```
/*  ********************************************************************************************
 Schalten von Funksteckdosen
 ATTINY 861A

 Jan Schoettler
 17.02.2013 Quellen zu Eclipse portiert
 26.02.2014 Watchdogtimer als Zeitbasis implementiert
 02.03.2013 PCINT als Alternative implementiert

 Verweise wie Seite X beziehen sich auf das entsprechende Datenblatt:
 Atmel AVR ATtiny 261/V, 461/V, 861/V Datasheet
 ******************************************************************************************** */
/*  --------------------------------------------------------------------------------------------
 Konstanten
 -------------------------------------------------------------------------------------------- */
#define funkPin     PB3     // Pin an dem der Sender angeschlossen ist
#define funkVcc     PA1     // Pin an dem die Stromversorgung des Funkmoduls angeschlossen ist
#define buttonPin_0 PB6     // Pin an dem der Taster_0 angeschlossen ist (INT0)
#define buttonPin_1 PA2     // Pin an dem der Taster_1 angeschlossen ist (INT1)
#define ledPin      PA3     // Pin an dem die LED angeschlossen ist¨
#define debugPin_0  PA6     // Pin für Messung von Zeiten, um Stromstaerken abzuschaetzen
#define debugPin_1  PA7     // Pin für Messung von Zeiten, um Stromstaerken abzuschaetzen
/*  --------------------------------------------------------------------------------------------
 Importierte Klassen und Librarys
 -------------------------------------------------------------------------------------------- */
#include <avr/io.h>         // IO Pins
#include <avr/interrupt.h>  // Unterstuetzung fuer Interrups
#include <avr/sleep.h>      // Unterstuetzung fuer sleep mode
#include <util/delay.h>     // Verzoegerungen
#include <avr/wdt.h>        // Watchdog Timer
#include "RCS.h"            // Klasse zum Schalten von Funksteckdosen
/*  --------------------------------------------------------------------------------------------
 Klassen und Variablen
 -------------------------------------------------------------------------------------------- */
rcs theRCS = rcs(&PORTB, funkPin, RCS_DIP, RCS_DIP_SYSTEM_1, RCS_DIP_UNIT_A);
volatile int remainingTime = 0;
/*  --------------------------------------------------------------------------------------------
 init
 -------------------------------------------------------------------------------------------- */
void init() {
    /*
     ClockPrescaler setzen:                                                         Seite 31 ff
     zuerst CLKPCE im Register CLKPR auf 1 setzen um eine Aenderung des
     ClockPrescalers zu ermoeglichen und dann innerhalb von 4 Taten CLKPR auf den entsprechenden Wert
     gemaess folgender Tabelle setzen.
     Basis ist hier der interne 8MHZ Oszillator
     In diesem Beispiel sind 500 kHz Takt ausreichend 8.000.000/500.000 = 16

     CLKPS3     CLKPS2  CLKPS1  CLKPS0 Clock Division Factor
     0      0       0       0               1
     0      0       0       1               2
     0      0       1       0               4
     0      0       1       1               8
     0      1       0       0               16
     0      1       0       1               32
     0      1       1       0               64
     0      1       1       1               128
     1      0       0       0               256
     */
    CLKPR = _BV(CLKPCE);                // Clock Prescaler aendern ermoeglichen
    CLKPR = _BV(CLKPS2);                // Clock Prescaler setzen (Faktor 16)

    //  alle IO Ports als Eingang mit PullUp konfigurieren um Strom zu sparen          Seite  ????
    DDRA = 0;                   // Alle Pins von PortA als Eingagng konfigurieren      Quelle ????
    PORTA = 0x77;               // Alle Pins von PortA Pullup aktivieren
    DDRB = 0;                   // Alle Pins von PortB als Eingagng konfigurieren
    PORTB = 0x77;               // Alle Pins von PortB Pullup aktivieren
```

```
68      //  genutzte IO PORTS konfigurieren
69      DDRA |= (1 << ledPin);      // LEDPin als Ausgang schalten
70      DDRA |= (1 << funkVcc);     // funkvcc als Ausgang schalten
71      PORTA |= (1 << funkVcc);    // Spannung fuer Funkmodul einschalten
72      PORTB |= ((1 << buttonPin_0) | (1 << buttonPin_1)); // ButtonsafPin_0 und ButtonPin_1 Pullup
73      //DDRA |= (1 << debugPin_0) | (1 << debugPin_1);// debugpins als Ausgang schalten
74      //PORTA &= ~((1 << debugPin_0) | (1 << debugPin_1));// debugpins auf low setzen
75      GIMSK |= (1 << INT0); // freigeben INT0                                          Seite 51 ff
76      GIMSK |= (1 << INT1) ;                  // freigeben INT1
77      MCUCR |= (0 << ISC00);                  // Interrupt, wenn INT0, INT1 LOW sind
78      //GIMSK |= (1 << PCIE1);                // freigeben PCINT Interrupt              Seite 52 ff
79      //PCMSK0 |= (1 << PCINT2);              // nur auf PCINT2 (PA2) reagieren
80      //  Verfuegbare Schlafmodi:
81      //  SLEEP_MODE_IDLE, SLEEP_MODE_ADC, SLEEP_MODE_PWR_DOWN, SLEEP_MODE_STANDBY       Seite 36 ff
82      set_sleep_mode (SLEEP_MODE_PWR_DOWN);   // Sleep Mode ausf Standby setzen
83      MCUCR |= (_BV(2)|_BV(7));               // deaktivieren Brown Out ermoeglichen    Seite 38
84                                              // Konstanten fehlen in iotnx61.h
85      MCUCR |= _BV(7);                        // deaktivieren Brown Out
86      PRR |= ((1 <<PRTIM1)|(1<<PRTIM0)|(1<<PRUSI)|(1<<PRADC));
87                                              // Timer0, Timer1, USI, ADC deaktivieren  Seite 39
88      sei();                                  // Interrupts freigeben
89      theRCS.on();                            // Funksteckdose einschalten
90      for (int i = 0; i < 5; i++) {           // hektisch blinken
91          PORTA |= (1 << ledPin);             // LED ein
92          _delay_ms(1);                       // 1 Milliekunden warten
93          PORTA &= ~(1 << ledPin);            // LED aus
94          _delay_ms(99);
95      }
96      theRCS.off();                           // Funksteckdose ausschalten
97  }
98  /*  ------------------------------------------------------------------------------------------
99   main
100  ------------------------------------------------------------------------------------------ */
101 int main(void) {
102     init();
103     // Endlosschleife
104     while (1) {
105         if (!(PINB & (1 << buttonPin_0))) {     // ist Button_0 gedrueckt?
