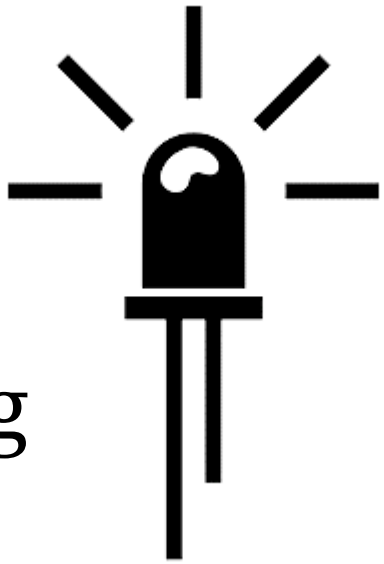
LED- Ansteuerung



TIM JOCHEN KICKER, RAFFAEL ZAFFIGNANI

Inhaltsverzeichnis

Kurzbeschreibung

In diesem Projekt soll eine LED mittels des ATMega16-Microcontrollers gesteuert werden. Dabei wird zwischen 3 Modis unterschieden: In einem ist es möglich, die Intensität der LED zu steuern, im anderen Modus wird dabei ein SOS-Signal ausgegeben und im dritten Modus können zwei LEDs angesteuert werden. Ersteres wurde mit dem Timer1 des ATMega16's realisiert.

Hardware

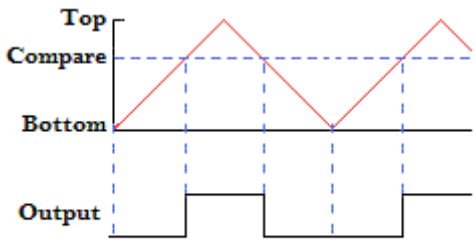
Hierbei wurde die von der HTL-Rankweil zur Verfügung gestellte MEGACARDv4 verwendet. Auf dieser ist der Microcontroller bereits angebracht und verfügt über weitere hilfreiche Schnittstellen, an welche die LED angebracht werden kann.

Verwendete Software

Software	Version
Microchip Studio	7.0.2542
Atmel Kits (Additional Package)	7.0.132
HID-Bootflash	1.0

Funktion

Der Grundbaustein des Programmes liegt im Timer1 des ATMega16's. Der Timer1 zählt dabei aufwärts bis zum maximalen Wert 1024. Sobald er den beinhalteten Wert des OCR1A Registers erreicht, wird ein Interrupt ausgelöst. Bei diesem wird jeweils die LED umgeschaltet. Legt man den Wert des genannten Registers niedrig, so erreicht die LED selbst eine sehr hohe Frequenz. Das gegensätzliche gilt in die andere Richtung. Somit lässt sich die Frequenz (bzw. Helligkeit) der LED individuell einstellen. Mit einem weiteren Taster kann eine rote LED zusätzlich aktiviert werden.



Phase Correct PWM

Beim SOS-Mode hingegen wird das PWM-Signal pausiert und die LED über einen Delay gesteuert. Die Abfolge für das SOS-Signal lautet dabei wie folgt:

