CMPE1250
Winter 2025
v1.0

ICA 4
GPIO and AD2 Measurements

1 Introduction

Your lectures and notes should have prepared you to initialize port AD for use with the buttons and LEDs by now. You should also be able to turn specific LEDs on and off by manipulating the `PT1AD1` register with bitwise operations.

You will be building on those skills in this assignment and incorporating some skills from your hardware classes. You will be creating software delays to toggle the LEDs at specific frequencies. To measure the LED frequency, you will need to use the oscilloscope function of the Analog Discovery. One of the goals with this assignment is for you to get comfortable using an oscilloscope to characterize and troubleshoot your code. It may seem like an unusual practice now, but an oscilloscope is a powerful diagnostic tool in embedded programming. We will also briefly explore the difference between blocking and non-blocking delays.

2 Outcomes

By the end of this assignment, you should be able to:

- Initialize the Port AD for use with the LEDs
- Manipulate single bits on `PT1AD1` with bitwise operations
- Create simple software delays
- Explain why you might make an integer variable long or unsigned
- Measure signals on the microboard with an Analog Discovery or oscilloscope
- Explain the difference between blocking and non-blocking delays

3 Assignment

The parts of the assignment are below. Please be aware that there are questions at the end of each part. To get full marks for each section, you must implement the required code **and** answer the questions. Make sure you have made a git commit indicating the completion of each section. This will allow your instructor to mark each part without having it affected by code from later sections.

3.1 Toggle an LED

For the first part of this assignment, you need to initialize the LEDs and write a main loop that toggles only the green LED on every loop iteration. To get full marks your code must be commented and be in the appropriate sections of the C template file; initializations need to the in the "One Time Initializations" section.

Don't forget to update the title block

3.1.1 Part 1 Questions

How does the LED appear while your code is running?
How can you explain this observation?

3.2 Frequency Measurement

Use an oscilloscope or your Analog Discovery to measure the waveform frequency on `Pin 78`. This is `PAD13`, which is the microcontroller pin connected to the LED. You can verify this with the board schematics available in your *\Datasheets* directory. Leave your oscilloscope or Analog Discovery connected to the board. You will be taking more frequency measurements.

3.2.1 Part 2 Questions

At what frequency is the LED toggling?
Include a screenshot of your measurement in your submission.

3.3 Toggle with a blocking delay

Create a `while()` loop that will do nothing 500,000 times. You will need to create a new variable at the start of your code for this to work. You cannot create new variables part way through a scope in C. Put your `while()` loop below the line that toggles your LED. Don't forget to reset your counter after it reaches the your setpoint.

This kind of delay is called a *blocking delay*. While the code is trapped in the loop, the microcontroller is not able to execute any other commands. This kind of delay *blocks* the execution of any other code. Our microcontroller is too busy counting to 500,000 to do anything else.

3.3.1 Part 3 Questions

Modify the variable type of your counter and fill out Table 1.
If the frequency changes between the data types, explain why it happens.

Data Type	Period [ms]	Frequency
int		
unsigned int		
unsigned long		

Table 1: Execution Time vs Data Type (500,000 Iterations)

3.4 Decrease the delay time

Reduce the delay count to 1,000 iterations; your LED should now be blinking much more quickly. Use your oscilloscope or AD2 to measure the frequency at which your LED is blinking.

3.4.1 Part 4 Questions

Modify the variable type of your counter and fill out Table 1.
If the frequency changes between the data types, explain why it happens.

Data Type	Period [ms]	Frequency
int		
unsigned int		
unsigned long		

Table 2: Execution Time vs Data Type (1,000 Iterations)

3.5 Set the frequency to 500Hz

Now that you have measured the period at which the LED is toggling, you should be able to calculate how how long each iteration of the loop takes. Remember that your LED toggles twice per period. It needs to turn off and back on to be a complete waveform cycle. Calculate how many delay iterations would be needed to create a 1ms delay. You should now have a waveform with a frequency of 500Hz on `Pin 78`.

3.5.1 Part 5 Questions

Include comments in your code showing how you calculated the correct number of loop iterations to create a 1ms delay. Include a screenshot of your 500Hz waveform.

3.6 Blink a second LED

To demonstrate the difference between a *blocking delay* and a *non-blocking delay*, you will now modify your code to blink a second LED. Modify your code to use an `if()` statement rather than a `while()` loop. You should have something like `if (counter == setpoint)`. Add a second `if()` statement that will toggle the red LED when a counter variable reaches $1,000$; you may need to add another counter variable.

This will change the timing of your green LED toggling, but we will see how to deal with that when we look at the *Real Time Interrupt* module. For now, just consider if you would have been able to make two LEDs blink at independent frequencies with `while()` loops.

3.6.1 Part 6 Questions

Was the microcontroller able to execute other commands while the counters were incrementing to their setpoints in this version of your code?
Describe the difference between a *blocking delay* and a *non-blocking delay*.