Name: **Lab Section:**

CprE 288 Fall 2018 – Homework 5
Due Sunday October 14 (11:59 on Canvas)

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: UART Basics (7pts)

a) Sketch the logic waveform appearing at the output of the UART when it transmits a character 'T' at a baud rate of 9,600. The sketch should show the bit durations in microseconds, in addition to the waveform. The frame format is of 1 start bit, 8 data bits, an odd parity bit, and 2 stop bits. [4 pts]

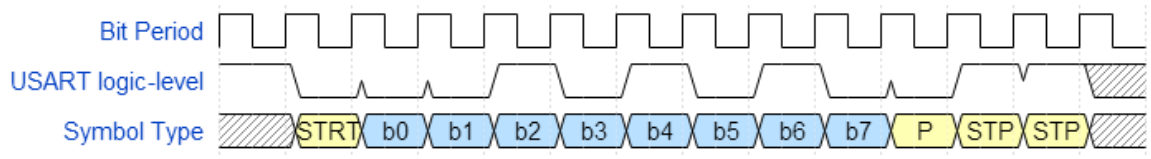
ASCII 'T' = 0x54 =0b0101 0100
Note: UART transmits the least significant bit first. **Gading Note: -1pts for incorrect bit order**

From datasheet an Odd Parity bit is computed as:

$$
\begin{aligned}
P_{odd} &= d_7 \oplus d_6 \oplus d_5 \oplus d_4 \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \\
&= 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \\
&= 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \\
&= 0 \oplus 1 \oplus 1 \\
&= 1 \oplus 1 \\
&= \underline{0}
\end{aligned}
$$

$$\mathbf{P_{odd} = 0}$$

Bit duration(Bit Period) in microseconds = 1/9600 = **104.17us**



Name: **Lab Section:**

b) What is the data rate of the UART configuration given in part a? [3pts]

Data rate is the fraction of the Baud Rate for transmitting data bits.
Number of bits in a frame: 1 start + 8 data + 1 parity + 2 stop = 12 bits

Data Rate = BAUD_RATE * (8 data_bits / 12 frame_bits)
Data Rate = 9600 * 8/12 = 9600 * .6667 = 6400 data bits / second

Question 2: Software vs. Hardware implemented UART (15pts)

a.) **Software:** Assume there is no UART hardware device. Complete functions `init_portB` and `serial_send(char my_txt)` to implement the UART protocol in software. [7pts]

Given: Assume you have available a function called `wait_us(float WaitTime)` that waits for `WaitTime` microseconds before continuing.

```
// Configure pin 3 of Port B as an GPIO output  [3pts]
void init_portB()
{

  // Start Port B clock
  SYSCTL_RCGCGPIO_R = SYSCTL_RCGCGPIO_R | 0b000010; // 0x02

  // Enable Digital functionality of Port B pin 3
  GPIO_PORTB_DEN_R = GPIO_PORTB_DEN_R | 0b0000_1000; // 0x04

  //Set direction of Port B pin3 to output
  GPIO_PORTB_DIR_R = GPIO_PORTB_DIR_R | 0b0000_1000; // 0x04

  // Set wire for GPIO (OK if missing this line for HW5)
  GPIO_PORTB_AFSEL_R = GPIO_PORTB_AFSEL_R & 0b1111_0111;

}
```

Name: **Lab Section:**

```
// Send my_txt out of Port B pin 3 encoded in the UART frame and
// speed specification given in Question 1. Since there is no
// UART device, your code must use the GPIO Data Register of
// Port B pin 3 to transmit the 8-bits of my_txt. [4pts]
void serial_send(char my_txt)
{

  int i = 0;
  int odd_parity = 1;
  float BitTime = 104.17;

  // Send Start bit
  GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R & 0b1111_0111;
  wait_us(BitTime);

  // Send Data bits
  for(i=0; i < 8; i++)
  {
    if( ((my_txt>>i) & 0x01) )
    {
       GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R | 0b0000_1000; //Send 1
    }
    else
    {
       GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R & 0b1111_0111; //Send 0
    }
    wait_us(BitTime);
  }


  // Compute and send Odd Parity bit
  for(i=0; i < 8; i++)
  {
    odd_parity = odd_parity ^ ((my_txt>>i) & 0x01);
  }

  if(odd_parity)
  {
    GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R | 0b0000_1000;
  }
  else
  {
    GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R & 0b1111_0111;
  }
  wait_us(BitTime);


  // Send 2 stop bits
  GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R | 0b0000_1000;
  wait_us(2*BitTime);
}
```

