ECE6483 EXAM 1- Spring 2023

NYU Tandon School of Engineering

Name: ______________________________________

ID: ______________________________________

Instructions:

This is a take home exam that is open book and open notes. You are not permitted to plagiarize any book or website and you are not permitted to post questions or solutions to any media site. You are not permitted to collaborate. Any indication of plagiarism or collaboration in any form will result in a zero score on this test.

This exam requires you to submit code. The code should be inline along with any other responses in a single PDF. Only 1 PDF file should be submitted with the answers to all questions (including necessary code, diagrams etc.)

1. Assembly Code and C

There are several sensor chips capable of measuring complex impedance. This is often achieved by providing 2 electrodes cable of sourcing (or sinking) current while measuring electrical potential of the electrode pair. For this exercise, assume that the code already exists to interface the sensor and the driver gives you access to the measurements.
The following function is available for you to use:

```
void GetData(short* ReIm, int nCount)

// ReIm is an array of shorts of size 2 *  nCount.  nCount is the number of measurements.
// Data returned in ReIm[] is stored as [Re1, Im1, Re2, Im2,…RenCount, ImnCount]
//Re1=Real part of first measurement, Im1=Imaginary part of first measurement
```

Write a main.cpp file that:

a. Defines the necessary array to hold 10 measurements made by the GetData function.
b. Defines an array of size nCount called Mags[] used to store the magnitude of each measurement
c. Imports an assembly function GetMags, as defined in d.
d. Calls an assembly function called GetMags that you pass in 3 parameters. The first is a pointer to the ReIm array, the second is a pointer to Mags array, and the third is the nCount. The assembly function calculates the magnitude of each of the nCount impedance measurements. No return value required.
e. Prints the magnitudes of each measurement to the terminal window.

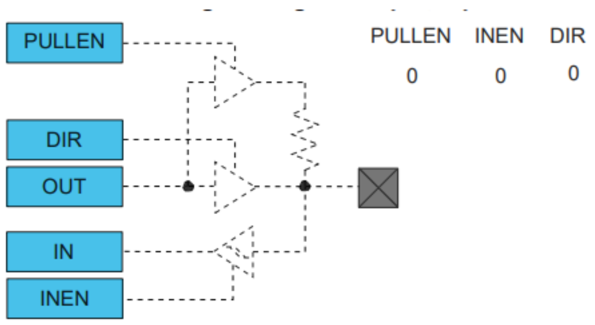
Write an assembly file implementing the GetMags function including comments on each line. Be sure to include any requirements necessary to interact with your C program.

2. GPIO, ATSAMD21 (ARM Cortex M0+)

The first section of the "Peripherals" lecture focusses on GPIO specifically for the M0+ microcontroller. Suppose you would like to design your own dev board for this microcontroller with access to all the GPIO pins on the "E" package. See the datasheet attached, section 5.5. You will notice that this chip only gives access to PORT A, pins [0-11,14-19,22-25,27,28,30,31].
Review carefully section 23 of the datasheet.

The goal of this assignment is to avoid using any framework (like EMBED) to perform basic GPIO functions. That means we can not rely on DigitalIn, DigitalOut, digitalRead etc.

Write 2 functions in C that ONLY manipulate the following parameters (meaning they read and write these i/o mapped registers) for the selected GPIO pin and performs the specified functions as described below:



.

a. Function 1:

   char digitalRead(int PinOnPortA, int pullEnable), where PinOnPortA is the pin on PORT A that you wish to read. The parameter pullEnable is a 0 = no pull, 1 = pull up, 2=pull down). The return value is a 1 or 0.

b. Function 2:

   void digitalWrite(int PinOnPortA, char val), where PinOnPortA is the pin on PORT A that you wish to write "val", which is a 1 or a 0.

3. ADC and Timers

For this exercise, you can use MBED implementations for ADCs and Timers like we did in our recitations. That is, you may use the class AnalogIn, Timer, Ticker etc. The objective here is to design a carbon monoxide detector that outputs to the terminal a status of “OK” or “BAD” every 1 second based on the MQ-7 CO detector, which outputs an analog signal in the range of 0 to 5V. Keep in mind your ADC has a V_{max} = 3.3V

Based on experimentation, you conclude that a reading of 2.5V or higher is the criteria for declaring “BAD”, while < 2.5V is considered “OK”.

a. Assuming the MQ-7 has 3 pins, Vin, GND, and V_{out}, sketch a schematic of how you will connect this to your board. Clearly show which analog channel you are using and show any circuitry if required.
b. Write code required to setup the analog channel defined in a.
c. Write the timing code required to read the sensor every 1 second.
d. Write the math logic that determines “OK” or “BAD” and outputs it to the terminal.

Good luck!