SCHALLORTUNG

```
;  =>FUSE LOW  BYTE:    2MHz  (0xE2)
;
;  +-------------------------------------------------------+
;  |FUSE LOW BYTE:                                (S. 30,223) |
;  |                                                       |
;  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |FREQ| HEX  |
;  |=======================================================|
;  |  1  |  1  |  1  |  0  |  0  |  0  |  0  |  1  |1MHz| 0xE1 |
;  |-----+-----+-----+-----+-----+-----+-----+-----+----------|
;  |  1  |  1  |  1  |  0  |  0  |  0  |  1  |  0  |2MHz| 0xE2 |
;  |-----+-----+-----+-----+-----+-----+-----+-----+----------|
;  |  1  |  1  |  1  |  0  |  0  |  0  |  1  |  1  |4MHz| 0xE3 |
;  |-----+-----+-----+-----+-----+-----+-----+-----+----------|
;  |  1  |  1  |  1  |  0  |  0  |  1  |  0  |  0  |8MHz| 0xE4 |
;  +-------------------------------------------------------+


;  BEMERKUNGEN:
;
;  -  Diff_Max_H:L
;           -  noch belassen, ggf. später für minimalen
;              Lautstärkepegel verwenden
;
;  -  Sample-Rate = 20KHz pro Kanal
;  -> Erkennung von Signalen bis ca. 10KHz
;  -  Aufnahme von 2 x 250 ADC-Werten im SRAM
;
;
;-------------------------------------------------------------


.include "m8def.inc"


;###  EQU-DEFINITIONEN  ###

;###  PORT B   ###
                              ;für LED's:
.equ  GRUEN          =  0     ;Kalibrationsphase; Winkelerkennung
.equ  ORANGE         =  1     ;Rechenphase
.equ  ROT            =  2     ;Aufnahmephase


;###  WrCOM_2  ###

.equ  TXD            =  4     ;TXD = MISO an PB4


;###  SONSTIGES   ###

.equ  Durchlauf      =  80    ;Anzeigedauer von Diff_Step
.equ  Pegeldiff      =  3     ;Pegeldifferenz zwischen Stille
                              ;und niedrigster Signalerkennung
                              ;(Ansprechempfindlichkeit)


;###  VARIABLEN-DEFINITIONEN  ###

.def  Sicher_1    =  r0       ;für  U-Programme
.def  Sicher_2    =  r1
```

```
.def  Sicher_ITR1 =  r2       ;für ITR's
.def  Sicher_ITR2 =  r3

.def  Diff_Min_H  =  r4       ;für Winkelbestimmung
.def  Diff_Min_L  =  r5
.def  Diff_Step   =  r6       ;für aktuelle Schallrichtung

.def  Diff_Max_H  =  r7       ;ggf. für Autokalibration
.def  Diff_Max_L  =  r8

.def  Step_alt    =  r9       ;Vorwert von Step
.def  Stille      =  r10      ;Pegel für Stille



.def  i           =  r16      ;Hilfsregister
.def  j           =  r17
.def  k           =  r18
.def  m           =  r19
.def  n           =  r20
.def  p           =  r21


.def  Delay       =  r22      ;für WrCOM_2
.def  Count       =  r23

.def  Step        =  r24
.def  Dauer       =  r25      ;für Anzeigedauer von Diff_Step



;  r29:r28 reserviert für Y!
;  r31:r30 reserviert für Z!


;#################################################################

rjmp  ANFANG

;.org    0x0000
;       rjmp  ITR-VEKTOR



ANFANG:

   cli

   ldi   i,LOW(RAMEND)        ;Stackpointer setzen
   out   SPL,i
   ldi   i,HIGH(RAMEND)
   out   SPH,i

;port b
   sbi   ddrb,TXD             ;Datenrichtung TXD für WrCOM (pb4)

   sbi   ddrb,GRUEN           ;LED's:  (pb0)
   sbi   ddrb,ORANGE          ;        (pb1)
   sbi   ddrb,ROT             ;        (pb2)


   sbi   ddrb,6               ;für 7-Segment-Anzeige (10er)
   sbi   ddrb,7               ;dto.

