

```

#####
; LED Thermometer mit 1-Wire Sensor DS18S20 und 4 Digit LED-Anzeige, Platine TN20150816
; Hardware: ATtiny2313A @8MHz interner RC-Takt, Fuses: Low=E4h, High=DFh, Lock, Ext=FFh
; LEDs mit gemeinsamer Anode: Digits an PD0-PD3, Segmente (Kathoden) A-G, DP an PB0-PB7
; Sensor an PD6.
;
; Erstellt mit AVR-Studio 4.17 (c) T. Neveling: 04/09/2015 (V1.60)
#####
.include "tn2313Adef.inc" ; Definitionsdatei für ATtiny2313A CPU einbinden
.cseg ; Codesegment
;
; Register-Zuordnung
.def CELSIUS = r13 ; Temperatur aus DS18S20 Scratchpad
.def NULL = r14 ; immer = 0
.def intreg = r15 ; Zwischenspeicher für SREG bei Interrupt
.def temp1 = r16 ; Universalvariable
.def DIGIT1 = r17 ; Nachkommastelle: 0 oder 5
.def DIGIT2 = r18 ; Einerstelle mit festem Dezimalpunkt
.def DIGIT3 = r19 ; linkes Digit: Zehnerstelle oder "-"
.def DigiCnt = r20 ; zählt Digits zum Multiplexen durch
.def Data IO = r21 ; Byte vom/zum 1-Wire Sensor
.def Counter1= r22 ; Zähler für Delays bzw. Sensorbits
.def Counter2= r23
.def Blink = r24 ; Blinkender Punkt signalisiert Messfolge
;
#define parasite ; falls Sensor ohne eigene Power-Pins betrieben wird.

; Interrupt Vektoren
.org 0 rjmp start ; Reset Handler, Startadresse 0
.org OVFOaddr rjmp t0int ; 2,048 ms Timer0 Interrupts zum Multiplexen der LEDs
;
; Initialisierung
start:
    ldi temp1, LOW(RAMEND) ; Stackpointer setzen
    out SPL, temp1 ; LOW-Byte der obersten RAM-Adresse, kein SPH!
;
; Port B initialisieren 0 = Eingang, 1 = Ausgang
    ldi temp1, 0b11111111 ; 8 Segmentausgänge (Kathoden)
    out DDRB, temp1
    out PORTB, temp1 ; auf 1 = aus initialisieren
;
; Port D initialisieren: Bits 0-3: Digit Ausgänge (Anoden), Bit 6 = Sensor-I/O
    out DDRD, temp1
    out PORTD, temp1 ; Nur Ausgänge, ebenfalls alles auf 1
;
; Timer0 (8 Bit) zum Multiplexen der LEDs alle 2,048ms
    ldi temp1, (0<<CS02) | (1<<CS01) | (1<<CS00) ; Prescaler 1/64
    out tccr0b, temp1
    ldi temp1, 1<<toie0 ; Überlauf-Interrupt für Timer0 aktivieren
    out timsk, temp1 ; läuft ohne Preload alle 64/8MHz * 256 = 2,048ms über
;
; Variablen initialisieren
    clr NULL
    ldi Blink, $FF ; Nur Bit7 wird nachfolgend getoggelt (0/1)
    clr DigiCnt
    ldi DIGIT1, 6
    ldi DIGIT2, 1 ; Firmwareversion Ax.y (A für Assembler)
    ldi DIGIT3, $0A
    sei ; globale Interrupt Freigabe
;
main:
    rcall ONEWIRE_RST ; ### Endlos Schleife: Sensor auslesen / LED Ausgabe ###
    ldi Data IO, $CC ; Resetimpuls & Presence-Echo vom 1Wire Sensor
    rcall ONEWIRE_WR ; skip ROM command, da nur 1 Sensor vorhanden
    ldi Data IO, $44 ; Convert temperature command
    rcall ONEWIRE_WR ; dauert bis zu 750ms (s. Datenblatt Seite 3)
;
#ifdef parasite
    sbi PORTD, 6 ; bei Parasite Power: "Strong Pullup" am 1Wire-Bus,
    sbi DDRD, 6 ; um Spannungszufuhr aufrecht zu erhalten (Output/HIGH)
#endif
;
    clr counter2 ; wird in t0int alle 8,192ms inkrementiert
Wait Conv T:
    cpi counter2, 244 ; 244 * 8,192ms = 2s (für >750ms auf 92 abfragen)
    brne Wait Conv T
    subi BLINK, 128 ; Bit 7 des Registers toggeln
;

```

```
#ifndef parasite                ; bei Parasite Power:
    cbi PORTD, 6                ; Bus wieder zurück auf Input/ext. Pullup
    cbi DDRD, 6
#endif
;
    rcall ONEWIRE RST           ; wie oben
    ldi Data_IO, $CC            ; Skip ROM
    rcall ONEWIRE WR
    ldi Data_IO, $BE            ; read scratchpad command
    rcall ONEWIRE WR
    rcall ONEWIRE RD            ; Get data (bis zu 9 Byte auslesen)
; Testwerte aus Datenblatt Seite 6 - Folgezeile regulär auskommentieren!