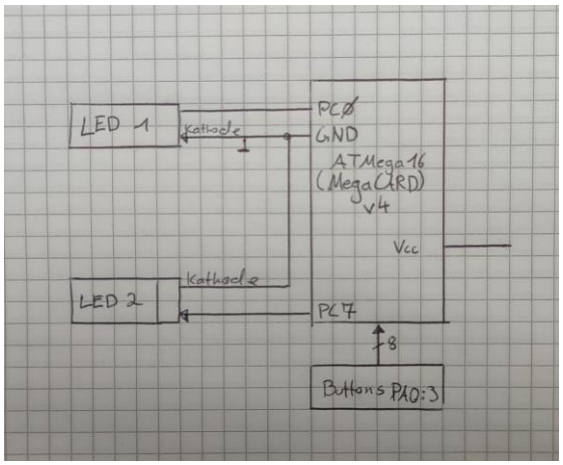
LANG – LANG – LANG – KURZ – KURZ – KURZ

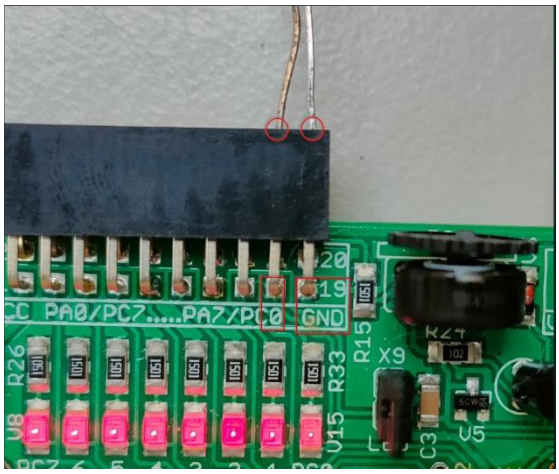
Tastenbelegung:

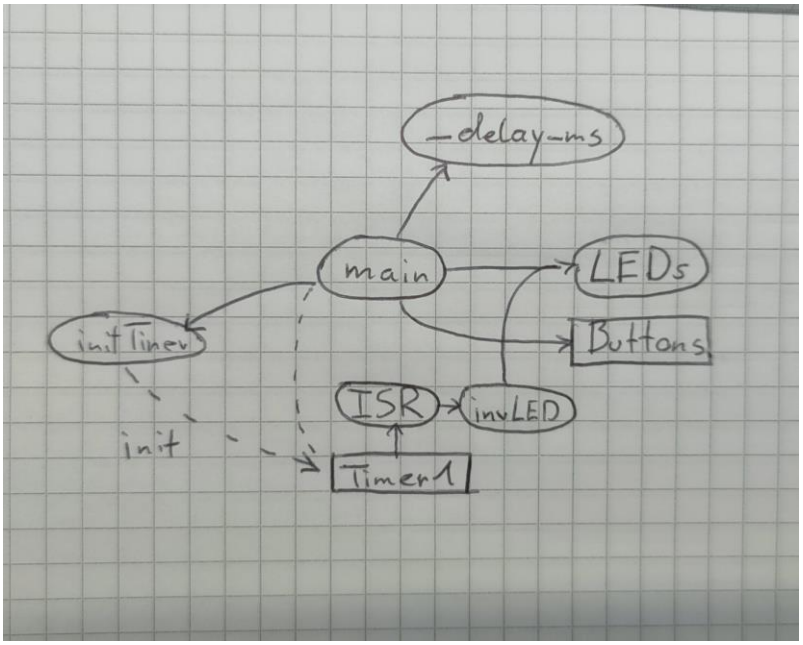
Taster	Funktion
PA0	LED Dimmen (LED-Controll Mode)
PA1	LED Erhellen LED-Controll Mode)
PA2	SOS-Modus
PA3	Ansteuerung von zwei LEDs

Aufbau – Blockschaltbild + Hirarchie

Beim Aufbau wird lediglich die Kathode mit dem GND-Pin (neben PC0) verbunden und die Anode mit dem PC0-Pin der MEGACARD. Die MEGACARD selbst muss per USB an eine Spannungsquelle angeschlossen werden. (Laptop reicht völlig aus)







Zeitplan

Arbeitstag	Arbeit
07.12.2021	Informationen über Timer 1 sammeln | Benötigte Register auswählen
14.12.2021	Testprogramme entwickeln | Start der Dokumentation
21.12.2021	Steuerung der Intensität fertigstellen
11.01.2022	SOS- Modus entwickeln | Dokumentation zum größten Teil fertigstellen
18.01.2022	Modus zur Ansteuerung zwei weiterer LEDs realisieren
25.01.2022	Dokumentation fertigstellen

Software (C-Code)

Das Programm wurde mithilfe von Microchip Studio in der Sprache C geschrieben und per HID-Bootflash auf die MEGACARD gespielt.

Bibliotheken:

- „Interrupt.h“: Interruptbibliothek für die Umschaltung der LEDs durch den Timer1
- „delay.h“: Delaybibliothek für die Pausierung zwischen den SOS-Signalen
- „stdbool.h“: Bibliothek für die Einbindung des Datentypes Boolean

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
```

Interrupts:

Die ISR für den Timer1 Compare wird ausgeführt, sobald der Timer den Wert des OCR1A Registers erreicht hat. Dabei werden jeweils die LEDs durch die Funtktion invertLEDs() invertiert.

