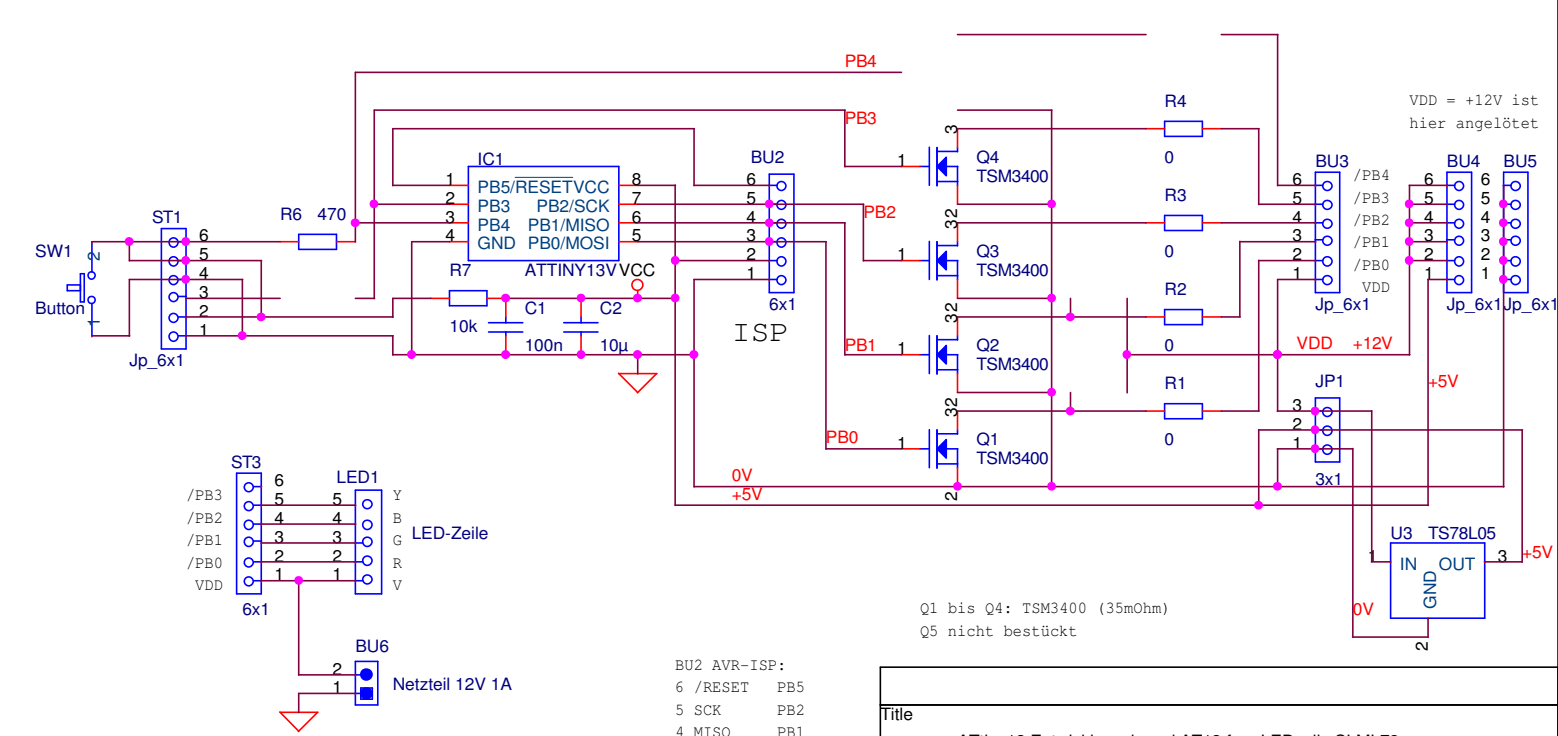
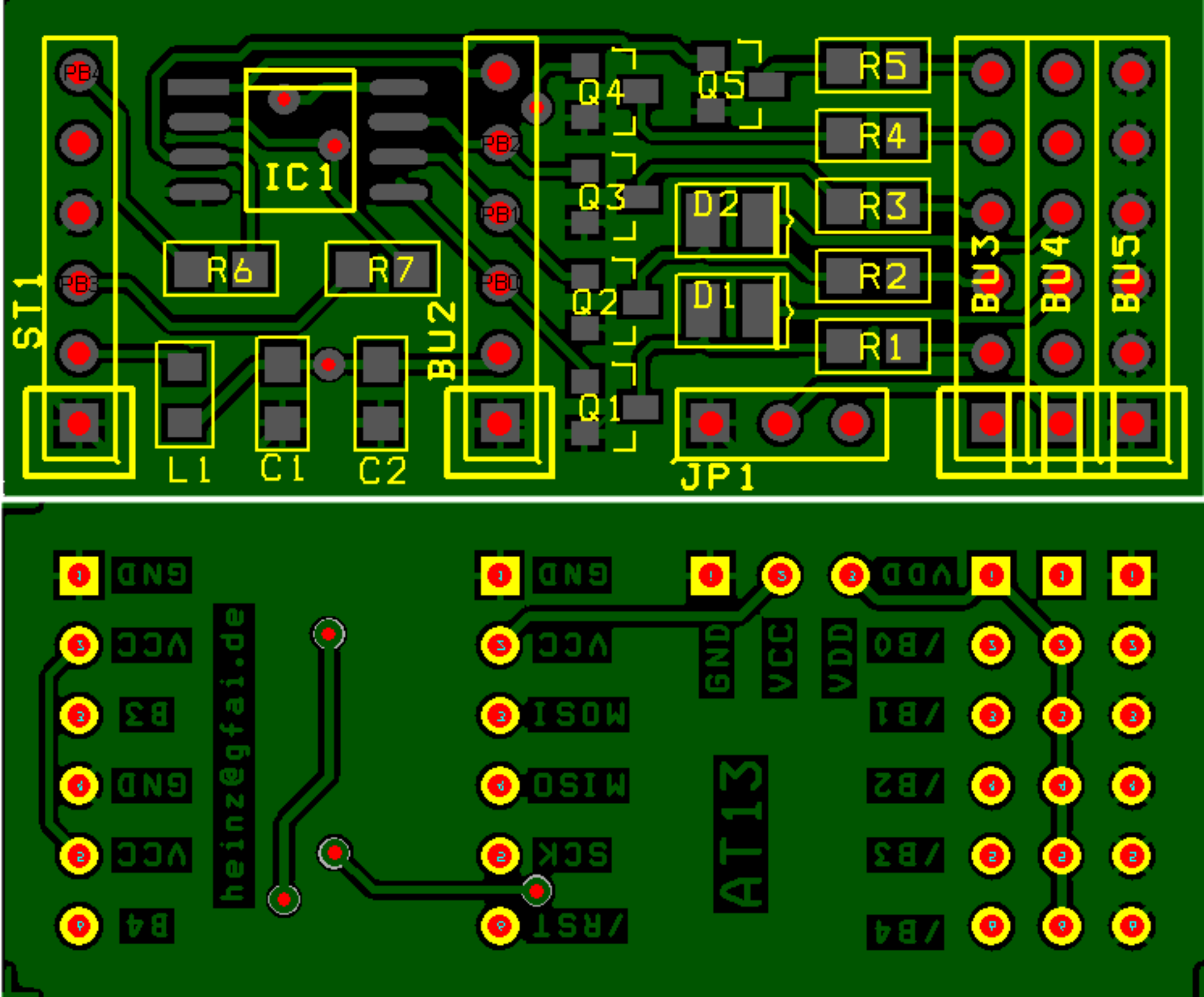
```
; Vierfarblicht für Board AT13
; mit 72 LED-Zeile aus SLML-72L-12V
; GH 2.1.2010, AVR-Studio 4.15
;
; Funktion:
;     Sobald Betriebsspannung anliegt, wird weisses Licht eingeschaltet.
;     Auf Knopfdruck (PB4) kann Farbmode gewechselt werden.
;     Mode wird in EEPROM gespeichert und startet
;     beim nächsten Einschalten wieder
;
; Moden:     (taste)
;
;     Knopfdruck  Mode
;           0x          statisch weiss
;           1x          statisch blau
;           2x          statisch blau/gruen
;           3x          statisch grün
;           4x          statisch rot/gelb/grün (warmweiss)
;           5x          statisch gelb
;           6x          statisch rot/gelb
;           7x          statisch rot
;           8x          statisch rot/blau (violett)
;           9x          Lauflicht
;           10x         Surfing (PWM, gleitende Farben, einer hell)
;           11x         Kamin (PWM, Zufall)
;           12x         Moving (PWM, gleitende Farben, einer dunkel)
;           13x         TVing Television-Effekt (PEWM, abprupte Änderungen)
;           14x         Walklight (hell, inverse Farben)
```



BU2 AVR-ISP:

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

Title	ATtiny13 Entwicklungsboard AT13 fuer LEDzeile SLML72	
Size A4	Document Number heinz@gfai.de	Rev 10


ST1
PB4
PB3
IC1
PB2
PB1
PB0
R6
R7
BU2
L1
C1
C2
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
/B0
/B1
/B2
/B3
/B4
VDD

```
;
; Vierfarblicht für Board AT13
; mit 72 LED-Zeile aus SLML-72L-12V
; heinz@gfai.de, 2.1.2010, AVR-Studio 4.15
;
; Funktion:
;   Sobald Betriebsspannung anliegt, wird weisses Licht eingeschaltet.
