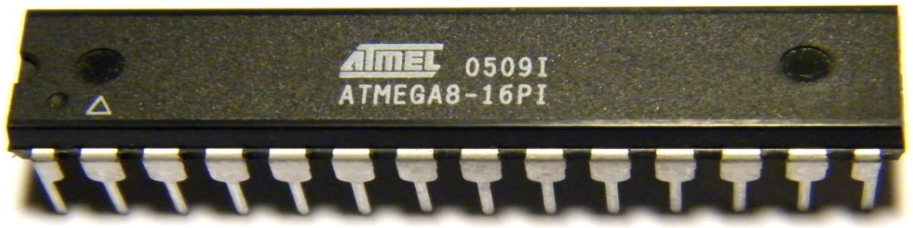
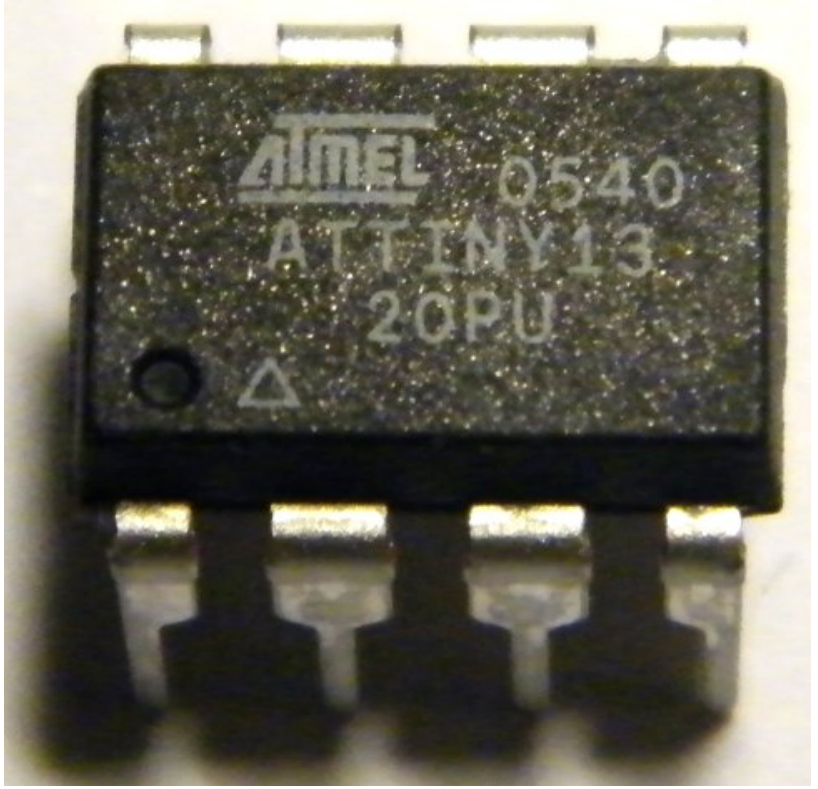
Lösungen zum Kurs "Mikrocontroller Hard- und Software

Gerhard Schmidt
Kastanienallee 20
64289 Darmstadt





http://www.avr-asm-tutorial.net

Lösung Aufgabe 2

- Quellcode:

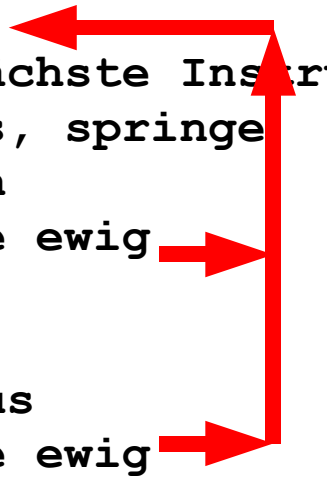
```
; Aufgabe 2
.nolist
.include "tn13def.inc"
.list
;
   sbi DDRB,PB0 ; 2 Takte
loop:
   sbi PORTB,PB0 ; 2 Takte
   cbi PORTB,PB0 ; 2 Takte
   rjmp loop ; 2 Takte
```

- Ausführungszeit:
  2+2+2 = 6 Takte
  6 / 1,2 MHz = 5 µs
- Ausführungsfrequenz:
  1 / 5 µs = 200 kHz
- Pulsweite:
  Ausgang=1: 2 Takt
  Ausgang=0: 4 Takte
  Pulsw.= 2 / 6 = 33%

