



AVR-Anwendungen

Stoppuhr mit ATmega8 in Assembler



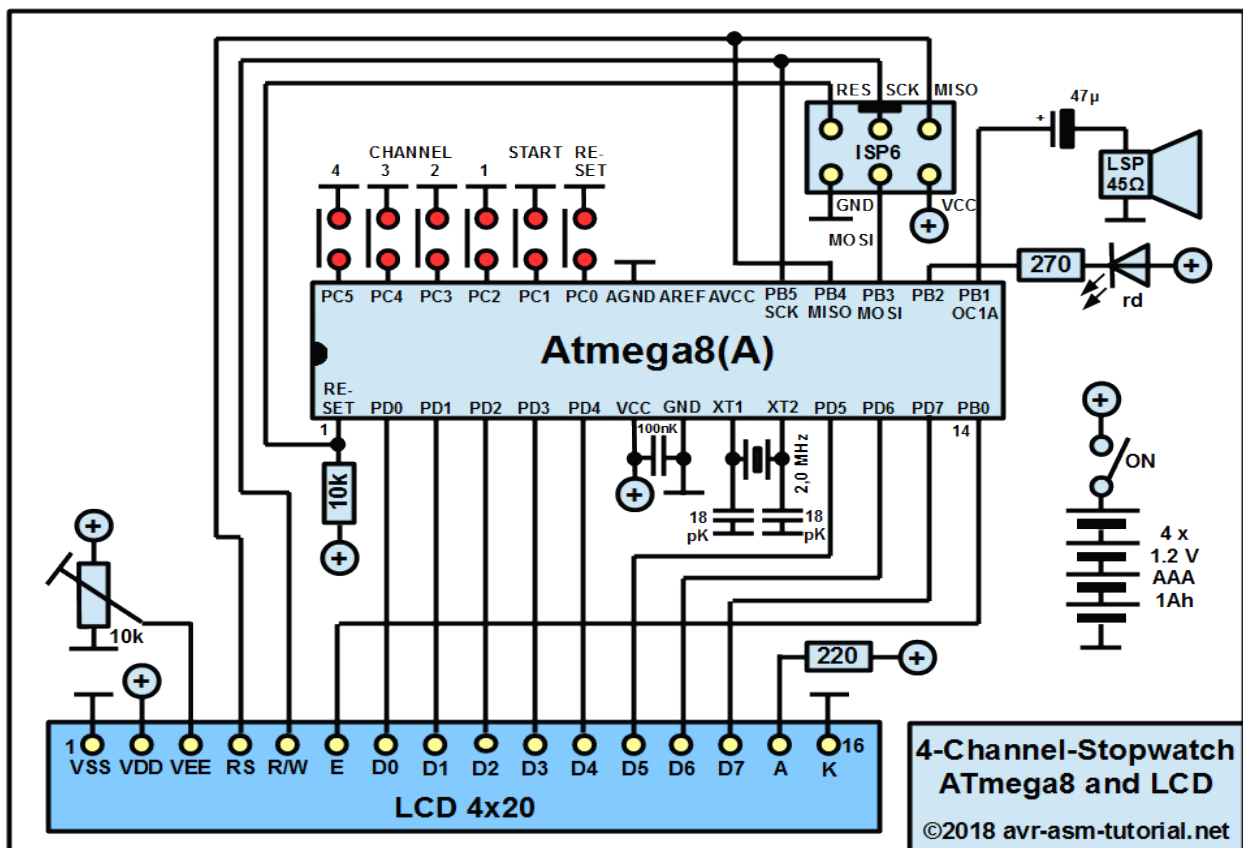
Stoppuhr mit einem ATmega8

Hier wird eine Stoppuhr mit den folgenden Eigenschaften beschrieben:

- Vier unabhängig stoppbare Kanäle
- Auflösung bei 1 ms
- Quarzstabiler Takt mit 50 ppm Genauigkeit
- Start- und Stop-Steuerung, Messpausen möglich
- Vierzeilige LCD zur Anzeige
- Einfacher Aufbau
- Assembler-Software mit Quellcode
- Software kompatibel mit 2,0 oder 2,048 oder 8,0 MHz-Quarzen

1 Hardware

Das ist die gesamte nötige Hardware.



Der ATmega8 wird mit einem 2,048 Mhz-Quarz getaktet. Verwendbar sind auch 2,0 und 8,0 MHz-Quarze (diese erfordern auch eine Änderung der Software).

Die Stromversorgung erfolgt aus vier hintereinander geschalteten 1,2V-Akkus. so dass eine 5V-LCD verwendet werden kann.

Die vierzeilige LCD ist im 8-Bit-Modus an den ATmega8 angeschlossen und wird mit dem R/W-Anschluss im Busy-Modus angesteuert.

Eine Taste setzt die Stoppuhr auf Null, eine Taste dient zum Starten und Anhalten der Zeit und vier Tasten dienen dem Stoppen der vier Kanäle.

Die ISP-Schnittstelle dient zum Programmieren in der Schaltung.

2 Software-Struktur

2.1 Anzeigeschema

Das Anzeigeschema demonstriert das Ausgabeschema

1. im gestoppten Zustand,
2. bei laufender Zeitmessung, und
3. bei abgeschlossener Messung in angehaltenen Kanälen.

Output format

Reset, not running

T	r	a	c	k	1					0	0	:	0	0	:	0	0	,	0		
T	r	a	c	k	2					0	0	:	0	0	:	0	0	,	0		
T	r	a	c	k	3					0	0	:	0	0	:	0	0	,	0		
T	r	a	c	k	4					0	0	:	0	0	:	0	0	,	0		

Running, no channel stopped

T	r	a	c	k	1					0	1	:	2	3	:	4	5	,	7		
T	r	a	c	k	2					0	1	:	2	3	:	4	5	,	7		
T	r	a	c	k	3					0	1	:	2	3	:	4	5	,	7		
T	r	a	c	k	4					0	1	:	2	3	:	4	5	,	7		

Running, channel 1 and 3 stopped

ZH	0	T	r	a	c	k	1	=		0	1	:	2	3	:	4	5	,	6	7	8
	1	T	r	a	c	k	2			1	2	:	3	4	:	5	6	,	7		
	2	T	r	a	c	k	3	=		0	1	:	2	3	:	4	5	,	6	7	8
	3	T	r	a	c	k	4			1	2	:	3	4	:	5	6	,	7		

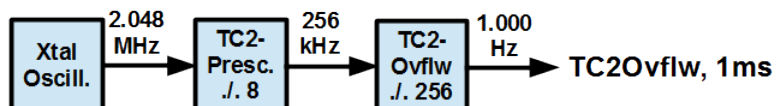
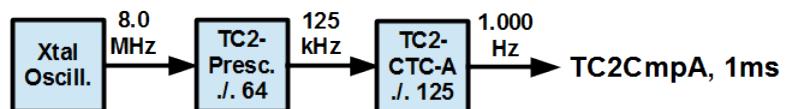
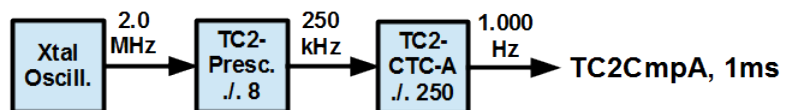
ZL 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

2.2 Taktung

Die Taktung erfolgt mit einem 2,0 oder 8,0 MHz-Quarz im TC2 mittels CTC und Compare-Match-Interrupt. Wird ein 2,048 MHz-Quarz verwendet, dann arbeitet TC2 im Normalmodus mit dem Overflow-Interrupt.

