Exercise Sheet 4 - Analog-To-Digital Converters

Note:
You will present your results of this exercise sheet to your tutor in a **colloquium**. Exact dates for each group will be announced by the individual tutors.

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 '**A**nalog-to-**D**igital **C**onverter' (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 a built-in ADC, which you can route to different pins by appropriate software configurations, in order to measure an analog signal applied to the corresponding pin.
In the following program an analog signal applied to `CON3:P1.7` is measured and then fowarded by the serial interface. For example, you could connect `CON3:P1.7` to the potentiometer `P1` (`U_POT`) and then observe a change in the value in the serial console when turning the potentiometer `P1`.

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

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

Task 1

a) Connect the potentiometer P1 to CON3:P1.7 and represent its setting angle using the LEDs D1 to D4. For this purpose, divide the maximum measuring range into five approximately equal sections. If the measured value is in the first section, no LED should be illuminated; if it's in the second section, only D1 should be illuminated; if it's in the third section, D1 and D2 be illuminated, if it's in the fourth section D1, D2 and D3 should be illuminated, and finally if it's in the fifth section, all LEDs from D1 to D4 should be illuminated. (**2 pts. for general function + 1 pt. for division into sections**)

b) Then, additionally connect LDR to CON3:P1.4, the red LED D6 to CON3:P3.0, the green LED D5 to CON3:P3.1 and the blue LED D7 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, so that it encloses the **l**ight **d**ependent **r**esistor (LDR) as well as the red, green and blue LEDs.

   At pin LDR, you can now measure 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. Therefore, make sure to keep the functionality of Task 1a). As soon as a chip is placed on the tube, its color shall be printed via serial interface, so that you can see it on your computer. Your program should be able to detect when no chip is placed on the tube. Further, it should be able to distinguish between white, black, red, green and blue chip colors. (**1 pt. for the detection of a chip being absent + 1 pt. for each chip color you can detect**)

Note:
You can measure the spectral component of a chip by illuminating individual LEDs while performing LDR measurements. Therefore, to measure the red component, turn on only the red LED and then perform a measurement. Do the same for blue and green accordingly.

Bonus:
Expand your program to detect another color. (**1 pt.**)

Hints:
The LDR always requires a few milliseconds to adapt to abrupt changes in light intensity. You don't have to optimize your code for a fast update of D1 to D4. Therefore, a change in the potentiometer P1 position may be followed by a delayed output.

Task 2

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 and alike. Import this text file `feedback.txt` in your **C**ode **C**omposer **S**tudio (CCS) project, so that you can upload it together with your software deliverable. (**1 pt.**)