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

Exercise sheet 4 - Analog-to-digital converters

While the signal flow within a microcontroller is always digital, most sensors provide analog data. In order to be able to read and process the output of those sensors, a so-called analog-to-digital converter (ADC) is required. This unit can convert an analog voltage to a digital value, related to a certain reference value also provided to the ADC.
The MSP430G2553 used in the exercise has an already built-in ADC, which you can route to different pins by appropriate software configuration to measure an analog signal applied to the corresponding pin.
In the following program an analog signal applied to pin 1.7 is measured and then fowarded by the serial interface. For example, you could connect pin 1.7 to the potentiometer (**X6:U_POT**) and then observe a change in the value in the serial console when turning the potentiometer.

Listing 1: Example Program for the use of the ADC.

```
/*
 * This programm will print the analog value at P1.7 to the serial console.
 *
 * Connect the following pins:
 * P1.7 (CON3) with U_POT (X6)
 */

#include <templateEMP.h>

void main(void) {
  initMSP();
  // Turn ADC on; use 16 clocks as sample & hold time (see p. 559 in the user guide)
  ADC10CTL0 = ADC10ON + ADC10SHT_2;
  // Enable P1.7 as AD input
  ADC10AE0 |= BIT7;
  // Select channel 7 as input for the following conversion(s)
  ADC10CTL1 = INCH_7;

  while (1)
  {
    // Start conversion
    ADC10CTL0 |= ENC + ADC10SC;
    // Wait until result is ready
    while(ADC10CTL1 & ADC10BUSY);
    // If result is ready, copy it to m1
    int m1 = ADC10MEM;
    // Print m1 to the serial console
    serialPrintInt(m1);
    serialPrintln(  );
  }
}
```

Task 1

a) Connect the potentiometer to `CON3:P1.5` and show the current value using the LEDs `D1` to `D4`. For this purpose, divide the maximum measuring range in 5 approximately equal sections. If the measured value is in the first section, no LED should be turned on; is it in the second section, only `D1` should light up; is it in the third, `D1` and `D2` should shine, in the fourth `D1`, `D2` and `D3`, and finally in the fifth section, all LEDs from `D1` to `D4` should be turned on (**2 pts. for general function + 1 pt. for division into sections**).

b) Then, additionally connect `K2:LDR` to `CON3:P1.4`, the red LED to `CON3:P3.0`, the green LED to `CON3:P3.1` and the blue LED to `CON3:P3.2`. Put the jumper `JP2` to `COL`. You should also use the black plastic tube from your development kit to place it in a way that it encloses the *light dependent resistor* (LDR) as well as the red, green and blue LED.

   On the pin `K2:LDR`, you can now meaure an analog signal, which is related to the intensity of the light reaching the LDR. Your task is now to extend your program from Task 1a) so that it recognizes the color of a plastic chip placed on top of the black tube (keep the functionality of the potentiometer!). As soon as a chip is placed on the tube, its color shall be printed to the serial interface, so that you can see it on the computer. Your program has only to be able to distinguish between white, black, red, green and blue (**1 pt. for each color + 1 pt. for the detection of a new chip being present**) .

   A few hints:

   1) You can measure the the spectral component of a chip by applying the corresponding LED and by performing an LDR measurement, i.e. to measure the red component,turn on the red LED and then perform a measurement. Do the same for blue and green.
   2) The LDR always requires a few milliseconds to adapt to abrupt changes in light intensity.
   3) It's okay if a change in the potentiometer output is followed by a delayed output; you don't have to optimize your code for a fast update of `D1` to `D4`.

c) **Bonus:** Expand your program to detect other colors (**1 pt.**).

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.