Bei 2,0 und 2,048 MHz wird der 8-Bit-Timer mit einem Prescaler von 8 betrieben, bei 8 Mhz mit einem solchen von 64.

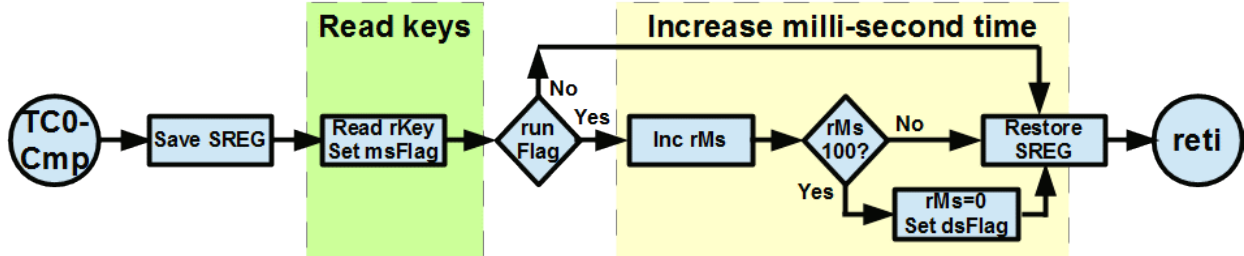
Milliseconds-timer



2.3 Interrupt-Service-Routine TC2

In der Interrupt-Service-Routine von TC2, die je nach verwendetem Quarz vom TC2OVF- oder vom TC2CMP-Interrupt angesteuert wird, wird zunächst der aktuelle Zustand am Tasteneingang eingelesen und die ms-Flagge gesetzt. Die weitere Verarbeitung des Tastenzustands erfolgt außerhalb der ISR in der Hauptprogrammschleife.

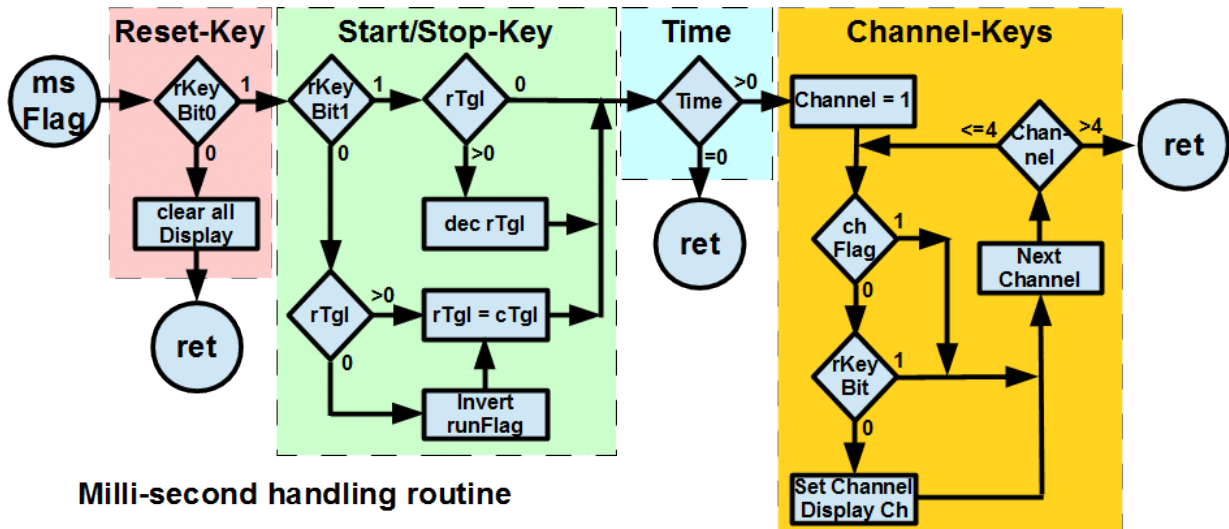
Milli-second timer interrupt



Danach wird die bRun-Flagge abgefragt und, falls der Timer läuft, die laufende Zeitmessung der Millisekunden aktualisiert. Erreicht der Millisekunden-Zähler 100, wird er neu gestartet und die ds-Flagge gesetzt. Die weitere Verarbeitung der Zeit erfolgt in der Hauptprogrammschleife. Die Verarbeitung der Millisekunden innerhalb der Service-Routine verhindert, dass bei langwierigen Ausgabeoperationen mit der LCD Millisekunden-Interrupts verpasst werden.

2.4 Behandlung der Millisekunden-Flagge

Mit der Millisekunden-Flagge werden alle Tasten daraufhin untersucht, ob sie aktiviert sind.



Milli-second handling routine

Zunächst wird die ms-Flagge wieder gelöscht, um die nächste Ausführung zu ermöglichen.

Ist die Reset-Taste gedrückt, wird unabhängig von allen anderen Operationen die Zeit angehalten, auf Null gesetzt und ausgegeben. Eine weitere Prüfung der anderen Tasten erfolgt dann nicht.

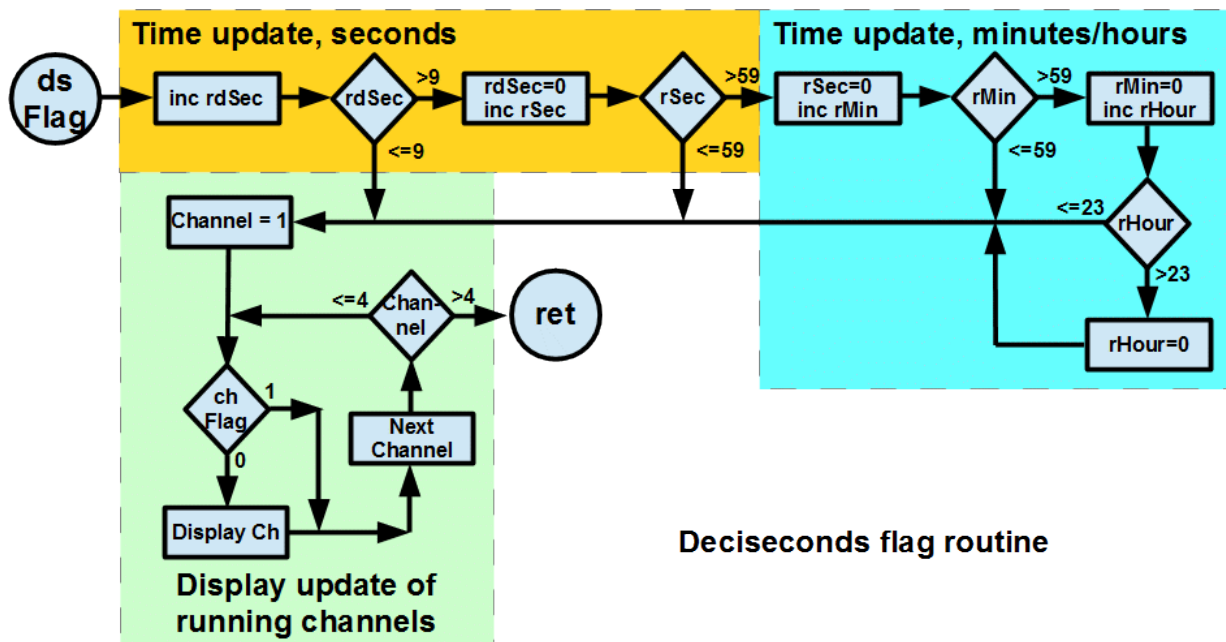
Die Start/Stop-Taste muss entprellt werden, da sie sonst zu wildem Starten und Anhalten der Zeit führen würde. Ist diese gedrückt (das vom Porteingang eingelesene Bit ist Null), wird zunächst überprüft, ob das Prellunterdrückungsregister rTgl Null ist. Wenn das nicht der Fall ist, wird das Prellunterdrückungsregister auf seinen Anfangswert gesetzt (Prellzeit in Millisekunden). Ist das Register auf Null, wird die Run-Flagge invertiert und das Register rTgl auf seinen Anfangswert gesetzt. Ist die Start/Stop-Taste nicht gedrückt und das Prellunterdrückungsregister nicht Null, wird es um Eins vermindert. Auf diese Weise sind cTgl ununterbrochene High-Zyklen nötig, bis wieder auf Nullen am Start/Stop-Eingang reagiert

wird.

