H ö h e r e T e c h n i s c h e B u n d e s l e h r a n s t a l t Salzburg

Abteilung für Elektronik

Übungen im
Laboratorium für Elektronik

Protokoll

für DIC Übungen

Gegenstand der Einheit

Pulsweitenmodulation

Name: Max Mustermann

Jahrgang: tbd

Übung am: tbd

Anwesend:	tbd

Inhaltsverzeichnis:

1) Aufgabenstellung
1.1.) Einführung:

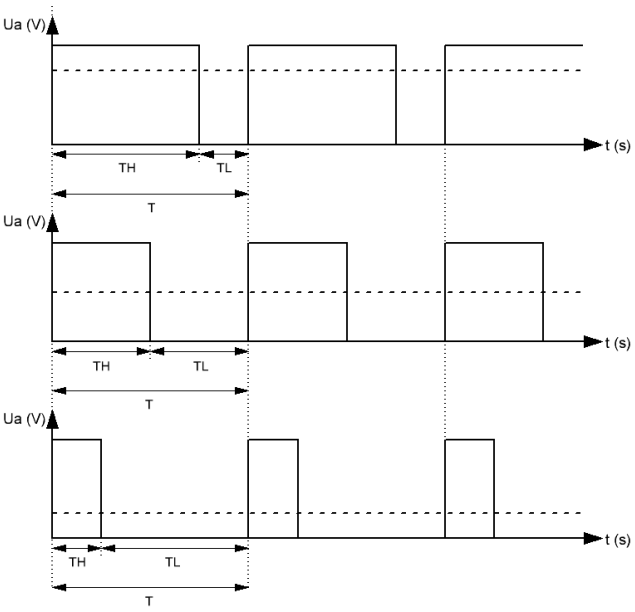

Abbildung 1: Funktionsprinzip PWM

In dieser Übung wurde mithilfe der Pulsweitenmodulation (fast PWM) vom Timer/Counter0 im Atmega644pa eine LED gedimmt. Bei der Pulsweitenmodulation wird an einem gewählten Pin (hier PB3) ein Rechtecksignal mit unterschiedlichen High- und Lowzeiten ausgegeben. Mit der Änderung des H/L Verhältnis kann die durchschnittliche Ausgangsspannung am Pin manipuliert werden, befindet sich nun am PWM Ausgang ¾ der Periode (T) eine 5V Spannung (TH) und ¼ der Periode (T) eine 0V Spannung (TL) beträgt die Ausgangsspannung am Pin ¾ von Ua (5V * 0,75 = 3,75V / Abb.1, oben).
Wie man noch an der Abbildung 1 erkennen kann verursacht eine größere Hightime (TH) am Ausgang (UA) eine größere Spannung und eine größere Lowtime (TL) eine kleinere Spannung (UA). Das im Punkt 3 angeführt Programm ändert die Hightime (TH) kontinuierlich zwischen 0 und T, sodass die LED langsam zwischen dunkel und hell und umgekehrt wechselt.

Um die Übung durchzuführen muss der Timer/Counter 0 im Atmega wie folgt konfiguriert werden:

- Timer Mode 3: Fast PWM mit TOP = 0xFF
- Prescaler auf „No prescaling“ ($T = \frac{größe\ von\ TCNT0}{f_{CPU}} = \frac{256}{16000000Hz} = 16\mu s$)
- PWM Signal Ausgabe auf OC0A (nicht invertiert)
- Aktivieren des Overflow Interrupt zur Aktualisierung der LED Helligkeit, welche ins OCR0A geschrieben wird.

Die Umsetzung wird genauer im Punkt 1.3 beschrieben.

