

```

/*****
ATTiny44A LED Doppelwürfel Firmware zu Platine TN20160427. Für jeden Würfel werden je 7
LEDs einzeln oder paarweise von 4 PORTA-Ausgängen angesteuert. Die LED leuchten bei Bit=0.
PB0 = Taste zum Würfeln, PB1 = Jumper: Wenn gesteckt dann 2 Würfel, sonst 1 Würfel.
Der Prozessor wird mit dem internen RC-Oszillator bei 1MHz betrieben => Default Fusebits
V0.10: 06/05/2016 - Rumpfpogramm mit Zählerarithmetik und Anzeige (c) T. Neveling 144 Byte
V1.00: 06/05/2016 - Volle Grundfunktion 1 oder 2 Würfel ohne Schnickschnack 144 Byte
V2.00: 21/05/2016 - Power Down Mode & Pin Change Interrupt mit Ruhestrom <0,1uA! 170 Byte
V2.10: 23/05/2016 - Besserer Zufallsgenerator Dauer des Tastendrucks entscheidet 170 Byte
*****/
;
.include "tn44Adef.inc" ; Definitionsdatei für ATtiny44A CPU einbinden
;
; Register-Zuordnung
.def temp = r16 ; Universalregister
.def Augen = r17 ; Augenzähler im packed BCD-Format für beide Würfel
.def Pattern = r18 ; entsprechendes Bitmuster zur LED-Ansteuerung
.def T1L = r19 ; zum Auslesen des 16 Bit Timers1 (Low/High)
.def T1H = r20
;
.cseg ; Codesegment
;
; Interrupt Vektoren
.org 0 rjmp start ; Reset Handler, Startadresse 0
.org PCIE1 rjmp PinChgInt1 ; Pin Change Interrupt (hier für PB0)
;
; Initialisierung: Der Stapelzeiger steht beim ATtiny44 per default auf RAMEND!
start:
    ldi temp, 0b11111111 ; Ports initialisieren:
    out DDRA, temp ; 8 LED Ausgänge für 2x7 Leuchtdioden
    ldi temp, 0b00000001 ; PortB: Nur Eingänge (default), PB0=Taste, PB1=Jumper
    out PortB, temp ; Pull-up Widerstand zunächst nur an PB0
;
; Variablen initialisieren: Pattern und Augen stehen nach dem LED-Test wieder auf 0.
    clr Pattern ; Register stehen bei Power-on nicht zwangsläufig auf 0!
    clr Augen ; Also hier erst mal löschen.
Loop:
    nop ; Delay (4x255 +4) x 255 = 261120 Zyklen = 261ms @1MHz
    dec Pattern ; 1
    brne Loop ; 1
    nop ; 2(1)
    dec Augen ; 1
    brne Loop ; 1
    ldi temp, $FF ; 2(1)
    out PortA, temp ; alle LED aus
;
    ldi temp, 1<<PCIE1 ; Pin Change Interrupt 1 aktivieren
    out GIMSK, temp
    ldi temp, 1<<PCINT8 ; PCINT8 = PB0 => Würfeltaste ist Auslöser
    out PCMSK1, temp
    sei ; Globale Interrupt Freigabe
;
    ldi temp, (1<<SE) | (1<<SM1) ; Sleep-Modus "Power Down" vorbereiten
    out MCUCR, temp
;
main:; ##### Hauptprogramm - Endlos-Schleife #####
    sleep ; Sleep aktivieren => Druck auf Würfeltaste abwarten
    rjmp main
;
LED Muster:
    .db 0b11101110, 0b11011101 ; 1, 2 Der Augenzähler läuft von 0 - 5.
