SCC.369 Assessment 1 – WK1-2: Lights and Switches

Aim of the Exercise

The aim of this exercise is to familiarize you with the MPLAB X development environment, the PIC assembler language, and some basic input/output. This exercise shouldn't be too taxing, so it is to be completed before start of WK3 and demonstrated in your Week 3 practical. If you have viewed all the blended lectures provided for WK1-2 and read the PIC16F1507 datasheet and instruction sheet, it should be easy to complete.

Software Configuration

You can develop these exercises initially in the MPLAB X simulator but remember that the simulator doesn't necessarily replicate all behaviour of a PIC.

You will need to use the default configuration for Configuration Words (see video regarding this), i.e. INTOSC, MCLRE enabled etc.

The list of SFRs that you should be aware of for this exercise are:

TRISB/TRISC, ANSELB/ANSELC, LATC, PORTB/PORTC, BSR, FSR0H, FSR0L, W

Hardware

You are also expected to demo the code on the actual PIC in Week 3. So be prepared to wire-up the circuits correctly during development and demo phases.

Components

1. 2 each of Red/Green/Yellow/Orange LEDs.
2. Additional 3-color RGB LED which is a common cathode LED (i.e. the longest leg is to be connected to GND and the three other pins individually produce one colour each).
3. 47R resistors. If you want, you can ask for extra 47R resistors.
4. M2M Jumper wires and breadboard
5. Micro:bit to flash program to PIC

Wiring Details

1. Micro:bit to PIC programming config
2. Eight LEDs to Port C Active high
3. Pushbutton 1 Port B.4 Active low
4. Pushbutton 2 Port B.5 Active low
5. Pushbutton 3 Port B.6 Active low

Task Description

This exercise consists of several subtasks, listed below. Implement each one of them **from first principles using pure PIC assembler (no pseudo-instructions),** within a SINGLE project. Write each subtask as a separate block of code, which can be run via a GOTO/CALL instruction from the entry point of your program. For all these exercises, style and comment your code properly as you go along. Remember - separate code into semantic blocks and comment every block of code and subroutine. We'll be marking on how elegant, clean, and efficient your code is. Do your best to make it perfect. Have fun, and don't forget to document everything you discover in your lab book!

Task 1: Turn all the PORTC LEDs on

When the PIC is powered on or reset, all the PORTC LEDs should be on.

Task 2: Flash your student id

Display your student id on PORTC one digit at a time and indicate which position it is at.

a. Store each digit of your student id in the lower nibbles of 8 consecutive registers. The upper nibbles of the registers should store the index of the digit. (e.g., if your student id's last digit is 9, the 8th register should store the value 0x79 i.e., index 7, value 9).
b. The program should start at the PORTC display state in Task 1.
c. When pushbutton on PortB.4 is pressed, the program should enter Task 2's subroutine and show the contents of the first register on PORTC.
d. After this, when pushbutton on PortB.5 is pressed, the contents of the next register should be displayed.
e. Once all 8 digits are shown, the program should return to Task 1.

Task 3: Bit Math Display

Implement bit-wise operations: Buffer (on PC0/1), NOT (on PC5/6), OR (on PC2), AND (on PC3), XOR (on PC4), NOR (on PC7) and display them for a two bit input

a. The program should start at the PORTC display state in Task 1.
b. When pushbutton on PortB.4 is pressed, the program should enter Task 3's subroutine
c. PORTC should now reflect the state (and bit-math results) of state of pushbuttons on PB5 & PB6 as follows:
    - PC0 is ON when PB5 pushbutton is pressed else off
    - PC1 is ON when PB6 pushbutton is pressed else off
    - PC2 is ON when either PB5 or PB6 is pressed else off
    - PC3 is ON when both PB5 and PB6 are pressed else off
    - PC4 is ON when either PB5 or PB6 is pressed but not when both are pressed/released
    - PC5 is ON when PB5 is NOT pressed else off
    - PC6 is ON when PB6 is NOT pressed else off
    - PC7 is ON when neither PB5 nor PB6 are pressed, else off
d. If pushbutton on PortB.4 is pressed again, the program should return to Task 1.

Task 4: Rolling Counter

Write code to display a full 8-bit binary count on the LEDs.

a. The program should start at the PORTC display state in Task 1.
b. When pushbutton on PortB.4 is pressed, the program should enter Task4's subroutine.
c. The code should count from 0 to 255, roll over to zero and begin again.
d. Each complete cycle from 0 to 255 should take approximately 30 seconds to complete.
e. If pushbutton on PortB.4 is pressed again, the program should return to Task 1.

