ATmega16

Mikroprofessor-

Wecker zum

Angeben

as

Designed and fabricated by

Schmidt Microsystems

GmbH & Co KG

Juchhee!

Herzlicher Gluckwunsch fur Kaufen sensatzjonelle Aufwachuhr von Schmidt Microsystems. Werden viel Vreude haben bei immer Aufwachen schone Melodie. Nix verschlafen oder mude in Tag, immer froh, singen und immer machen Spasze.

Fur Anmachen

Nur brauchen Strom aus Dose Plug-In. Stecken rein grosses Stecker. Nix Leuchten jetzt, weil Schaltung aus wenn gekriegen.

Musst erst schalten ein auf rechte Seite mit Umlegen kleines Schalter fur Strom. Ohne Einschalten nix Aufwachen und nix Stromverbrauch.

Wenn einschalten dann leuchten auf Zahlen ganz viel Nullen. Musst nix warten bis nachts, weil Stellen Uhr auf Zeit moglich.

Teile fur Bedien

Hier alles fur Bedien. Merken Name und Machen fur spater Beschreib.

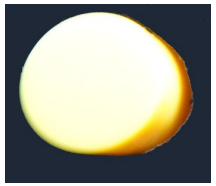
Fur stellen Uhr

Stellen immer alles mit Reihe Tasten. Musst merken alles in Kopf, sonst Durcheinander.

Fur stellen Uhr musst drucken blau taste von Reihe Tasten.

Wenn blau Taste erste Mal gedruckt, Stellrad links macht Stunde Uhr neu dauernd. Wenn Stunde hoch wie soll, blaue Taste nochmal drucken. Jetzt Stellrad links macht Minuten neu dauernd. Wenn Minute von Uhr richtig, dann blau Taste noch mal drucken. Jetzt Uhr fertig gestellt und laufen bis neu stellen.

Fur stellen Weckung

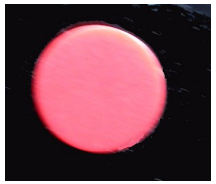


Fur stellen Weckung du drucken Taste gelb.

Wenn erste mal gelb, dann Stellrad links macht Stunde Weckzeit laufen. Wenn richtig drucke wieder gelbe Taste. Jetzt macht Stellrad links laufen Minuten. Wenn richtig dann wieder Taste gelb drucken. Jetzt Weckzeit gestellt und Alarm angemacht. Kann sehen an Lampen blink unten.

Fur Alarm

Wenn Zeit vergeht und Weckzeit gleich Stunde Minute, dann macht Musik. Wenn aufgewacht und wollen Aufstehen, Taste rot drucken.



Nach drucken Taste rot jetzt Alarm aus. Merken weil Lampen Weckzeit nix mehr blinkt. Wenn wieder Taste rot gedruckt, wieder blinkt.

Wenn nix rote Taste gedruckt, dann machen immer neu Musik nach Minute bis rot gedruckt.

Wenn wecken zu fruh und lieber noch bisschen schnarchen dann drucken nicht rote Taste aber gelbe Taste fur fuenf Minuten schnarchen oder blaue Taste fur zehn Minuten schnarchen. Wenn ganze Stunde weiter schnarchen, dann drucken blaue Taste fur sechs mal. Lampen unten musst weiter blinken, weil sonst nix Wecken.

Fur dunkel Lampen

Wenn Lampen zu hell, dann musst dunkel machen mit Stellrad rechts drehen. Kann fast ganz ausmachen Licht. Aber dann nix sehen Uhr und blinken.

Fur Musik

Musik macht acht verschiedene, mehr nix in Kaufpreis drin. Kann kaufen mit Geld moglich

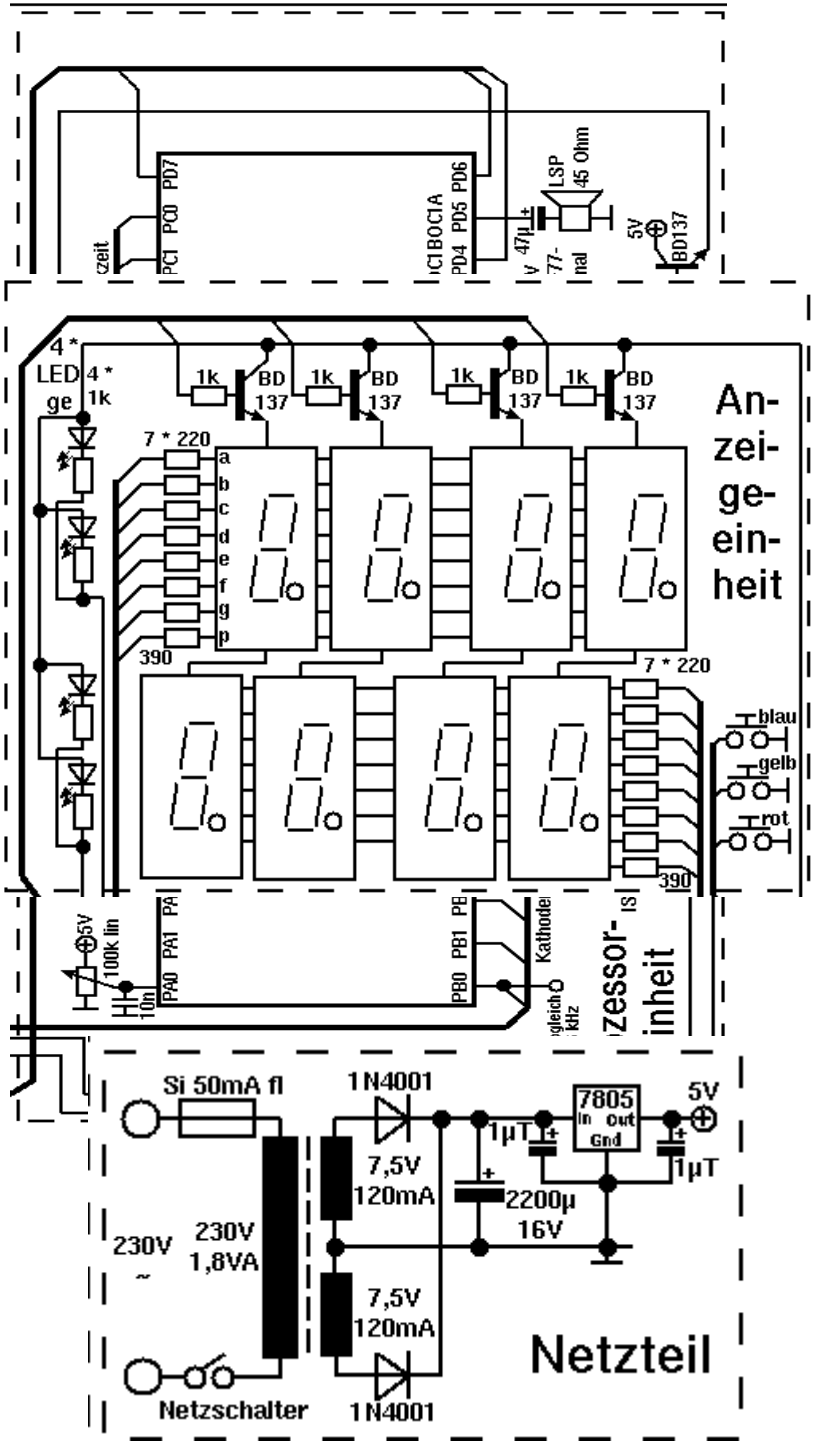
sechzehn Lied. Kann sehen nachste Lied an Dezimalpunkte von Weckzeit. Kann nix wahlen weil Programmierer faul und abgehauen.