;port c
```

```
            ;76543210
   ldi   i,0b00111100         ;für 7-Segment-Anzeige (10er)
   out   ddrc,i               ;pc0:    ADC-Eingang links
                              ;pc1:    ADC-Eingang rechts

;port d
            ;76543210
   ldi   i,0b01111111         ;für 7-Segment-Anzeige (1er)
   out   ddrd,i




   rcall ADC_START            ;ADC initialisieren


   clr   Step_alt
   clr   Dauer
   clr   Stille

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


LOOP_MAIN:

;STILLE EINMALIG KALIBRIEREN

   tst   Stille               ;Stille=0 nur direkt nach Reset!
   breq  WARTEN_STILLE
   rjmp  MESSWERTAUFNAHME


WARTEN_STILLE:                ;Warten wegen Schalter-Klick
                              ;LED-GRUEN während Kalibration
   sbi   portb,GRUEN          ;angeschaltet
   ldi   i,150
   rcall WARTEN


MESSWERTAUFNAHME:

   sbi   portb,ROT            ;LED-ROT: AUFNAHME-PHASE

   ldi   YH,HIGH(0x0060)      ;SRAM-Anfangsadresse -> Y
   ldi   YL,LOW(0x0060)

   clr   j                    ;j zählt die Samples (250/Kanal)

LOOP_INPUT:

   rcall ADC_Rd_LEFT          ;i <- ADC(left,8bit)
   st    Y+,i                 ;(Y) <- i, Y <- Y+1

   rcall ADC_Rd_RIGHT         ;i <- ADC(right,8bit)
   st    Y+,i                 ;(Y) <- i, Y <- Y+1

   inc   j                    ;j <- j+1
   cpi   j,250

brlo  LOOP_INPUT              ;nur 250 Werte/Kanal einlesen

   cbi   portb,ROT

;-----------------------
```

```
;EINMALIGE STILLE-PEGEL-BERECHNUNG

   tst   Stille               ;Stille=0 nur direkt nach Reset!
   breq  PEGEL_STILLE
   rjmp  TEST_MIN_PEGEL


PEGEL_STILLE:                 ;Berechnung der maximalen Amplitude
                              ;des Rauschens in Ruhe
   ldi   YH,HIGH(0x0060)      ;SRAM-Anfangsadresse -> Y
   ldi   YL,LOW(0x0060)

   clr   i
   clr   j


LOOP_STILLE:
   inc   i
   cpi   i,251                ;am Ende angelangt?
   brsh  STILLE_DIFF

   ld    j,Y+                 ;j <- (Y), Y <- Y + 1
   cp    Stille,j
   brsh  KEINE_AKTUAL         ;Stille >= (Y) ?

   mov   Stille,j             ;Stille <- (Y)


 KEINE_AKTUAL:
   ld    j,Y+                 ;j <- (Y), Y <- Y + 1
   cp    Stille,j
   brsh  LOOP_STILLE          ;Stille >= (Y) ?

   mov   Stille,j             ;Stille <- (Y)
   rjmp  LOOP_STILLE



STILLE_DIFF:                  ;Trennung des Maximalpegels
                              ;von Stille-Pegel
   mov   i,Stille
   cpi   i,255-Pegeldiff      ;Stille < 255-Pegeldiff ?
   brlo  STILLE_PLUS


STILLE_255:
   ldi   i,255
   mov   Stille,i             ;Stille <- 255
   rjmp  EXIT_STILLE_DIFF


STILLE_PLUS:
   ldi   i,Pegeldiff          ;Stille <- Stille + Pegeldiff
   add   Stille,i

EXIT_STILLE_DIFF:


   cbi   portb,GRUEN          ;LED-GRUEN aus!

;----------

TEST_MIN_PEGEL:               ;Lautstärke hoch genug?
                              ;es genügt ein einziger
                              ;Wert mit >= Stille
```

```
   ldi   YH,HIGH(0x0060)      ;SRAM-Anfangsadresse -> Y
   ldi   YL,LOW(0x0060)

   clr   i
   clr   j

LOOP_MIN_PEGEL:

   inc   i
   cpi   i,251                ;am Ende angelangt?
   brlo  MIN_PEGEL_WEITER

   ldi   j,0
   mov   Diff_Step,i          ;Diff_Step <- 0
   ldi   j,1
   mov   Step_alt,j           ;Step_alt  <- 1

   rjmp  EXIT_LAUFZEIT        ;KEINE AUSGABE!


