Lancaster University

SCC.369: Lab Exercise 1

Working with GPIO

Aim of the Exercise
The aim of this exercise is to familiarize you with a native C development environment, the Arm microcontroller architecture, and some basic input and output (often abbreviated IO or I/O). This exercise will give you an understanding of the fundamental characteristics of embedded systems.

Submission: **SCC.369 moodle submission, 16:00 Friday Week 2.**
Weighting: 25%

The Task

This task is all about controlling digital inputs and outputs. We'll be using 8 light-emitting diodes (LEDs) and simple push buttons to display and manipulate visible bit patterns. A classical first task, where you can follow in the footsteps of giants in the field. Like Steve. 😊

This exercise is split into a number of subtasks, listed below. Implement each one of them **from first principles using pure C.** This means that you are NOT permitted to use any library code or function calls that you have not written yourself. You can only use C keywords and operators.

Create a file called **Task1.cpp**, and write each subtask as a **separate function** in that file that performs the specified task when called. **Do not include a main() function in your submission** (although you will of course need to create one for your own testing). We will be testing these via an automated test harness, so be sure to use the specified function prototype for each task.

For all these exercises, be sure to style and comment your code properly as you go along. We'll also be marking on how elegant, clean and efficient your code is, so do your best to make it well written.

Remember to have fun, and use the labs to ask about anything you don't understand!

Hardware

Using the hardware kit provided, wire the eight LEDs and one button, to the edge connector pins specified below. Note that one resistor is necessary for every input and output; LEDs require a ~470R resistor in series and a button requires a ~1k-100k pull-up resistor.

Edge Connector Pin	Role	Description
P0	Red LED	Bit 7 (Most Significant)
P1	Red LED	Bit 6
P2	Orange LED	Bit 5
P8	Orange LED	Bit 4
P9	Yellow LED	Bit 3
P11	Yellow LED	Bit 2
P13	Green LED	Bit 1
P14	Green LED	Bit 0 (Least Significant)
P15	Button with 10k pull-up to 3.3V	User Input

Subtask Specification

Task 1: Turn all the LEDs on
Function prototype: **void turnOn();**
The title says it all – write a function that turns on all the LEDs...

Task 2: Different bit patterns
Function prototype: **void setLEDs(uint8_t value);**
Write a function that sets the bit pattern of the LEDs to match the unsigned 8-bit value provided as a parameter. A '1' in a bit position should turn the LED on, a '0' should turn the LED off. The edge connector pins don't map directly to the GPIO registers though… how inconvenient… Be sure to test different values, and check you get the bit pattern the right way around (its endian-ness), as described in the table above.

Task 3: Rolling counter
Function prototype: **void rollingCounter();**
Write code to display a full 8 bit binary count on the LEDs. The code should count from 0 to 255 and begin again. Write your code such that it takes approximately 30 seconds to complete one count from 0 to 255. This requires a little thought... Think how fast your CPU is operating and what you can do to meet these timing requirements. **Remember: FIRST PRINCIPLES!**

Task 4: Knight Rider
Function prototype: **void knightRider();**
Write a function that scrolls a single light across the LEDs. It should move side to side, then reverse direction when it reaches the end… Just like my favourite 1980s TV show when I was growing up (yes, I am that old). https://www.youtube.com/watch?v=WxE2xWZNfOc

Task 5: Count the number of button clicks
Function prototype: **void countClicks();**
Write a function to count the NUMBER OF TIMES that your button is clicked. Display this number in binary on the LEDs. The count should start from zero, and should increase *smoothly* each time the button is clicked…

Assessment

Marks will be awarded for each of the tasks completed, the quality of the code produced, and the quality of the comments written. This work will be assessed through offline marking and a code inspection of your work.

Marking Scheme

Your work will be marked based on the following four categories. Your final grade will be determined based on a weighted mean of the three weight-bearing grades shown in the table below.

Task 1: Turn all the LEDs on	10%
Task 2: Different bit patterns	20%
Task 3: Rolling counter	20%
Task 4: Knight Rider	10%
Task 5: Count the number of button clicks	20%
Code Style and Commenting	20%