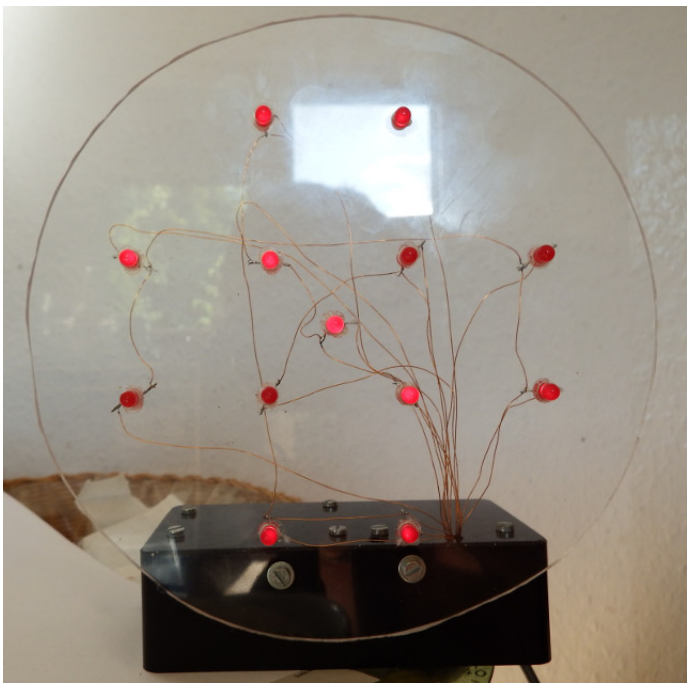
Benutzanleitung fur

Magic Circle



von

Schmidt Microsystems
Darmstadt/Germany
http://www.avr-asm-tutorial.net/avr_de/apps/magic_circle_tn2313/magic_circle.html

Mai 2017

Herzliche Gluckwunsch
fur Erwerb von Magic Circle!

Diese Gerat macht nix, nur Spass.

Ansicht von Gerat

Gerat macht Lichte an, aber nicht mit schnell an- und ausschalten. Hell kommt mit ganz leicht und steigt sanft. Bis Spitze hell, dann wieder langsam herunter bis Aus. Macht sonst kein ander Gerat, ist Erfindung von Schmidt Microsystems.

Gebrauch von Gerat

Stromversorgung:

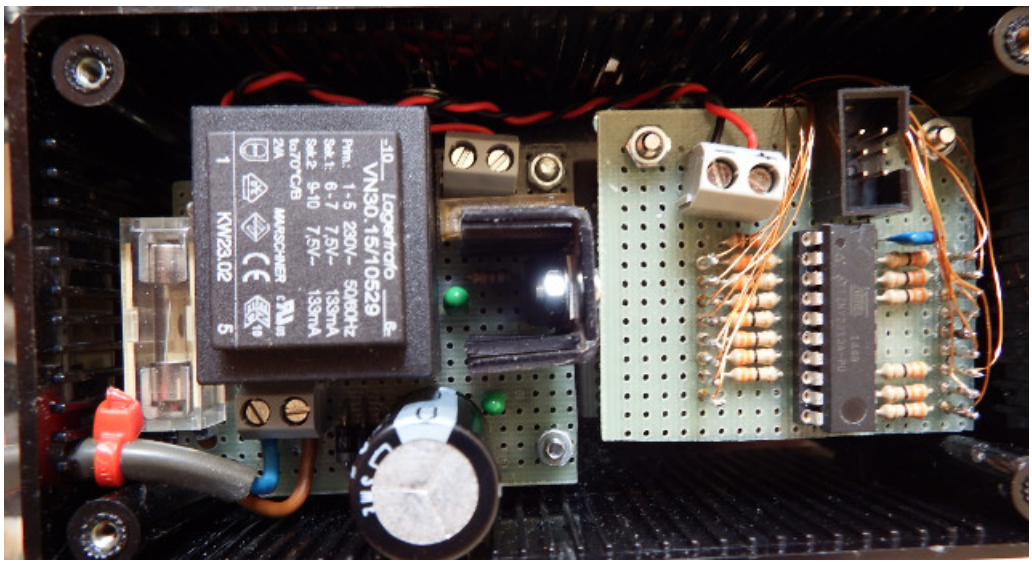
> Strom kommt aus Steckerdose 230 Volte. Musst nur Stecker rein, nix Schalter fur Ein. Verbrauch ist maximal 1 Watt, macht 8,760 Kilowattstunde fur ganze Jahr.