1.2.) Skizze zur Aufgabenstellung:

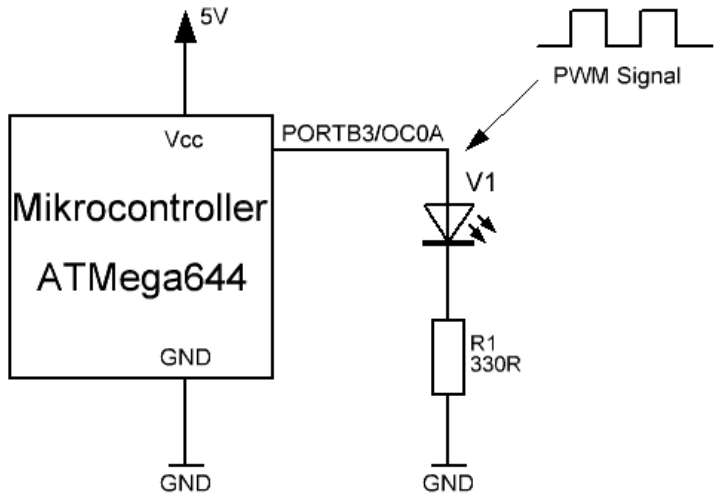

Abbildung 2: Schaltungsaufbau

1.3.) Verwendete Register:

Um den Timer zu konfigurieren ist es zunächst wichtig, die Bits im TCCR0A zu setzten, begonnen wird bei den WGM Bits zum Einstellen des Timermodes.

13.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Abbildung 3: Aufbau des TCCR0A Registers

Wie aus der Tabelle 13-8 entnommen werden kann, muss für den Fast PWM Mode das **WGM01** und **WGM00** Bit auf **1** gesetzt werden.
Praktisch für diese Übung sind auch die Bits COM0A1 und COM0A0, mit ihnen kann das PWM Signal direkt an den Pin PORTB3 ausgeben werden. Setzt man nach der Tabelle 13-3 das Bit **COM0A1** auf **1,** so wird das PWM Signal nichtinvertiert auf den PORTB3 ausgegeben.

13.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Abbildung 4: Aufbau des TCCR0B Registers

Auch von großer Bedeutung ist das TCCR0B Register, hier lässt sich der Prescaler einstellen. Da wir keinen Prescaler verwenden, muss man nach der Tabelle 13-9 nur das **CS00** Bit auf **1** setzten.

13.9.6 TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Abbildung 5: Aufbau des TIMSK0 Registers.

Zum aktivieren des Interrupts wird noch das TIMSK Register benötigt, hier verursacht das auf **1** setzten des **TOIE0** Bits einen Overflow Interrupt sobald das TCNT0 Register den Wert 255 erreicht. Das Setzten des TOIE0 Bits ist wichtig zum Aktualisieren der Helligkeit.

Figure 13-6. Fast PWM Mode, Timing Diagram

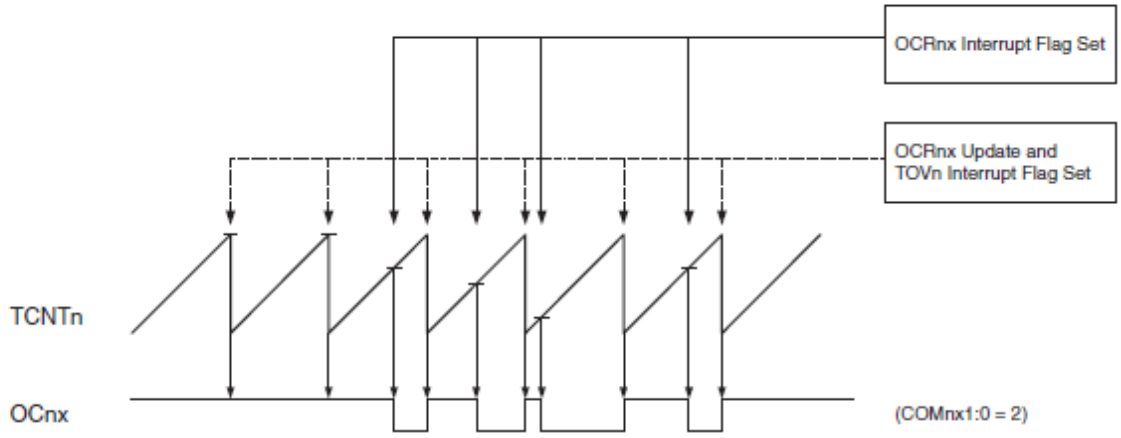

Abbildung 6: Darstellung Fast PWM Funktion (OCnx = OC0A/PORTB3)

