MP-P2
Gruppe 4

2021

Timer und Interrupts

15. Dezember 2021

Florian Tietjen 2519584
Emily Antosch 2519935

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einführung

Im dritten Praktikum wollen wir uns mit dem A/D-Wandler des TivaWare-Boards auseinandersetzen. Dabei wollen wir sowohl herausfinden, wie man mit externen Peripheriegeräten arbeiten kann, als auch das interne A/D-Modul effektiv nutzen. Darüberhinaus interessiert uns auch der PWM-Modus des Timers als mögliche Dimmung einer LED basierend auf dem analogen Signal eines Sensors (in unserem Beispiel verwenden wir einen Joystick).

2 Aufgabe 1: Externe D/A-Wandler mit Treppenverfahren

In unserer ersten Aufgabe benutzen wir das in der Vorlesung behandelte Treppenverfahren, um den Spannungswert eines Komparators zu ermitteln und diesen mittels BCD-Code auf einem Display auszugeben. Wir möchten zudem dann die Richtigkeit unseres Ergebnisses überprüfen, indem wir die verschiedenen Spannungen auf dem Oszilloskop anzeigen lassen. Die Triggerspannung erhalten wir dabei von Pin $PL(2)$.

Wir verwenden dabei folgenden Code:

```
/*
Mikroprozessortechnik P3 - Aufgabe 1
Autoren: Emily Antosch, Florian Tietjen
Beschreibung: Dieses Programm approximiert den Spannungswert der Spannung UE vom
    Komparator durch das Treppenverfahren.
*/
#include<stdio.h>
#include<stdint.h>
#include "/home/akku/ti/TivaWare_C_Series-2.2.0.295/inc/tm4c1294ncpdt.h"

// Prototypes
void init_port(void);
void init_adc(void);
unsigned int read_adc();


int main(void){
    // Initilization of the varibales
    unsigned short int da_increase = 0;
    unsigned int adc_value = 0;
    unsigned int last_adc_value = da_increase-1;

    while (1)
    {
        // ADC start
        ADC0_PSSI_R = 0x0001;
        // Save last value
        adc_value = read_adc();
        //delay(480);
        // If adc detectes a near zero value
        if(adc_value <= 0.1){
            //GPIO_PORTM_DATA_R = (last_adc_value \% 10) | (((last_adc_value / 10)
    \% 10) << 4);
            //GPIO_PORTL_DATA_R = ((last_adc_value / 100) \% 10) & 0x03;
            GPIO_PORTL_DATA_R |= 0x04;
            GPIO_PORTL_DATA_R = 0x02;
            GPIO_PORTM_DATA_R = ((int)(last_adc_value * 19.53125) / 1000 \% 10) << 4
     | ((((int)(last_adc_value * 19.53125) / 100) \% 10));
            GPIO_PORTL_DATA_R = 0x01;
            GPIO_PORTM_DATA_R = ((int)(last_adc_value * 19.53125) / 10 \% 10) << 4;
        }
        // If the button is not pressed
        if(GPIO_PORTD_AHB_DATA_R & 0x02 == 1){
            // Continue operation
            GPIO_PORTK_DATA_R = da_increase;
            da_increase++;
        }
        GPIO_PORTL_DATA_R = 0x00;
```

```
46      }
47  }
48
49
50  void init_port(void){
51      // Wait for clock for GPIO to be ready
52      SYSCTL_RCGCGPIO_R |= 0x0E08;
53      while((SYSCTL_PRGPIO_R & 0x0E08) == 0){}
54
55      // Initilization of the port D with ADC
56      GPIO_PORTD_AHB_DEN_R = 0x02;
57      GPIO_PORTD_AHB_AFSEL_R |= 0x01;
58      GPIO_PORTD_AHB_DEN_R &= ~0x01;
59      GPIO_PORTD_AHB_AMSEL_R |= 0x01;
60      GPIO_PORTD_AHB_PUR_R = 0x02;
61
62      // Initilization of the port K with the output
63      GPIO_PORTK_DEN_R = 0xFF;
64      GPIO_PORTK_DIR_R = 0xFF;
65      // Initilization of the port M for the BCD display
66      GPIO_PORTM_DEN_R = 0xFF;
67      GPIO_PORTM_DIR_R = 0xFF;
68      // Initilization of the port L for the BCD display and trigger pulse
69      GPIO_PORTL_DEN_R = 0x07;
70      GPIO_PORTL_DIR_R = 0x07;
71      GPIO_PORTL_DATA_R = 0x00;
72  }
73
74
75  void init_adc(void){
76      // Wait for clock for ADC to be ready
77      SYSCTL_RCGCADC_R |= 0x01;
78      while((SYSCTL_PRADC_R & 0x01) == 0){}
79      // Disable the ADC
80      ADC0_ACTSS_R &= ~0x0F;
81      // Set the corresponding Sequencer
82      ADC0_SSMUX0_R |= 0x0F;
83      // Set the corresponding Sequencer 0, length = 1
84      ADC0_SSCTL0_R |= 0x02;
85      // Enable the ADC
86      ADC0_ACTSS_R |= 0x01;
87  }
88  // Unused function for now
89  unsigned int read_adc(){
90      unsigned int result = 0;
91      while((ADC0_SSFSTAT0_R & (1<<8)) == 0){}
92
93      result = (unsigned int)ADC0_SSFIFO0_R * 5000 / 4095;
94      return result;
95  }
```

Listing 1: stepfnc.c zur Approximation der Spannung U_E mit dem Treppenverfahren

3 Aufgabe 3: Dimmen einer LED mithilfe des internen A/D-Wandlers

In der nächsten Aufgabe wollen wir uns mit dem internen A/D-Wandler befassen. Mit diesem messen wir die Ausgangsspannung eines Joysticks in der Y-Achse. Einen Pin des Ports M verbinden wir dann mit einer LED, um diese mit einem der Timer im PWM-Modus zu dimmen. Eine Bewegung führt daher zu einer Erhöhung oder Verringerung der Helligkeit der LED. Dafür verwenden wir den