Funktionieren:

> Gerat ist gebaut mit modernes Mikroprofessor ATtiny2313 von ATMEL, was mit 1 Million Takte in jede Sekunde Arbeit verrichtet. Hat viel Speicher mit 2.048 Bytes Programm (aber nur ein Drittel voll), hat sehr geringes Spannung (3,3 Volte) und Stromverbrauch (0,2 Milliampere), weil ist meiste Zeit nicht in Arbeit sondern in Schlaf wegen geniales und ausgebufftes Schmidt-Microsystems-Software-Design. Ist fast unkaputtbar, ausser wenn in Klo oder Badewanne oder aus Flugzeug werfen.

Repariere:

> Kannst offnen Schachtel fur Reparieren. Musst aber ziehen Stecker von Strom vorher. Gerat hat Sicherung vor zu viel Strom, kannst rausholen und anderes reinmachen (musst kaufen Sicherung 50 mA langsam). Sonst nix fur Reparieren in Schachtel.



Kwallidaed:

Hat getestet Dauerstress mit Stromverbrauch (100 mA ueber zwei Stunden heiss): nix Ausfall.
Nix ISO9000, nix Euro-Prufsiegel, bix TUV, weil sowieso nur Schmuh fur dumm Leut, aber Gerat solide gebaut und beachten Sicherheit.

Garantie:

Hast zwei Jahre Garantie auf richtig funktionieren. Wenn kaputt musst anrufen Hersteller auf kostenpflichtig Servicenummer (0900-1234567, kost 100 Euro pro Minut). Reparieren kostenlos, aber nur wenn nicht geschmissen vorher in Klo oder Badewanne oder aus Flugzeug.

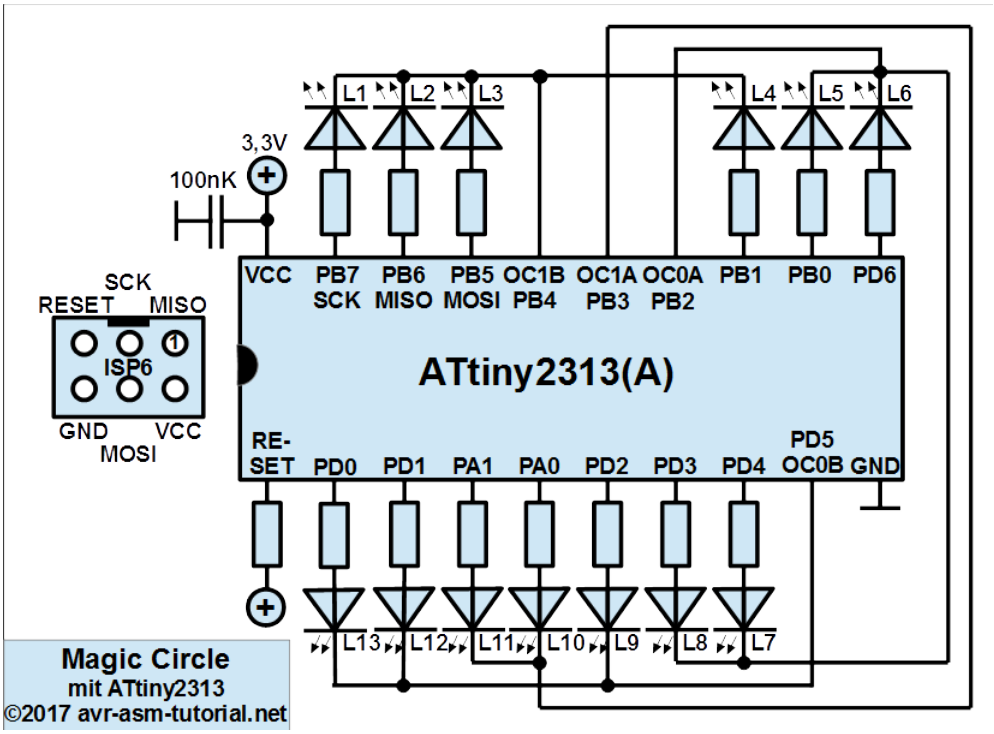
Updates:

Kannst neues Blinkdiagramm entwerfen auf Webseite Hersteller, wenn Seite von Hersteller fertig. Dann Hersteller machen Neuversion und brennen in Professor, wenn vorbeikommen mit Laptop, Programmiergerat und Schraubendreher Equipment. Kannst zugucken bei schnelles Update.

