Praktikum Embedded Controller

Gruppe 22
Jia Minggang, Sven Richter

Inhaltsverzeichnis

1 Einführung

1.1 Aufgabenstellung

Die Aufgabe (siehe Abbildung 1) bestand darin mit Hilfe des C166/167 ein Rechtecksignal zu erzeugen das in gewissen Grenzen durch 2 Potentiometer zu regeln ist.

Die beiden Potentiometer sind am Analog-Digitalwandler des C166/167 zu betreiben und damit die Parameter TI und TL des PWM Ausgangs zu variieren.

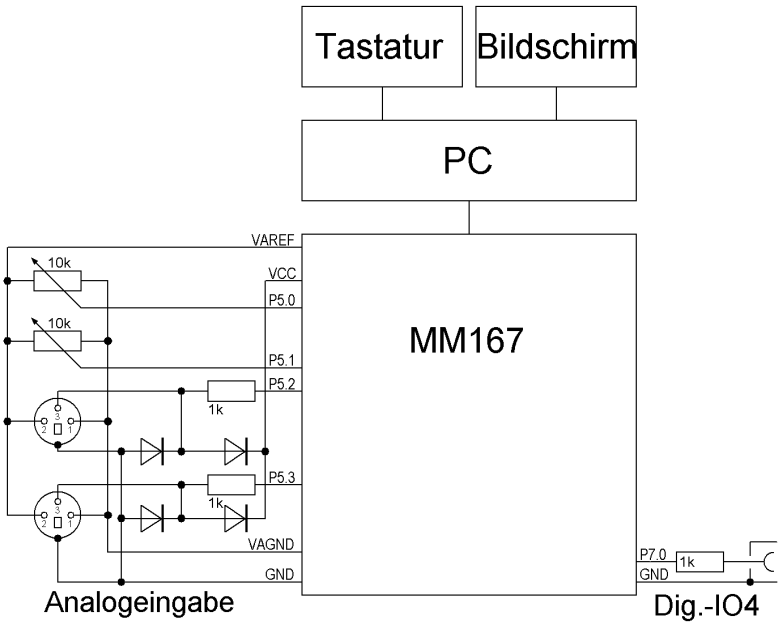
1.2 Realisierungsgedanken

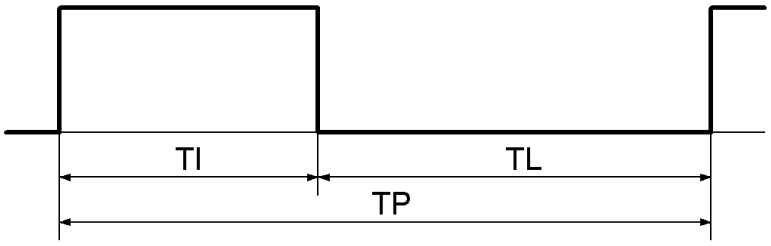
Das herauslesen der Parameter TI und TL sollte permanent erfolgen. Wenn sich einer der beiden Werte ändert werden die Parameter der PWM Einheit neu gesetzt.

Wir haben wie in der Aufgabenstellung vorgeschlagen ausgenutzt das der ADC einen Interrupt auslöst sobald er einen fertig gewandelten Wert im Register stehen hat. Zu diesen Zeitpunkt muss nur der Wert ausgelesen werden dem entsprechenden Port/Parameter zugeordnet werden und wird dann auf Veränderung geprüft.

Bei dieser Überprüfung trat das erste Problem mit dem wir während unserer Analyse nicht gerechnet hatten: auch wenn das Potentiometer nicht verändert wurde errechnete der AD-Wandler einen anderen Wert. Dies war auch nicht weiter überraschend da es zu geringen Störungen kommen kann die dann sich auf den gewandelten Wert auswirken. Wir haben dieses Problem gelöst indem wir nicht mehr auf Gleichheit geprüft haben sondern auf Übereinstimmung innerhalb einer Toleranz von 10.

Ansonsten lief die Implementierung problemlos.