Danach werden die Tasten der einzelnen Kanäle abgefragt. Zunächst wird geprüft, ob die Zeit auf Null steht. Falls das der Fall ist, werden die Tasten nicht weiter geprüft, weil das sinnlos wäre. Ist dies nicht der Fall, werden nacheinander alle Kanäle überprüft. Zunächst wird festgestellt, ob der betreffende Kanal bereits gestoppt ist. Ist dies nicht der Fall, wird das eingeleseene Portpin auf Null geprüft. Ist dieses Null, wird die Zeit im SRAM gespeichert, angezeigt und das entsprechende Kanalbit gesetzt. Sind auf diese Weise alle Kanalbits überprüft, wird die Routine beendet.

2.5 Dezisekunden-Behandlung

Ist der Ablauf einer Zehntel-Sekunde komplett (ds-Flagge in der ISR wurde gesetzt) wird die Zeit um eine Zehntel Sekunde erhöht. Entsprechend werden bei Erreichen der Obergrenzen auch die Sekunden, Minuten und Stunden angepasst. Erreichen die Stunden 24, beginnen diese wieder bei Null.



Nach der Erhöhung der Zeit wird für alle Kanäle geprüft, ob diese bereits angehalten sind. Wenn nicht, wird die aktuelle Zeit in der Kanalzeile ausgegeben (Stunden, Minuten, Sekunden und Zehntelsekunden).

2.6 Tonerzeugung

Die Tonerzeugung erfolgt mit dem 16-Bit-Timer TC1 im CTC-Modus mit einem Vorteiler von 8. Dadurch sind bei einer Taktfrequenz von 2,0 MHz Töne zwischen 3,8 Hz und 250 kHz, bei 8,0 MHz Takt 15,3 Hz bis 1 MHz möglich.

Die Tonausgabe erfolgt am Ausgang OC1A durch Torkeln des Ausgangspins beim Erreichen des Compare-Matches. Soll kein Ton ausgegeben werden, wird der Ausgang beim Compare-Match auf Clear gesetzt, wodurch der Lautsprecher-Elko entladen wird und beim Abschalten kein Entladestrom in den OC1A-Pin fließt.

Bei jedem Compare-Match wird ein Interrupt ausgelöst, der den 16-Bit-Zähler in R25:R24 um Eins vermindert. Erreicht dieser Null, wird der Clear-Modus am Ausgabepin OC1A eingestellt und damit der Ton beendet. Der Zähler R25:R24 bestimmt daher die Dauer des Tons.

Die Frequenzen und die Dauer der Töne sind in den Konstanten cTonexxx in Hz und cTonexxxDur in ms definierbar. Aus diesen Tonhöhen werden die Compare-Werte cCmpXXX für den CTC und die Zählerwerte cCtrXXX errechnet und in einer Tontabelle *ToneTable*: ab-

gelegt.

Der Compare-Wert ergibt sich aus der Taktfrequenz *clock* und der Tonhöhe, beides in Hz und dem Prescalerwert von 8 nach der Formel

$$\text{cmp} = \text{clock} / \text{Ton} / 8 / 2 - 1$$

Der Zählerwert ergibt sich aus der Frequenz in Hz und der Dauer in ms nach der Formel

$$\text{ctr} = \text{Ton} * 2 * \text{Dauer} / 1000$$

und ist daher unabhängig von der Taktfrequenz.

Die Routine *ToneStart*: startet den im Register *rmp* (R16) eingestellten Ton (0..10), indem er zuerst den Zählerwert und dann den Compare-Match-Wert dieses Tons ausliest und in die Register schreibt.

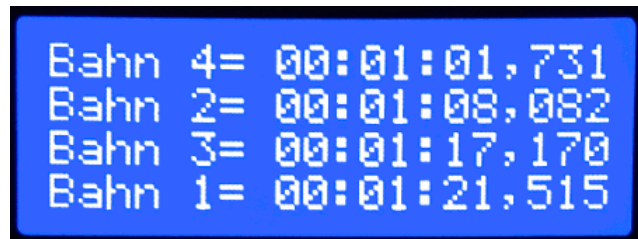
Für die Töne wurden Werte ausgewählt, die in unterschiedlichen Oktaven liegen: beim Init und bei den Tasten Reset und Start/Stop die Oktave 4, bei Sekunden, Minuten und Stunden die Oktave 3 und bei den Kanaltasten die Oktave 2.

2.7 LED-Ansteuerung

Die LED-Ansteuerung erfolgt mit dem Portpin PB2. Beim Init und bei inaktiver Stoppuhr ist diese dauerhaft an, bei laufender Stoppuhr nur bei den noch aktiven Kanälen im Rhythmus von Zehntelsekunden.

2.8 Sortierung nach Bestzeiten

Sind alle vier Kanäle angehalten, wird die Uhr angehalten und die vier Zeiten werden in der Reihenfolge ihrer Aktivierung ausgegeben. Die Reihenfolge des Anhaltens der vier Kanäle ist im SRAM abgelegt, die Zeiten ebenfalls.



Ist beim Assemblieren der Schalter *eep* auf 1 gesetzt, dann werden alle 20 Datenbytes und die vier Reihenfolgen in das EEPROM geschrieben und können von dort ausgelesen werden.

3 Software

Der Assembler-Quellcode der Stoppuhr mit einem ATmega8 kann [hier](#) heruntergeladen und ist als Anhang [hier](#) enthalten. Zum Assemblieren ist noch die Include-Datei [lcd.inc](#) erforderlich, die alle LCD-Routinen enthält.

Die Software ist per default auf folgende Eigenschaften gesetzt:

- Quarzfrequenz: 2,048 MHz, clock = 2048000
- LCD 4x20 8-Bit-Datenbus Busy-Modus
- EEPROM-Schreiben aus

Beim Programmieren des Chips ist dieser vom internen RC-Oszillator per Fuse auf High-Speed-Xtal (bei einem 8 MHz-Quarz) oder auf Medium-Speed-Xtal (bei 2,0 oder 2,048 MHz-Quarz) umzustellen, weil sonst die Zeiten und die Töne nicht stimmen.