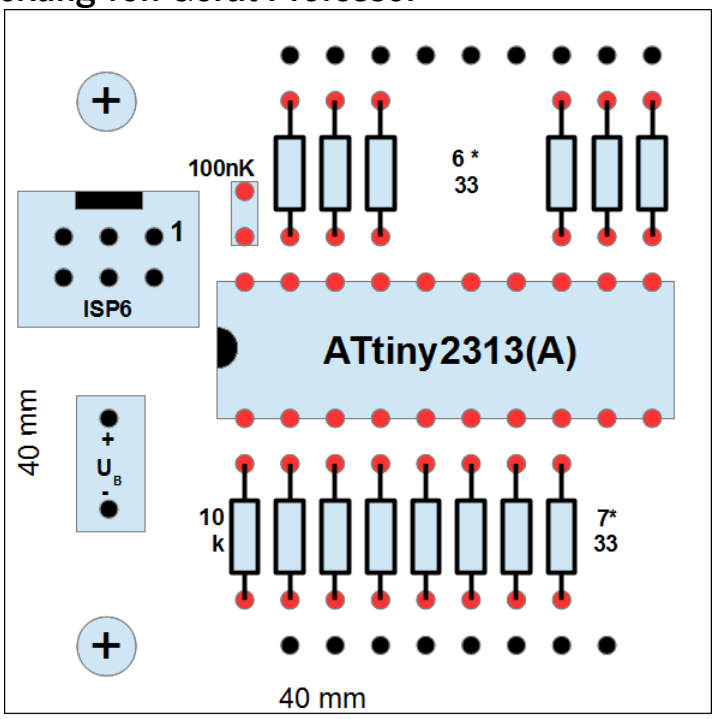
Anhang:

1. Schaltbild von Geraet Professor
2. Schaltbild von Lampen und Stecker
3. Schaltbild von Stromteil
4. Layout von Geraet und Netzteil
5. Programm von Mikroprofessor (Assembler)

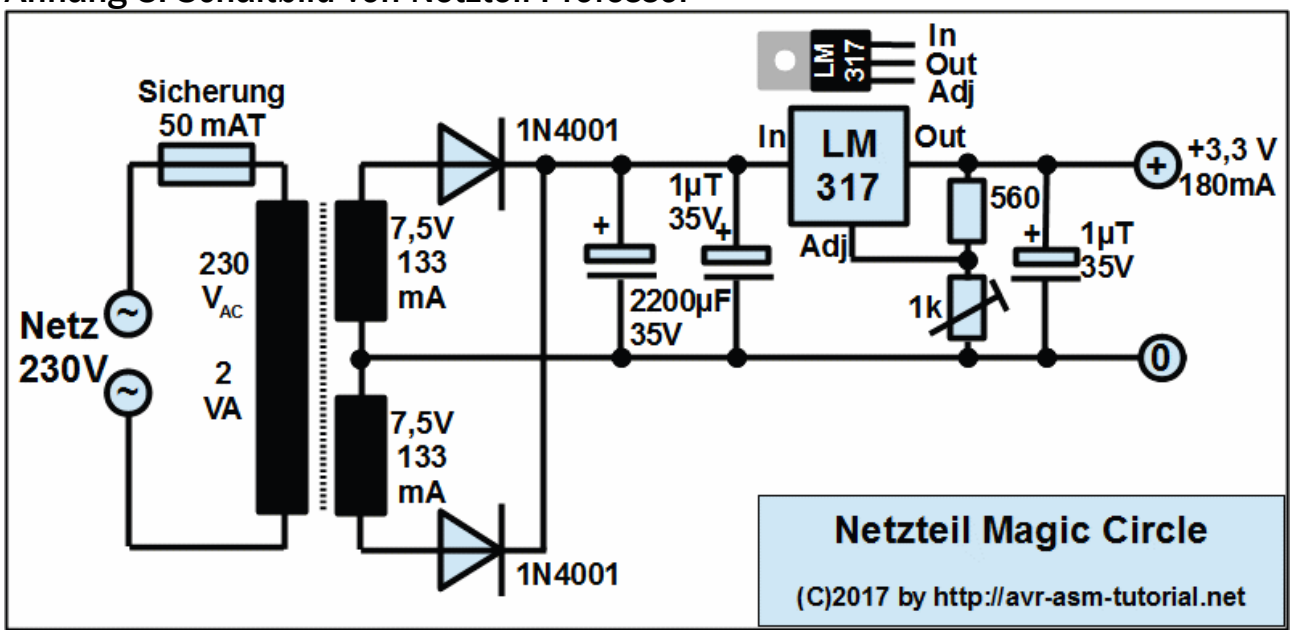
Anhang 1: Schaltbild von Geraet Professor