; ldi Data_IO, $FF             ; AAh/00h (sign) = +85C, FFh/FFh = -0,5C
;
    mov CELSIUS, Data_IO        ; 1. Byte = Temperatur (zwischenspeichern)
    rcall ONEWIRE RD            ; 2. Byte = Vorzeichen einlesen (nur 00 oder FF)
; Testwerte aus Datenblatt Seite 6 - Folgezeile regulär auskommentieren!
; ldi Data_IO, $FF             ; AAh/00h (sign) = +85C, FFh/FFh = -0,5C
;
    tst Data_IO                 ; Vorzeichenbyte = 0?
    breq Put_Out_Plus          ; wenn ja, positive Temperatur
;
    mov Data_IO, CELSIUS        ; NEGATIVE TEMPERATUR: Wert wieder zurückholen
    neg Data_IO                 ; Zweierkomplement rückgängig machen
    rcall Celsius2BCD           ; Temperatur mit 0,5C Auflösung den Digits 1-3 zuweisen
    ldi DIGIT3, $0B            ; vorne steht ein Minuszeichen => Grenze daher -9,5C
    rjmp main
;
Put_Out_Plus:                  ; POSITIVE TEMPERATUR
    mov Data_IO, CELSIUS        ; Temperatur wieder zurückholen
    rcall Celsius2BCD           ; Temperatur mit 0,5C Auflösung den Digits 1-3 zuweisen
    rjmp main
;
;*****
; * Timer0 Interrupt: Display alle 2,048ms multiplexen *
;*****
t0int:
    push temp1                 ; wird in Muxen verändert
    in intreg, sreg             ; CPU-Flags sichern
    rcall Muxen
    out sreg, intreg            ; CPU-Flags wiederherstellen
    pop temp1
    reti
;
Muxen:
    ldi temp1, 0b11111111
    out PORTB, temp1            ; alle Segmente aus
    cbi PORTD, 0                ; alle Digits aus
    cbi PORTD, 1
    cbi PORTD, 2
    cbi PORTD, 3
;
    inc DigiCnt                 ; 1 Digit pro Aufruf ansteuern
    cpi DigiCnt, 1
    brne Muxen1
    sbi PORTD, 0                ; Digit0 einschalten
    ldi temp1, 0b10011110       ; "c" (Celsius-Zeichen), Punkt oben ist hardwired
    and temp1, BLINK            ; Bit 7 = Dezimalpunkt dazu
    out PORTB, temp1            ; Segmentpattern ausgeben
    ret                         ; fertig
;
Muxen1:
    cpi DigiCnt, 2
    brne Muxen2
    sbi PORTD, 1                ; Digit1 einschalten
    mov temp1, DIGIT1           ; Wert holen
    rcall Anzeige               ; Segmentpattern ausgeben
    ret                         ; fertig
;
Muxen2:
    cpi DigiCnt, 3
    brne Muxen3
    sbi PORTD, 2                ; Digit2 einschalten
    mov temp1, DIGIT2           ; Wert holen
    rcall Anzeige               ; Segmentpattern ausgeben
    cbi PORTB, 7                ; Dezimalpunkt dazu
    ret                         ; fertig
;
Muxen3:
    sbi PORTD, 3                ; Digit3 einschalten
```

```

    mov temp1, DIGIT3          ; Wert holen
    rcall Anzeige              ; Segmentpattern ausgeben
    clr DigiCnt                ; Wieder bei Digit0 beginnen
    inc Counter2               ; für Warteschleife nach Convert-T in main
    ret                        ; fertig

;
Anzeige:
    ldi ZL, LOW(Ziffern*2)     ; Z-Pointer wird nur hier benutzt (kein push/pop)
    ldi ZH, HIGH(Ziffern*2)    ; zeigt auf Tabellenanfang
    add ZL, temp1              ; Zahl (0...9) als Offset
    adc ZH, NULL               ; ZH/ZL zeigt jetzt auf Pattern-Adresse
    lpm temp1, Z               ; Zugehöriges Pattern aus Tabelle zurück nach temp1
    out PORTB, temp1          ; Segmentmuster ausgeben
    ret

;
Ziffern:
; Bit 0 = a bis Bit 6 = g, Bit 7 = DP, Bit gelöscht => Segment leuchtet.