    .db 0b11001100, 0b10011001 ; 3, 4 Bit gelöscht => LED leuchtet
    .db 0b10001000, 0b00010001 ; 5, 6 gleiche Muster für beide Würfel
;
GetPattern:
    andi temp, 0b00000111 ; LED-Muster 0..5 in temp zurück nach temp
    ldi ZL, LOW (LED Muster*2) ; nur jeweils 6 Werte eines Würfels betrachten
    ldi ZH, HIGH (LED Muster*2)
    add ZL, temp ; der HIGH-Wert bleibt hier=0
    lpm temp, Z
    ret
;
Anzeige:
    mov temp, Augen
    swap temp ; Würfel 1 <=> Würfel 2
    rcall GetPattern ; LED-Muster für Würfel 1 holen
    ori temp, 0b11110000 ; Würfel 2 (oberes Nibble) aus, LED sind 0 aktiv

```

```
mov Pattern, temp                ; zwischenspeichern
sbic PINB,1                      ; 1 oder 2 Würfel?
rjmp a01
mov temp, Augen
rcall GetPattern                 ; LED-Muster für Würfel 2 holen
ori temp, 0b00001111            ; Würfel 1 (unteres Nibble) aus
and Pattern, temp               ; beide Ergebnisse in einem Byte
a01:out PortA, Pattern           ; LEDs einschalten
ret

;
wuerfeln:
clr temp                        ; Zeitschleife, damit die Würfel "rollen"
w01:dec temp                    ; 1
push Pattern                    ; 2
pop Pattern                     ; 2
tst temp                        ; 1
brne w01                        ; 2(1) => 8 x 255 Zyklen zu 1MHz = 2ms
out TCNT1H, temp                ; 4s Zähler für nachfolgende Anzeige löschen (temp=0)
out TCNT1L, temp

inc Augen                       ; beide Würfel hochzählen: 0-0, 0-1, 0-2, 0-3,
mov temp, Augen                 ; 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0..
andi temp, 0b00000111          ; Würfelergebnis in Arbeitsregister holen
cpi temp, 6                     ; Würfel 1 (untere 3 Bit betrachten)
brne wx1                        ; Wert = 6?
mov temp, Augen                ; kein Problem wenn nicht
andi temp, 0b11110000          ; Würfel 1 Überlauf, Würfelergebnis zurück holen
subi temp, -$10                 ; Würfel 1 löschen
mov Augen, temp                 ; Würfel 2 inkrementieren
cpi Augen, $60                  ; neuen Würfelstand speichern
brne wx1                        ; Überlauf Würfel 2?
clr Augen                       ; falls ja, beide Würfel auf Null (= Würfelmuster 1)
wx1:ret

;
PinChgInt1:
sbi PORTB, 1                    ; Die Würfeltaste an PB0 wurde betätigt.
sbic PINB,0                     ; Pull-up Widerstand an Jumpereingang (1 oder 2 Würfel)
rjmp p01                        ; Ein Pin Change tritt 2x pro Tastendruck auf!
                                ; beim Loslassen der Taste den Anzeigetimer starten.

;
loop1:
rcall wuerfeln                  ; Beim Drücken der Taste:
rcall Anzeige                   ; Würfeln solange Taste gedrückt bleibt
sbis PINB,0
rjmp loop1
rjmp p0x

;
p01:sei                          ; Interrupts für weitere Tastendrücke frei
ldi temp, (1<<CS10)|(1<<CS11)  ; 16 Bit Timer starten: Ergebnis anzeigen => LED aus
out TCCR1B, temp                ; 1:64 Prescaler => 64 x 65536 / 1MHz = 4,19s

loop2:
in T1L, TCNT1L
in T1H, TCNT1H                  ; Timer auslesen
cpi T1H, $FF                    ; FF00 = 65280 ~ 4s (65280 x 1us x 64 = 4,178s)
brlo loop2                      ; Würfelergebnis für 4s anzeigen

;
out PortA, T1H                  ; T1H = $FF => alle LED aus
clr temp
out TCCR1B, temp                ; Zähler stopp
cbi PORTB, 1                    ; Pull-up Widerstand für Jumper ausschalten (Power save!)
p0x:reti
```