Contact: Sebastian Stöcklin
sebastian.stoecklin@imtek.uni-freiburg.de

Winter term 2016/2017

Exercise sheet 2 - Digital I/O and interrupts

In Experiment 1, the digital I/O pins have only been used as output pins. In this experiment, these pins should also be used as an input. On the circuit board, you can find two buttons `PB5` and `PB6`, being located on the bottom side. They can be connected to the microcontroller via the header pins `X11` und `X12`. When you press these buttons, a physical connection to the ground potential is established. Once you configured the registers PxDIR and PxSEL, you can query the pin's logical state by reading the input register PxIN[1]. Please also consider the wiring of the buttons. The PxREN register[2] will allow you to internally connect pullup-/pulldown resistors to the I/O-ports.

Note:
*Unless noted otherwise, you should solve all tasks in this experiment by **polling**, i.e. actively read the corresponding pins instead of using an interrupt service routine.*

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 pts.**)

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

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 pts.**).

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 pt.**).

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 pt.**).

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 lines (**2 pts**).

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

Task 2

a) Create a file `feedback.txt` with a brief feedback statement, which contains specific problems and issues you experienced while solving the exercise, additional requests, positive remarks, etc. (**1 pt.**).

b) Import this text file `feedback.txt` in your Code Composer Studio project, so that you can upload it together with your software deliverable.

Listing 1: Example of the initialization and use of 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;
}
```