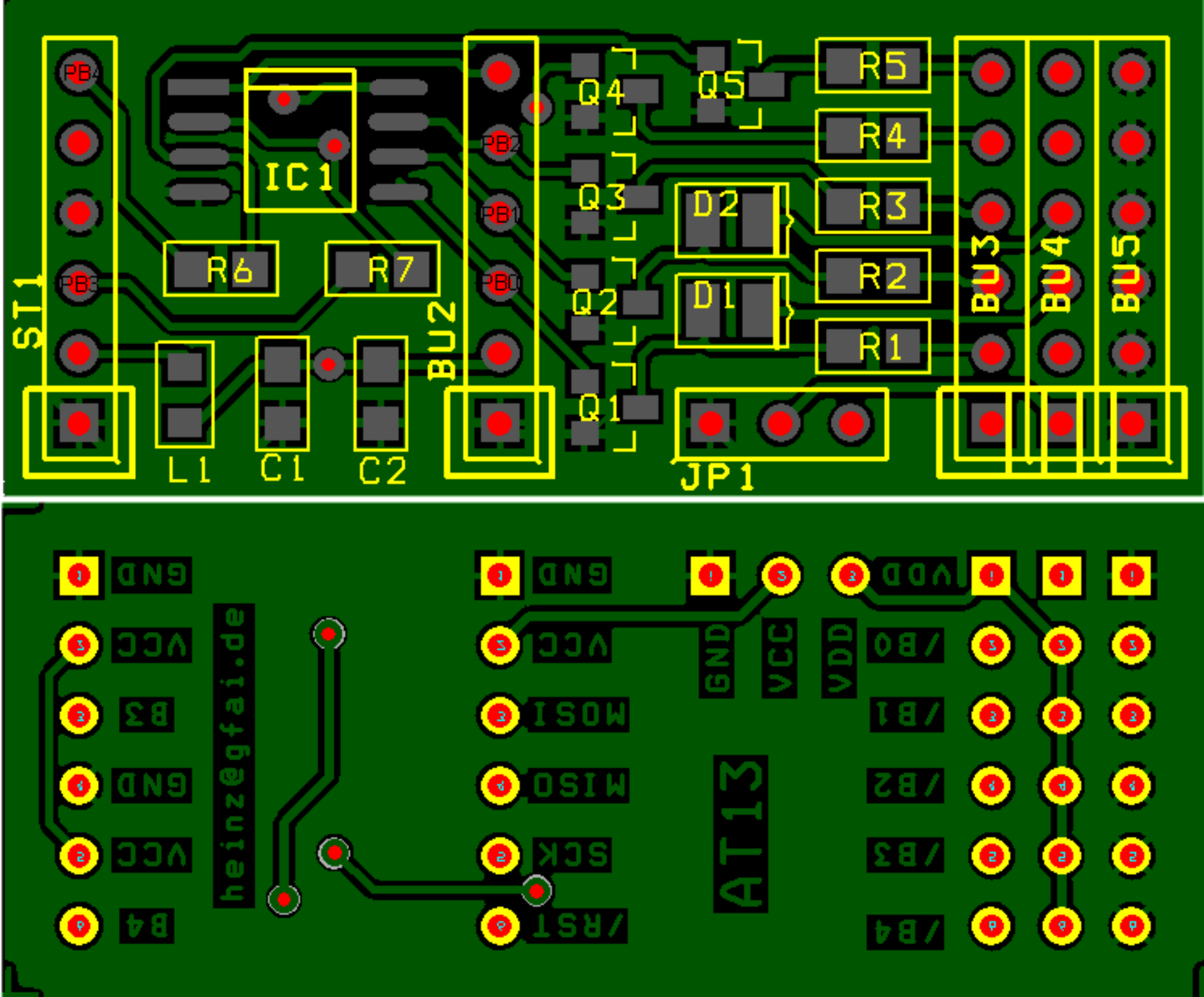
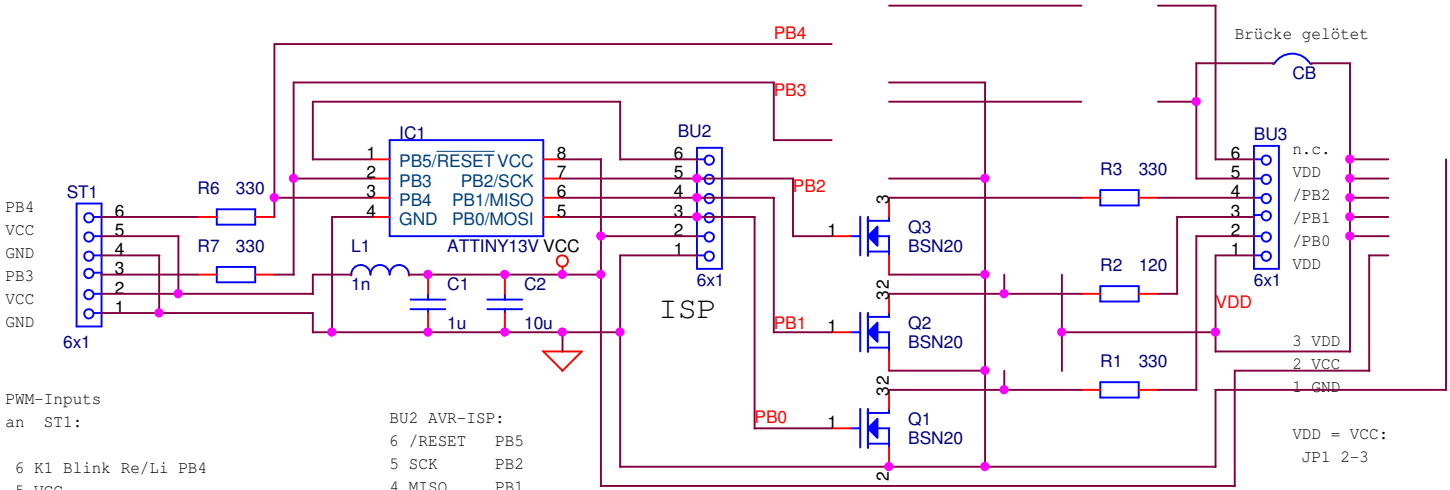

ST1
PB4
PB3
PB2
PB1
PB0
IC1
R6
R7
L1
C1
C2
BU2
Q4
Q3
Q2
Q1
Q5
D2
D1
R5
R4
R3
R2
R1
JP1
BU3
BU4
BU5
GND
VCC
B3
B4
heinz@gfai.de
MOSI
MISO
SCK
/RST
AT13
VDD
/B0
/B1
/B2
/B3
/B4



PWM-Inputs
an ST1:

6 K1 Blink Re/Li PB4
5 VCC
4 GND

3 K2 Licht PB3
2 VCC
1 GND

BU2 AVR-ISP:

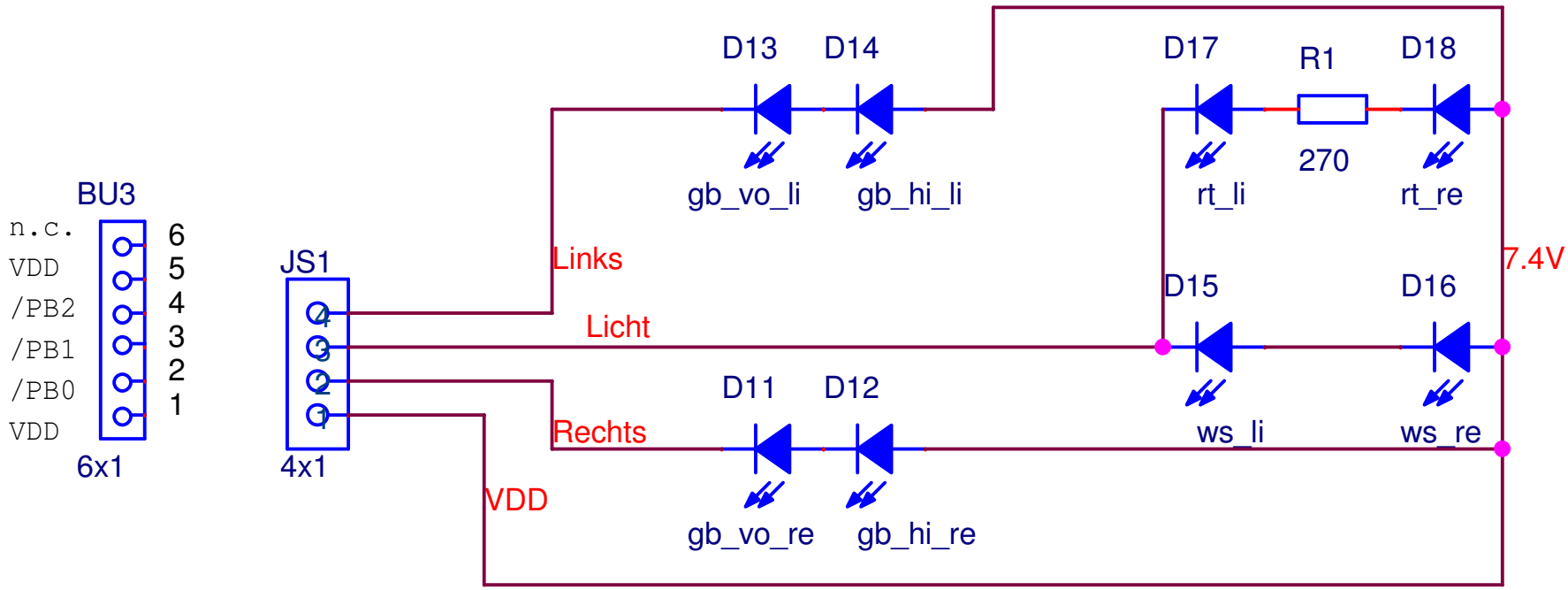
6	/RESET	PB5
5	SCK	PB2
4	MISO	PB1
3	MOSI	PB0
2	VCC	
1	GND	

VDD = VCC:
JP1 2-3

Links- rechts-Vertauschung am LED-Stecker JS1
5Volt-Versorgung über BEC des Empfängers

Q1 bis Q5: BSN20, BSS123, 2N7002, Si2312DS
BSN20: 5V/5V/1A; 5V/1V/0,5A

Title	ATtiny13 Entwicklungsboard AT13 Licht Mini-Cooper	
Size A4	Document Number heinz@gfai.de	Rev 10



Links- rechts-Vertauschung mit Stecker JS1 möglich

Title: LEDs im Mini-Cooper		
Size: A4	Document Number: dr.g.heinz@web.de	Rev: 1

```
;
; Blinkgeber und Lichtschalter für PWM-Signale (Mini Cooper)
; mit Effektlicht-Typen (US-Warnblink)
; mit Mikroprozessor ATtiny13
; für Boards BRBL10 oder AT13
; dr.g.heinz@web.de 10/2008
;
; Eingänge:     PB3 Licht-Kanal (Vor/Rück)
;               PB4 Blink-Kanal (Re/Li)
;
; Pulsbreite: 1,0...2,0 ms, positiver Impuls, Neutrallage: 1,5 ms
; Pulse können nacheinander, aber auch gleichzeitig erscheinen (Tests)
;
; Funktion:
;   Eine halbe Sekunde nach Einschalten der Betriebsspannung
;   wird die Pulsbreite beider Kanäle gemessen:
;   Maß wird als Neutralstellung interpretiert.
;
;   V/R-Knüppelrichtung festlegen: Blink-Kanal abziehen, bei
;   Vollrück Betriebsspannung einschalten bedeutet RESET.
;   Rück-Richtung wird dann im EEPROM abgelegt.
;
;   Rechts/Links Vertauschung Blinker ist an den Ausgängen möglich,
;   dazu LED-Stecker und VDD vertauschen/umdrehen.
;