Zuletzt ist es noch wichtig das **OCR0A Register** zu behandeln, da der Wert in diesem Register proportional zur Helligkeit der LED am Ausgang ist. Funktionstechnisch zählt das TCNT0 Register immer wieder von 0 bis 255 wo es den Overflow Interrupt auslöst und wieder von vorne beginnt. Dabei ist der Ausgangspin PORTB3 bei TCNT0 < OCR0A auf 5V (TH) und bei TCNT0 >OCR0A auf 0V (TL) beim nichtinvertierten Modus COM0A1:0 = 2 des Zählers. Durch das Verändern des OCR0A Registers, wie es bei dieser Übung in der ISR passiert, werden auch dementsprechend TH und TL verändert.

2) Flussdiagramm:

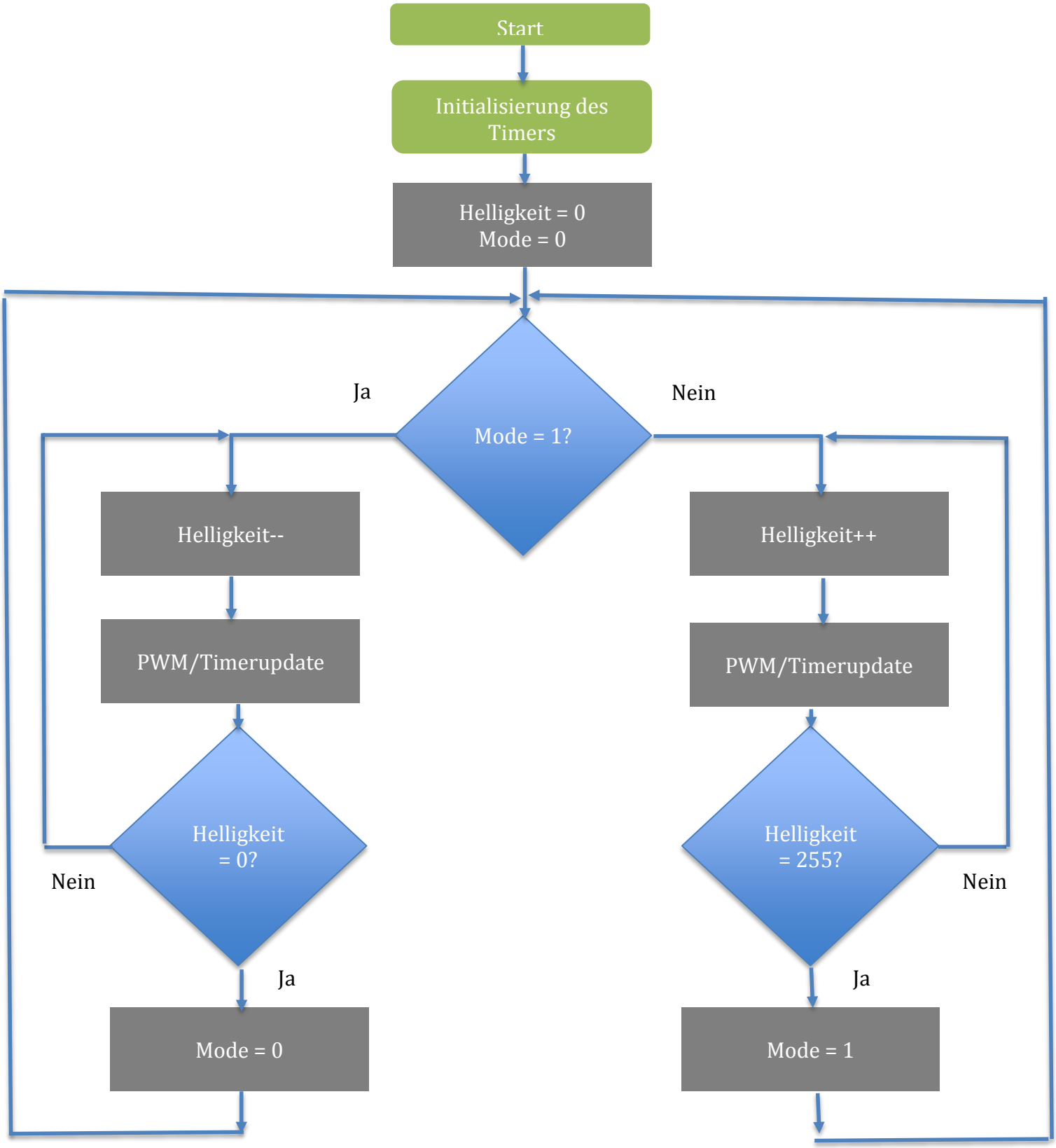

Abbildung 7: Darstellung Flussdiagramm.

3) Programmcode:

Code für die Messung:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t Helligkeit = 0;
int Modus = 0;  //die Variable "Modus" gibt an ob die Helligkeit gerade zu oder abnimmt

void inittimer(){
  TCCR0A |= (1<<WGM01) | (1<<WGM00) | (1<<COM0A1);   //fast PWM + non inverting
  TCCR0B |= (1<<CS00);    //no prescaler
  TIMSK0 |=(1<<TOIE0);   //Interrupt enable
  OCR0A = Helligkeit;    //default Wert wird zugewiesen
}

int main(void)
{
  sei();              //Global Interrupt enable
  DDRB |= (1<<DDB3);  //PWM Pin zum Ansteuern der LED als Ausgang definieren
  inittimer();        //Timer initialisieren
    while(1)
    {
    }
}

ISR(TIMER0_OVF_vect){
        OCR0A = Helligkeit; //Helligkeit wird aktualisiert
                if(Modus==0){      //LED wird langsam immer heller
                        Helligkeit= Helligkeit+5;   //Helligkeit wird erhöht
                        if(Helligkeit>=0xFF){ // Helligkeit ist am Maximum
                                Modus = 1;  //Helligkeit wird nun mit jedem Zyklus reduziert
