ELEC 335 - Lab #3

Objective

The objective of this lab is to

- practice C language, and write basic programs,
- understand the effects of optimization on code size and speed,
- work with interrupts and utilize external interrupts,
- work with seven segment displays.

Setup

Install STM32CubeIDE and implement problems on the breadboard. For that, use the **`blinky`** project from the g0 project repo (`https://github.com/fcayci/stm32g0`) and follow `https://youtu.be/UYk_NAnDJlw`. Do not use **standalone** for any problem / lab. That is just for demonstration purposes. You can use one of the undergraduate laboratories for using the scope and measuring the necessary voltages if needed.

- **For each coding problem**, you are required to create a flowchart that shows software operation and a block/connection diagram that shows how things are connected and what modules are in use. You can use online tools to do that. One example is lucidchart (`https://lucid.app`). Alternatively, you can use a software you are familiar with.
- **All codes should be written in C.**
- If a specific pin number is not stated, you are free to pick any appropriate pin to connect your components. Make sure to include this in your block/connection diagram.

Submission

You should submit the following items organized in a folder:

- **Lab report** - Written in English. PDF file. Should include
  - a cover page
  - for each problem
    - a flow chart
    - a block/connection diagram
    - pictures/photos if any/requested
    - code listing
    - at least one paragraph explanation/comment about that problem, code listing
  - final lab conclusion about what you've learned.
- **Source files** - should have proper comments and pin connections if there are any external components.
- **Elf files** - generated elf file with debug information.

Problems

Problem 1 [20 pts]. Write a program to blink an external LED at roughly 1 second intervals.

- Capture scope output.
- Is there any difference between the code size when you implemented this in assembly? What do you think accounts for that?
- Is the delay number different then the assembly implementation? Explain.
- Change the optimization to -O2, and try again, is there any change? If so, explain what happened. Is there any difference between the code sizes?

Problem 2 [20 pts]. Using **a state machine** blink the external LED at different intervals. Assign each speed to a mode, and attach a button to cycle through the modes. (Each button press will cycle through these modes.) You should do polling for the button press.

Modes:

- Mode 0 → No toggling, LED is off
- Mode 1 → LED is toggling at roughly 2 second intervals
- Mode 2 → LED is toggling at roughly 1 second intervals
- Mode 3 → LED is toggling at roughly .5 second intervals
- Mode 4 → LED is toggling at roughly .1 second intervals
- Mode 5 → No toggling, LED is on

Warnings:

- Define an enum for your states, and cycle through them in each button press.
- Depending on the state, define the delay.
- Your flowchart should be detailed enough and should overlap with your implementation.
- Make sure your code works when the optimization is defined as `-O2`.

Questions:

- What is the difference in code size when the optimization is enabled / disabled? How about the actual blinking speed of the LED? Is there any change? If so, what would be the difference?
- Compare the state machine approach to a regular super loop approach. What are the advantages / disadvantages? You need to give a pseudo-code for this comparison.

Problem 3 [20 pts]. Implement the same state machine in Problem 2, but this time use external interrupts to detect button press, and use the handler to change the state.

Questions:

- What is the difference between Problem 2 and Problem 3 in terms of scalability, clarity and responsiveness? Compare the oscilloscope outputs for both of them and explain.

Problem 4 [40 pts]. Connect the keypad to the microcontroller, and using external interrupts detect button presses. Use an SSD to display the pressed button. Your main loop should only be used to display the SSDs.

Requirements:

- You should create a function that will display the given number on SSDs.
- Each time button is pressed, the number should slide in from the right of the SSD.
- When there are 4 digits already, the next number should erase the oldest number.
    - For example, when you pressed the numbers 4, 2, 5, 7, 6 in that order, the SSD should display 4 → 42 → 425 → 4257 → 2576.

- There should be almost no difference in brightness between SSDs.

Questions:

- Try to figure out the processing delay of the interrupt looking at the scope output.
- Is there a brightness difference between the numbers Seven Segments? How did you solve it? Show the scope output of a single segment when you light up the same segment on all Seven Segments. What happens if you decrease the delay / increase the delay?