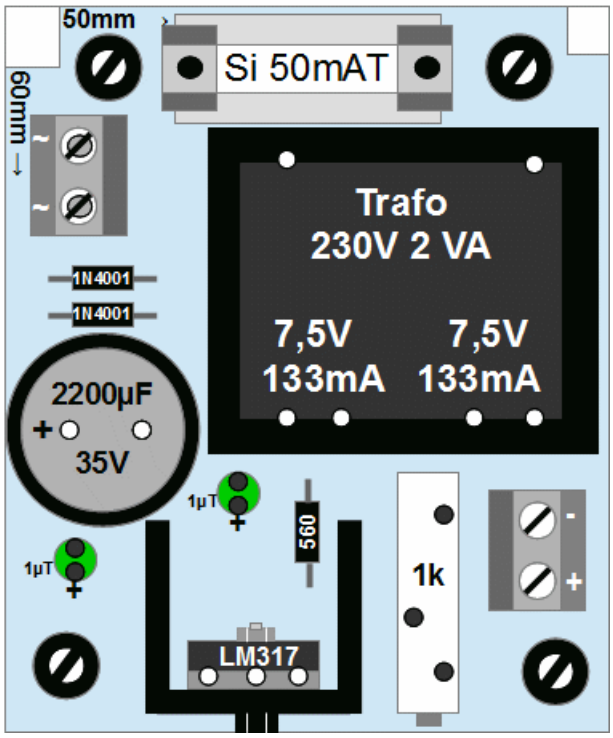
Anhang 2: Bestueckung von Gerat Professor



