Experiment 2: General Purpose Input Output

23.10.2023
Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

*M*usic is the pleasure the human mind experiences from counting without being aware that it is counting.

Gottfried Leibniz

1 -Introduction and Preliminary

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.

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
```

The following two instructions clear and set LED 5 respectively.

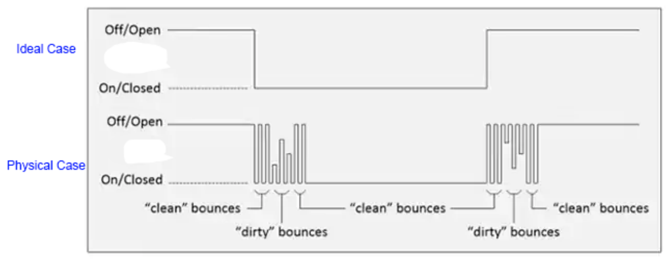
```
    ;read the switch at P1.2 and set flags
    bic.b   #00010000b,&P1OUT ;clear P1.4
    bis.b   #00010000b,&P1OUT ;set P1.4
```

Remember; bic and bis instructions are bit clear and bit set instructions respectively. bit instruction is bit test instruction. Note that bic and bis use masks and they are different than mov instruction. In MSP430 bit test compares two operands and sets status bits. Note that, this document is written to describe the objectives of the experiment. You are required to know(learn) how to use the MSP430. Examples in the first part are not sufficient enough to successfully complete the experiment. They can only give you some insights. You should read the necessary supplementary material before coming to the laboratory and in the laboratory.

2 (30 pts) - Part 1

In this part you will assign a register to count how many times a specific button is pressed. You can implement an endless loop to check the button presses. However, the most important thing here is switch debouncing.

Switch debouncing is a technique used in electronics and digital design to ensure that the electrical signals generated by a mechanical switch are interpreted correctly by digital circuits or microcontrollers. Mechanical switches, such as push buttons or toggle switches, can exhibit bouncing when they are pressed or released. This bouncing is caused by the physical properties of the switch contacts and can result in multiple open and close transitions in a very short period of time, even when you intend to make just one clean transition. An example for the physical world is shown in the figure below.



To not include "dirty bounces"in your calculation, you can use a small amount of waiting after each successful press. The operations will be checked by investigating the registers in the Debug mode.

3 (30 pts) - Part 2

Change the code in Part 1 to use three push buttons:

- Increase the variable X (Which is initially 0)
- Increase the variable Y (Which is initially 0)
- Reset the values of X and Y.

4 (40 pts) - Part 3

Assign another button that calculates $Z = X * Y$ and stores in another register. Remember that, there is no assembly command for this. Thus, you should design a loop.