Fur Verstehen Arbeit

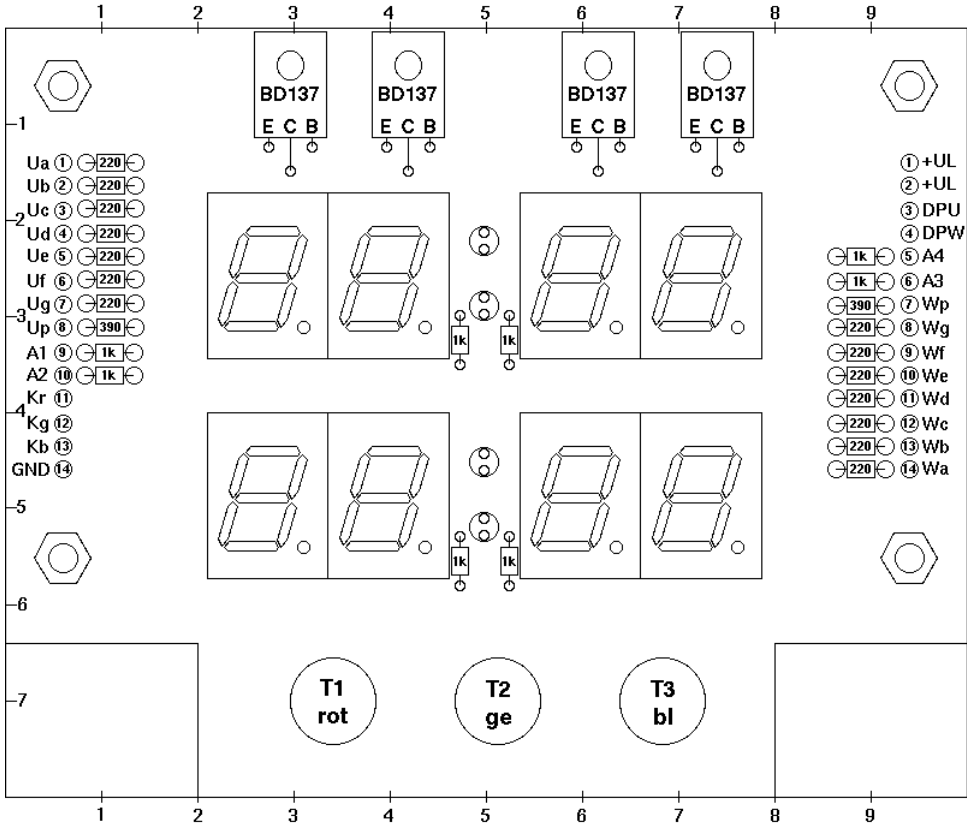
Teil gebaut mit Mikroprofessor ATMEL AVR Atmega16. Macht alles von alleinig:

- rennt mit 3.686.400 Befehle pro Sekunde aus Schwingen Quarz,
- alles mit Interrupt und meistens schlafen bis Aufwecken von Gerat Timer0, Timer1 oder ADC,
- treibt Lampen Siebensegment mit 100 mal in Sekunde an, damit nix sehen flimmern von an und aus,
- schaltet Lampen blink an,
- Timer0 stopt Sekunden, zahlt sechzig, rechnet Minuten, zahlt 24, rechnet Stunden und macht fertig fur Anzeige,
- Timer1 macht Musik nach Tabelle fur Noten,
- AD-Wandel macht Einstellung Stunden und Minuten mit malnehmen von 8-Bit-Wandel mit 8-Bit-Multiplikator.