AVR-Anwendungen

Stoppuhr mit ATmega8 in Assembler Quellcode



Assembler Quellcode Stoppuhr ATmega8

Dieser Assembler-Quellcode ist [hier im asm-Format](#) downloadbar. Die dazu gehörige lcd.inc gibt es [hier](#).

```
;
; *****
; * Vierkanal-Stoppuhr mit ATmega8 *
; * Version 1.0 Juli 2018 *
; * (C)2018 avr-asm-tutorial.net *
; *****
;
.nolist
.include "m8def.inc" ; Fuer ATmega8
.list
;
; *****
; D E B U G G I N G   C O D E
; *****
;
; Alle debug-Schalter aus (0) fuer Endversion
.equ Eep = 0 ; 1 schreibt alle Daten ins EEPROM
;
; *****
; H A R D W A R E
; *****
;
; Device: ATmega8, Package: 28-pin-PDIP
;
;          1 /-----| 28
; RESET o--|RES      PC5|--o Taste Ch 4
; LCD-D0 o--|PD0     PC4|--o Taste Ch 3
; LCD-D1 o--|PD1     PC3|--o Taste Ch 2
; LCD-D2 o--|PD2     PC2|--o Taste Ch 1
; LCD-D3 o--|PD3     PC1|--o Taste Start/Stop
; LCD-D4 o--|PD4     PC0|--o Taste Reset
; VCC o--|VCC       GND|--o GND
; GND o--|GND      AREF|--o NC
; Quarz o--|XTAL1  AVCC|--o NC
; 8 MHz o--|XTAL2  PB5|--o SCK/LCD-RW
; LCD-D5 o--|PD5     PB4|--o MISO/LCD-RS
; LCD-D6 o--|PD6     PB3|--o MOSI
; LCD-D7 o--|PD7     PB2|--o LED-Rot Kathode
```

```

; LCD-E o--|PB0    PB1|--o Lautsprecher
;          |_____|
;

; *****
; P O R T S    U N D    P I N S
; *****
;
; Rote Led
.equ pLedRO = PORTB ; Rot Led Output-Port
.equ pLedRD = DDRB ; Rote Led Richtungsport
.equ bLedRO = PORTB2 ; Rote Led Output-Portpin
.equ bLedRD = DDB2 ; Rote Led Richtungs-Portpin
;
; Lautsprecher
.equ pSpkO = PORTB ; Lautsprecher Output-Port
.equ pSpkD = DDRB ; Lautsprecher-Richtungsport
.equ bSpkO = PORTB1 ; Lautsprecher Output-Portpin
.equ bSpkD = DDB1 ; Lautsprecher Richtungs-Portpin
;
; Tasten
.equ pKeyO = PORTC ; Tasten Output-Port
.equ pKeyD = DDRC ; Tasten Richtungs-Port
.equ pKeyI = PINC ; Tasten Eingabe-Port
;
; Taktkonfiguration
; Moegliche Takte: 2,000 oder 8,000 MHz oder 2,048 MHz
; Wenn 2,048 MHz: TC2 Normaler Zaehler, Ueberlauf-Int
; Wenn 2/8 MHz: TC2 als CTC, Compare-Match-Int
.equ clock = 2048000 ; Taktfrequenz Controller in Hz
;
; LCD-Konfiguration fuer lcd.inc
; LCD-Groesse:
.equ LcdLines = 4 ; Anzahl Zeilen (1, 2, 4)
.equ LcdCols = 20 ; Anzahl Zeichen pro Zeile (8..24)
; LCD-Ansteuerung
.equ LcdBits = 8 ; Busbreite (4 oder 8)
; Wenn 4-Bit-Ansteuerung:
.equ Lcd4High = 1 ; Busnibble (1=Oberes, 0=Unteres)
.equ LcdWait = 0 ; Ansteuerung (0 mit Busy, 1 mit Warteschleifen)
; LCD-Datenports
.equ pLcdDO = PORTD ; Daten-Ausgabe-Port
.equ pLcdDD = DDRD ; Daten-Richtungs-Port
; LCD-Kontrollports und -pins
.equ pLcdCEO = PORTB ; Control E Ausgabe-Port
.equ bLcdCEO = PORTB0 ; Controll E Ausgabe-Portpin
.equ pLcdCED = DDRB ; Control E Richtungs-Port
.equ bLcdCED = DDB0 ; Control E Richtungs-Portpin
.equ pLcdCRSO = PORTB ; Control RS Ausgabe-Port
.equ bLcdCRSO = PORTB4 ; Controll RS Ausgabe-Portpin
.equ pLcdCRSD = DDRB ; Control RS Richtungs-Port
.equ bLcdCRSD = DDB4 ; Control RS Richtungs-Portpin
; Wenn LcdWait = 0:
.equ pLcdDI = PIND ; Daten-Input-Port
.equ pLcdCRWO = PORTB ; Control RW Ausgabe-Port
.equ bLcdCRWO = PORTB5 ; Control RW Ausgabe-Portpin
.equ pLcdCRWD = DDRB ; Control RW Richtungs-Port
.equ bLcdCRWD = DDB5 ; Control RW Richtungs-Portpin
; Dezimalausgabe einbinden
.equ LcdDecimal = 1 ; Dezimalwandlung an
; Hexadezimalausgabe einbinden
.equ LcdHex = 1 ; Hexadezimalwandlung aus
; Wenn nur Simulation im SRAM:
.equ avr_sim = 1 ; 1=Simulieren, 0=Nicht simulieren

```

```

;
; *****
;   J U S T I E R B A R E   K O N S T A N T E N
; *****
;
; Englisch oder Deutsche Version
.equ LangEn = 0 ; Englisch = 1, Deutsch = 0
;
.equ cToggle=60 ; Prellunterdrueckung fuer n ms
;
; Tonfrequenzen
.equ cToneInit = 3136 ; Ton bei Init, G7 (g4), in Hz
.equ cToneSec = 2794 ; Ton be Sekunden, F7 (f4), in Hz
.equ cToneMin = 2637 ; Ton bei Minuten, E7 (e4), in Hz
.equ cToneHour = 2349 ; Ton bei Stunden, D7 (d4), in Hz
.equ cToneReset = 1319 ; Ton bei Reset, E6 (e3), in Hz
.equ cToneStart = 1175 ; Ton bei Start, D6 (d3), in Hz
.equ cToneStop = 1047 ; Ton bei Stop, C6 (c3), in Hz
.equ cToneCh1 = 440 ; Ton bei Kanal 1, A4 (a1), in Hz
.equ cToneCh2 = 392 ; Ton bei Kanal 2, G4 (g1), in Hz
.equ cToneCh3 = 349 ; Ton bei Kanal 3, F4 (f1), in Hz
.equ cToneCh4 = 330 ; Ton bei Kanal 4, E4 (e1), in Hz
;
; Lautsprechertoene Dauer
.equ cToneSecDur = 100 ; Tondauer bei Sekunden, in ms
.equ cToneMinDur = 500 ; Tondauer bei Minuten, in ms
.equ cToneHourDur = 750 ; Tondauer bei Stunden, in ms
.equ cToneResetDur = 500 ; Tondauer bei Reset, in ms
.equ cToneStartDur = 400 ; Tondauer bei Start, in ms
.equ cToneStopDur = 300 ; Tondauer bei Stop, in ms
.equ cToneCh1Dur = 200 ; Tondauer bei Kanal 1, in ms
.equ cToneCh2Dur = 200 ; Tondauer bei Kanal 2, in ms
.equ cToneCh3Dur = 200 ; Tondauer bei Kanal 3, in ms
.equ cToneCh4Dur = 200 ; Tondauer bei Kanal 4, in ms
;
; *****
;   F E S T E   &   A B G E L E I T E T E   K O N S T
; *****
;
.if clock == 8000000
.equ cPresc2 = 64 ; Prescaler TC2 = 64
.else
.equ cPresc2 = 8 ; Prescaler TC2 = 8
.endif
;
.if clock != 2048000
.equ cCtcDiv2 = clock / cPresc2 / 1000 ; CTC-Teiler
.equ cCmp2A = cCtcDiv2 - 1 ; Compare-Wert TC2
.endif
;
; Lautsprechertoene in Compare-Werte
.equ cPresc1 = 8 ; Prescaler TC1
.equ cCmpInit = clock / cToneInit / cPresc1 / 2 - 1 ; Compare Init
.equ cCmpSec = clock / cToneSec / cPresc1 / 2 - 1 ; Compare Sekunde
.equ cCmpMin = clock / cToneMin / cPresc1 / 2 - 1 ; Compare Minute
.equ cCmpHour = clock / cToneHour / cPresc1 / 2 - 1 ; Compare Stunde
.equ cCmpReset = clock / cToneReset / cPresc1 / 2 - 1 ; Compare Reset
.equ cCmpStart = clock / cToneStart / cPresc1 / 2 - 1 ; Compare Start
.equ cCmpStop = clock / cToneStop / cPresc1 / 2 - 1 ; Compare Stop
.equ cCmpCh1 = clock / cToneCh1 / cPresc1 / 2 - 1 ; Compare Kanal 1
.equ cCmpCh2 = clock / cToneCh2 / cPresc1 / 2 - 1 ; Compare Kanal 2
.equ cCmpCh3 = clock / cToneCh3 / cPresc1 / 2 - 1 ; Compare Kanal 3
.equ cCmpCh4 = clock / cToneCh4 / cPresc1 / 2 - 1 ; Compare Kanal 4
;