                        }
                }
                else if(Modus == 1){  //LED wird langsam immer dunkler
                        Helligkeit = Helligkeit-5;       //Helligkeit wird reduziert
                        if(Helligkeit<=0x00){ //Helligkeit ist am Minimum
                                Modus = 0;  //Helligkeit wird nun mit jedem Zyklus erhöht
                        }
                }
}
```

Code zum Ansteuern der LED:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
uint8_t Helligkeit = 0;
int Modus = 0;  //die Variable "Modus" gibt an ob die Helligkeit gerade zu- oder abnimmt

void inittimer(){
  TCCR0A |= (1<<WGM01) | (1<<WGM00) | (1<<COM0A1);   //fast PWM + non inverting
  TCCR0B |= (1<<CS00);    //no prescaler
  TIMSK0 |=(1<<TOIE0);   //Interrupt enable
  OCR0A = Helligkeit;    //default Wert wird zugewiesen
}

int main(void)
{
  sei();              //Global Interrupt enable
  DDRB |= (1<<DDB3);  //PWM Pin zum Ansteuern der LED als Ausgang definieren
  inittimer();        //Timer initialisieren
    while(1)
    {
     _delay_us(20);     //Delay um Blinkgeschwindigkeit zu regulieren
     if(Modus==0){      //LED wird langsam immer heller
        Helligkeit++;   //Helligkeit wird erhöht
        if(Helligkeit==0xFF){ // Helligkeit ist am Maximum
          Modus = 1;  //Helligkeit wird nun mit jedem Zyklus reduziert
        }
      }
      else if(Modus == 1){  //LED wird langsam immer dunkler
        Helligkeit--;       //Helligkeit wird reduziert
        if(Helligkeit==0x00){ //Helligkeit ist am Minimum
          Modus = 0;  //Helligkeit wird nun mit jedem Zyklus erhöht
        }
      }
    }
}

ISR(TIMER0_OVF_vect){
  OCR0A = Helligkeit; //Helligkeit wird aktualisiert
}
```