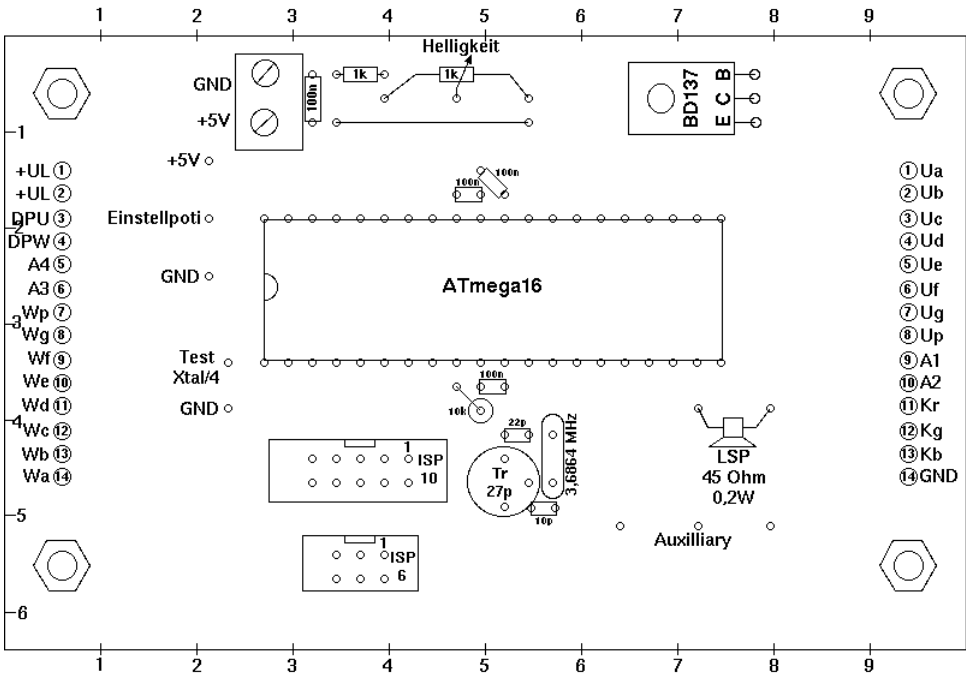
Fur Schaltbild



Fur Bestuckung Leuchtteil



Fur Bestuckung Prozessorteil



Fur Software

```
;
; ******************************************
; * Uhr mit 2 * 4 Digit 7-Segment Display  *
; * ATmega16, Interrupt-gesteuert          *
; * (C)2013 by http://www.avr-asm-tutorial.net *
; ******************************************
;
; Include-Datei fuer AVR-Typ ATmega16
.NOLIST
.INCLUDE "m16def.inc" ; Headerdatei fuer ATMEGA16
.LIST
;
; ==========================================
;   H A R D W A R E I N F O R M A T I O N E N
; ==========================================
;                              _____________
;                             /             |
;     Kathode Uhr a o--|PB0       PA0|--o Stellpotentiometer
;     Kathode Uhr b o--|PB1       PA1|--o frei
;     Kathode Uhr c o--|PB2       PA2|--o frei
;     Kathode Uhr d o--|PB3       PA3|--o frei
;     Kathode Uhr e o--|PB4       PA4|--o Anode 1
;      Uhr f + MOSI o--|PB5       PA5|--o Anode 2
;      Uhr g + MISO o--|PB6       PA6|--o Anode 3
;       Uhr p + SCK o--|PB7  AT   PA7|--o Anode 4
;             RESET o--|RES      AREF|--o +5V
;               +5V o--|VCC mega AGND|--o GND
;               GND o--|GND      AVCC|--o +5V
;  3,6864 MHz XTAL2 o--|XT2  16   PC7|--o Kathode Weckzeit p
;  3,6864 MHz XTAL1 o--|XT1       PC6|--o Kathode Weckzeit g
;     Uhr Doppelpkt o--|PD0       PC5|--o Kathode Weckzeit f
;    Weck Doppelpkt o--|PD1       PC4|--o Kathode Weckzeit e
;     DCF77 Eingang o--|PD2       PC3|--o Kathode Weckzeit d
;              frei o--|PD3       PC2|--o Kathode Weckzeit c
;         Taste rot o--|PD4       PC1|--o Kathode Weckzeit b
;       LSP Ausgang o--|PD5       PC0|--o Kathode Weckzeit a
;        Taste gelb o--|PD6       PD7|--o Taste blau
;                      |_____________|
;
```

```
; ============================================
;     P O R T S   U N D   P I N S
; ============================================
;
; Display-Ports
.EQU pKUhrOut  = PORTB
.EQU pKUhrDir  = DDRB
.EQU pKWeckOut = PORTC
.EQU pKWeckDir = DDRC
.EQU pAOut     = PORTA
.EQU pADir     = DDRA
.EQU pAIn      = PINA
.EQU pKDPOut   = PORTD
.EQU pKDPDir   = DDRD
.EQU bDPUhr    = PORTD0
.EQU bDPWeck   = PORTD1
; Tasten
.EQU pKeyOut    = PORTD
.EQU pKeyDir    = DDRD
.EQU pKeyIn     = PIND
.EQU bKeyRed    = PIND4
.EQU bKeyYellow = PIND6
.EQU bKeyBlue   = PIND7
.EQU cKeyMask =(1<<bKeyRed)|(1<<bKeyYellow)|(1<<bKeyBlue)
; Lautsprecher
.EQU pLspOut    = PORTD
.EQU pLspDir    = DDRD
.EQU bLsp       = PORTD5
;
; ================================================
; K O N S T A N T E N   Z U M  E I N S T E L L E N
; ================================================
;
; Taktung
.EQU cClock =  3686400 ; Quarzfrequenz
;
; Tasten aktiv
.EQU cKeyCnt = 28
;
; Auswertung ADC-Wert fuer Anzeige
.EQU cMult = 22 ; ca. 10 ms
;
; Timing TC0 (Uhr und Display-Mux):
;   Clock ==> Prescaler ==> Timer0 ==> CTC=144 ==> 4 Digits ==> Register
; 3.6864M        64          57600       400         100           1
;
; Tongenerator:
; Ton      Clock ==> Prescaler ==> Timer1 ==> CTC  ==> Frequ ==> ToggleOC1B
; c'     3.6864M         1        3.6864M    7045      523,26       261,63
; c''    3.6864M         1        3.6864M    3523     1046,38       523,19
; c'''   3.6864M         1        3.6864M    1761     2093,36      1046,68
; c''''  3.6864M         1        3.6864M     881     4184,34      2092,17
; c''''' 3.6864M         1        3.6864M     440     8378,18      4189,09
;
```

```
; ============================================
;   R E G I S T E R D E F I N I T I O N E N
; ============================================
; R0 verwendet fuer LPM ausserhalb Interrupts
; R1 verwendet fuer Multiplikationen
; Frei: R2..R8
.DEF rAlarmCnt = R9 ; zaehlt Anzahl Alarme
.DEF rLiedNr= R10 ; aktuelle Liednummer
.DEF rLiedL = R11 ; LSB Adresse Lied im Speicher
.DEF rLiedH = R12 ; dto. MSB
.DEF rDisp = R13 ; Display Transfer innerhalb Int
.DEF rAdc = R14 ; letzter Adc-Wert
.DEF rSreg = R15 ; Sichern SREG
.DEF rmp = R16 ; Vielzweckregister
.DEF rimp = R17 ; Vielzweckregister Interrupts
.DEF rSek = R18 ; Hunderstel Sekunden
.DEF rKey = R19 ; letzte Key-Kombination
.DEF rKeyN = R20 ; Anzahl Key-Gleich
.DEF rMult = R21 ; ADC-Multiplikationsausgabe
.DEF rFlg = R22 ; Flags
       .EQU bSek = 0 ; Flagge Sekunde vorbei
       .EQU bKeyOk = 1 ; Taste erkannt
       .EQU bAlarm = 2 ; Alarm aktiv
       .EQU bUSet  = 3 ; Setzen Uhrzeit aktiv
       .EQU bWSet  = 4 ; Setzen Weckzeit aktiv
       .EQU bSet2  = 5 ; Setzen zweite Zahl
       .EQU bMult  = 6 ; Multiplikationsausgabe
       .EQU bTonEnd = 7 ; Ton Melodieausgabe ist zu Ende