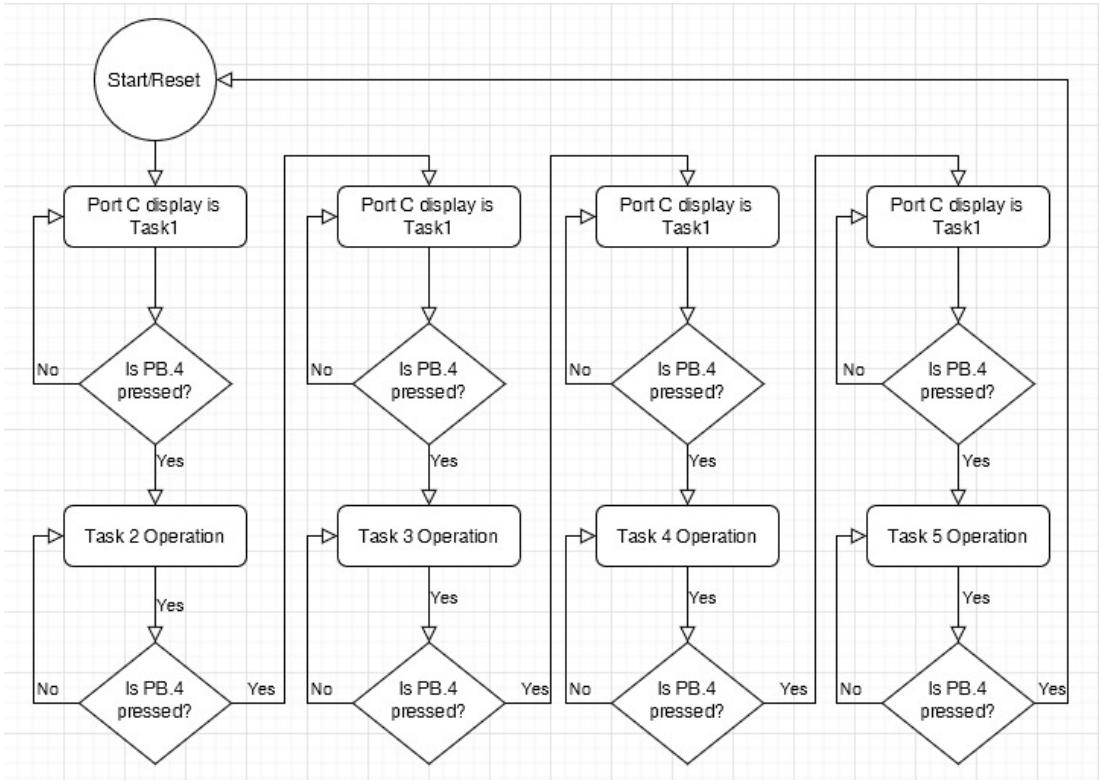
Task 5: LED Chaser

Write a code block to display a chasing LED pattern, from left to right and then back right to left. This is like the Knight Rider car (https://bit.ly/3zPbCuw) OR the Battlestar Galactica Cylon Eye (https://bit.ly/3m8STFG).

a. The program should start at the PORTC display state in Task 1.
b. When pushbutton on PortB.4 is pressed, the program should enter Task5's subroutine.
c. Once Task5 is running, you should display only a SINGLE LED at any point in time.
d. Write your code such that it takes approximately 2 seconds to complete one sweep before beginning again.
e. If pushbutton on PortB.4 is pressed again, the program should return to Task 1.

Task 6: Tying it all together

You may have noticed that we ask you to start with Task 1 and then press PortB.4 to enter the subroutine for Tasks 2-5. The final task is to tie all the 5 tasks together into a single system/program. The overall structure should follow this flow-chart:



Assessment

To gain marks for this assessment, you need to:

a. Complete the individual tasks up to specification.
b. Demonstrate quality of implementation through the code produced the comments written.
c. Submit the code by the Week 2 deadline (WK2 Friday 4pm).
d. Present a working practical demonstration in Week 3 during your own lab session.
e. Demonstrate understanding of PIC fundamentals through code walkthrough of your work.

Remember: NO THIRD PARTY CODE ALLOWED. (But it's okay to use the example from Week 1 as a starting point and think about the effect of modifying the delay counters for Task4)

Assessment Weights

Task	Task Description	Weights
1	Turn on all the PortC LEDs	10%
2	Flash your student id	20%
3	Bit Math Display	20%
4	Rolling Counter	20%
5	LED Chaser	15%
6	Tying it all together	15%

In all cases a grade descriptor (A, B, C, D, F) will be used to mark your work in each category.

NOTE:

Both code submission (due WK2 Friday 4pm) and demo/attendance during marking slot in WK3 are essential for grading. Missing either of these without a legitimate extension will result in a 0 or F4 for this piece of coursework.