Name: **Lab Section:**

b.) **Hardware:** Complete `serial_init` and `serial_send` making use of the UART hardware device. [8pts]

i) Complete the function `serial_init` to configure UART0 to: [5pts]

- Match the specifications given in Question 1
- Enable transmitting, Disable receiving
- No interrupts used (so ignore registers related to interrupts)
- Set up the GPIO registers to allow UART0 to transmit. Preserve all other GPIO settings.

```
// Initialize UART0 and associated GPIO Port/pins
void serial_init()
{
  // 1. Setup GPIO
  //A. Configure GPIO module associated with UART 0

    // i. Turn on clock for GPIO module A
    SYSCTL_RCGCGPIO_R = SYSCTL_RCGCGPIO_R | 0b000001; // 0x01

    // ii.Enable Alternate function and set Peripheral functionality
    GPIO_PORTA_AFSEL_R |= 0b0000_0010; // 0x02   UART0 TX
    GPIO_PORTA_PCTL_R |= 0x00000010;   // Set port A pin 1 to Tx

    // iii. set digital or analog mode, and pin directions
    GPIO_PORTA_DEN_R |= 0b0000_0010; // 0x02 enable pin 1 digital mode
    GPIO_PORTA_DIR_R |= 0b0000_0010; // 0x02 set pin 1 (Tx) to output

  // 2. Setup UART device
  // A) Configure UART functionality, frame format and Baud speed

     // Enable UART 0 clock
     SYSCTL_RCGCUART_R |= 0x01;

    //Disable uart0 device while we set it up
    UART0_CTL_R &= 0b1111_1111_1111_1110; // 0xFFFE

    // Set desired UART functionality
    UART0_CTL_R = 0b0000_0001_0000_0000;//Receive disabled, TX enabled

    //Set baud rate (9600 Baud)
    UART0_IBRD_R = 104; //16,000,000 / (16 * 9600) = 104.16666
    UART0_FBRD_R = 11;  // .1666*64+.5 = 11.16666
    UART0_CC_R = 0; //Use system clock as UART clock source

    //set frame format: 8 data bits, no FIFO, 2 stop bit, Odd parity
    UART0_LCRH_R = 0b0110_1010; or 0b0111_1010//0xCA or 0x7A

  // B) Setup UART 0 Interrupts
    No interrupts
```

Name: **Lab Section:**

```
  // 3. NVIC setup
  // A) Configure NVIC to allow UART interrupts
  No interrupts

  // B) Bind UART0 interrupt requests to User's Interrupt Handler
  No interrupts

  //re-enable uart
  UART0_CTL_R = UART0_CTL_R | 0x01;
}
```

ii) Complete function `serial_send` to transmit `my_txt` using hardware device UART0 [3pts]

```
// Send my_text using the hardware device UART0
serial_send(char my_txt)
{
  // Wait while TX FIFO is Full
  while(UART0_FR_R & 0b0010_0000) // 0x20
  {

  }

  UART0_DR_R = my_txt;
}
```

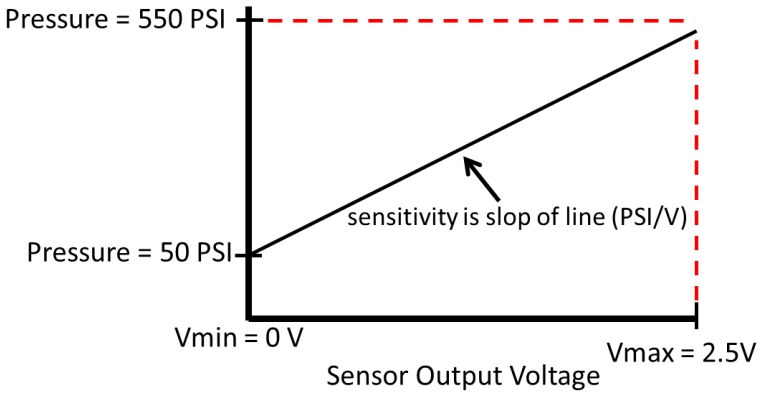
Name: **Lab Section:**

Question 3: ADC Design Principle (5 pts)