.DEF rAlarm = R23 ; Alarm Flags
       .EQU bAlarmAktiv = 0 ; Alarmierung ist aktiv
       .EQU bMusikAktiv =1 ; Musik noch nicht fertig
.DEF rTonL = R24 ; Anzahl Halbwellen Tongenerator, LSB
.DEF rTonH = R25 ; dto., MSB
; benutzt: R27:R26 = X außerhalb Interrupt
; benutzt: R29:R28 = Y innerhalb Interrupt fuer Multiplex-Zeiger
; benutzt: R31:R30 = Z außerhalb Interrupt: diverse Verwendungen
; ============================================
;      S R A M   D E F I N I T I O N E N
; ============================================
.DSEG
.ORG  0X0060
; Ausgabe der Siebensegmentanzeige oben
UhrMux:
.BYTE 4
; Ausgabe der Siebensegmentanzeige unten
WeckMux:
.BYTE 4
; Aktuelle Uhrzeit HH:MM:SS
Zeit:
.BYTE 3
; Weckzeit HH:MM
Weck:
.BYTE 2
; Verlaengerte Weckzeit HH:MM
WeckAktuell:
.BYTE 2
```

```
; ============================================
;   R E S E T   U N D   I N T   V E K T O R E N
; ============================================
;
.CSEG
.ORG $0000
       jmp Main ; Reset-Vektor
       reti ; INT0 Int Vektor 1
       nop
       reti ; INT1 Int Vektor 2
       nop
       reti ; TC2COMP Int Vektor 3
       nop
       reti ; TC2OVF Int Vektor 4
       nop
       reti ; TC1CAPT Int Vektor 5
       nop
       rjmp TC1CASr ; TC1COMPA Int Vektor 6
       nop
       reti ; TC1COMPB Int Vektor 7
       nop
       reti ; TC1OVF Int Vektor 8
       nop
       reti ; TC0OVF Int Vektor 9
       nop
       reti ; SPI/STC Int Vektor 10
       nop
       reti ; USART/RXC Int Vektor 11
       nop
       reti ; USART/UDRE Int Vektor 12
       nop
       reti ; USART/TXC Int Vektor 13
       nop
       rjmp IntAdc ; ADC Int Vektor 14
       nop
       reti ; EERDY Int Vektor 15
       nop
       reti ; ANACOMP Int Vektor 16
       nop
       reti ; TWI Int Vektor 17
       nop
       reti ; INT2 Int Vektor 18
       nop
       rjmp IntTC0Ctc ; TC0COMP Int Vektor 19
       nop
       reti ; SPM_RDY Int Vektor 20
       nop
;
; ========================================
;    I N T E R R U P T   S E R V I C E
; ========================================
;
; Multiplex-Interrupt
IntTC0Ctc:
       in rSreg,SREG ; sichern SREG
       in rimp,pAIn ; Lese Anodentreiber
       andi rimp,0XF0 ; losche niedriges Nibble
       lsl rimp ; Mux-Kanal links schieben
       brcc IntTC0Ctc1 ; Kein Neustart
       ldi YH,HIGH(UhrMux) ; Zeige auf Mux-Tabellenanfang
       ldi YL,LOW(UhrMux)
       ldi rimp,0b00010000 ; Neustart
       dec rSek
       brne IntTC0Ctc1 ; Sekunde nicht vorbei
       ldi rSek,100 ; Neustart Sekundenzaehler
       sbr rFlg,1<<bSek ; setze Sekundenflagge
IntTC0Ctc1:
       ld rDisp,Y ; lese naechsten Tabellenwert
       out pKUhrDir,rDisp
       ldd rDisp,Y+4 ; lese Zeitwert aus Tabelle
       out pKWeckDir,rDisp
       out pAOut,rimp ; schalte Mux ein
       adiw YL,1 ; naechste Position in Y
       out SREG,rSreg ; SREG wieder herstellen
       reti
;
; ADC-Interrupt
IntAdc:
       in rSreg,SREG ; sichere SREG
       in rAdc,ADCH ; lese oberstes Byte
       dec rMult ; Multiplikationsausgabe
       brne IntAdc1
       ldi rMult,cMult ; Neustart
       sbr rFlg,1<<bMult
IntAdc1:
       in rimp,pKeyIn ; lese Tasten
       andi rimp,cKeyMask ; maskiere Tasten
       cp rKey,rimp ; vergleiche mit letzter Taste
       brne IntAdc3 ; nicht gleich
       inc rKeyN ; gleiche Taste, erhoehe Zaehler
       brne IntAdc2 ; kein Ueberlauf
       dec rKeyN ; setze auf FF
       rjmp IntAdcRet
IntAdc2:
       cpi rKeyN,cKeyCnt ; Voreingestellter Wert?
       brne IntAdcRet
       sbr rFlg,1<<bKeyOk ; setze Flag erkannte Taste
       rjmp IntAdcRet
IntAdc3: ; Taste nicht gleich wie vorher
       clr rKeyN ; loesche Zaehler
       mov rKey,rimp ; kopiere Tastenkombination
IntAdcRet:
       ldi rimp,(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
       out ADCSRA,rimp ; Neustart Adc
       out SREG,rSreg ; stelle SREG wieder her
       reti
;
; TC1 Compare A Int
TC1CASr:
    in rSreg,SREG ; save SREG
       sbiw rTonL,1 ; Zaehle Tondauer abwaerts
       brne TC1CASrRet
       sbr rFlg,1<<bTonEnd
TC1CASrRet:
       out SREG,rSreg ; SREG wieder herstellen
       reti
;
```

```
; ============================================
;    H A U P T P R O G R A M M    I N I T
; ============================================
;
Main:
       rjmp Debug
;
; ************************************
;  P r o g r a m m s t a r t   N o r m a l
; ************************************
;
Start:
; Initiiere Stapel
       ldi rmp, HIGH(RAMEND) ; Initiiere MSB Stapel
       out SPH,rmp
       ldi rmp, LOW(RAMEND) ; Initiiere LSB Stapel
       out SPL,rmp
; Init Flaggen und RAM
       clr rFlg ; loesche Flaggen
       clr rAlarm