;   alle Moden werden gesteuert über Bewegung des Servoknüppels.
;
;
; Besonderheit:
;   Streng ablaufgesteuert, alle Pulse kommen nacheinander und werden
;   nacheinander abgearbeitet,
;   Zeitbasis 0,1 ms, d.h. 1ms ~ 10, 2ms ~ 20
;   Messungen relativ zu anfangs gemessener Neutralstellung
;
; Betriebsmoden:
;
;   PB3 Licht-Kanal (Vor/Rück): (i.a. Zusatzkanal)
;        siehe UP "Licht"
;       <1,2 ms Licht aus
;       >1,2 ms Licht Warnblink
;       >1,4 ms Licht Dauer
;       >1,6 ms Licht Flashblink
;       >1,8 ms Licht PacecarBlink
;
;       Umpolen der Stellrichtung Licht Licht-Kanal:
;       Blink-Kanal abziehen, mit Vollrück einschalten
;       Wert wird als Rückrichtung interpretiert
;
;   PB4 Blink-Kanal (Blinker Re/Li):
;       Blink R solange gedrückt
;       Blink L solange gedrückt
;
;       Umpolen der Stellrichtung Blink-Kanal: LED-Stecker wenden
;
; Ports:        PB0 (Pin5) Ausgabe Blinken re
;               PB1 (Pin6) Ausgabe Licht
;               PB2 (Pin7) Ausgabe Blinken li
;               PB3 (Pin2) Eingang Licht-Kanal
;               PB4 (Pin3) Eingang Blink-Kanal
; Buchsen:
;           BU3: LEDs Output
;               6   ---
;               5   VDD 7,4V        --- Akku (rot)
;               4   PB2 Blink re    \
;               3   PB1 Licht        |  LEDs
;               2   PB0 Blink li     |
;               1   VDD 7,4V        /
;
;           ST1: Servo Inputs
;               6   PB4 Blink   \
;               5   VCC +5V      |  Blink Blink-Kanal
;               4   GND         /
;               3   PB3 Licht   \
```

```
;               2   VCC +5V      |  Licht Licht-Kanal
;               1   GND          /
;
;           BU2: ISP-MKII
;               1 GND
;               2 VCC
;               3 MOSI      ~ PB0