Lösung Aufgabe 3

- Quellcode:

```
; Lösung von Aufgabe 3
.nolist
.include "tn13def.inc"
.list
;
   sbi DDRB,PB0 ; LED-Ausgangstreiber einschalten
   sbi PORTB,PB4 ; Pull-Up-Widerstand am Tasteneingang an
;
loop:
   sbic PINB,PB4 ; Überspringe nächste Instruktion wenn Taste an
   rjmp Taste_Aus ; Taste ist aus, springe
   cbi PORTB,PB0 ; Schalte LED an
   rjmp loop ; Fertig, wiederhole ewig
;
Taste_Aus:
   sbi PORTB,PB0 ; schalte LED aus
   rjmp loop ; Fertig, wiederhole ewig
```



Lösung Aufgabe 4

- Quellcode:

```
; Aufgabe 4: Tonerzeugung mit Schleife
.nolist
.include "tn13def.inc"
.list
.equ f = 1760 ; Kammerton a'''
.equ c = (600000 / f - 1) / 3 ; Berechne c
;
    sbi DDRB,PB3 ; Lautsprecherausgang an
    sbi PORTB,PB4 ; Pull-Up Taster an
;
loop:
    sbic PINB,PB4 ; Überspringe wenn Taster an
    rjmp loop ; Taste ist aus
;
    sbi PORTB,PB3 ; Lautsprecherausgang high
    ldi R16,c ; Lade Konstante
loop1:
    dec R16 ; zähle abwärts
    brne loop1
    cbi PORTB,PB3 ; Lautsprecherausgang low
    ldi R16,c ; Lade Konstante noch mal
loop2:
    dec R16 ; zähle wieder abwärts
    brne loop2
    rjmp loop ; wieder an den Anfang
```

- Kommentar
- Prozessordefinitionen
- Gewünschte Tonfrequenz
- Berechne Schleifenkonstante
- Inittiere Aus- und Eingänge
- Eingangsabfrage an PB4
- Lautsprecher auf High setzen
- 1. Verzögerungsschleife
- Lautsprecher auf Low setzen
- 2. Verzögerungsschleife
- Und alles wieder auf Anfang

Lösung Aufgabe 5

- Quellcode:

```
; Aufgabe 5: 1 Hz LED
.nolist
.include "tn13def.inc"
.list
;
.equ c23 = 8*65536-2; Verz. der 2.+ 3. Schleife
.equ c1 = (600000-c23-1)/4 ; Verz. für 1.Schleife
;
    sbi DDRB,PB0 ; LED-Ausgang an
    sbi PORTB,PB4 ; Pull-Up Taster an
loop:
    sbic PINB,PB4 ; Überspringe wenn Taster an
    rjmp loop ; Taste ist aus, lasse LED aus
    cbi PORTB,PB0 ; LED-Ausgang an
    ldi ZH,HIGH(c1) ; Lade Konstante
    ldi ZL,LOW(c1)
loop1:
    sbiw ZL,1 ; zähle abwärts
    brne loop1
loop2:
    sbiw ZL,1 ; zähle weiter abwärts
    brne loop2
loop3:
   sbiw ZL,1 ; zähle weiter abwärts
   brne loop3
   sbi PORTB,PB0 ; LED-Ausgang aus
   ldi ZH,HIGH(c1) ; Lade Konstante
   ldi ZL,LOW(c1)
loop4:
    sbiw ZL,1 ; zähle wieder abwärts
    brne loop4
loop5:
    sbiw ZL,1 ; zähle wieder abwärts
    brne loop5
loop6:
    sbiw ZL,1 ; zähle weiter abwärts
    brne loop6
    rjmp loop ; wieder an den Anfang
```

Lösung Aufgabe 6

- Quellcode:

```
; Sirene
.nolist
.include "tn13def.inc"
.list
.equ ton0 = 440 ; niedriger Ton
.equ ton1 = 880 ; hoher Ton
.equ presc = 8  ; Vorteiler
.equ c0 = 1200000 / presc / ton0 / 2
.equ c1 = 1200000 / presc / ton1 / 2
    sbi DDRB,PB0 ; Lautsprecherausgabe
    sbi PORTB,PB4 ; Pull-Up Taster
    ldi R16,c0 ; niedriger Ton in Compare A
    out OCR0A,R16 ; in Compare A
    ldi R16,(1<<COM0A0) | (1<<WGM01)
    out TCCR0A,R16 ; A = CTC und OC0A
    ldi R16,1<<CS01 ; Prescaler = 8
    out TCCR0B,R16
loop:
    sbic PINB,PB4 ; Springe bei Taster
    ldi R16,c0 ; Lade niedrigen Ton
    sbis PINB,PB4 ; Springe ohne Taster
    ldi R16,c1 ; Lade hohen Ton
    out OCR0A,R16 ; an Compare-Register A
    rjmp loop
```