;   Auf Knopfdruck (PB4) kann Farbmode gewechselt werden.
;   Mode wird in EEPROM gespeichert und startet
;   beim nächsten Einschalten wieder
;
; Moden:     (taste)
;
;   Knopfdruck  Mode
;       0x      statisch weiss
;       1x      statisch blau
;       2x      statisch blau/gruen
;       3x      statisch grün
;       4x      statisch rot/gelb/grün (warmweiss)
;       5x      statisch gelb
;       6x      statisch rot/gelb
;       7x      statisch rot
;       8x      statisch rot/blau (violett)
;       9x      Lauflicht
;       10x     Surfing (PWM, gleitende Farben, einer hell)
;       11x     Kamin (PWM, Zufall)
;       12x     Moving (PWM, gleitende Farben, einer dunkel)
;       13x     TVing Television-Effekt (PEWM, abprupte Änderungen)
;       14x     Walklight (hell, inverse Farben)

.equ tastmax    = 15    ; bei dieser Anzahl von Tastendrücken von vorn

; PWM:
;   Ausgabe von 4 Zeitfunktionen a 256 Schritten
;   als Pulsweitenmodulation für vier LEDs an PB0-3
;   Pushbutton an PB4 gegen GND, 10k nach VDD
;
; Prozessor: ATMEL ATtiny13 mit 5V und 9,6 MHz
;
; Fuses:
;   BODLEVEL = 4.3V     (wichtig bei Netzteilbetrieb)
;   CKDIV8 nicht gesetzt
;   SUT_CKSEL = 9.6 MHz, 64 ms

.include "tn13def.inc"  ; Prozessor-Aliases
.device attiny13        ; Assembler directive
;
; Pinout 8-PDIP/SOIC:
;                 --------
;   /RESET, PB5 |1  `´  8| VCC
;           PB3 |2      7| PB2
;           PB4 |3      6| PB1
;           GND |4      5| PB0
;                 --------
; Ports:
;
;   Port    Register    Farbe
;   PB0:    led1        Red
;   PB1:    led2        Green
;   PB2:    led3        Blue
;   PB3:    led4        Yellow
;   PB4:    Button gegen GND
;
; Endstufen:
;   an PB0...3 je ein n-Chl Enh. IRLML2502
;               20V / 33A / 35mOhm bei UGS=4,5V
;               (P = (0,5A)² * 35mOhm = 9mW)
;
; Regler:
;   TS78L05 SOT23 5V/100mA
;
; LEDs:
;   pro Farbe je 3 in Reihe und 6 parallel an Schaltnetzteil 12V/1A
```

```
;   LED-board enthält alle Vorwiderstände
;   interne Pullups an PB4 und PB5 funktionieren nicht,
;   sie werden durch sbi PORTB gestört

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

; Ausgabeports

.equ red    = 0b0000_0001   ; rot
.equ green  = 0b0000_0010   ; gruen
.equ blue   = 0b0000_0100   ; blau
.equ yellow = 0b0000_1000   ; gelb

.equ rtgb   = 0b0000_1001   ; rot gelb
.equ rtbl   = 0b0000_0101   ; rot blau
.equ rdyegn = 0b0000_1011   ; rot gelb gruen
.equ blgn   = 0b0000_0110   ; blau gruen

.equ invred     = 0b0000_1110   ; invers rot
.equ invgreen   = 0b0000_1101   ; invers gruen
.equ invblue    = 0b0000_1011   ; invers blau
.equ invyellow  = 0b0000_0111   ; invers gelb


.equ alle   = 0b0000_1111   ; 0-3 alle an
.equ dark   = 0b0000_0000   ; 0-3 dunkel
.equ vier   = 0b0000_1111   ; Datenrichtung (1: output)

.equ yred       = 0xb0  ; rot       Moving
.equ ygreen     = 0x80  ; gruen
.equ yblue      = 0x40  ; blau
.equ yyellow    = 0x00  ; gelb

.equ maxred     = 0xff  ; Begrenzung nach oben (Kamin)
.equ maxgreen   = 0x4f  ; Begrenzung nach oben
.equ maxblue    = 0x8f  ; Begrenzung nach oben
.equ maxyellow  = 0xff  ; Begrenzung nach oben

.equ minred     = 0xb0  ; Begrenzung nach unten (Kamin)
.equ mingreen   = 0x20  ; Begrenzung nach unten
.equ minblue    = 0x40  ; Begrenzung nach unten
