Practical Exercise – Microcontroller Techniques
Prof. Dr. L. Reindl
Laboratory for Electrical Instrumentation

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

Winter term 2016/2017

Exercise sheet 7 - Watchdog timers & general timer modules

In this experiment, we will see two means of avoiding a system to get stuck in certain conditions:

1. **Watchdog**
   The watchdog is a hardware timer that prevents or detects failures in microcontroller systems. Its principle: During regular operation, the software will periodically tell the watchdog timer that it's still alive and working properly (basically by resetting the watchdog timer's counter). When there is some non-intended error within the program, the watchdog will not be reset and its counter will overflow. In such cases, actions to restore the functionality of the system can be carried out (i.e. a system restart).

2. **Dynamic control systems**
   In case of errors (failure of the clock system, program bugs or partial failure of the supply voltage), an output pin of a microcontroller system can be stuck at any static state (high, low or intermediate values). External circuits connected to these pins (such as a relay) can therefore be driven in non-intended states. This can be partially prevented by dynamic control principles.

Note:
Connect the watchdog timer with the *Internal Very-Low-Power Low-Frequency Oscillator* (VLO) using the *Auxiliary Clock* module (ACLK). Make yourself familiar with the *Basic Clock Module+* in the *MSP430x2xx Family User's Guide*.

Task 1

a) Connect the button `PB5` with `P1.3` and `LED gr` with `P1.0`.

b) Let `LED gr` blink with a frequenzy of 4 Hz. If the button is pressed, the program should enter an endless loop, in which no code will be executed. The watchdog should recognize the inactivity of the microcontroller caused by the endless loop and reset the microcontroller after 4 seconds. (**2 pts.**).

Task 2

a) Additionally, connect `X3` with `X10` and activate the heating resistor via `JP4`. Also connect `X4` with `P3.5`, `X5` with `P3.4` and `LED rt` with `P3.2`. Moreover connect the thermistor `U_NTC` to `P1.5`.

b) Modify the watchdog configuration in that way that it resets the microcontroller after 20 seconds (**1 pt.**).

c) The NTC's resistance will change with temperature. Your task is to implement a thermometer using the NTC for measurement and the LEDs D1 to D4 as well as the red LED to display the temperature.
Therefore, determine the range of possible output voltages of the NTC voltage divider with the ADC for the complete temperature range you can apply with the heater: Activate the heating resistor R28 with the static input of the relay REL_STAT at X5 and capture the maximum and minimum values of the ADC for the complete heating period (it might take a few minutes till the temperature reaches its maximum). Divide the value range into six equal subranges, which are represened with the LEDs D1 to D4 for the five lower ranges and with the red LED on LED rt for the sixth range (the red LED indicates that the temperature is too high = overheating) (**1 pt.**).

d) The thermometer's visual LED output shall be refreshed every two seconds. This shall be accomplished by configuring a timer to trigger an interrupt every two seconds. Read and display the temperature within the interrupt service routine (**2 pts.**).
Note: Consider table 5 (Interrupt Sources, Flags, and Vectors) of the MSP430G2553 Datasheet as well as the *MSP430x2xx Family User's Guide*. Examine which of the two `Timer0_A3` interrupt types is required. Therefore, compare the values of the two macros `TIMER0_A0_VECTOR` and `TIMER0_A1_VECTOR` with the entries in the table and select the appropriate vector for your interrupt service routine.

e) Expand the main routine of your program to a simple controller which ensures that the temperature is staying always in the ranges 3 or 4 (**1 pt.**). Thereby, keep the functionality of task 1. When PB5 in pressed while the heater is warming up, you should end up in the overheating region (Oh no! The good thing is that the watchdog will restore your system at some point.).

f) With the given circuit board, there is a way of avoiding overheating of the NTC even if your controller program gets stuck (as you've seen in task 2d when pressing the button). Therefore, modify your controller of 2d) that it does not use X5, but X4 for triggering the heater (consider the schematic to see what has to be changed in the heating algorithm) (**2 pts.**). The thermometer app (using the interrupt) shall still run in the background without any modification. Test if you program avoids overheating when jumping into an infinite loop with PB5.

Task 3

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.