Anhang 3: Schaltbild von Netzteil Professor



Anhang 4: Bestuckung von Netzteil Professor



Anhang 5: Programm von Professor Assembler (musst nix verstehen)

```
;
; ****************************************
; * Magic circle mit ATtiny2313 und 13 LEDs *
; * Version 1 vom Juni 2017                *
; * (C)2017 by www.avr-asm-tutorial.net    *
; ****************************************
;
; Include-Datei fuer den betreffenden AVR-Typ
.NOLIST
.INCLUDE "tn2313def.inc" ; Headerdatei fuer ATtiny2313
.LIST
;
; ==========================================
;       D E B U G - S C H A L T E R
; ==========================================
;
; Final: alle Debugschalter = 0
.equ debug_Lampenstrom = 0
;
; ==========================================
;   H A R D W A R E I N F O R M A T I O N E N
; ==========================================
;
; Devices = ATtiny2313/4313
;                  _________________
;           1 /                 |20
;  Reset o--|reset          vcc|--o +3,3V (max 3,6V)
;           2|                  |19
;    L13 o--|pd0            pb7|--o L1
;           3|                  |18
;    L12 o--|pd1            pb6|--o L2
;           4|                  |17
;    L11 o--|pa1            pb5|--o L3
;           5|                  |16
;    L10 o--|pa0       oc1b/pb4|--o Kath L1+L2+L3+L4
;           6|                  |15
;     L9 o--|pd2       oc1a/pb3|--o Kath L10+L11
;           7|                  |14
;     L8 o--|pd3       oc0a/pb2|--o Kath L5+L6+L7+L8
;           8|                  |13
;     L7 o--|pd4            pb1|--o L4
; Kath      9|                  |12
; L9+L12 o--|pd5/oc0b       pb0|--o L5
; +L13    10|                  |11
;      0V o--|gnd            pd6|--o L6
;            |__________________|
;
; ==========================================
;              T I M I N G
; ==========================================
;
; Prozessortakt
;   Der Prozessor laeuft mit dem internen 8-MHz-
;   RC-Oszillator und gesetzter CKDIV-Fuse (ohne
;   Fuse-Aenderung, Default-Werkseinstellung),
;   daher mit 1 MHz Takt
;
; LED-Ansteuerung:
;   Timer TC0 und TC1 laufen synchron mit einem
;   Vorteiler von 8 (125 kHz) und steuern im 8-Bit-
;   PWM-Modus die LEDs an (488 Hz) die vier PWM-
```

```
;   Ausgaenge OCR0A, OCR0B, OCR1A und OCR1B an.
;   Alle A- und B-Vergleichswerte werden synchron
;   auf diesselben Werte gesetzt.
;   Alle Ausgaenge werden im Toggle-Modus betrieben
;   (244 Hz PWM-Frequenz).
;   LEDs, die ausgeschaltet sind, haben ihr Aus-
;   gangs-Treiber-Richtungsbit geloescht. Einge-
;   schaltete LEDs haben ihr Richtungsbit gesetzt,
;   und ihr Portausgang ist ebenfalls gesetzt.
;   Der PWM-Wert wird nach jedem Durchlauf im
;   Normalmodus um eins erhoeht, laeuft er ueber
;   wird entweder bei Null begonnen (Flagge bUpOnly
;   gesetzt) oder abwaerts bis Null gezaehlt.
;   Am Ende jedes Zyklus wird die naechste Bit-
;   kombination gelesen aud ausgewertet.
;
; Ablaufsteuerung:
;   Diese erfolgt mittels einer Tabelle. Sie
;   enthaelt jeweils vier Bytes:
;   1. Flaggenbyte: Die Flaggen koennen normal (0),
;      nur Aufwaertszaehlen (1), doppelte
;      Geschwindigkeit (2) und vierfache Ge-
;      schwindigkeit (4 plus 2) kodieren.
;   2. Richtungsbits fuer.Port D (Bit 6 ist immer
;      zu setzen)
;   3. Richtungsbits fuer Port B (Bits 2, 3 und 4
;      sind immer zu setzen)
;   4. Richtungsbits fuer Port A
;   Ist das Flaggenbyte mit folgenden Werten besetzt,
;   wird der Ablauf gesteuert und nur ein nachfolgen-
;   des Byte gelesen:
;   0xFF: Ende der Tabelle, Neubeginn am Anfang
;   0xFE: Wiederholung der nachfolgenden Eintraege,
;         zweites Byte ist die Anzahl Wiederholungen
;   0xFD: Zweite Wiederholungsschleife, zweites
;         Byte ist die Anzahl Wiederholungen
;   0xFC: Ende der ersten Wiederholungsschleife,
;         Beginn am Anfang der ersten Wiederholung
;   0xFB: Ende der zweiten Wiederholungsschleife,
;         Beginn am Anfang der zweiten Wiederholung
;
;
; ==========================================
;   R E G I S T E R D E F I N I T I O N E N
; ==========================================
;
; Frei: R0..R14
.DEF rSreg = R15 ; SREG-Sicherung
.DEF rmp = R16 ; Vielzweckregister
.DEF rimp = R17 ; Vielzweck innerhalb Int
.DEF rFlag = R18 ; Flaggenregister
      .equ bUpOnly = 0 ; Nur Aufwaerts
      .equ bFast = 1 ; Doppelte Geschwindigkeit
      .equ bVeryFast = 2 ; Vierfache Geschwindigkeit
;
      .equ bFall = 6 ; Abwaerts-Zyklus
      .equ bEndCycle = 7 ; Ende Zyklus
.DEF rRep1 = R19
.DEF rRep2 = R20
.DEF rPwm = R21
;
; ==========================================
;       S R A M   D E F I N I T I O N E N