;               4 MISO      ~ PB1
;               5 SCK       ~ PB2
;               6 /RESET    ~ PB5
;
;############### Prozessor ###############

.include "tn13def.inc"  ; ATMEL ATtiny13

;############### Adressen ################

rjmp RESET  ;


;   0x0000 rjmp RESET ; Reset Handler
;   0x0001 rjmp EXT_INT0 ; IRQ0 Handler
;   0x0002 rjmp PCINT0 ; PCINT0 Handler
;   0x0003 rjmp TIM0_OVF ; Timer0 Overflow Handler
;   0x0004 rjmp EE_RDY ; EEPROM Ready Handler
;   0x0005 rjmp ANA_COMP ; Analog Comparator Handler
;   0x0006 rjmp TIM0_COMPA ; Timer0 CompareA Handler
;   0x0007 rjmp TIM0_COMPB ; Timer0 CompareB Handler
;   0x0008 rjmp WATCHDOG ; Watchdog Interrupt Handler
;   0x0009 rjmp ADC ; ADC Conversion Handler
;
;   0x000A RESET: ldi r16, low(RAMEND); Main program start
;   0x000B out SPL,r16 ; Set Stack Pointer to top of RAM
;   0x000C sei ; Enable interrupts
;.org 0x000F    ; TIM0_OVF z.B. für Timer-Overflow freihalten, Tab.9-1, S.44
; rcall TIM0_OVF    ; Interrupt-Timer starten


RESET:
    ldi r16,low(RAMEND)     ; Main program start
    out SPL,r16             ; Set Stack Pointer to top of RAM
    ; sei                   ; Enable interrupts



;############### Namen ###################

; Register dynamic:
.def    a1      = r16   ; Albeit
.def    a2      = r17   ; Albeit
.def    a3      = r18   ; Albeit

; Register static
.def    modreg  = r19   ; Anz. der Tastendrücke
.def    delc1   = r20   ; Delay Licht-Kanal Vorzyklus i-1
.def    delc2   = r21   ; Delay Blink-Kanal Vorzyklus i-1
.def    prescal = r22   ; Teilerfaktor Prescaler
.def    leds    = r23   ; aktuelle Ausgabe an LEDs
.def    wait1   = r24   ; Wartezeit Zyklus 1
.def    wait2   = r25   ; Wartezeit Zyklus 2
.def    dela1   = r26   ; Delay Licht-Kanal aktuell i