Lösung Aufgabe 7

- Quellcode:

```
; EEPROM-Hochzähler
.nolist
.include "tn13def.inc"
.list
; Konstanten definieren
.equ cEep = 10 ; Zelle im EEPROM-Speicher
; Verzögerung durch Schleife
.equ c002 = (1200000/50-8)/4 ; 0,02 s Verzögerung
.equ c01 = (1200000/10-8)/4 ; 0,1 s Verzögerung
.equ c02 = (1200000/5-8)/4 ; 0,2 s Verzögerung
; Register definieren
.def rmp = R16 ; Multi-Purpose-Register
.def cnt = R17 ; Zähler
.def rdl = R24 ; Verzögerungszähler, LSB
.def rdh = R25 ; dto., MSB
; Initiieren
    ldi rmp,LOW(RAMEND) ; Stapelzeiger
    out SPL,rmp ; setzen
    sbi DDRB,PB0 ; LED-Treiber an
    sbi PORTB,PB0 ; LED aus
    sbi PORTB,PB4 ; Pull-Up Taster an
; Verzögerung um 0,02 s
    ldi rdh,HIGH(c002) ; 0,02 s
    ldi rdl,LOW(c002)
    rcall Delay ; rufe Unterprogramm
; Eingang abfragen
    clr cnt ; Zähler auf Null
    sbis PINB,PB4 ; Frage Taster ab
    rjmp Schreiben ; Schreibe Null in EEPROM
; EEPROM lesen
   ldi rmp,cEep ; Adresse der EEPROM-Zelle in Register
   out EEARL,rmp ; in EEPROM-Adressregister schreiben
   sbi EECR,EERE ; Setze Read Enable Bit
   in cnt,EEDR ; Lese Inhalt der Zelle in das Register
; Zähler um Eins erhöhen
   inc cnt ; increase Zähler
; Zähler in EEPROM schreiben
Schreiben:
   sbic EECR,EEPE ; Control ob Schreiben möglich
   rjmp Schreiben ; Schreiben noch nicht möglich
   ldi rmp,(0<<EEPM1) | (0<<EEPM0) ; Schreibmodus = 00
   out EECR,rmp ; in Kontrollregister
   ldi rmp,cEep ; Adresse der EEPROM-Zelle in Register
   out EEARL,rmp ; in EEPROM-Adressregister schreiben
   out EEDR,cnt ; Zähler in Datenregister
   sbi EECR,EEMPE ; setze Write Enable Bit
   sbi EECR,EEPE ; setze programming Enable
; Teste ob cnt gleich Null (sofort schlafen legen)
   tst cnt ; Vergleiche mit Null
   breq Schlafen ; bei Null gleich schlafen legen
; cnt nicht Null, LED cnt mal an- und ausschalten
LedAn:
   cbi PORTB,PB0 ; Led einschalten
   ldi rdh,HIGH(c01) ; Delay 0,1 s
   ldi rdl,LOW(c01)
   rcall Delay ; Delay ausführen
   sbi PORTB,PB0 ; LED ausschalten
   ldi rdh,HIGH(c02) ; Delay 0,2 s
   ldi rdl,LOW(c02)
   rcall Delay ; Delay ausführen
   dec cnt ; Zähler um Eins abwärts
   brne LedAn ; noch nicht Null, noch mal
```

Lösung Aufgabe 7 – Teil 2

```
; Prozessor ausschalten
Schlafen:
   ldi rmp,(1<<SE) | (1<<SM1) ; Power down mode
   out MCUCR,rmp ; in MCU Kontrollregister
   sleep ; Prozessor ausschalten
;
; Unterprogramm Verzögerungsschleife
Delay: ; LDI=2 Takte, RCALL=3 Takte
   sbiw rdl,1 ; zähle herunter, 2 Takte
   brne Delay ; noch nicht Null, 1 oder 2 Takte
   ret ; Rückkehr, 4 Takte
; Anzahl Takte  n = 5 + 4*(c-1) + 3 + 4
;               n = 4*c + 8
; Zählkonstante c = (n-8) / 4
;
; EEPROM beim Programmieren initiieren
.ESEG ; in das EEPROM-Segment schreiben
.ORG cEep ; Adresse auf Zelle setzen
.DB 0 ; Null in Zelle schreiben
;
```

Nach dem Kompilieren zuerst den Hexcode in das Flash programmieren.