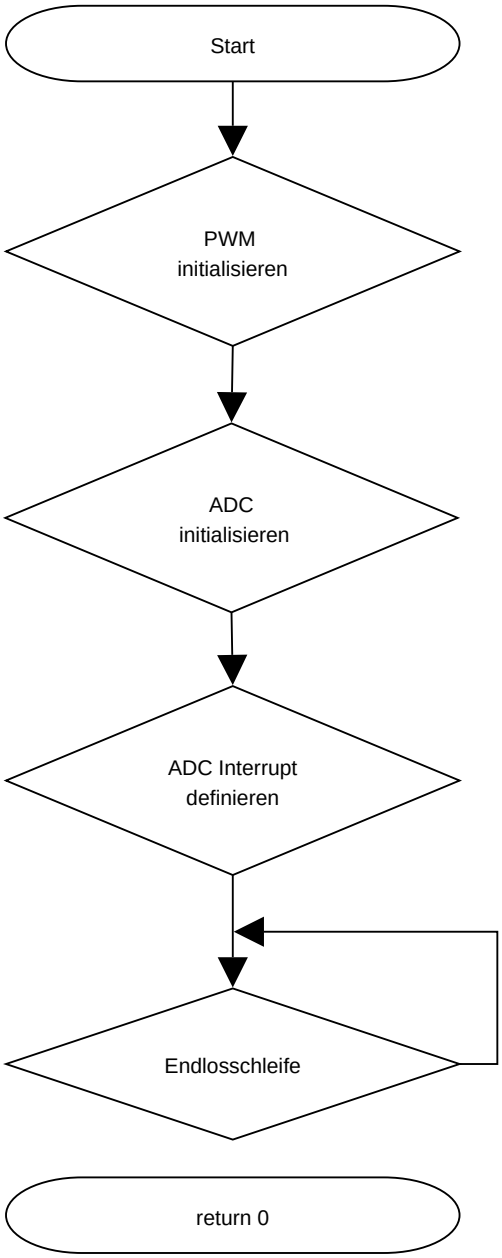
Erzeugung eines PWM-Signales mittels der PWM-Einheit an P7.0

Eingabe der Impulsbreite TI mit dem Potenziometer an den internen ADC (P5.0) im Bereich 10 bis 50 ms.
Eingabe der Impulspause TL mit dem Potenziometer an den internen ADC (P5.1) im Bereich 20 bis 100 ms.
Parameterübergabe: ISR ADCINT