       ldi ZH,HIGH(Zeit) ; Nullsetzen Zeit und Weckzeit
       ldi ZL,LOW(Zeit)
       clr rmp
       st Z+,rmp ; Zeit
       st Z+,rmp
       st Z+,rmp
       st Z+,rmp ; Weckzeit
       st Z+,rmp
       st Z+,rmp ; Aktuelle Weckzeit
       st Z+,rmp
       rcall Zeitausgabe
; Initiiere Anzeige
       clr rmp ; Nullen in Kathodentreiber-Ports schreiben
       out pKUhrDir,rmp ; Anzeigen aus
       out pKWeckDir,rmp
       out pKUhrOut,rmp
       out pKWeckOut,rmp
       cbi pKDPOut,bDPUhr ; Null in Doppelpunkt-Ausgabeports
       cbi pKDPOut,bDPWeck
       sbi pKDPDir,bDPUhr
       sbi pKDPDir,bDPWeck
; Initiiere Anodentreiber
       ldi rmp,0xF0 ; Richtung der Anodentreiber auf Ausgabe
       out pADir,rmp
       ldi rmp,0x10 ; erste Anzeige starten
       out pAOut,rmp
; Init Tastenauswertung
       sbi pKeyOut,bKeyRed ; enable Pull-ups
       sbi pKeyOut,bKeyYellow
       sbi pKeyOut,bKeyBlue
       cbi pKeyDir,bKeyRed ; set input
       cbi pKeyDir,bKeyYellow
       cbi pKeyDir,bKeyBlue
       ldi rKeyN,0xFF
; Init Timer 0 als Display- und Uhrzeittreiber
       ldi rmp,143 ; CTC-Wert festlegen
       out OCR0,rmp
       ldi rmp,(1<<WGM01)|(1<<CS01)|(1<<CS00) ; CTC, Prescale=64
       out TCCR0,rmp
       ldi rmp,0x80 ; Lampen auf letztes Digit
       out pAOut,rmp
; Init Timer 1 mit Lied 0
       sbi pLspDir,bLsp ; setze Ausgabebit Lautsprecher auf Null
       cbi pLspOut,bLsp
       clr rLiedNr ; starte mit Lied 0
       rcall StarteMusik
; ADC und Key-Auswertung initiieren
       ldi rmp,1<<ADLAR ; AREF, left adjust, channel 0
       out ADMUX,rmp
       ldi rmp,(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
       out ADCSRA,rmp ; Neustart Adc
; Sleep mode setzen
       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 rFlg,bSek ; Sekunde vorbei
       rcall IncSek ; naechste Sekunde
       sbrc rFlg,bMult ; Ausgabe Multiplikation
       rcall Mult ; Multiplikationsergebnis ausgeben
       sbrc rFlg,bTonEnd ; Ton ist vorbei
       rcall TonEnd ; Ton beendet, starte naechsten
       sbrc rFlg,bKeyOk ; Tastendruck
       rcall KeyEqual ; gleiche Taste
       rjmp loop ; Zurueck nach Loop
;
; ============================================
;     B E A R B E I T U N G    F L A G S
; ============================================
;
; Starte Musikausgabe
StarteMusik:
       mov rmp,rLiedNr
       ldi ZH,HIGH(2*Liedtabelle)
       ldi ZL,LOW(2*Liedtabelle)
       lsl rmp ; LiedNummer mal zwei
       add ZL,rmp
       clr rmp
       adc ZH,rmp
       lpm rLiedL,Z+
       lpm rLiedH,Z+
       ldi rmp,0x01 ; setze OCR1A auf Startwert
       out OCR1AH,rmp
       out OCR1AL,rmp
       ldi rTonH,0x0 ; Starte sofort
       ldi rTonL,1
       ldi rmp,(1<<WGM12)|(1<<CS10) ; CTC, Prescale = 1
       out TCCR1B,rmp
       ldi rmp,1<<COM1A0 ; Output control Bit auf Toggle
       out TCCR1A,rmp
```

```
    ; Timer Interrupt enable
       ldi rmp,(1<<OCIE1A)|(1<<OCIE0) ; Timer Int Mask, enable OCIE0 Ints
       out TIMSK,rmp
       sbr rAlarm,1<<bMusikAktiv
       ldi rmp,10 ; Anzahl Wiederholung Alarm
       mov rAlarmCnt,rmp
       rcall IncLied ; naechstes Lied
       rjmp Weckausgabe ; Weckzeit und Liednummer anzeigen
;
; Zeit um eine Sekunde erhoehen
IncSek:
       cbr rFlg,1<<bSek ; loesche Flagge
       ldi ZH,HIGH(Zeit+2) ; setze Zeiger auf Sekunden
       ldi ZL,LOW(Zeit+2)
       ld rmp,Z ; lese Sekunden
       inc rmp ; erhoehe Sekunde
       st Z,rmp ; schreibe zurueck
       sbrc rmp,0 ; blinken Uhrzeit
       sbi pKDPDir,bDPUhr
       sbrs rmp,0
       cbi pKDPDir,bDPUhr
       sbrs rFlg,bAlarm ; blinken Weckzeit
       rjmp AlarmOff
       sbrs rmp,0
       rjmp AlarmOff
       cbi pKDPDir,bDPWeck
       rjmp IncSekMin
AlarmOff:
       sbi pKDPDir,bDPWeck
IncSekMin:
       cpi rmp,60 ; Ende Minute?
       breq IncMin
       ret
IncMin:
       sbrc rFlg,bUSet ; Uhr stellen aktiv?
       ret ; ja, abbrechen
       clr rmp ; starte Sekundenzaehler neu
       st Z,rmp
       sbiw ZL,1 ; zeige auf Minuten
       ld rmp,Z ; lese Minuten
       inc rmp ; naechste Minute
       st Z,rmp ; schreibe Minute
       cpi rmp,60 ; Ende Stunde?
       brne CheckAlarmNeu
       clr rmp ; Minutenanfang
       st Z,rmp
       sbiw ZL,1 ; auf Stunden
       ld rmp,Z ; lese Stunden
       inc rmp ; naechste Stunde
       st Z,rmp ; schreiben
       cpi rmp,24 ; Tagende?
       brne CheckAlarmNeu
       clr rmp
       st Z,rmp
CheckAlarmNeu:
       sbrs rFlg,bAlarm ; Alarm nicht enabled
       rjmp Zeitausgabe
       sbrs rAlarm,bAlarmAktiv ; Ist Alarm aktiv?
       rjmp CheckWecken ; Alarm nicht aktiv
       sbrc rAlarm,bMusikAktiv
       rjmp CheckWecken
       dec rAlarmCnt ; Zaehler erniedrigen