; ==========================================
```

```
;
.DSEG
.ORG  0X0060
; Format: Label: .BYTE N ; Reserviere N Bytes fuer ...
;
; ============================================
;   R E S E T   U N D   I N T   V E K T O R E N
; ============================================
;
.CSEG
.ORG $0000
      rjmp Main ; Reset-Vektor
      reti ; INT0 Int Vektor 1
      reti ; INT1 Int Vektor 2
      reti ; TC1CAPT Int Vektor 3
      reti ; TC1COMPA Int Vektor 4
      reti ; TC1OVF Int Vektor 5
      rjmp Isr ; TC0OVF Int Vektor 6
      reti ; USART-RX Int Vektor 7
      reti ; USART UDRE Int Vektor 8
      reti ; USART TX Int Vektor 9
      reti ; ANACOMP Int Vektor 10
      reti ; PCINT Int Vektor 11
      reti ; TC1COMPB Int Vektor 12
      reti ; TC0COMPA Int Vektor 13
      reti ; TC0COMPB Int Vektor 14
      reti ; USI-START Int Vektor 15
      reti ; USI-OVERFLOW Int Vektor 16
      reti ; EEREADY Int Vektor 17
      reti ; WDT-OVERFLOW Int Vektor 18
;
; ========================================
;    I N T E R R U P T   S E R V I C E
; ========================================
;
Isr: ; TC0-Overflow
      in rSreg,SREG
      ldi rimp,0
      out OCR1AH,rimp
      out OCR1AL,rPwm
      out OCR1BH,rimp
      out OCR1BL,rPwm
      out OCR0A,rPwm
      out OCR0B,rPwm
      sbrs rFlag,bFall
      rjmp IsrUp
      sbrc rFlag,bVeryFast
      subi rPwm,2
      sbrc rFlag,bFast
      subi rPwm,1
      subi rPwm,1
      brne IsrRet
      cbr rFlag,1<<bFall
      sbr rFlag,1<<bEndCycle
      rjmp IsrRet
IsrUp:
      sbrc rFlag,bVeryFast
      subi rPwm,-2
      sbrc rFlag,bFast
      inc rPwm
      inc rPwm
      brne IsrRet
      sbrc rFlag,bUpOnly
      rjmp IsrUpOnly
```

```
      sbrc rFlag,bVeryFast
      subi rPwm,2
      sbrc rFlag,bFast
      dec rPwm
      dec rPwm
      sbr rFlag,1<<bFall
      rjmp IsrRet
IsrUpOnly:
      clr rPwm
      sbr rFlag,1<<bEndCycle
IsrRet:
      out SREG,rSreg
      reti
;
; =============================================
;    H A U P T P R O G R A M M    I N I T
; =============================================
;
Main:
; Initiire Stapel
      ldi rmp, LOW(RAMEND) ; Initiiere LSB Stapel
      out SPL,rmp
; Init Port A
      ldi rmp,0x03 ; Ausgabepins
      out PORTA,rmp
      clr rmp ; Alle Pins auf Input
      out DDRA,rmp
; Init Port B
      ldi rmp,0b11100011 ; Ausgabe-Pins Port B
      out PORTB,rmp
      ldi rmp,0b00011100 ; OC-Pins Output, LED-Pins Input
      out DDRB,rmp
; Init Port D
      ldi rmp,0b01011111 ; Ausgabe-Pins Port D
      out PORTD,rmp
      ldi rmp,0b00100000 ; OC-Pin Output, LED-Pins Input
      out DDRD,rmp
; Debuggen
.if debug_Lampenstrom == 1
      ldi rmp,0b00000011 ; alle Lampen Port A an
      out DDRA,rmp
      ldi rmp,0b11111111 ; alle Lampen Port B an
      out DDRB,rmp
      ldi rmp,0b01111111 ; alle Lampen Port D an
      out DDRD,rmp
Schleife:
      rjmp Schleife
      .endif
; Tabelle initiieren
      clr rFlag ; Flaggen auf Null
      rcall InitCycle
; Init Timer TC0
      clr rmp
      out OCR0A,rmp ; Compare A
      out OCR0B,rmp ; Compare B
      ldi         rmp,(1<<COM0A1)|(1<<COM0A0)|(1<<COM0B1)|(1<<COM0B0)|(1<<WGM01)|
(1<<WGM00)
      out TCCR0A,rmp
; Init Timer TC1
      clr rmp
      out OCR1AH,rmp ; MSB Compare A
      out OCR1AL,rmp ; LSB Compare A
      out OCR1BH,rmp ; MSB Compare B
      out OCR1BL,rmp ; LSB Compare B
```

```
      ldi rmp,(1<<COM1A1)|(1<<COM1A0)|(1<<COM1B1)|(1<<COM1B0)|(1<<WGM10)
      out TCCR1A,rmp
      ; Timer starten
      ldi rmp,(1<<WGM12)|(1<<CS11) ; Timer 1, Prescale = 8, 8-Bit-PWM
      out TCCR1B,rmp
      ldi rmp,1<<CS01 ; Timer 0, Prescaler = 8
      out TCCR0B,rmp
      ldi rmp,1<<TOIE0 ; Timer 0 Interrupt
      out TIMSK,rmp
; Sleep und Interrupt
      ldi rmp,1<<SE ; Enable sleep
      out MCUCR,rmp
      sei
;
; ==========================================
;     P R O G R A M M - S C H L E I F E
; ==========================================
;
Loop:
      sleep ; Schlafen legen
      nop ; Dummy fuer Aufwecken
      sbrc rFlag,bEndCycle
      rcall Cycle
      rjmp loop ; Zurueck nach Loop
;
; Naechster Zyklus
;
; Steuercodes
.equ cEnde = 0xFF
.equ cRepeat1 = 0xFE
.equ cRepeat2 = 0xFD
.equ cNext1 = 0xFC
.equ cNext2 = 0xFB
.equ cControl = 0xF0
;
Cycle:
      lpm rmp,Z+
      cpi rmp,cControl
      brcs Cycle1 ; normaler Zyklus
      cpi rmp,cRepeat1
      breq CycleRepeat1
      brcc InitCycle ; Neustart
      cpi rmp,cNext1
      breq CycleNext1
      cpi rmp,cRepeat2
      breq CycleRepeat2
      cpi rmp,cNext2
      breq CycleNext2
      lpm rmp,Z+
      rjmp Cycle
CycleNext2:
