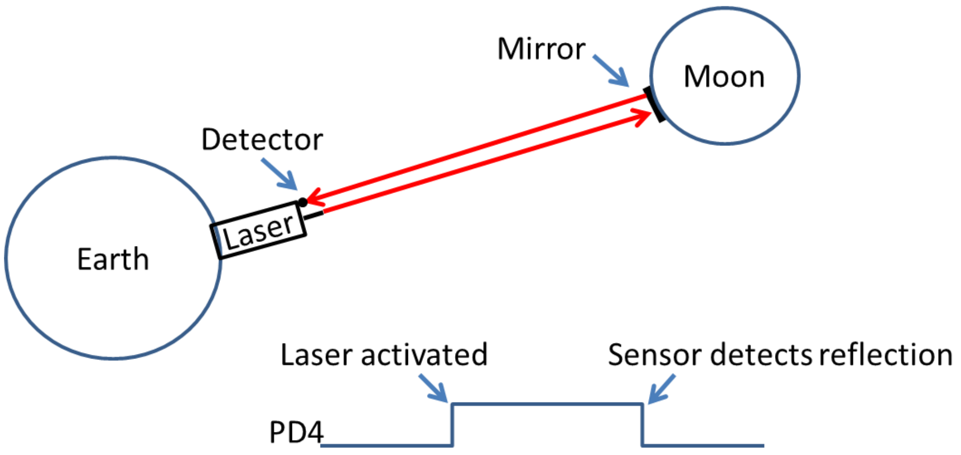
c) Let us assume we can configure the hardware for detecting either the positive edge or negative edge of an input by configuring the "Edge Select" bit of a configuration register (call it the Edge Select bit: ES). Draw the digital circuit to allow the Edge Detector to detect a positive edge when ES=1, and a negative edge when ES=0. You may now use multiplexers in addition to D-Flip Flops, and AND, OR, NOT gates (2 pts)



Name: **Lab Section:**

Question 5: Timer Accuracy (10 pts)

A laser that works similar to your Lab 7 Ping sensor is used to measure the distance to the Moon from Earth. The laser is fired at a mirror placed on the Moon, and a sensor attached to the laser detects when the reflection arrives back to the laser. As shown below. This is similar to an actual method used for measuring the Earth-Moon distance, see:
http://en.wikipedia.org/wiki/Lunar_Laser_Ranging_experiment



When the laser is activated, TIMER1's Input Capture pin is set to a 1. When the sensor detects the reflection, this pin is set to 0. A program has been written that uses Input Capture to compute the distance between the Earth and the Moon.

Given that Input Capture measures time in Timer ticks (i.e. clock cycles), how different can the programs calculation of the Earth-Moon distance be from the actual distance?

a) Explain what causes the error in the measured distance (5 pts)

Input Capture computes time from the number of clocks between events. If an event does not occur exactly on a positive clock edge, then there will be error in the measured time.

For example, if the "Sensor Detect" event happens right after a positive clock edge, then this time will be too long by nearly 1 clk cycle. And if the "Laser activated" event happens right after a positive edge, then this time will be too short by nearly 1 clk cycle.

Thus a worst case time measurement error will occur when "Laser activated" occurs exactly on a positive edge of the Timers clock, and "Sensor Detect" occurs right after a positive edge. This gives a maximum error in measuring time of 1 clk tick.

***Note: If a student says 2 clk cycles, then give full credit. They are thinking in the correct direction.**

Since: Distance = Velocity * Time, having error in the measurement of time directly impacts one's calculation of distance.

Name: **Lab Section:**

b) Compute the maximum error in distance for each the following speeds of the system clock used by TIMER1: 16MHz, 8MHz, 1MHz, 1KHz (5pts)

First compute the maximum error in time (i.e. period of one Timer tick) for each system clock speed, then compute how far light can travel in that amount of time (velocity of light is 3x10^8 m/s)

Note: The below errors should be divided by two since we are measuring round trip distance.

System Clock 16 MHz: Freq = 16 MHz, Period = 1 / 16 MHz
Distance = 3x10^8 * 1/16 MHz = 18.75 m // height of a 5 story building

System Clock 8 MHz: Freq = 8 MHz, Period = 1 / 8 MHz
Distance = 3x10^8 * 1/8 MHz = 37.5 m

System Clock 1 MHz: Freq = 1 MHz, Period = 1 / 1 MHz
Distance = 3x10^8 * 1/1 MHz = 300 m

System Clock 1 KHz: Freq = 1 KHz, Period = 1 / 1 KHz
Distance = 3x10^8 * 1/1 KHz = 300,000 m // 34 times the height of Mt. Everest

Side Note: The distance from the Earth to the Moon is about 238,900 miles (384 Million meters).

****Note: If students have ½ of the error, then that is fine and is more correct (i.e. round trip)**
****Note: If a student assumed worst case time is 2 clks it is fine also**

Collaboration Documentation

List the people (First and Last name) you collaborated with: ___________________________.

For each collaborator, describe the manner in which you collaborated:

1)

2)