       brne NeuStartAlarm
AlarmAus:
       cbr rFlg,1<<bAlarm ; setze Enable Flagge zurueck
AlarmStop:
       clr rmp ; Toene abschalten
       out TCCR1A,rmp
       ldi rmp,1<<OCIE0 ; Interrupts abschalten
       out TIMSK,rmp
       cbr rAlarm,1<<bAlarmAktiv ; keine weitere Musik
       ldi rmp,10 ; Neustart Zaehler
       mov rAlarmCnt,rmp
       rjmp Zeitausgabe ; Zeit ausgeben
NeustartAlarm:
       rcall StarteMusik
CheckWecken:
       sbrs rFlg,bAlarm ; Alarm aktiv?
       rjmp Zeitausgabe ; nein
       lds R0,Weckaktuell+1 ; lese Weckminute
       lds rmp,Zeit+1 ; lese aktuelle Minute
       cp rmp,R0 ; vergleiche
       brne Zeitausgabe
       lds R0,Weckaktuell ; lese Weckstunde
       lds rmp,Zeit ; lese aktuelle Stunde
       cp rmp,R0 ; vergleiche
       brne Zeitausgabe
       sbr rAlarm,1<<bAlarmAktiv
Wecken:
       rcall StarteMusik
       ldi rmp,1 ; eine Minute addieren
       rcall AlarmAdd
Zeitausgabe:
       ldi XH,HIGH(UhrMux) ; Zeiger auf Mux Zeit
       ldi XL,LOW(UhrMux)
       lds rmp,Zeit ; Lese Stunden
       rcall SchreibeZahl
       lds rmp,Zeit+1 ; Lese Minuten
; Schreibt Zahl in rmp in SRAM ab X
Schreibezahl:
       clr R0 ; Zehnerzaehler
       dec R0 ; auf 0xFF
Div10:
       inc R0 ; erhoehe Zaehler
       subi rmp,10 ; ziehe zehn ab
       brcc Div10 ; noch kein Unterlauf
       rcall SchreibeZiffer
       subi rmp,-10 ; addiere 10
       mov R0,rmp
SchreibeZiffer:
       ldi ZH,HIGH(2*Siebensegment) ; Siebensegment-Tabelle
       ldi ZL,LOW(2*Siebensegment)
       add ZL,R0 ; Zahl in R0 auf Tabelle addieren
       clr R0 ; R0 loeschen
       adc ZH,R0 ; Ueberlauf addieren
       lpm R0,Z ; Tabellenwert nach R0
       st X+,R0 ; und in Mux schreiben
       ret
```

```
; Erhoehe Liednummer
IncLied:
       inc rLiedNr ; erhoehe Liednummer
       mov rmp,rLiedNr ; teste zu hoch
       cpi rmp,LiedMax+1
       brcs IncLied1
       ldi rmp,1 ; zurueck auf Eins
       mov rLiedNr,rmp
IncLied1:
; Weckzeit ausgeben
Weckausgabe:
       ldi XH,HIGH(WeckMux) ; Zeiger auf Mux Weck
       ldi XL,LOW(WeckMux)
       lds rmp,WeckAktuell ; Lese Stunden
       rcall SchreibeZahl
       lds rmp,WeckAktuell+1 ; Lese Minuten
       rcall Schreibezahl
LiedNummer: ; Setzt Dezimalpunkte der Weckanzeige auf Liednummer
       ldi rmp,0x80 ; oberstes Bit setzen
       mov R0,rmp
       mov ZL,rLiedNr
       andi ZL,0x0F ; maskiere oberes Nibble
       ldi XH,HIGH(WeckMux+4) ; zeige auf letzte Ziffer
       ldi XL,LOW(WeckMux+4)
       ldi rmp,4 ; vier Ziffern
LiedNummer1: ; nachstes Bit
       sbiw XL,1 ; eine Anzeige rueckwaerts
       lsr ZL ; unterstes Bit rausschieben
       brcc LiedNummer2
       ld ZH,X ; lese Anzeige
       or ZH,R0 ; setze Dezimalpunkt
    st X,ZH ; schreibe Anzeige
       rjmp LiedNummer3
LiedNummer2:
       ld ZH,X
       andi ZH,0x7F ; loesche Dezimalpunkt
       st X,ZH
LiedNummer3:
       dec rmp ; schon vier Mal?
       brne LiedNummer1
       ret
;
; Siebensegmenttabelle
Siebensegment:         ;
.DB 0b00111111,0b00000110 ; 0,1         a
.DB 0b01011011,0b01001111 ; 2,3        ----
.DB 0b01100110,0b01101101 ; 4,5    f / g / b
.DB 0b01111101,0b00000111 ; 6,7     ----
.DB 0b01111111,0b01100111 ; 8,9  e /   / c
.DB 0b01110111,0b01111100 ; A,b   ----
.DB 0b00111001,0b01011110 ; C,d    d
.DB 0b01111001,0b01110001 ; E,F
;
; Tastendruck auswerten
KeyEqual:
       cbr rFlg,1<<bKeyOk ; loesche Flagge
       mov rmp,rKey ; Key lesen
       sbrc rFlg,bAlarm ; Alarm aus?
       rjmp KeyAlarm ; Taste bei Alarm on
       ; Alarm aus, werte Tasten aus
       sbrc rmp,bKeyRed ; rote Taste
       rjmp KeyYellow
       sbr rFlg,1<<bAlarm ; setze Alarm aktiv
       ret
KeyYellow:
       sbrc rmp,bKeyYellow ; gelbe Taste
       rjmp KeyBlue
       sbrc rFlg,bWSet ; Weckzeit-Einstellungsflagge
       rjmp KeyYellow1 ; gesetzt
       sbr rFlg,1<<bWSet ; setze input Flagge
       cbr rFlg,1<<bSet2 ; setze Stunden-Flagge
       ret
KeyYellow1:
       sbrc rFlg,bSet2 ; zweite Zahl?
       rjmp KeyYellow2 ; gesetzt
       sts Weck,R1 ; Weckzeitstunde setzen
       sts WeckAktuell,R1
       sbr rFlg,1<<bSet2 ; Flagge Minuten setzen
       ldi XH,HIGH(WeckMux) ; Weckzeit Stunden auf Anzeige
       ldi XL,LOW(WeckMux)
       mov rmp,R1
       rjmp Schreibezahl
KeyYellow2:
       sts Weck+1,R1 ; Weckzeitminute setzen
       sts WeckAktuell+1,R1
       cbr rFlg,(1<<bWSet)|(1<<bSet2)
       sbr rFlg,1<<bAlarm ; aktiviere Wecken
       rjmp Weckausgabe
;
KeyBlue:
       sbrc rmp,bKeyBlue ; blaue Taste
       ret
       sbrc rFlg,bUSet ; Uhr setzen?
       rjmp KeyBlue1
       sbr rFlg,1<<bUSet ; Uhr setzen aktivieren
       cbr rFlg,1<<bSet2 ; Stunden setzen