      lpm rmp,Z+
      dec rRep2
      breq Cycle
      mov ZH,YH
      mov ZL,YL
      rjmp Cycle
CycleNext1:
      lpm rmp,Z+
      dec rRep1
      breq Cycle
      mov ZH,XH
      mov ZL,XL
      rjmp Cycle
```

```
CycleRepeat1:
      lpm rRep1,Z+
      mov XH,ZH
      mov XL,ZL
      rjmp Cycle
CycleRepeat2:
      lpm rRep2,Z+
      mov YH,ZH
      mov YL,ZL
      rjmp Cycle
InitCycle:
      ldi ZH,High(2*Tab) ; Z auf Tabellenanfang
      ldi ZL,Low(2*Tab)
      rjmp Cycle
Cycle1:
      mov rFlag,rmp
      lpm rmp,Z+
      out DDRD,rmp
      lpm rmp,Z+
      out DDRB,rmp
      lpm rmp,Z+
      out DDRA,rmp
      ret
;
; Tabelle mit Ablauf
;
;
; Tabelle mit Konstanten
;
.include "konstanten.inc"
;
; Flaggen
.equ cUpOnly = 1<<bUpOnly
.equ cFast = 1<<bFast
.equ cVeryFast  = (1<<bVeryFast)|(1<<bFast)
;
Tab:
; Eine Runde sehr schnell
.set cX = cVeryFast+X1
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X2
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X7
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X8
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X4
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X3
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X6
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X5
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X9
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X10
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X12
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X11
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast+X13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
```

```
.set cX = X0
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
; Rueckwaerts schnell
.set cX = cVeryFast + X13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X11
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X12
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X10
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X9
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X5
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X6
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X3
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X4
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X8
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X7
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X2
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = cVeryFast + X1
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X0
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
; Zwei Lampen rotieren
.set cX = X1_2
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X2_7
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X7_8
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X4_8
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X3_6
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X5_6
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X5_9
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X9_10
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X10_12
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X11_12
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X11_13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X0
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
; Explode
.set cX = X13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X9bis13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = Xalle
```

```
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
; Implode
.set cX = X9bis13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.set cX = X13
.db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
.db cRepeat1,5 ; lange Pause
      .set cX = X0
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .db cNext1,0
; S Aufbau
.db cRepeat1,3
      .set cX = XS1
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS2
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS3
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS4
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS5
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS6
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .set cX = XS7
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .db cRepeat2,5
            .set cX = XS
            .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
            .db cNext2,0
      .db cNext1,0
; Lange Pause
.db cRepeat1,10
      .set cX = X0
      .db Byte1(cX),Byte2(cX),Byte3(cX),Byte4(cX)
      .db cNext1,0
.db 0xFF,0xFF
;
; Ende Quellcode
; Copyright
.db "(C)2017 by Gerhard Schmidt  " ; menschenlesbar
.db "C(2)17 0ybG reahdrS hcimtd  " ; wortgerecht
;
```

