Name: Sean Gordon **Lab Section:** F

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

Notes:

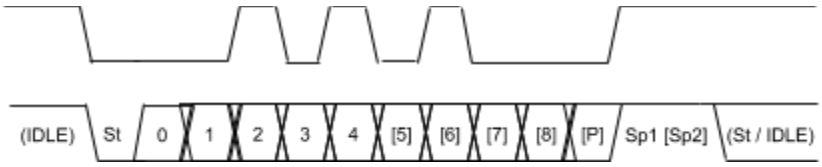
- Homework must be typed and submitted as a PDF or Word Document (i.e. .doc or .docx) only.
- If collaborating with others, you must document who you collaborate with, and specify in what way you collaborated (see last page of homework assignment), review the homework policy section of the syllabus: 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]

'T' = 0b01010100 Period = 1/9600 = 104.17us



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

9600 * 8/12 = 9600 * .6667 = 6400 bits/sec

Name: Sean Gordon **Lab Section:** F

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()
{
     SYSCTL_RCGCGPIO_R |= 0b000010;

     GPIO_PORTB_AFSEL_R |= BIT3;
     GPIO_PORTB_PCTL_R |= 0b00001000;

     GPIO_PORTB_DEN_R |= 0b0000_1000;
     GPIO_PORTB_DIR_R |= 0b0000_1000;
}

// 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)
{

     GPIO_PORTB_DATA_R &= 0;

     int i;
     for(i = 0; i < 8; i++){
          if(my_txt >> i == 1){
               GPIO_PORTB_DATA_R |= 0b0000_1000;
          }
          else{
               GPIO_PORTB_DATA_R &= 0;
          }
     }
}
```

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()
{
     SYSCTL_RCGCGPIO_R |= 0b000001;     //Provide clock to port
A

     GPIO_PORTA_AFSEL_R |= 0b0000_0010;      //UART0 TX
     GPIO_PORTA_PCTL_R |= 0x00000010;   //Port A pin 1 Tx

     GPIO_PORTA_DEN_R |= 0b0000_0010;   //Port A pin 1 digital
     GPIO_PORTA_DIR_R |= 0b0000_0010;   //Set pin 1 to output

     //UART
     UART0_CTL_R &= ~UART_CTL_UARTEN;   //Temp. disable UART0
     UART0_CTL_R &= 0xFFFF_3440;        //Preserve RES bits
     UART0_CTL_R |= 0x0000_0100;        //Enable only TX
                                        //(UART0 still disabled)
     //Set baud rate
     // 16,000,000 / (16*9600) = 104.16666 = 104  (IBRD)
     //    =>>   .1666*64 + .5 = 11.16666  = 11   (FBRD)
     UART0_IBRD_R = 104;
     UART0_FBRD_R = 11;

     UART0_CC_R &= 0xFFFF_FFF0;   //Preserve RES bits and
                                  //Configure to use system
                            clock

     // 8 data-bits, odd parity, 2 stop bits, disable FIFOs
     UART0_LCRH_R &= 0xFFFF_FF00; //Clear first 8 bits
     UART0_LCRH_R |= 0b0110_1010; //Final Configurations
     UART0_ICR_R |= 0b0001_0000;  //Clear RX interrupt status

     UART0_IM_R |= 0b0001_0000;   //Enable receive interrupts

     UART0_CTL_R |= 0b0000_0001;  //Re-enable UART0
}
```

Name: Sean Gordon **Lab Section:** F

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)
{

     while(UART0_FR_R & 0b0010_0000){

     }
     UART0_DR_R = my_txt;
}
```

Name: Sean Gordon **Lab Section:** F

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)

300/500 * 2.5 = 1.5v

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

1.5 / (5/4096) = 1228

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

(200 / 4096) * 5 + (5 / 4096)
.24414 + 1.2207 * 10^-3

.24414 -> .24536

Name: Sean Gordon **Lab Section:** F

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]

The volatile keyword should be used when a variable could be changed by a different thread, ensuring the value will never be cached locally and all reads will go directly to the 'main' memory. This is to help keep the value of the variable consistent, and ensure the program always uses the same instance of the variable.

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]

As compilers are designed to optimize higher level code before converting to machine code, the compiler may find that the programmer is waiting for clock_flag to be 0, but never changing it 'in the code', so it will just remove that section, or convert it to something else.