```



```

; Tondauer in Zaehlerwerte
.equ cCtrInit = 1 ; Zaehle bis zur ersten Interrupt-Ausfuehrung
.equ cCtrSec = cToneSec * 2 * cToneSecDur / 1000 ; Dauer Sekunde
.equ cCtrMin = cToneMin * 2 * cToneMinDur / 1000 ; Dauer Minute
.equ cCtrHour = cToneHour * 2 * cToneHourDur / 1000 ; Dauer Stunde
.equ cCtrReset = cToneReset * 2 * cToneResetDur / 1000 ; Dauer Reset
.equ cCtrStart = cToneStart * 2 * cToneStartDur / 1000 ; Dauer Start
.equ cCtrStop = cToneStop * 2 * cToneStopDur / 1000 ; Dauer Stop
.equ cCtrCh1 = cToneCh1 * 2 * cToneCh1Dur / 1000 ; Dauer Kanal 1
.equ cCtrCh2 = cToneCh2 * 2 * cToneCh2Dur / 1000 ; Dauer Kanal 2
.equ cCtrCh3 = cToneCh3 * 2 * cToneCh3Dur / 1000 ; Dauer Kanal 3
.equ cCtrCh4 = cToneCh4 * 2 * cToneCh4Dur / 1000 ; Dauer Kanal 4
;
; *****
;           R E G I S T E R
; *****
;
; frei: R0 bis R8
.def rmsBuf = R9 ; Ablage der Millisekunden
.def rLedCtr = R10 ; LED-Zaehlregister
.def rLed = R11 ; LED-Anzeigeregister
.def rTgl = R12 ; Prellzaehler
.def rKeyCmp = R13 ; Vergleichswert Tasten
.def rKey = R14 ; Tasteneingabe-Wert
.def rSreg = R15 ; Sichern/Wiederherstellung Status-Port
.def rmp = R16 ; Vielzweckregister
.def rimp = R17 ; Vielzweck innerhalb Ints
.def rFlag = R18 ; Flaggenregister
.equ bMs = 0 ; Millisekundenflagge
.equ bRun = 1 ; Run/Stop-Flagge
.equ bCh1 = 2 ; Kanal 1 Flagge
.equ bCh2 = 3 ; Kanal 2 Flagge
.equ bCh3 = 4 ; Kanal 3 Flagge
.equ bCh4 = 5 ; Kanal 4 Flagge
.equ bdS = 6 ; Dezisekundenflagge
.equ bSort = 7 ; Sortierflagge
.def rHour = R19 ; Stunden
.def rMin = R20 ; Minuten
.def rSec = R21 ; Sekunden
.def rdSec = R22 ; Dezisekunden
.def rmSec = R23 ; Millisekunden
.def rCtrL = R24 ; Zaehler fuer Tondauer, LSB
.def rCtrH = R25 ; dto., MSB
; Benutzt: XH:XL=R27:R26 als Zeiger auf sData
; Benutzt: YH:YL=R29:R28 als Zeiger fuer Stopreihenfolge
; Benutzt: ZH:ZL=R31:R30 fuer diverse Zwecke
;
; *****
;           S R A M
; *****
;
.dseg
.org SRAM_START
sData:
.byte 20 ; Reserviere 4*5 Bytes fuer Zeitinfo
sChannelRow:
.byte 4 ; Reihenfolgetabelle
sDataEnd:
;
; *****
;           C O D E   S E G M E N T
; *****
;
.cseg

```

```

.org 000000
;
; *****
; R E S E T   &   I N T - V E K T O R E N
; *****
rjmp Main ; Reset-Vector
reti ; INT0, nicht benutzt
reti ; INT1, nicht benutzt
.if clock == 2048000
    reti ; TC2 Compare A
    rjmp MilliSecIsr ; OVF2
.else
    rjmp MilliSecIsr
    reti ; OVF2
.endif
reti ; ICPL, nicht benutzt
rjmp OC1CmpIsr ; OC1A
reti ; OC1B, nicht benutzt
reti ; OVF1, nicht benutzt
reti ; OVF0, nicht benutzt
reti ; SPI, nicht benutzt
reti ; URXC, nicht benutzt
reti ; UDRE, nicht benutzt
reti ; UTXC, nicht benutzt
reti ; ADCC, nicht benutzt
reti ; ERDY, nicht benutzt
reti ; ACI, nicht benutzt
reti ; TWI, nicht benutzt
reti ; SPMR, nicht benutzt
;
; *****
; I N T - S E R V I C E   R O U T I N E N
; *****
;
; TC2 Millisekunden-Interrupt
; wird in jeder Millisekunde ausgefuehrt
; Liest die Tasteneingaenge und setzt die ms-Flagge
; Wenn bRun einschaltet: erhoehe Millisekunden,
; wenn 100: Neustart und Setzen der ds-Flagge
MilliSecIsr:
    in rSreg,SREG ; Sichere SREG
    in rKey,pKeyI ; Lese Tasteneingang
    sbr rFlag,1<<bmS ; Setze Millisekundenflagge
    sbrs rFlag,bRun ; Run-Flagge gesetzt?
    rjmp MilliSecIsr1 ; Nein, nicht erhoehen
    inc rmSec ; Erhoehe Millisekunden
    cpi rmSec,100 ; 100 ms erreicht?
    brcs MilliSecIsr1 ; Nein
    clr rmSec ; Neustart Millisekunden
    sbr rFlag,1<<bdS ; Setze Dezisekundenflagge
MilliSecIsr1:
    out SREG,rSreg ; SREG wieder herstellen
    reti
;
; TC1 Compare-Match-Interrupt
; wird bei Compare-Match A ausgefuehrt
; Ein halber Tonzyklus is beendet
; Zaehlt den Dauerzaehler abwaerts, wenn
; Null: Lautsprecher-Ausgang auf Clear
; mit dem Ende des naechsten Zyklus
OC1CmpIsr:
    in rSreg,SREG ; Sichern SREG
    sbiw rCtrl,1 ; Zaehler abwaerts
    brne OC1CmpIsr1 ; Nicht Null