;   0   1   2   3   4   5   6   7   8   9   A   -   <= LED-Anzeige
.DB $C0,$F9,$A4,$B0,$99,$92,$82,$D8,$80,$90,$88,$BF
;
;*****
;* 1WIRE Reset & Presence Echo: Timing siehe Sensor-Datenblatt Seite 13 *
;*****
;
; RESET |          //|          |////////////////////////|
;        |          |////////|          |////////////////////////|
;        <-----480µs-----><--66µs-->^<-- Wait Release 414µs-->
;
;
ONEWIRE RST:
;                Sample
    sbi DDRD,6
    cbi PORTD,6          ; 1Wire-Bus auf Output mit Pegel LOW
    ldi temp1,240
    rcall delayx2us      ; 480µs warten
    cbi DDRD,6          ; 1Wire-Bus auf Input (mit externem Pull-up-R)
    ldi temp1,33        ; 66µs warten
    rcall delayx2us      ; Nach spätestens 60µs zieht Sensor Signal für max 240µs auf 0
    clc                 ; CARRY-Flag löschen (wird hier nicht ausgewertet)
    sbis PIND,6         ; Sensor-Antwort einlesen (sollte LOW sein = PRESENCE Signal)
    sec                 ; Sensor vorhanden => CARRY-Flag gesetzt
    ldi temp1,207
    rcall delayx2us      ; 480-66µs warten bis Timeslot-Ende
    ret                 ; 1Wire-Bus Idle-State = Input (mit externem Pull-up-R)
;
;*****
;* 1 Byte aus Data IO an Sensor ausgeben: Timing siehe Sensor-Datenblatt Seite 14 *
;*****
;
; Write 1 |          |////////////////////////|
; -----|          |////////////////////////|
;        <-4µs-><-----66 µs----->
;
;
;        <---- sample typ. 60 µs -^
;
; Write 0 |          |////////|
; -----|          |////////|
;        <-----66µs-----><-4µs->
;
;
ONEWIRE WR:
    ldi counter1,8        ; 8 Bit schreiben
WR Loop:
    sbi DDRD,6
    cbi PORTD,6          ; 1Wire-Bus auf Output mit Pegel LOW
    sbrc Data_IO, 0      ; LSB zuerst: Unterscheidung ob Bit=0 oder 1
    rjmp Write_One       ; nächstes Bit ist "1"
    ldi temp1, 33        ; nächstes Bit ist "0"
    rcall delayx2us      ; 1Wire-Bus für 66µs auf LOW halten
    cbi DDRD,6          ; 1Wire-Bus auf Input (mit externem Pull-up-R)
    ldi temp1, 2         ; Rest des Timeslots abwarten
    rcall delayx2us      ; 4µs Pause
    rjmp WR_Next         ; nächstes Bit
Write One:
    ldi temp1, 2
    rcall delayx2us      ; 1Wire-Bus für 4µs auf LOW halten
    cbi DDRD,6          ; 1Wire-Bus auf Input (mit externem Pull-up-R)
    ldi temp1, 33        ; Rest des Timeslots abwarten
    rcall delayx2us      ; 66µs Pause
WR Next:
    lsr Data_IO          ; nächstes Bit in Position 0 schieben

```

```

    dec counter1                ; insgesamt 8 Bit
    brne WR_Loop
    ret                        ; 1Wire-Bus Idle-State = Input (mit externem Pull-up-R)
;
;*****
;* 1 Byte von Sensor nach Data IO einlesen: Timing siehe Sensor-Datenblatt Seite 14 *
;*****
;
; READ
; |//////////////////////////
; |////////////////DS18S20 zieht auf Low/High////////////////