Suppose that an TM4C123 is used with a pressure sensor to monitor the pressure exerted on a valve. The pressure sensor measures pressure from 50.0psi (pounds per square inch) to 550.0psi and converts it proportionally (i.e. linearly) to an electrical signal in the voltage range from 0V to 2.5V. Assume the reference voltage (i.e. max voltage) for the TM4C123 ADC is 5V.

a. If the gas pressure is 350.0 psi: [5pts]

i) what is the voltage level at the sensor's output? (1pts)



Solution:

Step 1: Find the sensitivity of the sensor. This is the slope (m) of the above figure

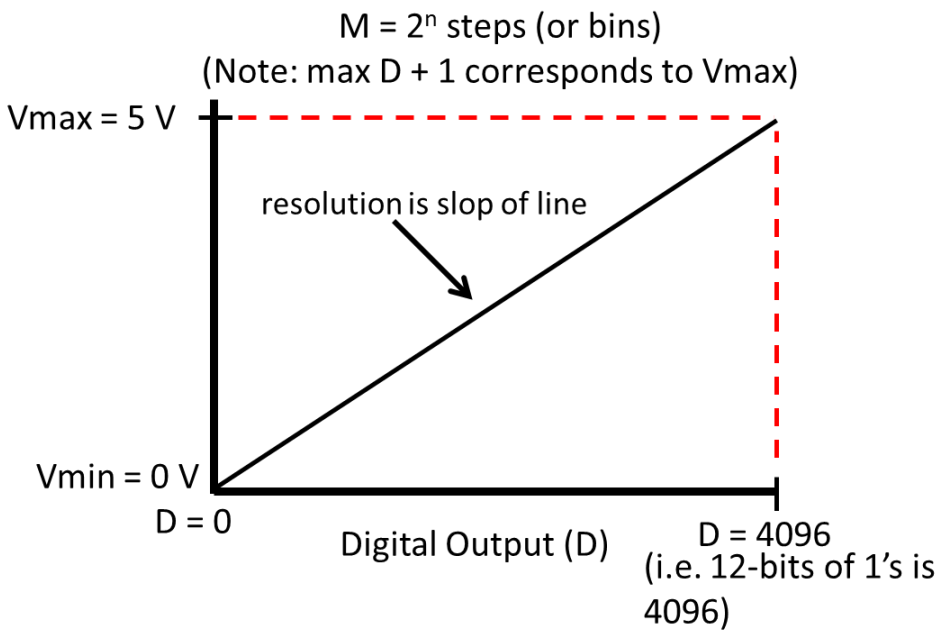
m = RISE/RUN = (550 – 50) / (2.5 – 0) = 500/2.5 = 200 PSI/V

Step 2: Use equation for a line: y = mx + b. Let y = pressure, m = slope of line, x = sensor output voltage, and b = y-intercept.

For this case: m = 200, and b = 50
So. y = 200x + 50, solve for x, x = (y -50)/200 = (350 – 50)/200 = 300/200 = **1.5 V**

Name: **Lab Section:**

ii) What is the digital reading from the TM4C123 ADC? (2 pts)



Background: The above figure can be drawn based on the TM4C123 ADC specifics: i) it is 12-bit, ii) having a Vref of 5.0V, means Vmax is 5.0V, iii) TM4C123 ADC has 2^n steps.

Step 1: Compute the resolution of the ADC. This is just the slope of the above graph

m = RISE/RUN = (5 – 0) / (4096 – 0) = 5/4096 = .0012207 V/bit

Step 2: Compute digital output

From part a.i), we know the input to the ADC is 1.5 V

Use equation for a line: y = mx + b. Let y = input Voltage, m = slope of line, x = digital output, and b = y-intercept = 0.
1.5 = .0012207x.Solve for x: x = 1.5/.0012207 = 1228.8. Since the digital output cannot have a fractional part, the digital output of the ADC is **1228 or 0x4CC or 0b0100_1100_1100 (12-bits)**

Grading Note: If $2^n - 1$ steps is used, but the answer is consistent with this assumption, then do not deduct any points.

iii) If the digital reading was 200, what is the range of possible analog values just read? (2pts)

The lowest voltage that will give a digital reading of 200 is:

y = mx + b = .0012207 * 200 + 0 = .24414 V.