 MIN_PEGEL_WEITER:
   ld    j,Y+                 ;j <- (Y), Y <- Y + 1
   cp    j,Stille             ;es genügt ein höherer Wert!
   brsh  INITIALISIERUNG

   ld    j,Y+                 ;j <- (Y), Y <- Y + 1
   cp    j,Stille             ;es genügt ein höherer Wert!
   brsh  INITIALISIERUNG

rjmp  LOOP_MIN_PEGEL



;BERECHNUNG LAUFZEITDIFFERENZ

INITIALISIERUNG:


   sbi   portb,ORANGE         ;LED-ORANGE: RECHEN-PHASE

   ldi   i,0xFF
   mov   Diff_Min_H,i         ;Diff_Min_H:L  <- 0xFFFF
   mov   Diff_Min_L,i

   clr   Diff_Max_H           ;Diff_Max_H:L  <- 0x0000
   clr   Diff_Max_L

   clr   Diff_Step            ;0 = keine Winkel-Position!
   clr   Step                 ;0 = Anfangswert

LOOP_LINKS:
                              ;Y(Links)  variiert Startposition
                              ;Z(Rechts) fest

   cpi   Step,12              ;if Step>=12 -> LOOP_RECHTS
   brsh  LOOP_RECHTS

   inc   Step                 ;Step <- Step + 1
   rcall Y_POSITION           ;Y    <- 0x60 + 24 - 2 x Step

   clr   k
   ldi   ZH,HIGH(0x0061)      ;Z <- 0x0061
   ldi   ZL,LOW(0x0061)

   clr   j                    ;[j:i] <- 0x0000
   clr   i
```

```
DIFF_LINKS:

   inc   k                    ;k <- k + 1
   cpi   k,240                ;if k>=240 -> AKT_DIFF_LI
   brsh  AKT_DIFF_LI

   ld    m,Y                  ;m <- (Y)      links
   ld    n,Z                  ;n <- (Z)      rechts


   cp    m,n                  ;[j:i] <- [j:i] + |(Y)-(Z)|
   brlo  M_KLEINER_LI

M_GROESSER_GLEICH_LI:

   sub   m,n                  ;m <- m - n
   clr   n                    ;n <- 0

   add   i,m                  ;[j:i] <- [j:i] + [0x00:m]
   adc   j,n

   rjmp  DIFF_RDY_LI

M_KLEINER_LI:

   sub   n,m                  ;n <- n - m
   clr   m                    ;m <- 0

   add   i,n                  ;[j:i] <- [j:i] + [0x00:n]
   adc   j,m

DIFF_RDY_LI:

   adiw  YL,2                 ;Y <- Y+2
   adiw  ZL,2                 ;Z <- Z+2


   rjmp  DIFF_LINKS


AKT_DIFF_LI:

;Diff_MAX_AKTUAL_LI           ;Diff_Max_H:L ggf. aktualisieren
                              ;NOCH NICHT VERWENDET!
   cp    Diff_Max_L,i
   cpc   Diff_Max_H,j

   brsh  DIFF_MIN_AKTUAL_LI   ;Diff_Max_H:L >= [j:i] ?

   mov   Diff_Max_L,i         ;Diff_Max_H:L <- [j:i]
   mov   Diff_Max_H,j


DIFF_MIN_AKTUAL_LI:           ;Diff_Min_H:L ggf. aktualisieren

   cp    i,Diff_Min_L
   cpc   j,Diff_Min_H

   brsh  LOOP_LINKS           ;[j:i] >= Diff_Min_H:L ?