```
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: Sean Gordon **Lab Section:** F

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()
{
      CAMERA_CONFIG &= 0b1000_0000;    //Preserve rsv bit
      CAMERA_CONFIG |= 0b0110_1011;    //Config camera

      CAMERA_INT_EN_CLR &= 0b11001100;      //Preserve rsv bits
      CAMERA_INT_EN_CLR |= 0b0011_0000;//Clear interrupts

      while(CAMERA_CMD_STAT & 0b0000_0001){}     //Wait for config to
finish
}
```

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
      int count = 0;        // Number of pictures taken
      int x, y;             // Used to grab pixels from CAMERA_DATA

      for(count = 0; count < 10; count++){       // Grabs 10 images
            CAMERA_CMD_STAT &= 0b1111_1001;      // Clear snap bit
            CAMERA_CMD_STAT |= 0b0000_0010;      // Take picture

            for(x = 0; x < 640; x++){
                  for(y = 0; y < 480; y++){
                        // When pixel is ready
                        while(~CAMERA_CMD_STAT & 0b0000_1000){}
                        image[x][y] = CAMERA_DATA; //Save pixel
                  }
            }
```

```
	}
}
```

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()
{
	CAMERA_CONFIG &= 0b1000_0000;	//Preserve rsv bit
	CAMERA_CONFIG |= 0b0110_1011;	//Config camera

	CAMERA_INT_EN_CLR &= 0b11001100;	//Preserve rsv bits
	CAMERA_INT_EN_CLR |= 0b0011_0001;//Clear interrupts & enable
data
                                        receive interrupt
	while(CAMERA_CMD_STAT & 0b0000_0001){}	//Wait for config to
finish

//Assume CAMERA_HANDLER has been bound to Camera Controller 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]
{
	if(CAMERA_CMD_STAT & 0b0100_0000) {
		int i, j;
		for(i = 0; i < 640; i++){
			for(j = 0; j < 480; j++){
				image[i][j] = CAMERA_DATA;
			}
		}
		CAMERA_INT_EN_CLR =CAMERA_INT_EN_CLR | 0b0001_0000;
	}
}
```

Name: Sean Gordon **Lab Section:** F

```
// Store the most recent image captured.  Assume image may be shared
volatile unsigned char image[640][480]; //between ISR and main.
[2pts]
main()
{
      int imgCount = 0;
      Camera_Configure();  // Configure the Camera

      while(1)
      {
            if(imgCount < 10){
                  CAMERA_CMD_STAT |= 0b0000_0010; //Enable snap bit
                  imgCount ++;
            }else
                  break;     //End while loop
      }
}
```

Name: Sean Gordon **Lab Section:** F

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()
{
     SYSCTL_RCGCGPIO_R |= 0b000001;     //Provide clock to port
     A

     GPIO_PORTA_AFSEL_R |= 0b0000_0011;      //UART0 TX and RX
     GPIO_PORTA_PCTL_R |= 0x00000011;   //Port A pins 0,1 Rx&Tx

     GPIO_PORTA_DEN_R |= 0b0000_0011;   //Port A pins 0,1
     digital

     GPIO_PORTA_DIR_R &= 0xFFFF_FFFE;   //Set pin 0 to input
     GPIO_PORTA_DIR_R |= 0x0000_0002;   //Set pin 1 to output


     UART0_CTL_R &= ~UART_CTL_UARTEN;   //Temp. disable UART0
     UART0_CTL_R &= 0xFFFF_3440;        //Preserve RES bits
     UART0_CTL_R |= 0x0000_0200;        //Enable only receive
                                        //(UART0 still disabled)



     //Set baud rate
     // 16,000,000 / (16*9600) = 104.16666 = 104  (IBRD)
     //    =>>   .1666*64 + .5 = 11.16666  = 11   (FBRD)
     UART0_IBRD_R = 104;
     UART0_FBRD_R = 11;
```

```
    UART0_CC_R &= 0xFFFF_FFF0;   //Preserve RES bits and
                                 //Configure to use system
                            clock

    // 8 data-bits, even parity, 2 stop bits, disable FIFOs
    UART0_LCRH_R &= 0xFFFF_FF00; //Clear first 8 bits
    UART0_LCRH_R |= 0b0110_1110; //Final Configurations
    UART0_ICR_R |= 0b0001_0000;  //Clear RX interrupt status

    UART0_IM_R |= 0b0001_0000;   //Enable receive interrupts

    UART0_CTL_R |= 0b0000_0001;  //Re-enable UART0

    //Setup NVIC
    NVIC_PRI1_R |= 0x0000_2000;
    NVIC_EN0_R |= 0x0000_0020;

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

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()
{
     if(UART0_MIS_R & 0b0001_0000){
          // Clear receive interrupts
          UART0_ICR_R = UART0_ICR_R | 0b0001_0000;

          my_char = UART0_DR_R;

          switch(my_char){
               case 'L':
                    GPIO_PORTB_DATA_R |= 0b0000_1000;
                    break;
               case 'O':
                    GPIO_PORTB_DATA_R &= 0b1111_0111;
                    break;
          }
     }
}
```

Name: Sean Gordon **Lab Section:** F

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 && (GPIO_PORTB_DATA_R & 0b0000_1000)){
          flag = 1;
          printf("LED turned ON");
     }
     else if(flag != 2 && !(GPIO_PORTB_DATA_R & 0b0000_1000)){
          flag = 2;
          printf("LED turned OFF");
     }
  }
  return 0;
}
```

Name: Sean Gordon **Lab Section:** F

Collaboration Documentation

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

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

1)

2)