       ret
KeyBlue1:
       sbrc rFlg,bSet2 ; Uhr Minuten setzen?
       rjmp KeyBlue2 ; ja
       sts Zeit,R1 ; Setze Stunden
       sbr rFlg,1<<bSet2 ; Minuten setzen aktivieren
       ldi XH,HIGH(UhrMux) ; Zeitausgabe Stunden
       ldi XL,LOW(UhrMux)
       mov rmp,R1
       rjmp Schreibezahl
KeyBlue2:
       sts Zeit+1,R1 ; Minuten speichern
       clr rmp ; Sekunden loeschen
       sts Zeit+2,rmp
       cbr rFlg,(1<<bUSet)|(1<<bSet2) ; Flaggen loeschen
       rjmp Zeitausgabe
;
; Tasten bei Alarm aktiv
KeyAlarm:
       sbrc rmp,bKeyRed ; rote Taste aktiv?
       rjmp KeyAlarmYellow
       lds rmp,Weck ; kopiere Original-Weckzeit
```

```
        sts WeckAktuell,rmp
        lds rmp,Weck+1
        sts WeckAktuell+1,rmp
        rcall AlarmAus ; aktive Alarme stoppen
        rjmp Weckausgabe
KeyAlarmYellow:
        sbrc rmp,bKeyYellow ; gelbe Taste aktiv?
        rjmp KeyAlarmBlue
        rcall AlarmStop ; Alarm anhalten
        ldi rmp,5 ; 5 Minuten addieren
        rjmp AlarmAdd
KeyAlarmBlue:
        sbrc rmp,bKeyBlue
        ret
        rcall Alarmstop ; Alarm abstellen
        lds rmp,WeckAktuell+1 ; lese aktuelle Weckzeit
        ldi rmp,10 ; addiere 10 Minuten
; Addiere Minuten in rmp zu aktueller Weckzeit
AlarmAdd:
        lds R0,WeckAktuell+1 ; lese aktuelle Weckzeit
        add R0,rmp ; Minuten addieren
        sts WeckAktuell+1,R0 ; speichern
        mov rmp,R0 ; Minuten lesen
        cpi rmp,60 ; Stunde ueberschritten
        brcs AlarmAdd1
        subi rmp,60 ; Minuten korrigieren
        sts WeckAktuell+1,rmp
        lds rmp,WeckAktuell ; Stunden erhoehen
        inc rmp
        sts WeckAktuell,rmp ; schreiben
        cpi rmp,24 ; Tagesende?
        brcc AlarmAdd1
        clr rmp ; Stunden Null setzen
        sts WeckAktuell,rmp
AlarmAdd1:
        rjmp Weckausgabe
;
; Multiplikationsergebnis ausgeben
Mult:
        cbr rFlg,1<<bMult ; Flagbit loeschen
        ldi XH,HIGH(UhrMux)
        ldi XL,LOW(UhrMux)
        sbrc rFlg,bUSet ; Uhr setzen?
        rjmp MultOut
        ldi XL,LOW(WeckMux)
        sbrs rFlg,bWSet ; Weckzeit setzen
        ret ; nein
MultOut:
        sbrc rFlg,bSet2 ; Minuten?
        adiw XL,2 ; ja
        ldi rmp,24 ; 24 Stunden
        sbrc rFlg,bSet2
        ldi rmp,60 ; 60 Minuten
        mul rmp,rAdc
        mov rmp,R1
        rjmp Schreibezahl
;
; *************************
;  S p i e l e   M u s i k
; *************************
;
; Ton zu Ende
TonEnd:
        cbr rFlg,1<<bTonEnd ; loesche Flagge
        mov ZH,rLiedH ; Adresse Lied lesen
        mov ZL,rLiedL
        lpm rmp,Z+
        cpi rmp,0xFF ; Ende Lied?
        breq TonEndEnd ; ja
        mov rLiedH,ZH ; Adresse naechste Note
        mov rLiedL,ZL
        mov R0,rmp ; Kopiere Note
        andi rmp,0x3F ; loesche Verzoegerungsbits
        ldi ZH,HIGH(2*TonTab)
        ldi ZL,LOW(2*TonTab)
        lsl rmp ; Note mal vier
        lsl rmp
        add ZL,rmp
        ldi rmp,0
        adc ZH,rmp
        lpm rTonL,Z+
        lpm rTonH,Z+
        out OCR1AH,rTonH
        out OCR1AL,rTonL
        cpi rTonH,HIGH(cPause) ; Pause?
        brne TonEnd1
        cpi rTonL,LOW(cPause)
        brne TonEnd1
        clr rmp ; Ausgabepin auf Null setzen
        out TCCR1A,rmp
        rjmp TonEnd2
TonEnd1:
        ldi rmp,1<<COM1A0 ; Toggle Ausgang
        out TCCR1A,rmp
TonEnd2:
        lpm rTonL,Z+ ; lese Tondauer
        lpm rTonH,Z
        mov rmp,R0
        cpi rmp,0x40 ; halbe Note?
        brcs TonEnd3
        lsr rTonH ; Dauer halbieren
        ror rTonL
        cpi rmp,0x80 ; viertel Note
        brcs TonEnd3
        lsr rTonH ; Dauer vierteln
        ror rTonL
        cpi rmp,0xC0 ; achtel Note
        brcs TonEnd3
        lsr rTonH ; Dauer achteln
        ror rTonL
TonEnd3:
;       rcall TonHex
        ret
TonEndEnd:
        ldi rmp,1<<COM1A1 ; Ausgabepin auf Null setzen
        out TCCR1A,rmp
        ldi rmp,1<<OCIE0 ; Keine Interrupts mehr
        out TIMSK,rmp
```

```
       cbr rAlarm,1<<bMusikAktiv ; loesche Musikausgabeflagge
       ret ; fertig mit Melodie, Ende
;
TonHex:
       ldi XH,HIGH(WeckMux)
       ldi XL,LOW(WeckMux)
       mov rmp,rTonH
       rcall Schreibehex
       mov rmp,rTonL
       rcall Schreibehex
       ldi XH,HIGH(UhrMux)
       ldi XL,LOW(UhrMux)
       in rmp,OCR1AH
       rcall Schreibehex
       in rmp,OCR1AL
Schreibehex:
       push rmp
       swap rmp
       rcall Schreibehex1
       pop rmp
Schreibehex1:
       andi rmp,0x0F
       ldi ZH,HIGH(2*Siebensegment)
       ldi ZL,LOW(2*Siebensegment)
       add ZL,rmp
       clr rmp
       adc ZH,rmp
       lpm rmp,Z
       st X+,rmp
       ret
;
.EQU cPause = 65530 ; Pausenton
;