.def    dela2   = r27   ; Delay Blink-Kanal aktuell i
.def    delk1   = r28   ; Delay Licht-Kanal Neutralstellung
.def    delk2   = r29   ; Delay Blink-Kanal Neutralstellung
.def    delb1   = r30   ; Delay Licht-Kanal Vorzyklus i-1
.def    delb2   = r31   ; Delay Blink-Kanal Vorzyklus i-1

;#################### Inits ######################

    ldi wait1,130       ; Schleifenzähler Licht (1...255) -> Blink- Pause1
    ldi wait2,255       ; Schleifenzähler Blink (1...255) -> Blink- Pause2
    ldi prescal,3       ; Teilerfaktor 011 -> /64 bei Takt 9,6 MHz/8 = 1,2 MHz
    ldi a1,7            ; 0000 0111     PB3,PB4 Eingang; PB0,PB1,PB2 Ausgang
```

```
    out ddrb,a1         ; (Datenrichtung 1 ist Ausgabe)

;#######################################################
;###################### Programm #######################
;#######################################################

    rcall Allein        ; alle LED ein xx00 0111
    rcall Pause2        ; erstmal warten bis Powerup stabil ist
    rcall Pause2        ; erstmal warten bis Powerup stabil ist
    rcall Pause2        ; erstmal warten bis Powerup stabil ist
    rcall Pause2        ; erstmal warten bis Powerup stabil ist
    rcall Pause2        ; erstmal warten bis Powerup stabil ist
    rcall Allaus        ; alle Lichter aus xx00 0000
    rcall Pause2        ;

;############### Ist Blink-Kanal gesteckt? ####################

; überprüfen, ob Stecker Blink-Kanal (r/l) PB4 gezogen wurde.
; Wenn nicht gesteckt, dann Richtungserkennung Licht,
; dazu Pullup für PB4 setzen und warten
    rcall Pullein   ; Pullups PB3, PB4 einschalten

; zweimal prüfen, ob Blink-Kanal = high mit Pause größer als max. Pulslänge
; nur wenn beidemale Blink-Kanal = high, dann ist Blink-Kanal-Stecker nicht gesteckt.