.equ minyellow  = 0xb0  ; Begrenzung nach unten

; Wartezeiten
.equ pwmdelay   = 0x01  ; PWM Taktzeit (größer null)
.equ dkamin     = 0x20  ; Kamin   Anzahl der PWM-Zyklen, bis nächste Zufallszahl gebild
et wird
.equ dsurf      = 0x3f  ; Surfing Anzahl der PWM-Zyklen, bis nächste Zufallszahl gebild
et wird
.equ dmoving    = 0x3f  ; Moving  Anzahl der PWM-Zyklen, bis nächste Zufallszahl gebild
et wird
.equ dauertv    = 0x8f  ; TVing  Anzahl der PWM-Zyklen, bis nächste Zufallszahl gebilde
t wird
.equ dauertv2   = 0x08  ; TVing  Zyklusverlängerung
.equ langsam    = 0xff  ; Lauflicht

; EEPROM-Adresse
.equ eeadr      = 0x10

; Register:
.def a1     = r16   ; Arbeit
.def a2     = r17   ; Arbeit
.def a3     = r18   ; Arbeit
.def delay  = r19   ; PWM Taktzeit (größer null)
.def zahl   = r20   ; Zufallszahl
.def pwm    = r21   ; PWM-Zähler

.def taste  = r22   ; Tastendruck-Anzahl
.def frei   = r23   ; frei
.def zykl   = r24   ; Zyklenzähler
.def leds   = r25   ; LED Outport

; alle Register belegt
```

```
; LED-Helligkeits-Register:
; YL, YH (Register r28...r29)   ; PWM-Dauer PB1,PB2
; ZL, ZH (Register r30...r31)   ; PWM-Dauer PB3,PB4

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

rjmp Reset

.org 0x0002 rjmp Tastendruck ; Pin Change Interrupt Handler, S.44


Reset:
    ldi r16,low(RAMEND)     ; Set Stack Pointer to top of RAM
    out SPL,r16                 ; an den Stackpointer (nur low)

.org 0x0010             ; Sprungadressen der anderen Interrupts auslassen
.cseg                   ; Codesegment folgt

rjmp Init               ; dort gehts weiter


; *********************************
;       Unterprogramme
; *********************************


Zufall: ; rückgekoppeltes Schieberegister (8 Bit random)
    mov a1,zahl     ; (inp: zahl); outp: a1
    clr a2
    clr a3
    lsr a1

    adc a2,a3
    lsr a1
    lsr a1
    lsr a1
    lsr a1

    adc a2,a3
    lsr a1
    adc a2,a3
    lsr a1
    adc a2,a3

    lsr zahl
    sbrc a2, 0
    sbr zahl,128
    mov a1,zahl     ; Zufallszahl auf Reg. a1 kopieren, damit arbeiten
ret


warte:              ; Rückzählen um max. 2^16-1
                    ; max a3 = 255, min a3 = 0
                    ; inp: delay
                    ; outp: Verzögerung
        mov a3,delay
        inc a3

        wart1:
        mov a2,delay    ; Grunddelay erzeugen
            wart2:
            dec     a2
            cpi     a2,0
            brne    wart2   ; branch if not equal
        dec     a3
        cpi     a3,0
        brne    wart1   ; branch if not equal
ret                     ; Rücksprung


Horu:       ; Sägezahn in Dreieck wandeln
            ; 0...127...255 in 0...254...0 umwandeln
```

```
            ; inp = outp = a1

    sbrc a1,7       ; überspringe, wenn MSB = 0, d.h. Zahl kleinergleich 127 (unsigned)
    com a1          ; alle Kleinen negieren (incl. MSB)
                    ; die Großen so lassen
    hoend:          ; alle Werte sind nun zwischen 128 und 255
    subi a1,0x80    ; 128 (=0x80) abziehen um auf 0...127 zu kommen
    rol a1          ; mal zwei um auf 0...254 zu kommen
ret


Dreieck:    ; inp: a1, outp: a1
            ; 0...127       -> 0...254 ausgeben (mal zwei)
            ; 128 bis 255   -> Null ausgeben
            ; Horu aufrufen


    cpi a1,0x80     ; liegt a1 zwischen 0...127?
