Lehrstuhl für Elektrische Mess- und Prüfverfahren
Prof. Dr. L. M. Reindl

Praktikum Mikrocomputertechnik

Exercise 2

While you have used the digital I/O pins as output within exercise 1, you will also use them as input pins within this exercise. These inputs will be controlled by the two pushbuttons `PB5` and `PB6`, which you find at the lower edge of your hardware board. They can be connected by the pin headers `X11` und `X12` to the microcontroller. While pressing the button, the header outputs will be tied to ground GND. After setting PxDIR and PxSEL, you can capture the logical level by reading the register PxIN[1]. Please note how the buttons are connected to the controller. By using the PxREN[2], you can enable the microcontroller's interal pull-up or pull-down resistors.

Please note:
If not mentioned differently, all tasks should be solved by using polling, i.e. you don't have to use interrupts, but just capture the pin states actively by your program.

Please note:
If not mentioned differently, solve all tasks of an exercise within the same project. All subtasks shall be processed together/successively in the same program.

Task 1

a) Connect the button `PB5` with `CON3:P1.3` and the button `PB6` with `CON3:P1.4`. Moreover, route the red LED (`K3 LED rt`) to the connector `CON3:P1.5` and the greed LED (`K4 LED gn`) to `CON3:P1.6`.

b) Write a program which is monitoring the button `PB5`. If the button is pressed, the red LED shall blink once. This means that the LED should not be activated several times once the button is kept pressed. However, if you release the button and press again, it shall blink again (**2 points**)

c) Add the following feature to you program: The green LED shall be activated while button `PB6` is pressed (**1 Pkt.**).

d) Connect the blue LED with `CON3:P1.0`. Add the following feature to you program: If both buttons are pressed, the blue LED shall be illuminated (**2 points**).

e) Connect `CON3:P1.7` to the yellow LED, which is placed next to the relay (right pin of `JP3`). Make the yellow LED glow each time the red LED is not turned on (**1 point**).

f) Export the code which makes the red LED blink into a separate function (you might also include the lines triggering the yellow LED). An example of how to use functions is shown in the C cheat sheet (**1 point**).

[1] see MSP430x2xx Family User's Guide: chapter 8.2.1
[2] see MSP430x2xx Family User's Guide: chapter 8.2.4

g) To read out PB5, use the function of an interrupt (see listing 1), but continue to poll PB6. Don't delete the polling code for button PB5, but just uncomment the code (**2 points**).

Task 2

a) Please generate a file `erfahrungen.txt` with a short feedback on the exercise sheet, i.e. name the challenges you had to face, which additional information should have been provided additionally, and so on. (**1 point**).

b) Import your text file to the Code Composer Studio project, export the project and upload the file to ILIAS.

Listing 1: Beispiel für die Initialisierung und Verwendung von Interrupts.

```
// Initialization
P1DIR &= ~BIT0;  // Set as input
P1REN |= BIT0;   // Enable pull-resistors
P1OUT |= BIT0;   // Set to pull-up
P1IE  |= BIT0;   // Enable interrupt
P1IES |= BIT0;   // High/Low-Edge
P1IFG &= ~BIT0;  // Clear interrupt flag

// ...other code

// Port 1 interrupt vector
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void) {
  // Do something (but keep in mind you're still in the interrupt,
  // so don't let it take TOO long).
  // Also note that all variables that you change within this function
  // must be declared 'volatile'.
  // Clear interrupt flag (here - as an example - the flag of P1.0).
  P1IFG &= ~BIT0;
}
```