konstanten.inc

```
; Definitionen
.equ X0=1843200 ; 00.1C.20.00
.equ X1=10231808 ; 00.9C.20.00
.equ X2=6037504 ; 00.5C.20.00
.equ X3=3940352 ; 00.3C.20.00
.equ X4=1974272 ; 00.1E.20.00
.equ X5=1908736 ; 00.1D.20.00
.equ X6=1859584 ; 00.1C.60.00
.equ X7=1847296 ; 00.1C.30.00
.equ X8=1845248 ; 00.1C.28.00
.equ X9=1844224 ; 00.1C.24.00
.equ X10=18620416 ; 01.1C.20.00
.equ X11=35397632 ; 02.1C.20.00
.equ X12=1843712 ; 00.1C.22.00
.equ X13=1843456 ; 00.1C.21.00
.equ X1_2=14426112 ; 00.DC.20.00
.equ X1_3=12328960 ; 00.BC.20.00
.equ X1_4=10362880 ; 00.9E.20.00
.equ X1_5=10297344 ; 00.9D.20.00
.equ X1_6=10248192 ; 00.9C.60.00
.equ X1_7=10235904 ; 00.9C.30.00
.equ X1_8=10233856 ; 00.9C.28.00
.equ X1_9=10232832 ; 00.9C.24.00
.equ X1_10=27009024 ; 01.9C.20.00
.equ X1_11=43786240 ; 02.9C.20.00
.equ X1_12=10232320 ; 00.9C.22.00
.equ X1_13=10232064 ; 00.9C.21.00
.equ X2_3=8134656 ; 00.7C.20.00
.equ X3_4=4071424 ; 00.3E.20.00
.equ X4_5=2039808 ; 00.1F.20.00
.equ X5_6=1925120 ; 00.1D.60.00
.equ X6_7=1863680 ; 00.1C.70.00
.equ X7_8=1849344 ; 00.1C.38.00
.equ X8_9=1846272 ; 00.1C.2C.00
.equ X9_10=18621440 ; 01.1C.24.00
.equ X10_11=52174848 ; 03.1C.20.00
.equ X11_12=35398144 ; 02.1C.22.00
.equ X12_13=1843968 ; 00.1C.23.00
.equ X1_2_3=16523264 ; 00.FC.20.00
```

```
.equ X2_3_4=8265728 ; 00.7E.20.00
.equ X3_4_5=4136960 ; 00.3F.20.00
.equ X4_5_6=2056192 ; 00.1F.60.00
.equ X5_6_7=1929216 ; 00.1D.70.00
.equ X6_7_8=1865728 ; 00.1C.78.00
.equ X7_8_9=1850368 ; 00.1C.3C.00
.equ X8_9_10=18623488 ; 01.1C.2C.00
.equ X9_10_11=52175872 ; 03.1C.24.00
.equ X10_11_12=52175360 ; 03.1C.22.00
.equ X11_12_13=35398400 ; 02.1C.23.00
.equ X2_7=6041600 ; 00.5C.30.00
.equ X4_8=1976320 ; 00.1E.28.00
.equ X3_6=3956736 ; 00.3C.60.00
.equ X5_9=1909760 ; 00.1D.24.00
```

```
.equ X10_12=18620928 ; 01.1C.22.00
.equ X11_13=35397888 ; 02.1C.21.00
.equ X9bis13=52176640 ; 03.1C.27.00
.equ Xalle=67075840 ; 03.FF.7F.00
.equ XS1=6037504 ; 00.5C.20.00
.equ XS2=14426112 ; 00.DC.20.00
.equ XS3=14491648 ; 00.DD.20.00
.equ XS4=14491904 ; 00.DD.21.00
.equ XS5=14492416 ; 00.DD.23.00
.equ XS6=14494464 ; 00.DD.2B.00
.equ XS7=14625536 ; 00.DF.2B.00
.equ XS=16723712 ; 00.FF.2F.00
```