Lösung 07: Watchdog, Energiesparmodi, Temperatursensor

Aufgabe 1: Watchdog

```
#include <avr/wdt.h>

void setup() {
        Serial.begin(9600);
        Serial.println("System restart");

        // start watchdog
        cli();
        wdt_reset();
        // preparation for configuration: write logic one to WDCE and WDE, p61
        WDTCSR |= (1<<WDCE) | (1<<WDE);
        // within 4 clock cycles: set timeout to 4 seconds and start watchdog
        // Hint: WDCE bit is implicitly cleared by "="
        WDTCSR = (1<<WDP3) | (1<<WDE);

        // configure interrupt
        DDRD &= ~(1 << DDD0);          // configure PD0 as input
        PORTD |= (1 << PORTD0);  // pull up, write to PORT when in INPUT mode, p68
        EIMSK |= (1 << INT0);          // turn on INT0
        EICRA |= (1 << ISC01);         // set INT0 to trigger on falling edge
        sei();                                    // globally activate
interrupts
}

void loop() { }                        // empty!

ISR (INT0_vect) {
             wdt_reset();
             Serial.println("ResetWDT");
}
```

Hinweise:

- Die internen Pull-Up Widerstände werden beim ATmega2560 standardmäßig aktiviert, wenn ein Pin als Eingang konfiguriert ist.
- Konfiguriert man für den Watchdog Timeout 4s, so beobachtet man auf der Konsole vermutlich etwas mehr als 4 Sekunden bis man erneut „System Restart“ auf der seriellen Konsole liest. Der Grund ist, dass der Restart selbst Zeit benötigt. Man misst also eigentlich 4s + für den Restart benötigte Zeit.

Aufgabe 2: Power-Down Mode

```
#include <avr/sleep.h>

void setup()
{

       Serial.begin(9600);
       delay(2000);
       Serial.println("System restart");

       // activate interrupt on pin0 of PORT D (INT0)
       DDRD &= ~(1 << DDD0);
```

```
        PORTD |= (1 << PORTD0);
        EIMSK |= (1 << INT0);
        EICRA |= (1 << ISC01);
        sei();

        // configure power-down mode
        SMCR |= (1 << SM1);
}

void loop() {
        // enter sleep mode
        Serial.println("Going to sleep");
        delay(200);
        sleep_mode();

        Serial.println("Waking up");
        delay(1000);

}

// ISR: can be empty, but must exist!
ISR (INT0_vect) {
}
```

Aufgabe 3: Temperatursensor

Im folgenden Code wird gleich der Free Running Mode eingesetzt. Immer nach Beenden einer A/D Umsetzung wird sofort die neue Umsetzung gestartet.

```
void setup() {
        Serial.begin(9600);

        // enable ADC functionality
        ADCSRA |= (1 << ADEN);

        // use /128 prescaler, see manual p271
        ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

        // select autotrigger, to use free-running mode
        ADCSRA |= (1 << ADATE) | (1 << ADSC);

        // use free-running mode
        ADCSRB &= ~((1 << ADTS2) | (1 << ADTS1) | (1 << ADTS0));

        // select ADC2 as input pin
        ADMUX |= (1 << MUX1);

        // use reference voltage 2,56 V, manual p281
        ADMUX |= (1 << REFS1) | (1 << REFS0);


}

void loop()
{
        // note: conversion is automatically triggered in free running mode
        // read analog value, first LOW then HIGH register
```

```
        unsigned int read = ADCL + 256 * ADCH;

        // convert integer value into temperature
        double temperature = 0.25 * read - 50;
        Serial.println(temperature);
        delay(1000);
}
```