EEL 4742C: Embedded Systems
Homework 4

QUESTION 1. **(10 points)**

Part a) Write a piece of code that configures the backchannel UART of the basic LaunchPad (G2553). You only need to divert the pins to UART functionality. The documents of the basic LaunchPad can be found on the main page of Webcourses.

Part b) What is the oversampling scheme in UART?

Part c) Is it possible to start with SMCLK = 1 MHz and configure 9600 baud with oversampling. Show how you obtained your answer.

Part d) Is it possible to start with ACLK = 32 KHz and configure 9600 baud with oversampling. Show how you obtained your answer.

Part e) What are the divider and modulator of the UART clock configuration? Where are these values found?

QUESTION 2. **(10 points)**

Part a) What is the most popular UART configuration?

Part b) Draw the UART signal that corresponds to transmitting the data byte (0101 0011 binary) based on the most popular configuration.

Part c) Configure the eUSCI module for the following settings. Write the code. You don't have to divert the pins.

Source clock:	SMCLK 1 MHz calibrated (calibrated: 1.000000 MHz, raw: 1.048,576 MHz)
Baud rate:	38,400
Data size:	7-bit
First bit:	MSB
Parity:	Even
Stop-bit:	2-bit
Flow control:	none

Part d) What happens if the microcontroller's and the terminal's (at PC) UART configurations don't match?

QUESTION 3. **(10 points)**

Part a) Explain the bus topology that is used by I2C.

Part b) What is the configuration of the I2C bus lines? What value do they read by default?

Part c) What are the Start and Stop signals? By design, what requirement should they adhere to?

Part d) Is it possible to have multiple I2C masters? If yes, what is required to make this work?

Part e) Draw the I2C transmission signals for the bits 0101. Draw the start and stop signals and the bits in between. Show how the lines transition between low and high.

Part f) Repeat the previous question for the bits 1010.

Part g) Discuss one approach on how an I2C address is configured for a sensor.

QUESTION 4. **(10 points)**

Part a) Show the I2C transmission when the master reads from I2C address 0x22. The data read is 0x1234 (MSB sent first).

Part b) Show the I2C transmission where the master writes 0x5678 (MSB sent first) to I2C address 0x33.

Part c) Show the I2C transmission where the master reads from register 0xCD on I2C address 0xEF. The data read is 0x6789 (MSB sent first).

Part d) Show the I2C transmission where the master writes to register 0x12 on I2C address 0x34. The data written is 0xABCD (MSB sent first).

QUESTION 5. **(10 points)**

Part a) Draw the configuration where an SPI master is connected to two devices that are individually selected.

Part b) Draw the configuration where an SPI master is connected to two devices in a daisy chain.

Part c) What is the 3-pin SPI configuration? What’s the difference between 3-pin SPI and 3-wire serial?

QUESTION 6. **(10 points)**

The analog-to-digital converter (ADC) module uses a clock signal that varies in the range [10 -20] Hz due to the operating conditions and error. The sample-and-hold time of the signal we're converting is three seconds. The ADC allows selecting a number of cycles from the list below.

Number of cycles: 10, 40, 80, 110, 150, 200, 250, 300

Part a) Show the analysis for choosing the suitable number of cycles.

Part b) The conversion takes 20 cycles. What is the total operation time (that includes the sample-and-hold time and the conversion)?

Note: the numeric values in this question are chosen to simplify the computations; an ADC usually uses a much higher clock frequency and the sample-and-hold time is usually in the micro-seconds range.

QUESTION 7. **(10 points)**

An SAR ADC has a 3-bit resolution (result). The reference voltages are Vr-=0 and Vr+=Vr. The signal we're converting is: Vin = 3.5/8 Vr+. Show all the voltage comparisons that are done by the ADC and show the conversion's binary result.

QUESTION 8. **(10 points)**

Part a) Describe the graphics software stack in embedded systems.

Part b) What services does the display driver provide? To what party does it provide these services? Comment on the compatibility of a driver with multiple displays.