    rcall Warnb     ; Test: Erster Warnblink
    sbis pinb,4     ; Pinb4 (PB4) abfragen, wenn 1, dann nächsten Befehl überspringen
    rjmp Steckt     ; Pinb4 war 0: PB4-Stecker ist gesteckt -> normal weiter
    rcall Pause2    ; warten, falls das gerade ein Puls war
    rcall Pause2    ; warten, falls das gerade ein Puls war
    rcall Pause2    ; warten, falls das gerade ein Puls war
    rcall Pause2    ; warten, falls das gerade ein Puls war
    rcall Pause2    ; warten, falls das gerade ein Puls war

    rcall Warnb     ; Test: Zweiter Warnblink
    sbis pinb,4     ; Pinb4 (PB4) abfragen, wenn 1, dann nächsten Befehl überspringen
    rjmp Steckt     ; Pinb4 war 0: PB4-Stecker ist gesteckt -> normal weiter

    rcall Warnb     ; Test: Dritter Warnblink
    ; zweimal war PB4 = Blink-Kanal = 1 --> Stecker ist nicht gesteckt

;############ Messung Delay der Rückrichtung Licht-Kanal #############

    ; erwartet wird hier, daß der Rückhebel gezogen ist vor power-up
    ; und daß der PB4-Stecker nicht gesteckt ist
    rcall Flank1    ; Flankendetektion von Licht-Kanal, warte auf HL-Flanke
    rcall Messk1    ; Pulsbreite von Licht-Kanal bei voll zurück messen
    in  delb1,TCNT0 ; Pulsbreite in Register delb1 schreiben
    rcall Wrflash   ; Delay in Flash speichern
    ldi delb1,5     ; zum Test delb1 verändern
    rcall Rdflash   ; Delay aus Flash zurücklesen (Test)
    mov delk1,delb1 ; kopieren auf delk1
    rcall Test1     ; Testausgabe - Licht blinkt halb so oft, wie Wert in delk1
    rcall Pause2    ;   1ms = 10; 1,5ms = 15; 2ms = 20 Blinker (1 blink = 0,1 ms)
    rcall Halt      ; Dauerleuchtend verweilen
    ; --> Ende der Messung, externes Reset wird erwartet


;############### Normalbetrieb ########################
;############### Neutrallage bestimmen #################

Steckt:     ; Blink-Kanal-Stecker ist als gesteckt erkannt

    rcall Brbl      ; Erster Lichtblink
    rcall Brbl      ; Zweiter Lichtblink

    ; Neutrallage Licht-Kanal messen, Licht-Kanal v/r PB3
    ; auf Licht-Kanal -> LH warten, dazu
    rcall Flank1    ; Flankendetektion von Licht-Kanal
    ; Meßende bei HL-Flanke
    rcall Messk1    ; Neutrallage von Licht-Kanal messen
    in  delk1,TCNT0 ; Zeit der Neutralstellung Licht-Kanal in Register delk1 schreiben
```

```
    ; Neutrallage Blink-Kanal messen, Blink-Kanal r/l PB4
    ; auf Blink-Kanal -> LH warten
    rcall Flank2    ; Flankendetektion von Blink-Kanal
    ; Meßende bei HL-Flanke
    rcall Messk2    ; Neutrallage von Blink-Kanal messen
    in  delk2,TCNT0 ; Zeit der Neutralstellung PB4 in Register delk2 schreiben

    ; Testausgaben Licht-Kanal und Blink-Kanal
    ;rcall Testall  ; Testausgabe delk1 auf PB0,PB1,PB2 (alle blinken)
    ;rcall Test1    ; Testausgabe delk1 auf PB1 (Licht blinken)
    ;rcall Test2    ; Testausgabe delk2 auf PB0,PB2 (blinker blinken)

;############### Blink- und Licht #################

    rcall Warnb     ; Warnblink zu Beginn

Main:   ; Hauptschleife

    ; Licht an PB3 messen:
    rcall Flank1    ; Warte auf nächste HL-Flanke an PB3 Licht-Kanal
    rcall Messk1    ; Licht-Kanal messen
    in dela1,TCNT0  ; Meßwert in Arbeitsregister delay-a1 ablegen

    ; Blinken an PB4 messen:
    rcall Flank2    ; Warte auf nächste HL-Flanke an PB4 Blink-Kanal
    rcall Messk2    ; Blink-Kanal messen