The lowest voltage that will give a digital reading of 201 is:

y = mx + b = .0012207 * 201 + 0 = .24536 V.

So the range of voltages that could give a digital reading of 200 is:

.24414 V < Input Voltage < .24536 V

Question 4: `Volatile` keyword (5 pts)

When developing software for an embedded system, the keyword `volatile` is often used. Read through the articles below and answer the following:

Jones, Nigel. "Introduction to the Volatile Keyword" Embedded Systems Programming, July 2001: http://www.embedded.com/electronics-blogs/beginner-s-corner/4023801/Introduction-to-the-Volatile-Keyword

Wikipedia Article: https://en.wikipedia.org/wiki/Volatile_(computer_programming)

a) Give a summary of the conditions under which the `volatile` keyword should be used within a C based embedded system, **and why**. [3pts]

In general the keyword volatile is used for a variable when its value can change in a manner for which the compiler cannot track. Specific examples:

i) Memory Mapped registers: Since these values can often be modified by hardware other than the CPU, the complier cannot track when the values will change.

ii) Global variables shared between an ISR and primary code: Compliers cannot typically track when values are updated within an ISR with respect to being updated in the primary code.

iii) Global variables within multi-threaded applications: Compilers often have an issue tracking changes of variables shared between "threads".

Name: **Lab Section:**

b) Explain what unwanted behavior could occur if the `volatile` keyword was NOT used for the variable `clock_flag`, for the following code segment. [2pts]

If the `volatile` keyword is not used in this case, then the complier may assume since `clock_flag` is initialized to `0`, and then never modified by the primary code. Then it may "optimize away" the `if` statement and the associated body as it will assume `if(clock_flag == 1)` will always evaluate to FALSE.

```
volatile int clock_flag = 0;//Indicate Timer interrupt has fired

// Timer 1 ISR that will be activated once per second
My_TIMER1_HANDLER()
{
  clock_flag = 1;
}

// Drive robot and manage printing time to the LCD screen
int main()
{
  Timer1_configure();//Configure Timer 1 to fire once per second

  while(1)
  {
    if(clock_flag == 1)
    {
      clock = 0;
      // advance the clock, print new time to LCD screen
    }

    // Code for driving the robot
    ...
    ...
    ...
  }
}
```

Name: **Lab Section:**

Question 5: Polling-based Device Interaction (7 pts)

For this problem, and Question 5, assume variables (more accurately MACROs) called `CAMERA_DATA`, `CAMERA_CONFIG`, `CAMERA_CMD_STAT`, and `CAMERA_INT_EN_CLR` have been defined for you to use for accessing the corresponding Memory Mapped registers of the "CPRE 288 Datasheet Trainer".

a) Complete the function `Camera_Configure()` so that it updates the configuration of the "CPRE 288: Datasheet Trainer" Camera Controller as follows, **and** does not return until the Camera configuration update has completed. [4pts]

- `Color Mode: Color`
- `Resolution: 640x480`
- `Speed: 240 FPS`
- `No interrupts enabled`

```
Camera_Configure()
{

  // Assuming RSV has no effect
  CAMERA_CONFIG = 0b0110_1001;  // or 0x69
  CAMERA_INT_EN_CLR = 0b0011_0000; //or 0x30 (Clear Interrupts)

  // Wait configuration update to complete
  while(CAMERA_CMD_STAT & 0b0000_0001) // or 0x01
  {
  }

}
```

Name: **Lab Section:**

b) Without using interrupts and using "One shot Mode", complete `main()` so that it takes 10 pictures and the most recent image is always stored in the array `image`. Use additional variables if you would like. [3pts]

```
unsigned char image[640][480]; // Store the most recent image captured

main()
{

  Camera_Configure();  // Configure the Camera

  // Snap Mode = 0
  CAMERA_CMD_STAT = CAMERA_CMD_STAT & 0b1111_1001;


  while(1)
  {
    // Collect 10 images
    for(num_images = 0; num_images < 10; num_images++)
    {
      // Take an image
      CAMERA_CMD_STAT = CAMERA_CMD_STAT | 0b0000_0010; //or 0x02

      // Read 640x480 pixels
      for(i=0; i < 640; i++)
      {
        for(j=0; j<480; j++)
        {
          // Wait for a pixel
          while(~CAMERA_CMD_STAT & 0b0000_1000)  // or 0x08
          {
          }
          image[i][j] = CAMERA_DATA; // Store a pixel
        }
      }
    }  // End for image num_images


  }
}
```