106             _delay_ms(10);                      // etwas warten (Entprellen)
107             if (!(PINB & (1 << buttonPin_0))) { // Button_0 immer noch gedrueckt?
108                 cli();                          // Interrups sperren, wegen des Timings der Funksignale
109                 theRCS.on();                    // Funksteckdose anschalten
110                 remainingTime = 8;              // fuer 4 Sekunden einschalten
111                 wdt_reset();// Watchdog zuruecksetzen                               Seite 47 ff
112                 WDTCR = _BV(WDIE) | _BV(WDP2) | _BV(WDP1);  // Watchdog starten ~ 1,2 s bei 3,8 V
113             }
114         }
115         if (!(PINA & (1 << buttonPin_1))) {     // ist Button_1 gedrueckt?
116             _delay_ms(10);                      // etwas warten (entprellen)
117             if (!(PINA & (1 << buttonPin_1))) { // Button_1 immer noch gedrueckt?
118                 cli();
119                 // Interrups sperren, wegen des Timings der Funksignale
120                 theRCS.off();                   // Funksteckdose ausschalten
121                 remainingTime = 0;              // Restzeit loeschen
122                 WDTCR = 0;                      // Watchdog ausschalten
123             }
124         }
125         sleep_enable(); // Sleep Befehl freigeben
126         sei();          // Interrups freigeben, sonst wacht der Controller nie wieder auf!
127         sleep_mode();   // Gute Nacht / wartet auf irgendeinen Interrupt, z.B. INT0, INT1, Watchdog
128         sleep_disable();// wieder wach! / Sleep Befehl sperren
129     };
130     return 0;           // sollte nie erreicht werden
131 }
132 /*  ------------------------------------------------------------------------------------------
133  ISR fuer den Interrupt INT0
134  ------------------------------------------------------------------------------------------ */
```

```
135 ISR(INT0_vect) {// hier gibt es nix zu tun, der Interrupt INT0 soll den Controller nur aufwecken
136 }
137 /*  ------------------------------------------------------------------------------------------------
138  ISR fuer den Interrupt INT1
139  ------------------------------------------------------------------------------------------------ */
140 ISR(INT1_vect) {// hier gibt es nix zu tun, der Interrupt INT1 soll den Controller nur aufwecken
141 }
142 /*  ------------------------------------------------------------------------------------------------
143  ISR fuer denInterrupt PCINT (Pin Change Interupt)
144  ------------------------------------------------------------------------------------------------ */
145 ISR(PCINT_vect) {// hier gibt es nix zu tun, der Interrupt PCINT soll den Controller nur aufwecken
146 // das GIFR (General Interrupt Flag Register) wird automatisch zurueckgesetzt
147 }
148 /*  ------------------------------------------------------------------------------------------------
149  ISR fuer den Watchdog Interrupt
150  ------------------------------------------------------------------------------------------------ */
151 ISR(WDT_vect) {
152     remainingTime--;                    // Restzeit reduzieren
153     wdt_reset();                        // Watchdog neu starten
154     PORTA |= (1 << ledPin);             // LED ein                  kurz blinken
155     _delay_ms(1);                       // 1 Millisekunde warten
156     PORTA &= ~(1 << ledPin);            // LED aus
157     if (remainingTime == 0) {           // Zeit ist agelaufen?
158         WDTCR = 0;                      // Watchdog ausschalten
159         theRCS.off();                   // Steckdose ausschalten
160     }
161 }
162
```


AKKU
1
2
+
-
J1
1
2
3
-
ISP
2
4
6
1
3
5
R_RESET
27k
IC1
LED1
150
R_LED
BUTTON_1
RESET
BUTTON_0
5
15
VCC
AVCC
11
12
13
14
17
18
19
20
PA7_(ADC6/AIN1/PCINT7)
PA6_(ADC5/AIN0/PCINT6)
PA5_(ADC4/AIN2/PCINT5)
PA4_(ADC3/ICP0/PCINT4)
PA3_(AREF/PCINT3)
PA2_(ADC2/INT1/USCK/SCL/PCINT2)
PA1_(ADC1/DO/PCINT1)
PA0_(ADC0/DI/SDA/PCINT0)
6
16
GND
AGND
1
2
3
4
7
8
9
10
(MOSI/DI/SDA/~OC1A/PCINT8)_PB0
(MISO/DO/OC1A/PCINT9)_PB1
(SCK/USCK/SCL/~OC1B/PCINT10)_PB2
(OC1B/PCINT11)_PB3
(ADC7/~OC1D/CLKI/XTAL1/PCINT12)_PB4
(ADC8/OC1D/CLKO/XTAL2/PCINT13)_PB5
(ADC9/INT0/T0/PCINT14)_PB6
(ADC10/~RESET/PCINT15)_PB7
ATTINY861A-PU