```
ISR(TIMER1_COMPA_vect) //Display signal on onboard LEDs
{
    invertLEDs(); //LEDs umschalten
}
```

invertLEDs Methode

Diese Funtkion dient nur zur Invertierung der LEDs. Dabei wird der Status der roten LED berücksichtigt

```
void invertLEDs(){
    if ((PORTC == 0xFF ) | (PORTC == 0xFC)){ //Sind LEDs derzeit an?
        PORTC = 0x00; //LEDs ausschalten
    }
    else {
        if (redLight == 1){ //Rotes Licht aktiviert?
            PORTC = 0xFF; //Alle LEDs an
        }
        else {PORTC = 0xFC; } //Nur weiße LED ansteuern
    }
}
```

Timer1-Setup Methode:

In dieser Funktion wird ein Setup des Timer1 durchgeführt. Die einzelnen Code-Zeilen werden in den jeweiligen Kommentaren erklärt.

```
void timerSetup(){

    OCR1A = 512;                        // Set Dutycycle to 50% (1024/2)

    TCCR1A |= (1<<COM1A0);                  // Set Timer1 in Phase Correct Mode 10 Bit

    TCCR1A |= ((1<<WGM10)|(1<<WGM11));      // Phasen Richtiger Modus
                                            //mit Compare-Match bei OCR1A
    TCCR1B |= (1<<CS10);                    // Vorteiler auf 1

    TIMSK |= (1<<OCIE1A);    // Enable Timer 1 Overflow
                            // and Compare interrupt

}
```

Main-Methode:

Anfangs dieser Methode werden die Ports/Portrichtungen definiert und die Setup-Methode aufgerufen.

```
    DDRC = 0xFF;    //LEDs output
    DDRA = 0x00;   //Taster Ports auf input
    PORTC = 0x00; //LEDs aus
    PORTA = 0x0F;

    cli(); //Disable Interrupts
    timerSetup(); //Execute timer-setup method
    sei(); //Enable interrupt
```

In der Arbeitsschleife wird zuerst nach Betätigung eines Tasters abgefragt und die dementsprechende Tasterfunktion durchgeführt. Dies ist in den jeweiligen Kommentaren genauer beschrieben.

```
while (1) { //Arbeitsschleife


        if((OCR1A < 1020) && (!(PINA&(1<<PA0)))){ //Wenn Taster 0
gedrückt und max nicht erreicht
            OCR1A++; //Duty Cicle inkrementieren
            _delay_ms(10);
        }
        else if( (OCR1A > 5) && (!(PINA&(1<<PA1)))){ //Wenn Taster
1 gedrückt und min nicht erreicht
            OCR1A--; //Duty Cicle veringern
            _delay_ms(10);
        }
        else if (!(PINA&(1<<PA3))){ //Rote-LED aktivieren /
deaktivieren
            if (redLight == 0){redLight = 1;}
                else{redLight = 0;}
            _delay_ms(40); //Warten um Tastenprellen zu verhindern
        }
        else if ((!(PINA &(1<<PA0))) && (!(PINA &(1<<PA1)))){

        }

         else if( (!(PINA &(1<<PA2)))){ //SOS-Mode
            OCR1A =6;
            while(1)
            {
                PORTC=0xFF; // Lang
                _delay_ms(5000);
                PORTC=0x00; // Pause
                _delay_ms(5000);
                PORTC=0xFF; // Lang
                _delay_ms(5000);
                PORTC=0x00; // Pause
                _delay_ms(5000);
                PORTC=0xFF; // Lang
                _delay_ms(5000);
                PORTC=0x00; // Pause
                _delay_ms(5000);
                PORTC=0xFF; // Kurz
                _delay_ms(1000);
                PORTC=0x00; // Pause
```

Demo-Video

➔ Ein Demonstrations-Video mit allen Funktionen befindet sich in der ZIP-Datei.

Notiz: Die Helligkeit der Roten LED ist auf dem Video nicht zu erkennen, da selbst die maximale Helligkeit dieser LED sehr schwach ist. Die Änderung ist jedoch an den LEDs an der MEGACARD erkennbar.