   mov   Diff_Min_L,i
   mov   Diff_Min_H,j         ;Diff_Min_H:L <- [j:i]

   mov   Diff_Step,Step       ;Winkel aktualisieren
```

```
   rjmp  LOOP_LINKS


;--------

LOOP_RECHTS:
                              ;Y(Links)  fest
                              ;Z(Rechts) variiert Startposition

   cpi   Step,23              ;if Step>=23 -> EXIT_LAUFZEIT
   brsh  EXIT_LAUFZEIT

   inc   Step                 ;Step <- Step + 1
   rcall Z_POSITION           ;Z    <- 0x60 - 23 + 2 x Step

   clr   k
   ldi   YH,HIGH(0x0060)      ;Y <- 0x0060
   ldi   YL,LOW(0x0060)

   clr   j                    ;[j:i] <- 0x0000
   clr   i


DIFF_RECHTS:

   inc   k                    ;k <- k + 1
   cpi   k,240                ;if k>=240 -> AKT_DIFF_RE
   brsh  AKT_DIFF_RE

   ld    m,Y                  ;m <- (Y)      links
   ld    n,Z                  ;n <- (Z)      rechts


   cp    m,n                  ;[j:i] <- [j:i] + |(Y)-(Z)|
   brlo  M_KLEINER_RE

M_GROESSER_GLEICH_RE:

   sub   m,n                  ;m <- m - n
   clr   n                    ;n <- 0

   add   i,m                  ;[j:i] <- [j:i] + [0x00:m]
   adc   j,n

   rjmp  DIFF_RDY_RE

M_KLEINER_RE:

   sub   n,m                  ;n <- n - m
   clr   m                    ;m <- 0

   add   i,n                  ;[j:i] <- [j:i] + [0x00:n]
   adc   j,m

DIFF_RDY_RE:

   adiw  YL,2                 ;Y <- Y+2
   adiw  ZL,2                 ;Z <- Y+2

   rjmp  DIFF_RECHTS


AKT_DIFF_RE:
```

```
;Diff_MAX_AKTUAL_RE          ;Diff_Max_H:L ggf. aktualisieren
                            ;NOCH NICHT VERWENDET!
   cp    Diff_Max_L,i
   cpc   Diff_Max_H,j

   brsh  DIFF_MIN_AKTUAL_RE  ;Diff_Max_H:L >= [j:i] ?

   mov   Diff_Max_L,i        ;Diff_Max_H:L <- [j:i]
   mov   Diff_Max_H,j


DIFF_MIN_AKTUAL_RE:         ;Diff_Min_H:L ggf. aktualisieren

   cp    i,Diff_Min_L
   cpc   j,Diff_Min_H

   brsh  LOOP_RECHTS         ;[j:i] >= Diff_Min_H:L ?

   mov   Diff_Min_L,i
   mov   Diff_Min_H,j        ;Diff_Min_H:L <- [j:i]

   mov   Diff_Step,Step      ;Winkel aktualisieren

   rjmp  LOOP_RECHTS

;-----------------------

EXIT_LAUFZEIT:

   cbi   portb,ORANGE

;STEP_CHECK

   cp    Step_alt,Diff_Step
   breq  MATCH_STEP
   rjmp  NO_MATCH_STEP


NO_MATCH_STEP:              ;anderer Step-Wert !

   mov   Step_alt,Diff_Step ;Step_alt aktualisieren

   cpi   Dauer,Durchlauf    ;Durchläufe, bevor alter
   brsh  DISP_CLR           ;Winkel wieder gelöscht wird

DISP_BLEIBT:

   inc   Dauer              ;Dauer <- Dauer + 1
   rjmp  ENDE_STEP_CHECK


DISP_CLR:

   cpi   Dauer,Durchlauf+1  ;ja: Anzeige wurde schon gelöscht
   breq  ENDE_STEP_CHECK

   ldi   Dauer,Durchlauf+1  ;Dauer <- Durchlauf + 1

   ldi   i,0                ;7-Segment-Anzeige einmalig löschen
   rcall LED_DISPLAY

   ldi   i,0                ;Anzeige einmalig löschen
   rcall WrCOM_2
                            ;Ausgabe-Pause
   rjmp  ENDE_STEP_CHECK
```

```
MATCH_STEP:

   sbi   portb,GRUEN          ;LED-GRUEN: WINKEL ERKANNT!

