1. Introduction

The purpose of this lab is to explore the design of firmware for to drive a more complicated component with a parallel interface. Structs should be used to control access to the required GPIO port. A driver will be constructed for the LCD Character to display. To demonstrate the drive a “Maze” game will be coded using the three buttons from Lab2.

2. Prerequisites

- StmCubeIDE is installed on your machine.
- Your MSOE Development Board with the following components assembled:
  - STM32 Microcontroller
  - LED Bar Array (with drivers)
  - Resistor Bars, Rotary Encoder, and Analog Control Stick (with drivers)
  - LCD Display

3. Activities

3.1 Create Lab2Match Project

- Copy your working Lab2Match Project within CubeIDE in the ProjectExplorer tab by copy-and-pasting.
- Name your project Lab3Maze
- Make any improvements to existing drivers based on instructor feedback.

3.2 Add starter APIs

- Download the *lcd_starter.h* and *lcd_starter.c* from the canvas assignment.
- Place them in the “inc” and “src” directories respectively.
- Rename them *lcd.h* and *lcd.c*.

3.3 Create your LCD APIs

3.3.1 gpio.h

- Create a GPIO structure and constants

3.3.2 lcd.h

- Find constant values in class with your instructor from the schematic and datasheets.

3.3.3 lcd.c

- Create a main.c to exercise your API.
- Implement the interface one method at time testing as you go.
- Try to get a blinking cursor by the end of the lab period.

3.4 Maze

- The goal is to make it from the top left of the screen to the bottom right of the screen while avoiding obstacles on the way.
    - You may use any character as your player. (I like 0xEF)
    - You may use any character or combination of characters as obstacles.
    - You may use any character as the objective.
- Write a driver program that will use your API to implement the “maze” game with the following mechanics.
    - Always start in the top left of the screen.
    - The goal should always be in the bottom right of the screen.
    - There should be several obstacles in the way.
    - There must be a viable path through to the objective.
    - Start with 5 lives shown on the LED Bar.
- Movement:
    - Use the Rotary Encoder to move the player right.
    - Use the Controller Button to move the player left.
    - Use the User Button to toggle between rows.
    - Assume you can’t move beyond the right and left edge of the screen.
- Obstacles
    - If you land on an obstacle change show a collision character in that location
    - Drop a life on the LED Bar and reset back to the start position.
        - If there are no lives left print a “Loss” message on line1 and “Rst 2 Restart” message on line2.
- Getting to the goal
    - Quickly fill the screen with an animated celebration
    - Print a “Win” message on line1 and “Rst 2 Restart” message on line2.
- Hints:
    - It is helpful to keep track of your position in the program.
    - Remember the cursor will automatically move to the next position after writing a character. (This may now work in our favor during game play.)
    - You may have to remember you last position to delete the previous printing of the character before drawing in the new position.

4. Deliverables

- All code should include a header block with:
    - Your name
    - Course number and section
    - Assignment name
    - File name
    - List of any dependencies.
    - Description
- Print out you document code in the following order:
    - main.c (driver/game mechanics)

    - gpio.h
    - gpio.c
    - lcd.h
    - lcd.c
- Print in Light mode with Line numbers, filenames, and time and date using CubeIDE or Notepad++
- Staple together packet in the top left, in order, with rubric cover sheet.
- Phase one target – Blinking Cursor (Week5 Lab)
- Phase two target - full application with switch control. (Week6 Lab)
- Final packet and demonstration due at the beginning of Week7 Lab.