Part c) What services does the graphics library provide? To what party are these services provided? Comment on the compatibility of a graphics library with multiple displays.

Part d) What is a graphics context?

Part e) What is the clipping region of a graphics context?

QUESTION 9. **(10 points)**

Part a) There are multiple levels of programming microcontrollers that are shown in the list below. Which level did we use in this course?

- Assembly level
- Register level
- Library level
- Operating system level

Part b) The Arduino line of microcontrollers is very popular thanks to the programming library known as the Wiring API. This API provides simple functions that can be used by the programmer. These functions are implemented for all the Arduino microcontrollers. Therefore, the user can use the same (portable) code for any Arduino device chosen. The Wiring API has been implemented for MSP430 microcontrollers in the Energia IDE.

To give you an idea of how an API is implemented, let's write a function that's similar in style to the Wiring API. The function finds the first available channel of Timer_A. A channel is considered available if its CCIE bit is disabled. This is the header of the function. It returns the first available channel out of Channels 0, 1, 2, in this order. If all the channels are used, the function returns -1. Write the code.

```
int get_available_timer_channel();
```

QUESTION 10. **(10 points)**

ARM 32-bit microcontrollers have a 32-bit CPU and a 32-bit memory address. The 32-bit address can reference 2^32 = 4 GB of memory, which is far more memory than a microcontroller usually has. Therefore, there is an abundance of unused addresses, which has inspired the bit-banding technique.

Modifying a single bit in the memory takes three assembly instructions, e.g. (load-OR-store to set a bit). There is interest in making this type of operation faster since it's used frequently. The bit-banding technique takes unused addresses and maps them to bits of a variable. As shown below, the variable Data is at address 100. Accessing address 100 references the whole 8 bits in Data. The bit-banding technique takes the non-used addresses in the 2000 range and maps them to individual bits of Data. That is, address 2000 maps to bit #0 of Data, address 2001 maps to bit #1 of Data, etc. With this technique, the load-OR-store operation to modify a bit can be reduced to a store operation. Writing 1 to address 2000 results in writing 1 to bit #0 of Data, leaving the other bits unchanged.

```
         Data:  __  __  __  __  __  __  __  __
Address: 100    #7  #6  #5  #4  #3  #2  #1  #0

Address:      2007          ...         2000
```

Part a) Write a piece of code that reads bit #4 of Data (into the variable temp) using the bit-banding technique.
Part b) Write a piece of code that sets bit #5 of Data using the bit-banding technique.
Part c) Write a piece of code that counts how many bits in Data are equal to 1, using the bit-banding technique.

Practice Questions

Do not submit; these questions were done in the class or lab.

PRACTICE 1.

Write a code that turns on the LED (P1.0, active high) for 3 seconds when the button (P1.3 active low) is pushed. The interval is non-renewing, i.e., the button is ignored if it's pushed again during the three second interval. Configure the button via interrupt. Time the 3-second interval using Timer_A with interrupt (preferred mode: continuous mode with the timer running all the time). Use ACLK at 32 KHz.

PRACTICE 2.

Write a code that debounces the push button (P1.3, active low). When the first button edge is detected, wait for the maximum bounce duration (15 ms) and, if the button is still pushed, take the action, which is toggling the LED (P1.0, active high). Configure the button via interrupt. Time the duration using Timer_A with interrupt (preferred mode: continuous with the timer running all the time). Use ACLK at 32 KHz.

PRACTICE 3.

The programming model of a communication module is based on the variables below. An 8-bit data is transmitted or received at a time.

TXFLAG bit	1: ready to transmit; 0: transmission in progress
RXFLAG bit	1: data received; 0: no data received
FLAGS register	Contains the flags
TXBUFFER register	8-bit transmit data; writing to this register pulls TXFLAG low and Tx begins
RXBUFFER register	8-bit received data; reading this register clears RXFLAG

Part a) Write a function that transmits a byte.

Part b) Write a non-block function that receives a byte.