   clr   Dauer                ;Dauer <- 0

   mov   i,Step_alt           ;7-Segment-Anzeige: Step_alt
   rcall LED_DISPLAY

   mov   i,Step_alt           ;OUTPUT: Step_alt
   rcall WrCOM_2


   rjmp  ENDE_STEP_CHECK



ENDE_STEP_CHECK:

   ;ldi  i,5                  ;hier ggf. Verzögerung der
   ;rcall   WARTEN            ;nächsten Messung

   cbi   portb,GRUEN

rjmp  LOOP_MAIN




ENDE:
rjmp  ENDE


;=============================================================

WrCOM_2:                   ;### Für 2MHz !!! ###

        in    Sicher_1,SREG

        sbi   portb,TXD    ;Senden (Übergabewert i)

        ldi   Delay,33
D4:     dec   Delay
        brne  D4

        ldi   Delay,33     ;zzgl. für 2MHz
D4_2:   dec   Delay
        brne  D4_2

        ldi   Count,8
L2:     sbrc  i,0
        rjmp  OFF
        rjmp  ON
ON :    sbi   portb,TXD
        rjmp  BitD
OFF:    cbi   portb,TXD
        rjmp  BitD
BitD:   ldi   Delay,33
D5:     dec   Delay
        brne  D5

        ldi   Delay,33     ;zzgl. für 2MHz
D5_2:   dec   Delay
```

```
        brne  D5_2

        lsr   i
        dec   Count
        brne  L2
        cbi   PORTB,TXD

        ldi   Delay,33
D6:     dec   Delay

        brne  D6

        ldi   Delay,33     ;zzgl. für 2MHz
D6_2:   dec   Delay
        brne  D6_2

        out   SREG,Sicher_1

        ret

;-----------------------------------------------------------
ADC_START:

   in    Sicher_1,SREG

            ;76543210
   ldi   i,0b01100000
   out   ADMUX,i              ;b7=0 & b6=1: AVCC 5,0V stab.
                              ;b5(ADLAR)=1: left adjust AN!
                              ;b4:          keine Funktion
                              ;b3..0:
                              ;       0000: ADC0 (links)  AN
                              ;       0001: ADC1 (rechts) AUS

            ;76543210
   ldi   i,0b10000000
   out   ADCSRA,i             ;b7=1(ADEN) :ADC enable ein !
                              ;b6=0(ADSC) :No first Conv.
                              ;b5=0(ADFR) :free running aus!
                              ;b4=0(ADIF) :ADC-ITR-Flag(read)
                              ;b3=0(ADIE) :ADC-ITR-Enable aus!

                              ;b2..0=001  :2MHz/2=1MHz

   out   SREG,Sicher_1
ret
;-----------------------------------------------------------
ADC_Rd_LEFT:

   in Sicher_1,SREG           ;Ausgabe: i

            ;76543210
   ldi   i,0b01100000         ;LINKER KANAL
   out   ADMUX,i

   sbi   ADCSRA,ADSC          ;Start Conversion!

LEFT_RDY:
   sbic  ADCSRA,ADSC          ;Conversion beendet?
   rjmp  LEFT_RDY
                              ;ggf. weitere Conversion abwarten..!
   in    i,ADCH

   out   SREG,Sicher_1
ret
```

```
;-----------------------------------------------------------
ADC_Rd_RIGHT:

   in Sicher_1,SREG           ;Ausgabe: i

            ;76543210
   ldi   i,0b01100001         ;RECHTER KANAL
   out   ADMUX,i

   sbi   ADCSRA,ADSC          ;Start Conversion!

RIGHT_RDY:
   sbic  ADCSRA,ADSC          ;Conversion beendet?
   rjmp  RIGHT_RDY
                              ;ggf. weitere Conversion abwarten..!
   in    i,ADCH

   out   SREG,Sicher_1
ret

;-----------------------------------------------------------
WARTEN:
                              ;Warteschleife
                              ;mit i(Übergabewert),j,k
   in    Sicher_1,SREG

   ;ldi  i,255

LOOP_i:
   ldi   j,255
   LOOP_j:
      ldi   k,20
      LOOP_k:
         dec   k
      brne  LOOP_k
      dec   j
   brne  LOOP_j
   dec   i
brne  LOOP_i

   out   SREG,Sicher_1
ret
;-------------------------------------------------------------

Y_POSITION:                   ;Y <- 0x0060 + 24 - 2 x Step

   in    Sicher_1,SREG

   ldi   YH,HIGH(0x0060)      ;Y <- 0x0060
   ldi   YL,LOW(0x0060)

   adiw  YL,24                ;Y <- Y + 24

   mov   i,Step               ;i <- Step

Y_SUB:
   tst   i                    ;i <= 0 ?