```

```

    ldi rimp,1<<COM1A1 ; Lautsprecherausgang Clear
    out TCCR1A,rimp
OC1CmpIsr1:
    out SREG,rSreg ; SREG wieder herstellen
    reti
;
; *****
; H A U P T P R O G R A M   I N I T
; *****
;
Main:
#ifdef SPH
    ldi rmp,High(RAMEND) ; MSB auf RAMEND
    out SPH,rmp ; Init MSB Stapelzeiger
#endif
    ldi rmp,Low(RAMEND) ; LSB auf RAMEND
    out SPL,rmp ; Init LSB Stapelzeiger
; Initiierung Tastereingaenge
    ldi rmp,0x3F ; Alle sechs Output-Pins High
    out pKeyO,rmp ; Einschalten der Pull-Ups
    clr rmp ; Richtung auf Low
    out pKeyD,rmp ; im Richtungsregister
; Initiieren der roten Led
    sbi pLedRD,bLedRD ; Richtung Portpin = Output
    cbi pLedRO,bLedRO ; Ausgang Low, Led an
    ldi rmp,0b11000011 ; LED-Register Startwert
    mov rLed,rmp ; in LED-Register
    ldi rmp,8 ; LED-Zaehler-Register auf acht
    mov rLedCtr,rmp
; Lautsprecherausgang initiieren
    cbi pSpkO,bSpkO ; Ausgang Low
    sbi pSpkD,bSpkD ; Richtung = Output
; Lautsprecher-Init-Ton
    ldi rCtrH,High(cCtrInit) ; Lade Dauerzaehler
    ldi rCtrL,Low(cCtrInit)
    ldi rmp,High(cCmpInit) ; Lade Compare-Wert MSB
    out OCR1AH,rmp ; MSB zuerst
    ldi rmp,Low(cCmpInit) ; dto., LSB
    out OCR1AL,rmp ; LSB danach
    ldi rmp,1<<COM1A0 ; Toggele OC1A-Ausgabe-Pin bei Compare Match
    out TCCR1A,rmp
    ldi rmp,(1<<WGM12)|(1<<CS11) ; TC1=CTC, Prescaler = 8
    out TCCR1B,rmp
; Starte die LCD
    rcall LcdInit ; LCD initiieren
    ldi ZH,High(2*LcdInit1) ; Starttext ausgeben
    ldi ZL,Low(2*LcdInit1)
    rcall LcdText
    ldi rmp,40 ; Warte zwei Sekunden
Start1:
    rcall LcdWait50ms ; Warte 50 ms
    dec rmp ; Abwaerts zaehlen
    brne Start1 ; Weiter warten
    clr ZH ; Position erste Zeile
    clr ZL ; Position erste Spalte
    rcall LcdPos ; Position setzen
    ldi ZH,High(2*LcdInit2) ; Zweite Maske ausgeben
    ldi ZL,Low(2*LcdInit2)
    rcall LcdText
; Initiiere die SRAM-Daten
    clr rmSec ; Init Millisekunden
    clr rdSec ; Init Dezisekunden
    clr rSec ; Init Sekunden
    clr rMin ; Init Minuten

```

```

clr rHour ; Init Stunden
clr rFlag ; Loesche alle Flaggen
rcall ClearData ; Loesche SRAM-Daten
rcall Display ; Alle Kanale anzeigen
; Initiiere TC2 als Millisekunden-Timer
.if clock == 2048000
; TC2 als normalen Timer mit Precaler 8 und Ueberlauf-Int
ldi rmp,(1<<CS21) ; Prescaler = 8
out TCCR2,rmp
ldi rmp,(1<<OCIE1A)|(1<<TOIE2) ; TC1=Comp Match Int, TC2=Overflow Int
out TIMSK,rmp
.else
ldi rmp,cCmp2A ; Setze CTC-Vergleichswert
out OCR2,rmp ; im Compare-Port
.if clock == 8000000
ldi rmp,(1<<WGM21)|(1<<CS22) ; CTC-Mode, Prescaler=64
.else
ldi rmp,(1<<WGM21)|(1<<CS21) ; CTC-Mode, Prescaler=8
.endif
out TCCR2,rmp ; in TC2 control port
ldi rmp,(1<<OCIE1A)|(1<<OCIE2) ; TC1-Comp Match Int, TC2-Comp Match Int
out TIMSK,rmp
.endif
; Schlafmodus
ldi rmp,1<<SE ; Schlafmodus an, Idle-Modus
out MCUCR,rmp
; Interrupt Enable
sei ; Interrupts an
;
; *****
; P R O G R A M M S C H L E I F E
; *****
;
Loop:
sleep ; Schlafen
nop ; Aufwachen
sbrc rFlag,bmS ; Millisekundenflagge?
rcall Millisecond ; Millisekunde bearbeiten
sbrc rFlag,bdS ; Dezisekundenflagge?
rcall Dezisecond ; Dezisekunde bearbeiten
rjmp loop ; Schleife
;
; Eine Millisekunde ist vorbei, pruefe Tasten
Millisecond:
cbr rFlag,1<<bmS ; Loesche Flagge
mov rmsBuf,rmSec ; Millisekundenstand in Puffer
ldi rmp,0b00000001 ; Starte mit Taste 1
mov rKeyCmp,rmp ; in Vergleichsregister
and rmp,rKey ; Pruefe Portbit
brne ChkStartStop ; Bit ist high, teste Taste 2
; Reset-Taste gedrueckt
clr rFlag ; Loesche alle Flaggen
clr rmSec ; Starte Zeit bei Null
clr rdSec
clr rSec
clr rMin
clr rHour
rcall ClearData ; Loesche SRAM-Daten
ldi rmp,4 ; Ton #4
rcall ToneStart ; Starten
clr ZH ; Erste Zeile
clr ZL ; Erste Spalte
rcall LcdPos ; LCD-Position
ldi ZH,High(2*LcdInit2) ; Maske 2 ausgeben

```

```

ldi ZL,Low(2*LcdInit2)
rcall LcdText
rjmp Display ; Alle Kanäle anzeigen
ChkStartStop:
; Start/Stop-Taste?
lsl rKeyCmp ; Nächstes Bit
mov rmp,rKey ; Tastenstand in rmp
and rmp,rKeyCmp ; Mit Bitmaske vergleichen
brne ChkToggle ; Eins, prüfe Toggel-Zähler
tst rTgl ; Prüfe Toggelzähler
brne ChkStartStopRestartToggle ; Nicht Null, Neustart
ldi rmp,1<<bRun ; bRun-Bit in rmp
eor rFlag,rmp ; Flagge umkehren
ChkStartStopRestartToggle:
; Toggel-Zähler neu starten
ldi rmp,cToggle ; Toggle-Wert
mov rTgl,rmp ; in Toggle-Zähler
rjmp ChkChannels ; Weiter mit Kanaltasten
ChkToggle:
; Eingang high, teste Toggelzähler
tst rTgl ; Toggelzähler Null?
breq ChkChannels ; Ja, weiter mit Kanaltasten
dec rTgl ; Toggelzähler vermindern
;
; Alle Kanaltasten prüfen
; rKeyCmp zeigt auf Vergleichswert Start/Stop
ChkChannels:
mov rmp,rmSec ; Zeit = Null?
or rmp,rdSec
or rmp,rSec
or rmp,rMin
or rmp,rHour
breq ChkChannels4 ; Ja, Tasten nicht auswerten
clr ZH ; Beginne in LCD-Zeile 1
ldi XH,High(sData) ; Zeiger auf ersten Datensatz
ldi XL,Low(sData)
ChkChannels1:
lsl rKeyCmp ; Nächster Tastenvergleich
mov rmp,rKeyCmp ; in rmp
and rmp,rKey ; Ist diese Taste gedrückt?
brne ChkChannels2 ; Taste nicht gedrückt
mov rmp,rKeyCmp ; Lese Tastenvergleich
and rmp,rFlag ; Ist dieser Kanal angehalten?
brne ChkChannels2 ; Ja, zur nächsten Taste
; Kanaltaste ist erstmals gedrückt
; Disable interrupts to avoid ms increase
; during copy operation
st X+,rHour ; Speichern Zeit
st X+,rMin
st X+,rSec
st X+,rdSec
st X+,rmsBuf ; Millisekunden aus Puffer!
sbiw XL,5 ; An den Pufferanfang
or rFlag,rKeyCmp ; Setze Kanalflagge
st Y+,ZH ; Kanal in Reihenfolgeliste
push ZH ; Zeile retten
ldi ZL,6 ; Ausgabeposition
rcall LcdPos ; in LCD wählen
rcall DisplayX ; Zeile aus Zeiger X anzeigen
ldi rmp,7 ; Ton auswählen
pop ZH ; Zeile vom Stapel holen
push ZH ; und zurück auf den Stapel
add rmp,ZH ; Addiere den Zeilenzähler
rcall ToneStart ; Spiele den Ton