    in dela2,TCNT0  ; Meßwert in Arbeitsregister delay-a2 ablegen

    ; Ausgabe an Blinklichter PB0/PB2:
    rcall Blon      ; Blinklicht einschalten
    rcall Pause2
    rcall Bloff     ; Blinklicht ausschalten
    rcall Pause2

    ; Ausgabe Licht an PB1:
    rcall Licht     ; Licht ein/ausschalten prüfen

rjmp Main   ; hier endlos verweilen


;#########################################################
;#################### Unterprogramme #####################
;#########################################################


Wrflash:    ; delb1 im Flash speichern
    ; Pulsbreite von Licht-Kanal bei voll zurück speichern
    ; Wait for completion of previous write
    EEPROM_write:
    sbic EECR,EEPE
    rjmp EEPROM_write

    ; Set Programming mode
    ldi a1, (0<<EEPM1)|(0<<EEPM0)   ; xx00 xxxx erase and write as one op. (atomic)
    out EECR,a1     ; (I/O-Register --> out statt mov)

    ; Write address to address register EEARL
    ldi a1,15
    out EEARL,a1

    ; Write data to data register EEDR
    out EEDR,delb1

    ; Write logical one to EEMPE (alt EEMWE = 2)
    sbi EECR,EEMPE  ; set bit in I/O-register

    ; Start eeprom write by setting EEPE (alt EEWE = 1)
    sbi EECR,EEPE
ret


Rdflash:    ; und als delk1 aus Flash lesen
    ; Wait for completion of previous write
```

```
    EEPROM_read:
    sbic EECR,EEPE  ; EEPE alt EEWE (1)
    rjmp EEPROM_read

    ; Set up address in address register:
    ldi a1,15
    out EEARL, a1

    ; Start eeprom read by writing EERE (0)
    sbi EECR,EERE

    ; move data to register delb1
    in delb1,EEDR
ret


Halt:
    rcall Brei      ; Licht ein, verweilen (ext. Reset wird erwartet)
    rjmp  Halt      ; Prozeß anhalten -> fertig, Rückstellung gefunden
ret


Pullein:
    ldi a1,$18      ; 0001 1000     Pullups an PB3,PB4 einschalten
    out portb,a1    ; Pullup init (jeweiliges ddrb muß null sein)
ret


Pullaus:
    ldi a1,$00      ; 0000 0000     Pullups an PB3,PB4 ausschalten
    out portb,a1    ; Pullup init (jeweiliges ddrb muß null sein)
ret


Pause1:
    mov a2,wait1
    pause1two:
        mov a1,wait1
        pause1one:
        dec a1
        cpi a1,0
        brne pause1one
    dec a2
    cpi a2,0
    brne pause1two
ret


Pause2:
    mov a2,wait2
    pause2two:
        mov a1,wait2
        pause2one:
        dec a1
        cpi a1,0
        brne pause2one
    dec a2
    cpi a2,0
    brne pause2two
ret


Flank1:         ; Warte auf Puls an Licht-Kanal
    ; LH-Flankendetektion an Licht-Kanal
    ; Warten, bis Licht-Kanal erste L/H-Flanke erhält; Abfrage v/r PB3

    flank1eins:     ; warten, bis Licht-Kanal null ist (falls gerade auf 1)
    sbic    pinb,3  ; Pinb3 (Licht-Kanal) abfragen: weiter bei 0
    rjmp    flank1eins  ; bei 1 in Schleife verweilen

    flank1null: ; warten, bis Licht-Kanal auf high geht
    sbis    pinb,3  ; Pinb3 (Licht-Kanal) abfragen, weiter bei 1
    rjmp    flank1null  ; bei 0 in Schleife verweilen
```

```
    ; Flanke Licht-Kanal 0 -> 1 detektiert
ret


Flank2:         ; Warte auf Puls an Blink-Kanal
    ; LH-Flankendetektion an Blink-Kanal
    ; Warten, bis PB4 erste L/H-Flanke erhält; Abfrage r/l PB4

    flank2eins:     ; warten, bis PB4 null ist (falls gerade auf 1)
    sbic    pinb,4  ; Pinb4 (PB4) abfragen: weiter bei 0
    rjmp    flank2eins  ; bei 1 in Schleife verweilen

    flank2null: ; warten, bis Licht-Kanal auf high geht
