2

General Purpose Input Output

2.1 Introduction

This lab aims to help students to gain more experience in the MSP430 Education Board, MSP430G2553 microcontroller and its assembly language. Students are recommended both to read the supplementary material **Supplementary_Chapter_6_General_purpose_IO** on Ninova. Also, if preferred, they could bring their own computers to the laboratory on which Texas Instruments Code Composer Studio IDE is installed (For installation instructions, review **CCS_Installation.pdf**).

2.2 Background Information

The general purpose input and output (GPIO) using the ports of MSP430G2553 (i.e., Port 1 and 2) can be performed by configuring and reading/setting the corresponding registers of the selected port. The following two instructions read P1.2 and conditionally branch depending on the state of the button.

```
    ;read the switch at P1.2 and set flags
    bit.b #00000100b,&P1IN
    jnz     ON          ;if P1.2, branch to the label OFF
```

The following two instructions clear and set LED 5 respectively.

```
    bic.b #00010000b,&P1OUT ; clear P1.4
    bis.b   #00010000b,&P1OUT ; set P1.4
```

Note that, *this document is written to describe the objectives of the experiment. Thus, it only contains short background information to simply remind you the basics. The given background information in the experiment sheets is not sufficient enough to successfully complete the experiments, you should read necessary supplementary material before coming to the laboratory.*

2.3 Experiment

2.3.1 Part 1 Controlling single LED via Push Button

Write an assembly program which controls LED 2 on Port 1 using the push button 3 on Port 2. Your program should toggle LED 2 whenever P2.2 is pressed. In order to do so, the written program should read the status of the button through P2.2 in an infinite loop and update the output accordingly.

2.3.2 Part 2 Counting Push Button Events

Write an assembly program that counts how many times the push button 3 on Port 2 is pressed. In this manner, you should first define a **variable in memory rather than using an accumulator** to store how many times the button pressed. Then, your program should display the result (value stored in memory) on the LEDs connected through Port 1 in binary form. (Hint: variables can be declared and initialized in the *data* section of the assembly code. More information can be found in 1_MSP430_Introduction document.)

2.3.3 Part 3 Adding Reset Mechanism to Counter

In this part of the experiment, you are asked to improve your code from **Part 2** by adding Reset Mechanism to your program. In this manner, you should utilize another push button from Port 2 and clear the content of the counter when that button is pressed.

2.4 Report

Your report should contain your program code (with explanations) for Part 1, Part 2 and Part 3 and explain the anomalies (if countered) with their possible reasons while testing your program.