```

```

    pop ZH ; Wiederherstellen Zeilenzaehler
    rjmp ChkChannels3 ; Naechste Taste
ChkChannels2:
    ; Kanal inaktiv, passe Zeiger an
    adiw XL,5 ; Zeige auf naechsten Datensatz
ChkChannels3:
    inc ZH ; Erhoehe Zeilenzaehler
    cpi ZH,4 ; Ende erreicht?
    brcs ChkChannels1 ; Pruefe naechsten Kanal
    ldi rmp,0b00111100 ; Alle Kanaele
    and rmp,rFlag ; isolieren
    cpi rmp,0b00111100 ; Alle Kanaele Eins?
    breq DisplaySorted ; Ja, gib sortiert aus
ChkChannels4:
    ret
;
; Gib die Zeiten sortiert aus
DisplaySorted:
    .if Eep == 1
        rcall Write2Eep ; Schreibe die Daten ins EEPROM
    .endif
    sbr rFlag,1<<bSort ; Setze Sortiertflagge
    cbr rFlag,1<<bRun ; Halte Uhr an
    clr ZH ; Zeilenzaehler, beginne bei Zeile 1
    ldi YH,High(sChannelRow) ; Y auf sortierte Liste
    ldi YL,Low(sChannelRow)
DisplaySorted1:
    ld rmp,Y ; Lese Kanalnummer aus sortierter Liste
    cpi rmp,0xFF ; Kanal nicht angehalten?
    brne DisplaySorted3 ; Kanal angehalten
    ; Loesche diese Zeile
    ldi ZL,0 ; Spalte = 0
    rcall LcdPos ; LCD-Position setzen
    ldi ZL,20 ; 20 Leerzeichen
DisplaySorted2:
    ldi rmp,' ' ; Leerzeichen
    rcall LcdChar ; auf LCD
    dec ZL ; Weitere Leerzeichen?
    brne DisplaySorted2 ; Ja, weiter
    rjmp DisplaySorted6 ; Naechster Kanal
DisplaySorted3:
    ldi XH,High(sData) ; Zeiger auf Anfang
    ldi XL,Low(sData)
    tst rmp ; Datensatz = 0?
    breq DisplaySorted5 ; Ja
DisplaySorted4:
    adiw XL,5 ; Naechster Datensatz
    dec rmp ; Abwaerts
    brne DisplaySorted4 ; Weiter
DisplaySorted5:
    ldi ZL,5 ; Spaltenposition
    rcall LcdPos ; Position auf LCD
    ld rmp,Y ; Lese Kanalnummer erneut
    subi rmp,-'0'-1 ; Addiere ASCII-Eins
    rcall LcdChar ; Kanalnummer ausgeben
    push ZH ; Zeilenzaehler retten
    rcall DisplayX ; Datensatz ausgeben
    pop ZL ; Zeilenzaehler wieder herstellen
DisplaySorted6:
    adiw YL,1 ; Naechster Listeneintrag
    inc ZH ; Naechste Zeile
    cpi ZH,4 ; Alle ausgegeben?
    brcs DisplaySorted1 ; Nein, weiter
    ret

```

```

;
; Zeige Zeile aus Seicher an
; X zeigt auf Datensatz
DisplayX:
    ldi rmp,'=' ; Mit Gleichheitszeichen beginnen
    rcall LcdChar
    ldi rmp,' ' ; und ein Leerzeichen
    rcall LcdChar
    ld rmp,X+ ; Lade Stunden
    rcall LcdDec2 ; und zeige an
    ldi rmp,':' ; Doppelpunktrenner
    rcall LcdChar ; anzeigen
    ld rmp,X+ ; Lade Minuten
    rcall LcdDec2 ; und zeige an
    ldi rmp,':' ; Doppelpunktrenner
    rcall LcdChar ; anzeigen
    ld rmp,X+ ; Lade Sekunden
    rcall LcdDec2 ; und zeige an
    .if LangEn == 1
        ldi rmp',' ; Dezimalpunkt
    .else
        ldi rmp',' ; Dezimalkomma
    .endif
    rcall LcdChar ; anzeigen
    ld rmp,X+ ; Lade Dezisekunden
    subi rmp,-'0' ; In ASCII
    rcall LcdChar ; und anzeigen
    ld rmp,X+ ; Lade Millisekunden
    rjmp LcdDec2 ; und zeige an
;
; Debugging, schreibe SRAM-Inhalt in das EEPROM
Write2Eep:
    ldi XH,High(sData) ; SRAM-Zeiger
    ldi XL,Low(sData)
    clr ZH ; EEPROM Adresszeiger
    clr ZL
Write2Eep1:
    out EEARH,ZH ; Adresse schreiben
    out EEARL,ZL
    ld rmp,X+ ; SRAM-Byte lesen
    out EEDR,rmp ; in Datenport
    ldi rmp,1<<EEMWE ; Master Write Enable
    cli ; Interrupts abschalten
    out EECR,rmp ; Starte EEPROM Write Enable
    ldi rmp,1<<EWE ; Write Enable
    out EECR,rmp ; Starte EEPROM Write
    sei ; Interrupts wieder zulassen
Write2Eep2:
    sbic EECR,EWE ; Fertig geschrieben?
    rjmp Write2Eep2 ; Nein, warte weiter
    adiw ZL,1 ; Naechste Adresse
    cpi XL,Low(sDataEnd+4) ; Alles geschrieben?
    brne Write2Eep1 ; Nein, weiter
    ret
;
; Eine Zehntelsekunde ist vorbei, erhoehe Zeit
Dezisecond:
    cbr rFlag,1<<bdS ; Loesche Flagge
    inc rdSec ; Erhoehe Dezisekunde
    cpi rdSec,10 ; Schon bei 10?
    brcs Dezisecond2 ; Nein, weiter
    clr rdSec ; Neustart Dezisekunde
    inc rSec ; Erhoehe Sekunden
    cpi rSec,60 ; 60 Sekunden?