Name: **Lab Section:**

Question 6: Interrupt-based Device Interaction (6 pts)

a) Modify the function `Camera_Configure()`, from the previous question, so that it updates the configuration of the "CPRE 288: Datasheet Trainer" Camera Controller as follows, **and** does not return until the Camera configuration update has completed. [2pts]

- `Color Mode: Color`
- `Resolution: 640x480`
- `Speed: 240 FPS`
- `Interrupts: Only enable Data received interrupts`

```
Camera_Configure()
{
  // Assuming RSV has no effect
  CAMERA_CONFIG = 0b0110_1001;  // or 0x69
  CAMERA_INT_EN_CLR = 0b0011_0001; //or 0x31 (Clear/Enable Ints)

  // Wait configuration update to complete
  while(CAMERA_CMD_STAT & 0b0000_0001) // or 0x01
  {
  }
  //Assume CAMERA_HANDLER has been bound to Camera interrupts
}
```

b) Using interrupts and "One shot Mode", complete `CAMERA_HANDLER()`, and `main()` so that it takes 10 pictures and the most recent image is always stored in the array `image`. Use additional variables if you would like. **Remember, keep your ISRs short**. [4pts]

`CAMERA_HANDLER() //ISR that services Camera Controller interrupts` [2pts]

```
{

  //Check if the interrupt type received was Data received
  if(CAMERA_CMD_STAT & 0b0100_0000)
  {
    image[i][j] = CAMERA_DATA; // Store a pixel
    i = (i + 1) % 640; // advance i pixel index
    j = (j + 1) % 480; // advance j pixel index

    // Raise a flag indicating a full image has been read
    if( (i==0) && (j==0) )
    {
      received_full_image = 1;
    }
    // Clear Data Received interrupt
    CAMERA_INT_EN_CLR =CAMERA_INT_EN_CLR | 0b0001_0000;//or 0x10
  }
}
```

Name: **Lab Section:**

```
// Store the most recent image captured.  Assume image may be shared


volatile unsigned char image[640][480]; //between ISR and main. [2pts]

volatile int received_full_image = 0; //Flag indicates image received
int i=0,j=0; // track pixels of an image

main()
{

  int num_images = 0; // count number of images captured

  Camera_Configure();  // Configure the Camera

  CAMERA_CMD_STAT |= 0b0000_0010; //or 0x02, Take an image

  while(1)
  {
    while(num_images < 10)
    {
      // Check if a full image has been received
      if(received_full_image == 1)
      {
        received_full_image = 0; // Reset flag
        num_images++;

        // Take another image
        CAMERA_CMD_STAT = CAMERA_CMD_STAT | 0b0000_0010; //or 0x02
      }
     ... // Do other tasks
    }
  }
}
```

Name: **Lab Section:**

Question 7: Interrupt based UART data processing (15pts)

In this question data received by UART 0 will be processed by an Interrupt Service Routine.

a) Write the `serial_init()` function to initialize UART0 as follows [5pts]:

- **Receive Only**
- **9,600 baud rate**
- **8 data bits**
- **Even parity**
- **2 stop bit**
- **Disable FIFOs**
- **Enable UART Receive interrupts only**

You may initialize unrelated control bits as you wish.

```
// Initialize UART0
void serial_init()
{
  // 1. Setup GPIO
  //A. Configure GPIO module associated with UART 0

    // i. Turn on clock for GPIO module A
    SYSCTL_RCGCGPIO_R = SYSCTL_RCGCGPIO_R | 0b000001; // 0x01

    // ii.Enable Alternate function and set Peripheral functionality
    GPIO_PORTA_AFSEL_R |= 0b0000_00001; // 0x01 UART0 TX and RX
    GPIO_PORTA_PCTL_R |= 0x00000001; //set port A pins0 to Rx

    // iii. set digital or analog mode, and pin directions
    GPIO_PORTA_DEN_R |= 0b0000_0001; //enable pin 0 digital mode
    GPIO_PORTA_DIR_R &= 0b1111_1110; // 0xFE set pin 0 (Rx) to input

  // 2. Setup UART device
  // A) Configure UART functionality, frame format and Baud speed

     // Enable UART 0 clock
     SYSCTL_RCGCUART_R |= 0x01;

    //Disable uart0 device while we set it up
    UART0_CTL_R &= 0b1111_1111_1111_1110; // 0xFFFE

    // Set desired UART functionality
    UART0_CTL_R = 0b0000_0010_0000_0000; //Receive enable, TX disabled
```