    brsh dreig      ; branch if same or higher
    mov a2,a1
    rjmp dreik

    dreig:          ; a1 liegt zwischen 128...255
    ldi a2,0        ; a2 löschen

    dreik:          ; a2 mal zwei
    lsl a2          ; a2 liegt nun zwischen 0 und 255

    mov a1,a2       ; a2 auf a1 kopieren
    rcall Horu
ret


Maximum:    ; Begrenzung von a1 auf einen Wert in a2
            ; inp: a1, a2,  outp: a1
    cp a1,a2
    brlo limx
    mov a1,a2
    limx:
ret


Minimum:    ; Erhöhung von a1 auf einen Wert in a2
            ; inp: a1, a2,  outp: a1
    cp a2,a1    ; a2 <-> a1 vertauscht!
    brlo minx
    mov a1,a2
    minx:
ret



EepromWrite:        ; taste im EEPROM speichern

    ; Wait for completion of previous write
    eepromww:       ; wait for write enable
    sbic EECR,EEPE  ; EEPE alt EEWE (Bit 1)
    rjmp eepromww

    ; Set Programming mode (PM = 00) in Control Register
    in a1,EECR
    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,eeadr
    out EEARL,a1

    ; Write data to data register EEDR
    out EEDR,taste

    ; Master write enable: 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


EepromRead:     ; taste aus EEPROM lesen

    ; Wait for completion of previous write (wird hier nicht gebraucht)
    eepromrw:       ; wait for read enable
    sbic EECR,EEPE  ; (Bit 1)
    rjmp eepromrw

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

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

    ; Read data to register taste
    in taste,EEDR
ret


InitPinChangeInt:       ; Pin Change Interrupt aktivieren
                        ; für Flankenerkennung am Eingang PB4

    ; MCUCR MCU Control Register, Interrupt Sense Control, S.46
    ;   ISC0x:
    ;       00  low level       L       0x00
    ;       01  any edge    HL / LH     0x01
    ;       10  falling edge    HL      0x02
    ;       11  rising edge     LH      0x03

    ldi a1, 0x01        ; ISC0x - auf any edge einstellen im
    out MCUCR, a1       ; CPU/MCU Control Register, S.46

    ldi a1, 0x10        ; PCINT4 - Pin Change Interrupt für Pin4 setzen im
    out PCMSK, a1       ; Pin Change Mask Register, S.47

    ldi a1, 0x20        ; PCIE - Pin Change Interrupt Enable an das
    out GIMSK, a1       ; General Interrupt Mask Register, S.46

    ; alternativ wären denkbar:
    ; sbi MCUCR, ISC00  ; geht nicht
    ; sbi PCMSK, PCINT4 ; geht
    ; sbi GIMSK, PCIE   ; geht nicht
ret


;############### Interrupts ##############


Tastendruck:    ; Pin Change Interrupt an PB4 bei Tastendruck
                ; in/out: taste

    cli ; wichtig, um weitere Preller zu unterdrücken (ist eigtl. automatisch)
    push a1         ; sichern
    in a1, sreg     ; SREG sichern
    push a1         ; SREG gesichert

    ; Tastenentprellung
    push delay      ; altes delay sichern
    ldi delay,0x80  ; Preller abwarten
    rcall warte     ; delay ist input
    pop delay       ; delay wiederherstellen

    ; jetzt Pinabfrage
    sbis  PINB,4        ; Pin4 abfragen, Skip if H
    rjmp  tastenpuls    ; Pin ist Low (Taste ist gedrückt)

    ; Pin = H           ; Taste nicht gedrückt
```

```
    rjmp tastenend

    ; Taste wurde gedrückt:
    tastenpuls:         ; Pin ist Low (Taste wurde gedrückt)
    inc taste           ; taste erhöhen
    cpi taste,tastmax   ; tastmax Tastendrucke sind erlaubt
    brlt tastenwei      ; wenn kleiner, dann überspringen
    ldi taste,0         ; wenn tastmax, dann rücksetzen
    tastenwei:          ; weiter

    ; neuen Wert in EEPROM sichern
    rcall EepromWrite   ; Einstellung sichern über taste

    ; Taste wurde nicht gedrückt:
    tastenend:
    pop a1          ; SREG rückholen
    out sreg,a1     ; SREG wiederhergestellt
    pop a1          ; a1 wiederhergestellt
    ; sei           ; doppelt, passiert mit reti
reti


; *********************************
;           Start, Init
; *********************************

Init:

; Port
    ldi a1,vier     ; DDR init
    out DDRB,a1     ; Datenrichtung Ausgabe

    ldi a1,alle     ; Portinit
    out PORTB,a1    ; alle an

; Zufall
    ldi zahl,1      ; init Zufallszahl
    ldi zykl,0      ; init Zyklus
    ldi pwm,0       ; lange Schleife
    ldi XH,0

; Timerregister Init Mittelstellung 0x80 = 128 dez
    ldi YL, 0x30    ; ws
    ldi YH, 0x30    ; rt
    ldi ZL, 0x30    ; ws
    ldi ZH, 0x30    ; gb

; Interrupt init
    rcall InitPinChangeInt

; Taste aus EEPROM lesen
    rcall EepromRead    ; EEPROM-Wert auf Register "taste"

sei ; Interrupt-Freigabe

; *********************************
;           Mains
; *********************************

Main:

Rotgelb:
    cpi taste,6
    brne rtgbRET
    ldi a1,rtgb ; Leds ein
    out PORTB,a1
    rjmp Rotgelb
rtgbRET:


Rotblau:
    cpi taste,8
    brne rtblRET
```

```
    ldi a1,rtbl ; Leds ein
    out PORTB,a1
    rjmp Rotblau
rtblRET:


RYG:    ; red yellow green (warmweiss)
    cpi taste,4
    brne rygRET
    ldi a1,rdyegn   ; Leds ein
    out PORTB,a1
    rjmp RYG
rygRET:


Bluegreen:
    cpi taste,2
    brne blgnRET
    ldi a1,blgn ; Leds ein
    out PORTB,a1
    rjmp Bluegreen
blgnRET:


Weiss:
    cpi taste,0
    brne weissRET
    ldi a1,alle ; Leds ein
    out PORTB,a1
    rjmp Weiss
weissRET:


Blau:
    cpi taste,1
    brne blauRET
    ldi a1,blue ; Leds ein
    out PORTB,a1
    rjmp Blau
blauRET:


Gruen:
    cpi taste,3
    brne gruenRET
    ldi a1,green    ; Leds ein
    out PORTB,a1
    rjmp Gruen
gruenRET:


Gelb:
    cpi taste,5
    brne gelbRET
    ldi a1,yellow   ; Leds ein
    out PORTB,a1
    rjmp Gelb
gelbRET:


Rot:
    cpi taste,7
    brne rotRET
    ldi a1,red  ; Leds ein
    out PORTB,a1
    rjmp Rot
rotRET:


Lauflicht:  ; bei Aufruf hier verweilen
    cpi taste,9     ;
    brne laufRET
```

```
    ldi delay,langsam   ; entweder periodisch,
    ; inc delay         ; oder variieren (19 sec)

    ldi a1,red
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,green
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,blue
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,yellow
    out PORTB,a1
    rcall warte
    rcall warte

    rjmp Lauflicht
laufRET:


Walklight:  ; bei Aufruf hier verweilen
    cpi taste,14    ;
    brne walkRET

    ldi delay,langsam   ; entweder periodisch,
    ; inc delay         ; oder variieren (19 sec)

    ldi a1,invred
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,invgreen
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,invblue
    out PORTB,a1
    rcall warte
    rcall warte

    ldi a1,invyellow
    out PORTB,a1
    rcall warte
    rcall warte

    rjmp Walklight
walkRET:


Kamin:
    cpi taste,11        ;
    brne kaminRET

    ldi delay,pwmdelay  ; PWM-pwmdelay

    inc zykl            ; Zyklenzähler erhöhen
    cpi zykl, dkamin    ; nur bei Gleichheit neu einlesen
    brne kaminnext      ; ungleich: nichts verändern
    ldi zykl,0          ; weiter (nur bei zykl=0 neue Zufallszahl erzeugen)

        rcall Zufall    ; Zufallszahl erzeugen (output = a1)
        ldi a2,maxred
        rcall Maximum
        ldi a2,minred
```

```
        rcall Minimum
        mov YL,a1

        rcall Zufall    ; Zufallszahl erzeugen (output = a1)
        ldi a2,maxgreen
        rcall Maximum
        ldi a2,mingreen
        rcall Minimum
        mov YH,a1

        rcall Zufall    ; Zufallszahl erzeugen (output = a1)
        ldi a2,maxblue
        rcall Maximum
        ldi a2,minblue
        rcall Minimum
        mov ZL,a1

        rcall Zufall    ; Zufallszahl erzeugen (output = a1)
        ldi a2,maxyellow
        rcall Maximum
        ldi a2,minyellow
        rcall Minimum
        mov ZH,a1

    kaminnext:

    ; PWM erzeugen
        ldi pwm,0           ; init PWM-Zähler
        ldi a1,0xff         ; LED-Portout Register einstellen
        out PORTB,a1        ; alle LEDs ein

    kamloop:
        ; PWM: for pwm = 1 to 255 (für jede Zeitstufe)
        ; LED-Bits nach Zeitdauer in YL...ZH einschalten

        mov a1,YL
        cp a1,pwm   ; Zeit für LED01 abgelaufen?
        brsh w1     ; nicht same or higher? Dann drüber
        cbi PORTB,0 ; Clear Bit in I/O: Bit R
        w1:

        mov a1,YH
        cp a1,pwm
        brsh w2
        cbi PORTB,1 ; Bit G
        w2:

        mov a1,ZL
        cp a1,pwm
        brsh w3
        cbi PORTB,2 ; Bit B
        w3:

        mov a1,ZH
        cp a1,pwm
        brsh w4
        cbi PORTB,3 ; Bit Y
        w4:

        ; warten
        rcall warte     ; nächster PWM-Schritt
        inc pwm         ; nächster PWM-Zeitschritt
        cpi pwm, 0      ; PWM abgelaufen, ganz von vorn
        breq Kamin      ; bei 0 neue Werte holen
        rjmp kamloop    ; in Schleife bleiben

        rjmp Kamin      ; eigentlich überflüssig
kaminRET:



Surfing:
    ldi delay,pwmdelay  ; PWM-delay laden
```

```
    ldi YL,yred         ; Inits laden
    ldi YH,ygreen
    ldi ZL,yblue
    ldi ZH,yyellow

    ; Zyklus init
    surfweit:

    cpi taste,10
    brne surfRET        ; Ausstieg, falls falsche Taste

    inc zykl            ; Zyklenzähler erhöhen
    cpi zykl, dsurf ; nur bei Gleichheit neu einlesen
    brlo surfneu        ; ungleich: nichts verändern
    ldi zykl,0          ; weiter (nur bei zykl=0 neue Zufallszahl erzeugen)

    inc YL
    inc YH
    inc ZL
    inc ZH


    ; PWM erzeugen
    surfneu:
        ldi pwm,0           ; init PWM-Zähler
        ldi a1,0xff         ; LED-Portout Register einstellen
        out PORTB,a1        ; alle LEDs ein

    surfloop:
        ; PWM: for pwm = 1 to 255 (für jede Zeitstufe)
        ; LED-Bits nach Zeitdauer in YL...ZH einschalten

        mov a1,YL
        rcall Horu      ; Format in hoch/runter wandeln
        cp a1,pwm       ; Zeit für LED01 abgelaufen?
        brsh surfw1     ;
        cbi PORTB,0     ; Lösche Bit in I/O: Rot
        surfw1:

        mov a1,YH
        rcall Horu
        cp a1,pwm
        brsh surfw2
        cbi PORTB,1     ; Bit Gruen
        surfw2:

        mov a1,ZL
        rcall Horu
        cp a1,pwm
        brsh surfw3
        cbi PORTB,2     ; Bit Blau
        surfw3:

        mov a1,ZH
        rcall Horu
        cp a1,pwm
        brsh surfw4
        cbi PORTB,3     ; Bit Yellow
        surfw4:

        ; warten
        rcall warte     ; einen PWM-Schritt lang warten (inp: delay)

        inc pwm         ; PWM-Zeitschrittzähler erhöhen
        cpi pwm, 0      ; PWM abgelaufen, ganz von vorn?
        breq surfweit   ; von vorn
    rjmp surfloop       ; in Schleife bleiben
    rjmp Surfing        ; eigentlich überflüssig
surfRET:
```

```
Moving:
    ldi delay,pwmdelay  ; PWM-delay laden

    ldi YL,yred         ; Inits laden
    ldi YH,ygreen
    ldi ZL,yblue
    ldi ZH,yyellow

    ; neuer PWM-Zyklus, init
    movweit:
    cpi taste,12
    brne movRET     ; Ausstieg, falls falsche Taste

    inc zykl            ; Zyklenzähler erhöhen
    cpi zykl, dmoving   ; neu einlesen?
    brlo movneu         ; dann nichts verändern

        ; neue Werte
        ldi zykl,0          ; nur bei zykl=0 neue Werte erzeugen
        inc YL
        inc YH
        inc ZL
        inc ZH


    ; PWM erzeugen
    movneu:
        ldi pwm,0           ; init PWM-Zähler
        ldi a1,0xff         ; LED-Portout Register einstellen
        out PORTB,a1        ; LEDs an

    movloop:
        ; PWM: for pwm = 1 to 255 (für jede Zeitstufe)
        ; LED-Bits nach Zeitdauer in YL...ZH einschalten

        mov a1,YL
        rcall Dreieck       ; Format in hoch/runter wandeln
        cp a1,pwm       ; Zeit für LED01 abgelaufen?
        brsh movw1      ;
        cbi PORTB,0     ; Clear Bit in I/O: Rot
        movw1:

        mov a1,YH
        rcall Dreieck
        cp a1,pwm
        brsh movw2
        cbi PORTB,1     ; Bit Gruen
        movw2:

        mov a1,ZL
        rcall Dreieck
        cp a1,pwm
        brsh movw3
        cbi PORTB,2     ; Bit Blau
        movw3:

        mov a1,ZH
        rcall Dreieck
        cp a1,pwm
        brsh movw4
        cbi PORTB,3     ; Bit Yellow
        movw4:

        ; warten
        rcall warte     ; einen PWM-Schritt lang warten (inp: delay)

        inc pwm         ; PWM-Zeitschrittzähler erhöhen
        cpi pwm, 0      ; PWM abgelaufen, ganz von vorn?
        breq movweit    ; von vorn
    rjmp movloop        ; in Schleife bleiben
    rjmp Moving     ; eigentlich überflüssig
movRET:
```

```
TVing:
    ldi delay,pwmdelay  ; PWM-delay laden

    ldi YL,yred         ; Inits laden
    ldi YH,ygreen
    ldi ZL,yblue
    ldi ZH,yyellow

    ; neuer PWM-Zyklus, init
    tvweit:
    cpi taste,13
    brne tvRET          ; Ausstieg, wenn falsche Taste

    ; Zyklengenerator

    inc XH              ; Zyklus um dauertv2 verlängern
    cpi XH,dauertv2
    brne tvneu
        ldi XH,0
        inc YL              ; Farbe dabei bissel ändern
        dec YH

    inc zykl            ; Zyklenzähler erhöhen
    cp zykl,XL          ; neu einlesen?
    brne tvneu          ; hier nichts verändern

        ; neue Werte einlesen
        ldi zykl,0          ; nur bei zykl=0 neue Werte erzeugen

        ldi a2,dauertv      ; minimale Zeitkonstante vorgeben
        mov XL,a2           ; auf XL sichern

        rcall Zufall        ; Ausgabe auf a1
        add XL,a1           ; Zufall auf XL addieren (incl. Überlauf)

        ; stabilen Zyklus verhindern
        cpi XL,1
        brne tvwei
        ldi XL,0
        tvwei:

        ; Ausgabe auf Farbregister YL...ZH
        rcall Zufall        ; Ausgabe auf a1
        mov ZH,ZL
        mov ZL,a1

        rcall Zufall        ; Ausgabe auf a1
        mov YH,YL
        mov YL,a1

    ; PWM erzeugen
    tvneu:
        ldi pwm,0           ; init PWM-Zähler
        ldi a1,0xff         ; LED-Portout Register einstellen
        out PORTB,a1        ; LEDs an

    tvloop:
        ; PWM: for pwm = 1 to 255 (für jede Zeitstufe)
        ; LED-Bits nach Zeitdauer in YL...ZH einschalten

        mov a1,YL
        rcall Horu      ; Format in hoch/runter wandeln
        cp a1,pwm       ; Zeit für LED01 abgelaufen?
        brsh tvw1       ;
        cbi PORTB,0     ; Clear Bit in I/O: Rot
        tvw1:

        mov a1,YH
        rcall Horu
        cp a1,pwm
        brsh tvw2
```

```
        cbi PORTB,1     ; Bit Gruen
        tvw2:

        mov a1,ZL
        rcall Horu
        cp a1,pwm
        brsh tvw3
        cbi PORTB,2     ; Bit Blau
        tvw3:

        mov a1,ZH
        rcall Horu
        cp a1,pwm
        brsh tvw4
        cbi PORTB,3     ; Bit Yellow
        tvw4:

        ; warten
        rcall warte     ; einen PWM-Schritt lang warten (inp: delay)

        inc pwm         ; PWM-Zeitschrittzähler erhöhen
        cpi pwm, 0      ; PWM abgelaufen, ganz von vorn?
        breq tvweit ; von vorn
    rjmp tvloop     ; in Schleife bleiben
    rjmp TVing      ; eigentlich überflüssig
tvRET:



nop
nop
nop

rjmp Main       ; nächste Tastenübereinstimmung suchen



;****************************************************************
```