; Musiktabelle
TonTab:
.DW cPause, 16; Pause, 28 Hz, n=0
.DW 28178, 39; C, 65 Hz, n=1
.DW 26596, 42; Des, 69 Hz, n=2
.DW 25104, 44; D, 73 Hz, n=3
.DW 23697, 47; Es, 78 Hz, n=4
.DW 22365, 49; E, 82 Hz, n=5
.DW 21110, 52; F, 87 Hz, n=6
.DW 19925, 56; Ges, 93 Hz, n=7
.DW 18807, 59; G, 98 Hz, n=8
.DW 17751, 62; As, 104 Hz, n=9
.DW 16755, 66; A, 110 Hz, n=10
.DW 15815, 70; B, 117 Hz, n=11
.DW 14927, 74; H, 123 Hz, n=12
.DW 14090, 78; c, 131 Hz, n=13
.DW 13299, 83; des, 139 Hz, n=14
.DW 12552, 88; d, 147 Hz, n=15
.DW 11848, 93; es, 156 Hz, n=16
.DW 11183, 99; e, 165 Hz, n=17
.DW 10555, 105; f, 175 Hz, n=18
.DW 9962, 111; ges, 185 Hz, n=19
.DW 9403, 118; g, 196 Hz, n=20
.DW 8875, 125; as, 208 Hz, n=21
.DW 8377, 132; a, 220 Hz, n=22
.DW 7907, 140; b, 233 Hz, n=23
.DW 7463, 148; h, 247 Hz, n=24
.DW 7044, 157; c', 262 Hz, n=25
.DW 6649, 166; des', 277 Hz, n=26
.DW 6276, 176; d', 294 Hz, n=27
.DW 5923, 187; es', 311 Hz, n=28
.DW 5591, 198; e', 330 Hz, n=29
.DW 5277, 210; f', 349 Hz, n=30
.DW 4981, 222; ges', 370 Hz, n=31
.DW 4701, 235; g', 392 Hz, n=32
.DW 4437, 249; as', 415 Hz, n=33
.DW 4188, 264; a', 440 Hz, n=34
.DW 3953, 280; b', 466 Hz, n=35
.DW 3731, 296; h', 494 Hz, n=36
.DW 3522, 314; c'', 523 Hz, n=37
.DW 3324, 333; des'', 554 Hz, n=38
.DW 3137, 352; d'', 587 Hz, n=39
.DW 2961, 373; es'', 622 Hz, n=40
.DW 2795, 396; e'', 659 Hz, n=41
.DW 2638, 419; f'', 698 Hz, n=42
.DW 2490, 444; ges'', 740 Hz, n=43
.DW 2350, 470; g'', 784 Hz, n=44
.DW 2218, 498; as'', 831 Hz, n=45
.DW 2094, 528; a'', 880 Hz, n=46
.DW 1976, 559; b'', 932 Hz, n=47
.DW 1865, 593; h'', 988 Hz, n=48
.DW 1760, 628; c''', 1047 Hz, n=49
.DW 1661, 665; des''', 1109 Hz, n=50
.DW 1568, 705; d''', 1175 Hz, n=51
.DW 1480, 747; es''', 1245 Hz, n=52
.DW 1397, 791; e''', 1319 Hz, n=53
.DW 1318, 838; f''', 1397 Hz, n=54
.DW 1244, 888; ges''', 1480 Hz, n=55
.DW 1175, 941; g''', 1568 Hz, n=56
.DW 1109, 997; as''', 1661 Hz, n=57
.DW 1046, 1056; a''', 1760 Hz, n=58
.DW 987, 1119; b''', 1865 Hz, n=59
.DW 932, 1185; h''', 1976 Hz, n=60
.DW 880, 1256; c'''', 2093 Hz, n=61
.DW 830, 1330; des'''', 2217 Hz, n=62
.DW 784, 1410; d'''', 2349 Hz, n=63
;
; Tontabelle
.EQU TPause=0
.EQU TtC=1
.EQU TtDes=2
.EQU TtD=3
.EQU TtEs=4
.EQU TtE=5
.EQU TtF=6
.EQU TtGes=7
.EQU TtG=8
.EQU TtAs=9
.EQU TtA=10
.EQU TtB=11
.EQU TtH=12
.EQU TC=13
.EQU TDes=14
.EQU TD=15
.EQU TEs=16
```

```
.EQU TE=17
.EQU TF=18
.EQU TGes=19
.EQU TG=20
.EQU TAs=21
.EQU TA=22
.EQU TB=23
.EQU TH=24
.EQU TC1=25
.EQU TDes1=26
.EQU TD1=27
.EQU TEs1=28
.EQU TE1=29
.EQU TF1=30
.EQU TGes1=31
.EQU TG1=32
.EQU TAs1=33
.EQU TA1=34
.EQU TB1=35
.EQU TH1=36
.EQU TC2=37
.EQU TDes2=38
.EQU TD2=39
.EQU TEs2=40
.EQU TE2=41
.EQU TF2=42
.EQU TGGes2=43
.EQU TG2=44
.EQU TAs2=45
.EQU TA2=46
.EQU TB2=47
.EQU TH2=48
.EQU TC3=49
.EQU TDes3=50
.EQU TD3=51
.EQU TEs3=52
.EQU TE3=53
.EQU TF3=54
.EQU TGes3=55
.EQU TG3=56
.EQU TAs3=57
.EQU TA3=58
.EQU TB3=59
.EQU TH3=60
.EQU TC4=61
.EQU TDes4=62
.EQU TD4=63
.EQU TEnde=0xFF
;
; Tondauern
.EQU DH = 0x40 ; Verkuerzung auf die Haelfte
.EQU DV = 0x80 ; Verkuerzung auf ein Viertel
.EQU DA = 0xC0 ; Verkuerzung auf ein Achtel
;
; Pausendauern
.EQU TP1=TPause
.EQU TP2=TPause+DH
.EQU TP4=TPause+DV
.EQU TP8=TPause+DA
;
; Lieder
;
; Tonleiter
LiedT:
.DB TtH,TC,TD,TE,TF,TG,TA,TH
.DB TC1,TD1,TE1,TF1,TG1,TA1,TH1,TC2
.DB TD2,TE2,TF2,TG2,TA2,TH2,TC3,TEnde
;
Lied0:
;   Au- ro- ra      mit     dem     Son-    nen-    stern
.DB TA2,TF2,TD2,TP4,TD2,TP8,TA2,TP8,TA2,TP8,TF2,TP8,TC2,TEnde
;
Lied1: ; Avanti popolo
;   A-  van-ti  po-    po- lo      Al- la  ris-cos-sa
.DB TA1,TE2,TF2,TGGes2,TG2,TG2,TP1,TF2,TE2,TF2,TG2,TG2
;       Ban-die-ra      ros-sa      A-  van-ti
.DB TP1,TG2,TA2,TG2,TP4,TG2,TF2,TP1,TA1,TE2,TF2,TP8
;   po-   po-     lo      al- la  ris-cos-    sa
.DB TGGes2,TG2,TP8,TG2,TP1,TF2,TE2,TF2,TG2,TP8,TG2,TP1
;   Ban-die-ra      ros-sa      tri-on- fe- ra
.DB TG2,TA2,TG2,TP2,TG2,TF2,TP1,TA2,TG2,TF2,TE2,TEnde
;
Lied2: ; Voelker hoert die Signale
;   Voel-         ker    hoert      die           Sig-
.DB TD3+DH,TD3+DV,TC3+DV,TA2+DH,TP1,TE2+DH,TE2+DV,TC2+DV
;   na-    le!     Auf           zum    letz-
.DB TF2+DH,TD2,TP8,TC3+DH,TC3+DV,TA2+DV,TG2+DH,TP8
;   ten    Ge- fecht!     Die In-        ter-
.DB TF2+DH,TE2,TD3+DH,TP1,TD3,TF3+DH,TP8,TF3+DH
;   na- tio-na-           le      er-    kaempft
.DB TE3,TD3,TC3+DH,TD3+DH,TE3,TP1,TE3+DH,TA2+DH
;   das    Men-schen- recht
.DB TA2+DH,TC3,TG2+DH,TA2+DH,TEnde,TP1
;
Lied3:
.DB TA2,TEnde
;
Lied4:
.DB TH2,TEnde
Lied5:
.DB TC3,TEnde
Lied6:
.DB TA3,TEnde
Lied7:
.DB TH3,TEnde
;
.EQU LiedMax = 7
;
; Liedertabelle
Liedtabelle:
.DW 2*Lied0
.DW 2*Lied1
.DW 2*Lied2
.DW 2*Lied3
.DW 2*Lied4
.DW 2*Lied5
.DW 2*Lied6
.DW 2*Lied7
```