Name: **Lab Section:**

```
    //Set baud rate (9600 Baud)
    UART0_IBRD_R = 104; //16,000,000 / (16 * 9600) = 104.16666
    UART0_FBRD_R = 11;  // .1666*64+.5 = 11.16666
    UART0_CC_R = 0; //Use system clock as UART clock source

    //set frame format: 8 data bits, no FIFO, 2 stop bit, Even parity
    UART0_LCRH_R = 0b0110_1110;


  // B) Setup UART 0 Interrupts
    UART0_ICR_R |= 0b0001_0000; // Clear RX interrupt status flag
    UART0_IM_R  |= 0b0001_0000; // Enable RX interrupts

  // 3. NVIC setup
  // A) Configure NVIC to allow UART interrupts (OK if Pri not set)
  NVIC_PRI1_R |= 0x0000_2000;//Set UART0 IRQ Pri to 1. Grp1 bits 15-13
  NVIC_EN0_R |= 0x0000_0020; //Enable UART0 IRQ (ie. IRQ 5): set bit 5

  // B) Bind UART0 interrupt requests to User's Interrupt Handler
     IntRegister(INT_UART0, My_UART0_Handler);
     IntMasterEnable();//Globally allows CPU to service interrupts

  //re-enable uart
  UART0_CTL_R = UART0_CTL_R | 0x01;

  //Binds UART0 interrupt requests to My_UART0_RX_Handler
  IntRegister(INT_UART1, My_UART0_Handler);
  IntMasterEnable();//Globally allows CPU to service interrupts

}
```

Name: **Lab Section:**

b) Write code for `My_UART0_Handler` to implement the Interrupt Service Routine (ISR) that processes the occurrence of a UART0 received data interrupt. In addition, within this ISR turn on an LED connected to GPIO Port B pin 3 when an `'L'` (for Light) is received by UART0 by writing a 1 to the LED, and turn off this LED when an `'O'` (for Off) is received by writing a 0 to the LED. [5pts]

```
// UART0 ISR
void My_UART0_Handler()
{
  // 1) Check if Handler called due to a RX event
  if(UART0_MIS_R & 0b0001_0000)
  {

  // 2) Clear the RX Interrupt Status
  UART0_ICR_R = UART0_ICR_R | 0b0001_0000;

  // 3) Application specific functionality

     // Get byte from UART
     my_char = UART0_DR_R;

     // Check if an 'O' or 'F' ASCII charter was received
       // Light LED
     if(my_char == 'L')
     {
        GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R | 0b0000_1000;
       flag = 1;
     }
       // Turn off LED
     if(my_char == 'O')
     {
        GPIO_PORTB_DATA_R = GPIO_PORTB_DATA_R & 0b1111_0111;
       flag = 2;
     }
  }
}
```

Name: **Lab Section:**

c) Complete main() to print "LED turned ON" and "LED turned OFF" <u>once</u> each time the LED is turned on or off respectively. [5pts]

```
void My_UART0_Handler();
void serial_init(void);

volatile int flag = 0; // Helper variable

int main()
{
  init_portB(); // Assume implemented correctly in Question 2
  serial_init();

  while (1)
  {

    //Print each time the LED is turned ON or OFF
    //Hint: make use the helper variable flag declared above.
    // YOUR CODE HERE
    if(flag == 1)
    {
      flag = 0; // Reset flag
      lprintf("LED turned ON");
    }

    if(flag == 2)
    {
      flag = 0; // Reset flag
      lprintf("LED turned OFF");
    }

  }

  return 0;
}
```

<u>Collaboration Documentation</u>

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

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

1)

2)