; |<-2µs-><--16µs-->^<-----70µs----->
;
;
;                               Sample
ONEWIRE RD:
    ldi counter1,8              ; 8 Bit lesen
RD_Loop:
    sbi DDRD,6                  ; 1Wire-Bus auf Output mit Pegel LOW
    ldi temp1,1
    rcall delayx2us              ; 2µs Pause
    cbi DDRD,6                  ; 1Wire-Bus auf Input (mit externem Pull-up-R)
    ldi temp1, 8
    rcall delayx2us              ; nach 16µs 0 oder 1 einlesen
    clc
    sbic PIND,6                  ; 1 Bit lesen
    sec                          ; wenn Bit=0 => CARRY nicht gesetzt
    ror Data_IO                  ; 0/1 aus CARRY von links einschieben
    ldi temp1, 35                ; Timeslot-Ende abwarten
    rcall delayx2us              ; 70µs Pause
    dec counter1                ; insgesamt 8 Bit
    brne RD_Loop
    ret                        ; 1Wire-Bus Idle-State = Input (mit externem Pull-up-R)
;
;*****
;* Zeitschleifen für 8MHz Taktfrequenz: 0,125µs pro Takt, 8 Takte = 1µs *
;*****
delayx2us:
    push counter1                ; temp1 x 2µs verzögern (max 510µs)
    mov counter1,temp1           ; 1 Takt
    mov counter1,temp1           ; 1 Takt
Loop_2us:
    push counter1                ; Zeitschleife: 1 Durchlauf = 2µs Dauer (16 Takte)
    pop counter1                 ; 2 Takte
    push counter1                ; 2 Takte
    pop counter1                 ; 2 Takte
    push counter1                ; 2 Takte
    pop counter1                 ; 2 Takte
    push counter1                ; 2 Takte
    pop counter1                 ; 2 Takte
    nop                          ; 1 Takt
    dec counter1                 ; 1 Takt
    brne Loop_2us                ; 2 Takte => Summe 16 Takte = 2µs
    pop counter1                 ; 1 Takt
    ret                          ; 1 Takt
;
;*****
;* Binärwert aus Data IO in 3 Dezimalzahlen DIGIT1-3 umwandeln: *
;* Bit0 = Fixkomma (0 oder 5), Bits1-7 = Temperatur (2^0 bis 2^6) *
;*****
Celsius2BCD:
    BST Data_IO, 0                ; Temperatur-Register Format siehe Datenblatt Seite 3
    LSR Data_IO                  ; Fixkomma-Stelle (Bit0) retten => T-Flag
    ; jetzt Bit0 = 2^0 - Bit6 = 2^6, Bit7 = 0
;
    ldi DIGIT3, -1                ; Startwert zum nachfolgenden Inkrementieren
    ldi DIGIT2, 10                ; Startwert zum nachfolgenden Dekrementieren
D3Loop:
    inc DIGIT3                    ; Zehner-Stelle ermitteln => DIGIT3
    subi Data_IO, 10              ; DIGIT3 zählt Subtraktionen bis zum Negativ-Überlauf
    brcc D3Loop
    add DIGIT2, Data_IO           ; 10 plus Negativ-Überlauf = DIGIT2 (Einer-Stelle)
;
    brts C2BCD_x                 ; T-Flag gesetzt => DIGIT1 = 5 (Fixkomma-Stelle)
    ldi DIGIT1, 0                ; sonst DIGIT1 = 0
    ret
C2BCD_x:
    ldi DIGIT1, 5
    ret

```