    sbis    pinb,4  ; Pinb4 (PB4) abfragen, weiter bei 1
    rjmp    flank2null  ; bei 0 in Schleife verweilen
    ; Flanke PB4 0 -> 1 detektiert
ret


Messk1:         ;  Messe Dauer von Puls auf Licht-Kanal
    ; Timer-Init Licht-Kanal v/r PB3
    ldi a1,$00      ; Timer rücksetzen
    out TCNT0,a1    ; TCNT0 rücksetzen, siehe S.73 ATtiny13

    ; Vorteiler einstellen   siehe Tab.11-9, S.73: clk/64
    out TCCR0B,prescal  ; TCCR0B mit prescal laden
    ; ldi prescal,3     ; Teilerfaktor 011 -> /64 bei Takt 9,6 MHz/8 = 1,2 MHz
    ; Init Steuerregister -> Start Timer (Signal CS02)

    ; warte auf Licht-Kanal -> low:
    messk1null:
    sbic    pinb,3  ; Pinb3 (Licht-Kanal) abfragen, warte auf Pulsende HL
    rjmp    messk1null  ; bei 1 in Schleife verweilen
    ldi a1,$00      ; Timer 0 stop
    out TCCR0B,a1   ; Steuerregister löschen -> Stop Timer
    ; gemessene Zeit steht in TCNT0
ret


Messk2:         ; Messe Dauer von Puls auf Blink-Kanal
    ; Timer-Init Blink-Kanal r/l PB4
    ldi a1,$00      ; Timer löschen
    out TCNT0,a1
    ; Vorteiler einstellen
    out TCCR0B,prescal  ; Init Steuerregister -> Start Timer (Signal CS02)
    ; warte auf Licht-Kanal -> low
    messk2null:
    sbic    pinb,4  ; Pinb4 (PB4) abfragen, warte auf Pulsende HL
    rjmp    messk2null  ; in Schleife verweilen
    ldi a1,0        ; Timer 0 stop
    out TCCR0B,a1   ; Steuerregister löschen -> Stop Timer
ret


Testall:        ; Wert im Register delk1 anzeigen
    ; Testausgabe delk1 auf PB0, PB1 und PB2
    ; so oft blinken, wie Wert im Register
    mov a1,delk1
    testallnoch:
    ldi a2,$07      ; 0000 0111 PB0-3 auf 1
    out portb,a2    ; Eins ausgeben
    rcall Pause2
    ldi a2,$00      ; 0000 0000 PBs auf 0
    out portb,a2    ; Null ausgeben
    rcall Pause2
    dec a1
    cpi a1,0        ; ist a1 schon auf Null?
    brge testallnoch        ; weiterblinken
ret


Test1:      ; Wert im Register delk1 anzeigen
    ; Testausgabe delk1 auf PB1
```

```
    ; so oft blinken, wie Wert im Register delk1
    mov a3,delk1
    test1noch:
    ldi a2,$02      ; 0000 0010 PB0 auf 1
    out portb,a2    ; Eins ausgeben
    rcall Pause2
    ldi a2,$00      ; 0000 0000 PB0 auf 0
    out portb,a2    ; Null ausgeben
    rcall Pause2
    dec a3          ; decrement
    dec a3          ; zweimal, um Skala "1 blink = 0,1 ms" festzulegen
    cpi a3,0        ; ist a3 schon kleiner gleich Null?
    brge test1noch      ; weiterblinken, wenn größer
ret


Test2:      ; Wert im Register delk2 anzeigen
    ; Testausgabe delk2 auf PB0, PB2
    ; so oft blinken, wie Wert im Register
    mov a3,delk2
    test2noch:
    ldi a2,$05      ; 0000 0101 PB2 auf 1
    out portb,a2    ; Eins ausgeben
    rcall Pause2
    ldi a2,$00      ; 0000 0000 PB2 auf 0
    out portb,a2    ; Null ausgeben
    rcall Pause2
    dec a3
    cpi a3,0        ; ist a3 schon auf Null?
    brge test2noch      ; weiterblinken
ret


Licht:  ; Lichtmode ermitteln
                    ; dela1: aktueller Meßwert
                    ; delb1: Rückwert aus EEPROM
                    ; delk1: Neutralstellung (beim Einschalten gemessen)
    ; aus EEPROM lesen
    rcall Rdflash   ; lese Wert delb1 aus EEPROM
    cp delb1,delk1  ; vergleiche Neutralstellung delk1 mit EEPROM delb1
    brlo licht_norm     ; wenn EEPROM kleiner Neutral, Lichtrichtung o.k.