Kommentar: Der Code für die Messung dient zur Darstellung der PWM Veränderungen mithilfe des Logicanalyzers, hier wurde im Gegensatz zum LED Code eine um einiges geringere Durchlaufdauer (Schritt 2-5 Flussdiagramm) gewählt.

4) Messung:

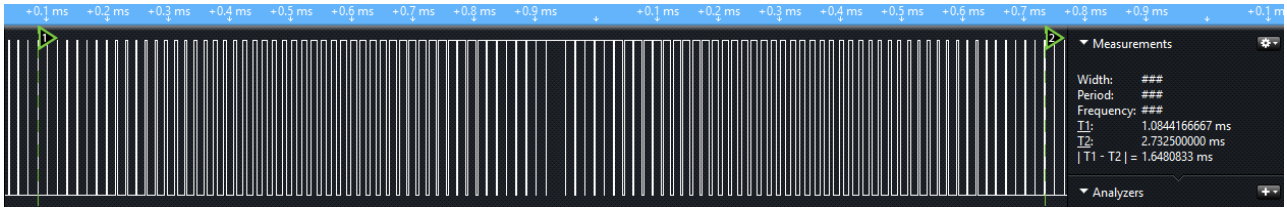

Abbildung 8: Messung einer kompletten Durchlaufes

Kommentar: Die Abbildung zeigt wie zunächst das OCR0A Register einen geringen Wert besitzt (bei T1) welcher Richtung Mitte zunimmt und gegen Ende wieder abnimmt (bei T2). Die Dauer eines kompletten Durchlaufs nimmt 1,64ms in Anspruch, dieser entsteht da wir in der ISR den Helligkeitswert um 5 (Schrittgröße) pro Overflowinterrupt alle 16µs (siehe 1.1/Prescaler) erhöhen.

$(T_{Durchlauf} = 2 * T * \frac{256}{Schrittgröße} = 2 * 16\mu s * \frac{256}{5} = 1638{,}4\mu s)$.

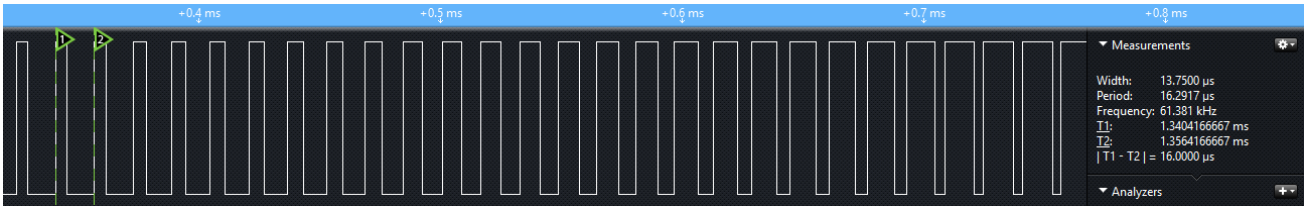

Abbildung 9: Darstellung der PWM-H/L Veränderung

Kommentar: Die Abbildung zeigt wie die Hightime immer mehr zunimmt und die Lowtime proportional abnimmt. Gleichzeitig wurde eine Messung durchgeführt um die Dauer einer Periode (T) zu bestimmen (|T1 – T2|). Sie beträgt 16µs wie bereits bei 1.1.) berechnet wurde.

Salzburg, am 03. März 2016 Unterschrift des Schülers: ……

(Unterschrift des Schülers bestätigt die eigenhändige Ausfertigung)

Datum:	Note:	Unterschrift: