```
;
; RC-Blinkgeber und Bremsschalter für
; Proportional-Funkfernsteuerung
; mit Mikroprozessor Atmel ATtiny13
; dr.g.heinz@web.de
; 28.8.2008
;
; Eingänge:          K1 (Gas/Bremse)
;                                        K2 (Rechts/Links)
;
; Pulsbreite: 1,0...2,0 ms, positiver Impuls, neutral: 1,5 ms
; Grundlagen siehe http://www.sprut.de/electronic/modell/allgemein/index.htm
;
; Funktion:
;          Etwa eine halbe Sekunde nach Einschalten der Betriebsspannung
;          wird die Pulsbreite beider Kanäle gemessen.
;          Lenkung: Maß wird als Mittelstellung interpretiert.
;          Gas/Bremse: Maß wird als Leerlaufstellung interpretiert.
;          Um die Gas-Knüppelrichtung festzulegen: Stecker Kanal K2 abziehen, bei
;          Vollbremsung Receiver-Betriebsspannung einschalten.
;          Diese Richtung wird im EEPROM abgelegt.
;          Rechts/Links Vertauschung Blinker ist an den Ausgängen möglich.
;          Bremse leuchtet bei Rücknahme des Gasknüppels.
;          Blinker leuchten nach Gasfahrt bei Lenkeinschlag auf.
;
; Ports:             PB0 (Pin5) Ausgabe Blinken rechts
;                                        PB1 (Pin6) Ausgabe Bremse
;                                        PB2 (Pin7) Ausgabe Blinken links
;                                        PB3 (Pin2) Eingang K1 Gas/Bremse
;                                        PB4 (Pin3) Eingang K2 Links/Rechts
;
; Sonst:             +5V       (Pin8)
;                                        GND       (Pin4)
;                                        RES (Pin1)
;
; 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 ################

.org 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

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

rjmp	anfang		; hier kann Boot eingefügt werden

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

anfang:

; Register dynamic:
.def	a1		= r16	; Albeit
.def	a2		= r17	; Albeit
.def	a3		= r18	; Albeit
.def	a4		= r19	; Albeit
; Register static
.def	prescal	= r20	; Teilerfaktor Prescaler
.def	leds	= r21	; aktuelle Ausgabe an LEDs
.def	wait1	= r22	; Wartezeit Zyklus 1
.def	wait2	= r23	; Wartezeit Zyklus 2
.def	dela1	= r24	; Delay K1 aktueller Meßwert
.def	dela2	= r25	; Delay K2 aktueller Meßwert
.def	delk1	= r26	; Delay K1 Neutralstellung
.def	delk2	= r27	; Delay K2 Neutralstellung
.def	delb1	= r28	; Delay K1 Bremsrichtung aus EEPROM
.def	delb2	= r29	; Delay K2 Hilfsregister
.def	delc1	= r30	; Delay K1 Vorzyklus i-1
.def	delc2	= r31	; Delay K2 Vorzyklus i-1

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

		ldi wait1,255			; Schleifenzähler für Pausen (Simulation: 1,
Burn: 255)
		mov wait2,wait1			; Schleifenzähler wait2 von wait1 übernehmen
		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 Lichter ein xx00 0111
        rcall pause                             ; erstmal warten bis Powerup stabil ist
        rcall pause                             ; erstmal warten bis Powerup stabil ist
        rcall pause                             ; erstmal warten bis Powerup stabil ist
        rcall allaus                    ; alle Lichter aus xx00 0000
        rcall pause                             ;

;################ Ist K2 gesteckt? ####################

; überprüfen, ob Stecker K2 (r/l) PB4 gezogen wurde.
; Wenn nicht gesteckt, dann Richtungserkennung Gas/Bremse,
; dazu Pullup für PB4 setzen und warten
        rcall pullein       ; Pullups PB3, PB4 einschalten

; zweimal prüfen, ob K2 = high mit Pause größer als max. Pulslänge
; nur wenn beidemale K2 = high, dann ist K2-Stecker nicht gesteckt.
        ;rcall warnb                    ; Auftakt erster Warnblink
        sbis pinb,4                     ; Pinb4 (K2) abfragen, wenn 1, dann nächsten
Befehl überspringen
        rjmp steckt                     ; Pinb4 war 0: K2-Stecker ist gesteckt
        rcall pause                     ; warten, falls das gerade ein Puls war
        rcall pause                     ; warten, falls das gerade ein Puls war
        rcall pause                     ; warten, falls das gerade ein Puls war
        ;rcall warnb                    ; Zweiter Warnblink
        sbis pinb,4                     ; Pinb4 (K2) abfragen, wenn 1, dann nächsten
Befehl überspringen
        rjmp steckt                     ; Pinb4 war 0: K2-Stecker ist gesteckt
        ;rcall warnb        ; Dritter Warnblink

        ; zweimal war PB4 = K2 = 1 --> Stecker ist nicht gesteckt

;############ Messung Delay der Bremsrichtung K1 #############

        ; erwartet wird hier, daß der Bremshebel gezogen ist vor power-up
        ; und daß der K2-Stecker nicht gesteckt wurde
        rcall flank1        ; Flankendetektion von K1, warte auf HL-Flanke
        rcall messk1        ; Pulsbreite der Vollbremsung von K1 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 - Brake blinkt halb so oft, wie Wert in delk1
        rcall pause                     ;   1ms = 10; 1,5ms = 15; 2ms = 20 Blinker (1
blink = 0,1 ms)
```

```
        rcall halt              ; Dauerleuchtend verweilen --> Ende der Messung

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

steckt:                 ; K2-Stecker wurde als gesteckt erkannt

        rcall brbl              ; Erster Brakeblink
        rcall brbl              ; Zweiter Brakeblink

; Neutrallage K1 messen, K1 v/r PB3
        ; auf K1 -> LH warten
        rcall flank1            ; Flankendetektion von K1
        ; Meßende bei HL-Flanke
        rcall messk1            ; Neutrallage von K1 messen
        in      delk1,TCNT0     ; Zeit der Neutralstellung K1 in Register delk1
schreiben

; Neutrallage K2 messen, K2 r/l PB4
        ; auf K2 -> LH warten
        rcall flank2            ; Flankendetektion von K2
        ; Meßende bei HL-Flanke
        rcall messk2            ; Neutrallage von K2 messen
        in      delk2,TCNT0     ; Zeit der Neutralstellung K2 in Register delk2
schreiben

; Testausgaben K1 und K2
        ;rcall testall          ; Testausgabe delk1 auf PB0,PB1,PB2 (alle blinken)
        ;rcall test1 ; Testausgabe delk1 auf PB1 (brake blinken)
        ;rcall test2 ; Testausgabe delk2 auf PB0,PB2 (blinker blinken)

;############## Blink- und Bremslicht #################

        rcall warnb                     ; Warnblink

schleife:

; Bremse an PB3 messen:
        rcall flank1            ; Warte auf nächste HL-Flanke an PB3 K1
        rcall messk1            ; K1 erneut messen
        in dela1,TCNT0          ; Meßwert in Arbeitsregister ablegen
        ; Ausgabe an Bremslicht PB1:
        rcall brake             ; Bremslicht ein/ausschalten

; Blinken an PB4 messen:
        rcall flank2            ; Warte auf nächste HL-Flanke an PB4 K2
        rcall messk2            ; K2 erneut messen
        in dela2,TCNT0          ; Meßwert in Arbeitsregister ablegen
        ; Ausgabe an Blinklichter PB0/PB2:
        rcall blink             ; Blinklicht einschalten
        rcall pause
        rcall blau              ; Blinklicht ausschalten
```

```
        rcall pause

rjmp schleife


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


wrflash:    ; delb1 im Flash 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:    ; 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              ; Bremslicht ein
        rjmp halt               ; Prozeß anhalten -> fertig, Bremsstellung 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


pause:
        mov a2,wait1
        twomore:
                mov a1,wait2
                onemore:
                dec a1
                cpi a1,0
                brne onemore
        dec a2
        cpi a2,0
        brne twomore
ret


flank1:                         ; Warte auf Puls
        ; LH-Flankendetektion an K1
        ; Warten, bis K1 erste L/H-Flanke erhält; Abfrage K1 v/r PB3

        eins1:                  ; warten, bis K1 null ist (falls gerade auf 1)
        sbic    pinb,3          ; Pinb3 (K1) abfragen: weiter, wenn 0
        rjmp    eins1           ; bei 1 in Schleife verweilen

        null1:  ; warten, bis K1 auf high geht
        sbis    pinb,3          ; Pinb3 (K1) abfragen, weiter, wenn 1
        rjmp    null1           ; bei 0 in Schleife verweilen
        ; Flanke K1 0 -> 1 detektiert
ret


flank2:                         ; Warte auf Puls
        ; LH-Flankendetektion an K2
        ; Warten, bis K2 erste L/H-Flanke erhält; Abfrage K2 r/l PB4
```

```
            eins2:                  ; warten, bis K2 null ist (falls gerade auf 1)
            sbic        pinb,4      ; Pinb4 (K2) abfragen: weiter, wenn 0
            rjmp        eins2       ; bei 1 in Schleife verweilen

            null2:      ; warten, bis K1 auf high geht
            sbis        pinb,4      ; Pinb4 (K2) abfragen, weiter, wenn 1
            rjmp        null2       ; bei 0 in Schleife verweilen
            ; Flanke K2 0 -> 1 detektiert
ret


messk1:                             ;  Messe Dauer von Puls auf K1
            ; Timer-Init K1 v/r PB3
            ldi a1,$00              ; Timer rücksetzen
            out TCNT0,a1            ; Siehe S.73 ATtiny13
            ; Vorteiler einstellen   siehe Tab.11-9, S.73
            out TCCR0B,prescal; Init Steuerregister -> Start Timer (Signal CS02)
            ; warte auf K1 -> low
            k1null:
            sbic        pinb,3      ; Pinb3 (K1) abfragen, warte auf Pulsende HL
            rjmp        k1null      ; 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 K2
            ; Timer-Init K2 r/l PB4
            ldi a1,$00              ; Timer löschen
            out TCNT0,a1
            ; Vorteiler einstellen
            out TCCR0B,prescal; Init Steuerregister -> Start Timer (Signal CS02)
            ; warte auf K1 -> low
            k2null:
            sbic        pinb,4      ; Pinb4 (K2) abfragen, warte auf Pulsende HL
            rjmp        k2null      ; in Schleife verweilen
            ldi a1,0                ; Timer 0 stop
            out TCCR0B,a1           ; Steuerregister löschen -> Stop Timer
ret


testall:
            ; Testausgabe delk1 auf PB0, PB1 und PB2
            ; so oft blinken, wie Wert im Register
            mov a1,delk1
            noch0:
            ldi a2,$07              ; 0000 0111 PB0-3 auf 1
            out portb,a2            ; Eins ausgeben
            rcall pause
            ldi a2,$00              ; 0000 0000 PBs auf 0
```

```
        out portb,a2        ; Null ausgeben
        rcall pause
        dec a1
        cpi a1,0            ; ist a1 schon auf Null?
        brge noch0                  ; weiterblinken
ret


test1:
        ; Testausgabe delk1 auf PB1
        ; so oft blinken, wie Wert im Register delk1
        mov a3,delk1
        noch1:
        ldi a2,$02          ; 0000 0010 PB0 auf 1
        out portb,a2        ; Eins ausgeben
        rcall pause
        ldi a2,$00          ; 0000 0000 PB0 auf 0
        out portb,a2        ; Null ausgeben
        rcall pause
        dec a3                      ; decrement
        dec a3                      ; zweimal, um Skala 1 blink = 0,1 ms
festzulegen
        cpi a3,0            ; ist a3 schon kleiner gleich Null?
        brge noch1                  ; weiterblinken, wenn größer
ret


test2:
        ; Testausgabe delk2 auf PB0, PB2
        ; so oft blinken, wie Wert im Register
        mov a3,delk2
        noch2:
        ldi a2,$05          ; 0000 0101 PB2 auf 1
        out portb,a2        ; Eins ausgeben
        rcall pause
        ldi a2,$00          ; 0000 0000 PB2 auf 0
        out portb,a2        ; Null ausgeben
        rcall pause
        dec a3
        cpi a3,0            ; ist a3 schon auf Null?
        brge noch2                  ; weiterblinken
ret


brake:  ; Ausgabe Bremslicht an PB1
                                        ; dela1: aktueller Meßwert
                                        ; delb1: Bremsrichtung aus EEPROM
                                        ; delk1: Neutralstellung
        ; Prüfung der Bremsrichtung
        ; aus EEPROM lesen
        rcall rdflash       ; lese Wert delb1 aus EEPROM
        cp delb1,delk1      ; vergleiche Neutralstellung delk1 mit Bremswert aus
```

```
EEPROM delb1
        brlo normal                 ; wenn Vergleich kleiner null, Bremsrichtung
normal

        invert:                     ; wenn delb1 größer delk1, ein/aus vertauschen:
        subi dela1,2        ; Empfindlichkeit verringern, dazu Nullbereich vergrößern
        cp dela1,delk1      ; Meßwert mit Neutralstellung delk1 vergleichen
        brge bron           ; wenn Meßwert größer i+1: Bremslicht einschalten,
        rjmp broff          ; sonst ausschalten

        normal:
        ldi a1,1            ; Empfindlichkeit verringern, dazu Nullbereich vergrößern
        add dela1,a1        ;
        cp dela1,delk1      ; Meßwert mit Neutralstellung delk1 vergleichen
        brlo bron           ; wenn Meßwert kleiner i-1: Bremslicht einschalten
        rjmp broff          ; sonst ausschalten

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

        bron:                       ; Einschalten PB1:
        sbr leds,(1<<PB1)   ; xxxx xx1x PB1 Bremslicht einschalten
        out portb,leds
ret


blink:  ; Blinken rechts: PB0; Blinken links: PB2
                                            ; dela2: aktueller Meßwert
                                            ; delb2: Hilfsregister
                                            ; delk2: Neutralstellung

 ; Nichts tun 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

 ; Blinker einschalten für alle anderen Werte
        blein:                      ; links  einschalten für kleiner als i-1
```

```
        mov delb2,dela2     ; duplizieren
        dec delb2           ; i-1
        cp delk2,delb2      ; Meßwerte kleiner i-1 mit Neutralstellung vergleichen
        brlo brein          ; 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
        brein:                      ; rechts einschalten
                sbr leds,(1<<PB0)           ; xxxx xxx1 PB0 ein
                out portb,leds      ; PB0 ein (R)
        endb:
ret


brbl:   ; Brake-LEDs blinken
        rcall brei   ; brake ein
        rcall pause
        rcall brau  ; brake aus
        rcall pause
ret


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


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


warnb:                      ; Warnblink
        rcall blei          ; Blinklicht einschalten
        rcall pause
        rcall blau          ; Blinklicht ausschalten
        rcall pause
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
```

```
blau:                           ; 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

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 anfang                             ; sicher ist sicher
```