   breq  Y_SUB_EXIT
   SBIW  YL,2                 ;Y <- Y - 2
   dec   i                    ;i <- i - 1
   rjmp  Y_SUB

Y_SUB_EXIT:

   out   SREG,Sicher_1
```

```
ret

;---------------------------------------------------------------

Z_POSITION:                     ;Z <- 0x0060 - 23 + 2 x Step

   in    Sicher_1,SREG

   ldi   ZH,HIGH(0x0060)        ;Z <- 0x0060
   ldi   ZL,LOW(0x0060)

   sbiw  ZL,23                  ;Z <- Z - 23

   mov   i,Step                 ;i <- Step

Z_ADD:
   tst   i                      ;i <= 0 ?
   breq  Z_ADD_EXIT
   ADIW  ZL,2                   ;Z <- Z + 2
   dec   i                      ;i <- i - 1
   rjmp  Z_ADD

Z_ADD_EXIT:

   out   SREG,Sicher_1

ret

;---------------------------------------------------------------


LED_DISPLAY:

   in    Sicher_1,SREG

   cpi   i,0                    ;Anzeige dunkel
   brne  GROESSER_30
   ldi   j,0
   out   portd,j

   cbi   portc,2
   cbi   portc,3
   cbi   portc,4
   cbi   portc,5

   cbi   portb,6
   cbi   portb,7

   rjmp  ENDE_LED_DISPLAY

GROESSER_30:
   cpi   i,30
   brlo  GROESSER_20

;ERROR
   ldi   j,0b00001100           ;[E][r]
   out   portd,j

   sbi   portc,2
   sbi   portc,3
   sbi   portc,4
   cbi   portc,5

   sbi   portb,6
   sbi   portb,7
```

```
   rjmp  ENDE_LED_DISPLAY

GROESSER_20:
   cpi   i,20
   brlo  GROESSER_10

   sbi   portc,2              ;[2]
   sbi   portc,3
   sbi   portc,4
   sbi   portc,5

   cbi   portb,6
   sbi   portb,7

   subi  i,20
   rjmp  STELLE_1er

GROESSER_10:
   cpi   i,10
   brlo  KLEINER_10

   cbi   portc,2              ;[1]
   cbi   portc,3
   cbi   portc,4
   cbi   portc,5

   sbi   portb,6
   sbi   portb,7

   subi  i,10
   rjmp  STELLE_1er

KLEINER_10:
   cbi   portc,2              ;[ ]
   cbi   portc,3              ;(keine führende Null)
   cbi   portc,4
   cbi   portc,5

   cbi   portb,6
   cbi   portb,7

   rjmp  STELLE_1er



STELLE_1er:

ZAHL_9:
   cpi   i,9
   brne  ZAHL_8
   ldi   j,0b01111011         ;[9]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_8:
   cpi   i,8
   brne  ZAHL_7
   ldi   j,0b01111111         ;[8]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_7:
   cpi   i,7
   brne  ZAHL_6
   ldi   j,0b01010010         ;[7]
```

```
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_6:

   cpi   i,6
   brne  ZAHL_5
   ldi   j,0b01101111         ;[6]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_5:
   cpi   i,5
   brne  ZAHL_4
   ldi   j,0b01101011         ;[5]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_4:
   cpi   i,4
   brne  ZAHL_3
   ldi   j,0b00111010         ;[4]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_3:
   cpi   i,3
   brne  ZAHL_2
   ldi   j,0b01011011         ;[3]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_2:
   cpi   i,2
   brne  ZAHL_1
   ldi   j,0b01011101         ;[2]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_1:
   cpi   i,1
   brne  ZAHL_0
   ldi   j,0b00010010         ;[1]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY

ZAHL_0:
   ldi   j,0b01110111         ;[0]
   out   portd,j
   rjmp  ENDE_LED_DISPLAY


ENDE_LED_DISPLAY:

   out   SREG,Sicher_1

ret

;-------------------------------------------
```