```



```

ldi rmp,1 ; Ton #1
brcs Dezisecond1 ; Nein, starte Sekundenton
inc rmp ; Naechster Ton
clr rSec ; Neustart Sekunden
inc rMin ; Erhoehe Minuten
cpi rMin,60 ; 60 Minuten?
brcs Dezisecond1 ; Nein, starte Minutenton
inc rmp ; Naechstr Ton
clr rMin ; Neustart Minuten
inc rHour ; Erhoehe Stunden
cpi rHour,24 ; 24 Stunden?
brcs Dezisecond1 ; Nein
clr rHour ; Neustart Stunden
Dezisecond1:
rcall ToneStart ; Spiele Ton
DeziSecond2:
sbrs rFlag,bSort ; Anzeige im Sortiermodus?
rcall Display ; Nein, zeige Zeit an
; LED-Aktionen
cbi pLedRO,bLedRO ; LED an
sbrs rFlag,bRun ; Nicht aktiv?
ret ; Inaktiv
lsr rLed ; Schiebe niedrigstes Bit in Carry
brcc DeziSecond3 ; Null, lasse LED an
sbi pLedRO,bLedRO ; Eins, LED aus
DeziSecond3:
dec rLedCtr ; Erniedrige LED-Zaehler
brne DeziSecond4 ; Nicht bei Null
mov rmp,rFlag ; Kopiere Flaggen
ori rmp,0b11000011 ; Ausser Flaggen alles aus
mov rLed,rmp ; In LED-Register
ldi rmp,8 ; Neustart LED-Zaehler
mov rLedCtr,rmp
DeziSecond4:
ret
;
; Zeit auf allen aktiven Kanaelen anzeigen
Display:
ldi rmp,0b00000100 ; Vergleichswert Kanal 1
mov rKeyCmp,rmp ; Kopiere in Vergleichsregister
ldi ZH,0 ; Beginne mit Kanal 1 in Zeile 1
ldi ZL,8 ; Spalte 8
Display1:
push ZH ; Sichere Z
push ZL
mov rmp,rFlag ; Kopiere Flaggen in rmp
and rmp,rKeyCmp ; Ist Kanalflagge gesetzt?
brne Display2 ; Bit ist gesetzt, nicht anzeigen
rcall LcdPos ; Setze LCD-Cursor
rcall DisplayLine ; Zeige Zeile an
Display2:
lsl rKeyCmp ; Naechster Kanal
pop ZL ; Hole Spalte vom Stapel
pop ZH ; Hole Zeile vom Stapel
inc ZH ; Naechste Zeile
cpi ZH,4 ; Ende der Zeilen?
brcs Display1 ; Nein, wiederhole
ret
;
; Zeigt eine Zeile mit aktueller Zeit an
; LCD-Adresse voreingestellt
DisplayLine:
mov rmp,rHour ; Lese Stunden
rcall LcdDec2 ; Zeige an

```

```

ldi rmp, ':'
rcall LcdChar
mov rmp, rMin ; Lese Minuten
rcall LcdDec2 ; Zeige an
ldi rmp, ':'
rcall LcdChar
mov rmp, rSec ; Lese Sekunden
rcall LcdDec2 ; Zeige an
.if LangEn == 1
    ldi rmp, '.' ; Dezimalpunkt
.else
    ldi rmp, ',' ; Dezimalkomma
.endif
rcall LcdChar ; Zeichen ausgeben
mov rmp, rdSec ; Lese Dezisekunden
subi rmp, -'0'
rjmp LcdChar
;
; Loesche Daten im SRAM
ClearData:
    ldi ZH, High(sData) ; Z an den Datenanfang
    ldi ZL, Low(sData)
    clr rmp ; Schreibe Nullen
ClearData1:
    st Z+, rmp ; Schreibe Byte in Datenpuffer
    cpi ZL, Low(sDataEnd) ; Pufferende?
    brne ClearData1 ; Nein, weiter
    ldi YH, High(sChannelRow+4) ; Setze Reihenfolge-Zeiger
    ldi YL, Low(sChannelRow+4)
    ldi rmp, 0xFF ; Fuelle Reihenfolge mit FF
    st -Y, rmp ; In letztes Byte
    st -Y, rmp ; In vorletztes
    st -Y, rmp ; In vorvorletztes
    st -Y, rmp ; In Vorvorvorletztes
    ret
;
; Starte einen Ton
; Tonnummer in rmp
ToneStart:
    lsl rmp ; Multipliziere mit 2
    lsl rmp ; Multipliziere mit 4
    ldi ZH, High(2*ToneTable) ; Tontabelle in Z
    ldi ZL, Low(2*ToneTable)
    add ZL, rmp ; Addiere Tonnummer*4 in LSB
    ldi rmp, 0 ; Zum Carry addieren
    adc ZH, rmp ; Carry in MSB
    lpm rCtrL, Z+ ; Lese LSB Tondauer
    lpm rCtrH, Z+ ; Lese MSB Tondauer
    lpm rmp, Z+ ; Lese LSB CTC-Wert
    lpm ZH, Z ; Lese MSB CTC-Wert
    out OCR1AH, ZH ; Schreibe MSB in Compare A
    out OCR1AL, rmp ; Schreibe LSB in compare A
    ldi rmp, 1 << COM1A0 ; Lausprecherausgang torkeln
    out TCCR1A, rmp ; Starte Ton
    ret
;
; Tontabelle
ToneTable:
    .dw cCtrInit, cCmpInit ; Ton 0
    .dw cCtrSec, cCmpSec ; Ton 1
    .dw cCtrMin, cCmpMin ; Ton 2
    .dw cCtrHour, cCmpHour ; Ton 3
    .dw cCtrReset, cCmpReset ; Ton 4
    .dw cCtrStart, cCmpStart ; Ton 5

```

```

.dw cCtrStop,cCmpStop ; Ton 6
.dw cCtrCh1,cCmpCh1 ; Ton 7
.dw cCtrCh2,cCmpCh2 ; Ton 8
.dw cCtrCh3,cCmpCh3 ; Ton 9
.dw cCtrCh4,cCmpCh4 ; Ton 10
;
; Include der Lcd-Routinen
.include "lcd.inc"
;
; Lcd-Init Text
.if LangEn == 1
; Init Text englisch
LcdInit1:
.db " Stop watch ATmega8",0x0D
.db " Four channels at",0x0D,0xFF
.db " one millisecond",0x0D,0xFF
.db " (C)2018 by DG4FAC",0xFE
;
; Maske englisch
LcdInit2:
.db "Track1 00:00:00.0 ",0x0D,0xFF
.db "Track2 00:00:00.0 ",0x0D,0xFF
.db "Track3 00:00:00.0 ",0x0D,0xFF
.db "Track4 00:00:00.0 ",0xFE,0xFF
;
01234567890123456789
.else
; Init Text deutsch
LcdInit1:
.db " Stoppuhr ATmega8",0x0D,0xFF
.db " Vier Kanaele mit",0x0D,0xFF
.db " einer Millisekunde",0x0D
.db " (C)2018 by DG4FAC",0xFE
;
; Maske deutsch
LcdInit2:
.db "Bahn 1 00:00:00.0 ",0x0D,0xFF
.db "Bahn 2 00:00:00.0 ",0x0D,0xFF
.db "Bahn 3 00:00:00.0 ",0x0D,0xFF
.db "Bahn 4 00:00:00.0 ",0xFE,0xFF
;
01234567890123456789
.endif
;
; Copyright Info
.db "(C)2018 by avr-asm-tutorial.net " ; Normaler Text
.db "C(2)10 8yba rva-mst-turoai.len t" ; Wortweise
;
; Ende Quellcode
;

```

Lob, Tadel, Fehlermeldungen, Genöle und Geschimpfe oder Spam bitte über das [Kommentarformular](#) an mich.