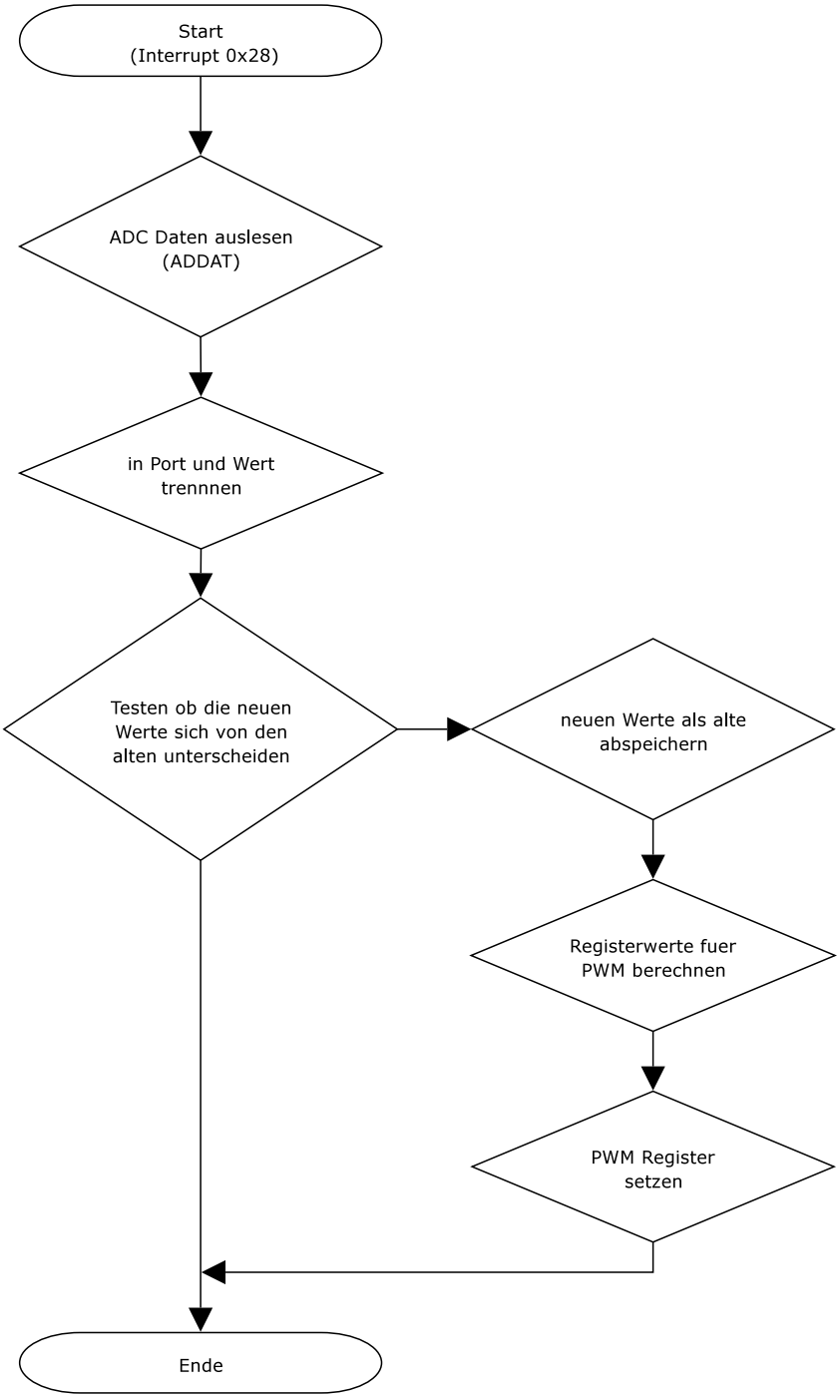
Abbildung 1: Aufgabenstellung

2 PAP

2.1 Hauptprogramm "main()"



2.2 Interruptroutine ”adc_int()”



3 Quelltext - a422.C

```
//Praktikum EC A422
//Bearbeiter Sven Richter, Jia Minggang
//Datum

#include <reg167.h>
#include <stdio.h>


unsigned long adc[2];
unsigned long oldresult;

// Interupt wenn AD Wandlung fertig
void  adc_int() interrupt 0x28 {

  unsigned long Result;   // Aus dem Ergebniss Register des ADC
gelesen
  unsigned long wert;    // das Resultat
  unsigned long a0result, a1result;   // Zwischenspeicher um
Multiplikation
                    // zu sparen

  unsigned char n;    // der Port

  Result = ADDAT;    // rauslesen

  // welcher Port ? entsprechente Ergebnisswert ausrechnen
  if (Result >= 0x1000) {
    n = 1;
    wert = Result - 0x1000 + 10;
  } else {
    n= 0;
    wert = Result + 10;
  }

  /* ueberpruefen ob der aktuell gelesene Wert sich vom
   * alten Wert unterscheidet (mit Toleranz +-10
   */

  if ((adc[n] - 9) > wert || (adc[n] +  9) < wert ) {
    adc[n] = wert;

    //PP0=0xb71b; //TP
    //PW0=0x249f; //TI
```

```
    a0result = (adc[0]-10)*12;
    a1result = (adc[1]-10)*24;
    PW0=a0result+3120;        // TI
    PP0=a0result+a1result+9375;    // TP
   }
}

void main(){

   DP7=1;    // port 7 als Ausgang definieren
  P7=0x1;    // xor am port 7.0 auf 1 setzen
        // (damit mit HIGH Flanke angefangen wird)

  PWMCON1=0x1;  // PWM auf "Edge-Allign-Modus" stellen
  PWMCON0=0x10;   /* PTI0=1*/
  // PWM erst auf ein paar sinnvolle Werte initializieren
  PP0=3120;   // TP
  PW0=9375;  // TI
  // die ADC MERKER entsprechent dem PWM initialisieren
  adc[0]=10:
  adc[1]=10;

  PT0=0x0;  // Timer erstmal auf 0 setzten
  PTR0=0x1;   // Timer starten

  /* ADCH=0x1 start bei channel 1..0
    ADM=0x3 auto scan continous conversion
    ADST=0x1 Start
  */
  ADCON=0xb1;
  ADCIC=0x44;    // Interuptlevel definieren


 while(1){
    ;
  }//end while
}//end main
```