   licht_inv:   ; wenn EEPROM größer Neutral
    ; Meßwert oben/unten vertauschen:
    mov a2, delk1   ; Zweifachen Mittelwert (Neutralstellung)
    lsl a2          ; auf a2 parken
    sbc dela1,a2    ; von Meßwert 2* Mittelwert abziehen (mit Vorzeichen) -> negativ
    neg dela1       ; Meßwert wieder positiv machen mit Zweierkomplement
                    ; z.B.: aus 10 wurde 20, aus 20 wurde 10, aus 15 wurde 15

   licht_norm:          ; Meßwert normieren
    ldi a1,15           ; Neuralwert Sollvorgabe
    mov modreg,dela1    ; aktuellen Meßwert laden
    add modreg,a1       ; zum Meßwert Soll addieren
    sub modreg,delk1    ; Neutralwert abziehen
    ; Die Neutralstellung sollte jetzt immer 15 = 1,5 ms sein

    ; Vergleich
    rcall Lioff ; Init

    cpi modreg,18   ; 1,8 ms Licht Pacecar
    brlo lie1
    rcall Gocar
    ret

    lie1:
    cpi modreg,16   ; 1,6 ms Licht Flashblink
    brlo lie2
    rcall Flabli
    ret

    lie2:
    cpi modreg,14   ; 1,4 ms Licht einschalten
```

```
    brlo lie3
    rcall Lion
    ret

    lie3:
    cpi modreg,12   ; 1,2 ms Licht Warnblink
    brlo lie4
    rcall Warnb
    ret

    lie4:
    rcall Lioff
ret


Lion:       ; PB1 ein
    sbr leds,(1<<PB1)   ; xxxx xx1x PB1 Licht einschalten
    out portb,leds
ret


Lioff:      ; PB1 aus
    com leds            ; leds negieren
    sbr leds,(1<<PB1)   ; xxxx xx0x PB1=1 ausschalten
    com leds            ; leds negieren
    out portb,leds      ; an Port übertragen
ret


Flabli:     ; Flashblink
    rcall Brbl
    rcall Brbl
    rcall Brbl
    rcall Brbl
ret


Gocar:      ; Pacemaker blink
    rcall Allbl
    rcall Warnb
    rcall Warnb
    rcall Brbl
    rcall Brbl
    rcall Brbl  ; nach jedem Blinkzyklus wird wieder Kanal abgefragt
ret


Blon:   ; Blinken ein
    ; Blinken rechts auf PB0; Blinken links auf PB2
    ; Ausschalten für i-1,i,i+1:
    ; i               Neutralstellung i testen
    cp delk2,dela2  ; Meßwert mit Neutralstellung vergleichen
    breq endb       ; if not equal: gehe zu Marke
    ; i+1
    mov delb2,dela2 ; duplizieren
    inc delb2       ; i+1 testen
    cp delk2,delb2  ; Meßwert mit Neutralstellung vergleichen
    breq endb       ; if not equal: gehe zu Marke
    ; i-1
    mov delb2,dela2 ; duplizieren
    dec delb2       ; i-1 testen
    cp delk2,delb2  ; Meßwert mit Neutralstellung vergleichen
    breq endb       ; if not equal: gehe zu Marke

    ; Blinkwert testen
    mov delb2,dela2 ; duplizieren
    dec delb2       ; i-1
    cp delk2,delb2  ; Meßwerte kleiner i-1 mit Neutralstellung vergleichen
    brlo blre       ; wenn kleiner i-1, dann rechts blinken, sonst links

    sbr leds,(1<<PB2)       ; xxxx x1xx PB2 ein
    out portb,leds  ; an Port übertragen
    ret             ; PB2 ein (L)
```

```
    ; Rest muß rechts sein
    blre:           ; rechts einschalten
        sbr leds,(1<<PB0)   ; xxxx xxx1 PB0 ein
        out portb,leds  ; PB0 ein (R)
    endb:
ret


Bloff:      ; Blinken aus, PB0 und PB2 ausschalten
    com leds                    ; negieren
    sbr leds,(1<<PB0)+(1<<PB2)  ; xxxx x0x0 PB0=0 und PB2=2 aus
    com leds                    ; negieren
    out portb,leds  ; an Port übertragen
ret             ; aus


Brbl:   ; Licht-LEDs blinken PB1
    rcall Brei  ; Licht ein
    rcall Pause1    ; kurz
    rcall Brau  ; Licht aus
    rcall Pause1
ret


Brei:   ; Licht ein PB1
    sbr leds,(1<<PB1)       ; xxxx xx1x PB1 ein
    out portb,leds  ; PB1 ein
ret


Brau:   ; Licht aus PB1
    com leds                ; negieren
    sbr leds,(1<<PB1)       ; xxxx xx0x PB1 aus
    com leds                ; negieren
    out portb,leds  ; PB1 ein
ret


Warnb:          ; Warnblink zu Beginn
    rcall Blei      ; Blinklicht einschalten
    rcall Pause1
    rcall Bloff     ; Blinklicht ausschalten
    rcall Pause1
ret


Blei:           ; PB0 und PB2 einschalten
    sbr leds,(1<<PB0)+(1<<PB2)  ; xxxx x1x1 PB0=0 und PB2=2 ein
    out portb,leds              ; an Port übertragen
ret


Allbl:
    rcall Allein
    rcall Pause1
    rcall Allaus
    rcall Pause1
ret


Allein:
    ldi leds,$07        ; alle Lichter ein xx00 0111
    out portb,leds      ; Portausgabe
ret


Allaus:
    ldi leds,0          ; alle Lichter aus xx00 0000
    out portb,leds      ; Portausgabe
ret
```

```
rjmp Reset          ; sicher ist sicher
```