Dann im Programmierfenster unter „EEPROM“ die Hexdatei „eeprom.eep“ auswählen und in das EEPROM schreiben.

Lösung Aufgabe 8

- Quellcode:

```
; ; LED-Dimmer
.nolist
.include "tn13def.inc"
.list
;
    sbi DDRB,PB0 ; LED-Ausgabeport
    clr R16 ; compare A auf LED aus
    out OCR0A,R16
    ldi R16,(1<<COM0A1)|(1<<COM0A0)|(1<<WGM01)|(1<<WGM00)
    out TCCR0A,R16 ; Fast PWM mode, A invertierter Output
    ldi R16,1<<CS00 ; Prescaler = 1
    out TCCR0B,R16 ; Timer ein
    ldi R16,(1<<ADLAR)|(1<<MUX1)|(1<<MUX0)
    out ADMUX,R16 ; links adjust, AD-Kanal PB3
loop:
    ldi R16,(1<<ADEN)|(1<<ADSC)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
    out ADCSRA,R16
Warte:
    sbic ADCSRA,ADSC ; warte bis ADC fertig
    rjmp Warte ; noch nicht fertig
    in R16,ADCH ; lese Ergebnis
    out OCR0A,R16 ; in Compare Register
    rjmp loop ; und immer so weiter
```

Lösung Aufgabe 9 - Teil 1

- Quellcode:

```
; Drei Mal blinken pro Tastendruck, interrupt-gesteuert
.nolist
.include "tn13def.inc"
.list
; Timer: Takt 1,2 MHz, Vorteiler 1024, Overflow bei 256
; Dauer: 1024*256/1,2 = 218.453 µs = 218 ms
; Registerdefinitionen
.def rmp = R16 ; Multipurpose-Register
.def rCnt = R17 ; Blinkzähler-Register
; Reset- und Int-Vektoren
    rjmp start ; Reset-Vektor
    reti ; INT0-Vektor
    rjmp pcint ; PCINT0-Vektor
    rjmp tc0ovf ; TC0-Overflow-Vektor
    reti ; EE_RDY-Vektor
    reti ; ANA_COMP-Vektor
    reti ; TIM0_COMPA-Vektor
    reti ; TIM0_COMPB-Vektor
    reti ; WDT-Vektor
    reti ; ADC-Vektor
pcint:
    sbis PINB,PB4 ; überspringe wenn Taste nicht gedrückt
    ldi rCnt,3 ; Zähler setzen
    reti ; fertig mit Interrupt
```

Lösung Aufgabe 9 – Teil 2

```
; TC0 Overflow interrupt service routine
tc0ovf:
     sbic PORTB,PB0 ; überspringe, wenn LED-Ausgang = 0
     rjmp tc0ovf1 ; LED-Ausgang = 1
     sbi PORTB,PB0 ; schalte LED-Ausgang auf 1
     reti ; fertig, warte auf nächsten Interrupt
tc0ovf1:
     tst rCnt ; Blinkzähler = 0 ?
     brne tc0ovf2 ; nicht Null, noch einmal blinken
     reti ; Null, nicht mehr blinken
tc0ovf2:
     dec rCnt ; Zähler um Eins niedriger setzen
     cbi PORTB,PB0 ; LED anschalten
     reti ; fertig
; Hauptprogramm-Start
Start:
     ldi rmp,LOW(RAMEND) ; init Stapel
     out SPL,rmp
     sbi DDRB,PB0 ; LED-Ausgang einschalten
     sbi PORTB,PB0 ; LED ausschalten
     sbi PORTB,PB4 ; Pullup-Widerstand am Tasteneingang einschalten
     ldi rmp,1<<PCINT4 ; externe Int-Maske an Tasteneingang einschalten
     out PCMSK,rmp
     ldi rmp,1<<PCIE ; externer Int einschalten
     out GIMSK,rmp
     ldi rCnt,3 ; am Anfang drei mal blinken
     ldi rmp,0 ; Timer-Mode auf Normal
     out TCCR0A,rmp
     ldi rmp,(1<<CS02)|(1<<CS00) ; Timer Prescaler auf 1024
     out TCCR0B,rmp
     ldi rmp,1<<TOIE0 ; Timer-Interrupts ermöglichen
     out TIMSK0,rmp
     sei ; Interrupts generell einschalten
Schleife:
     rjmp Schleife ; alles weitere interrupt-gesteuert
```