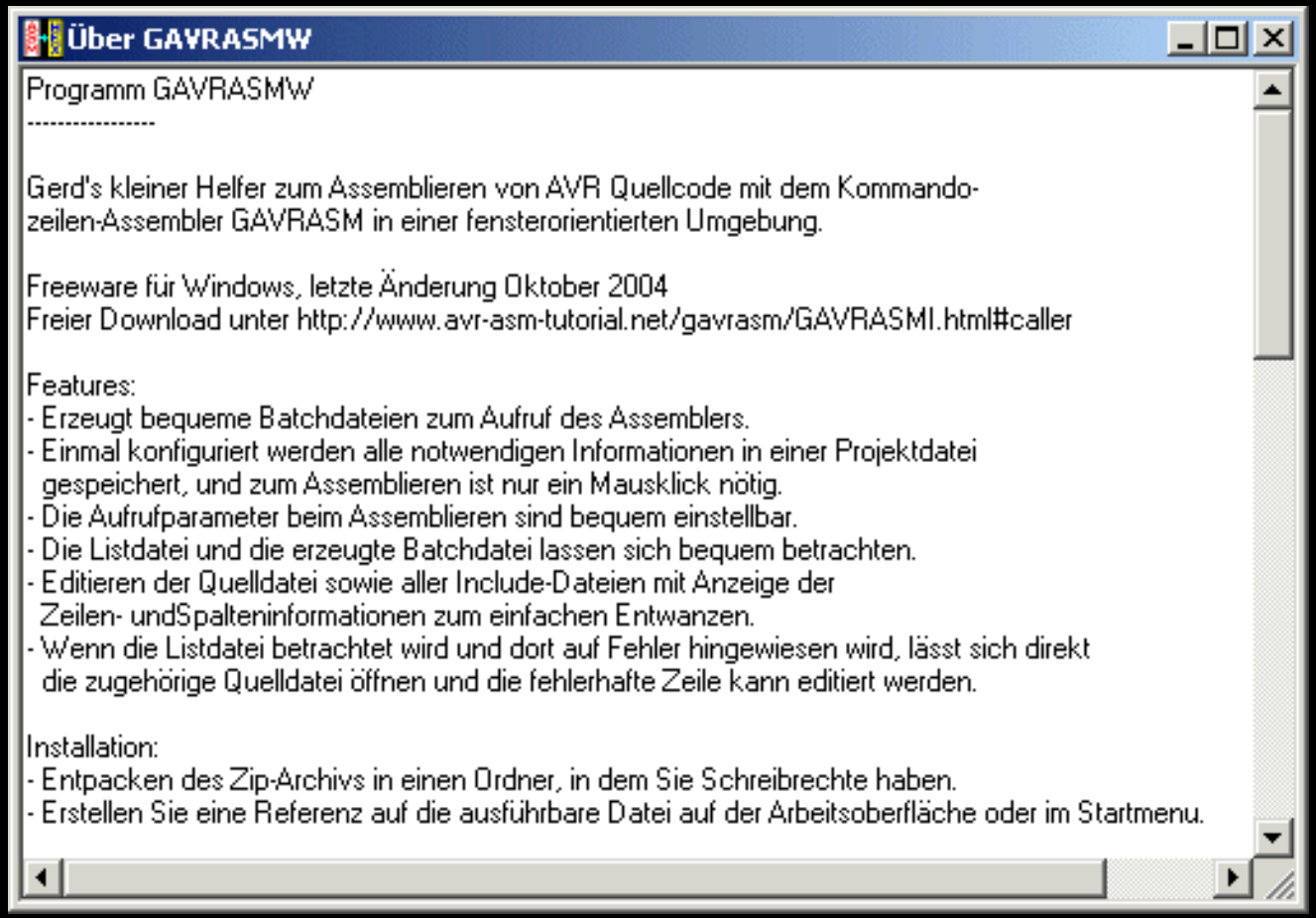
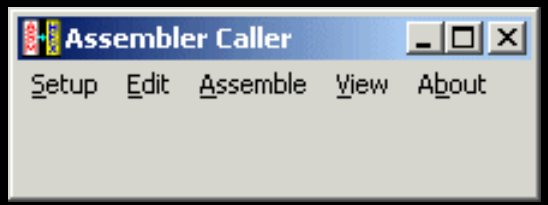
Programmierung von ATMEL AVR Mikroprozessoren am Beispiel des ATtiny13

Eine Einführung in Aufbau, Funktionsweise,
Programmierung und Nutzen von Mikroprozessoren

Teil 8: gavrasmw und weitere Beispiele

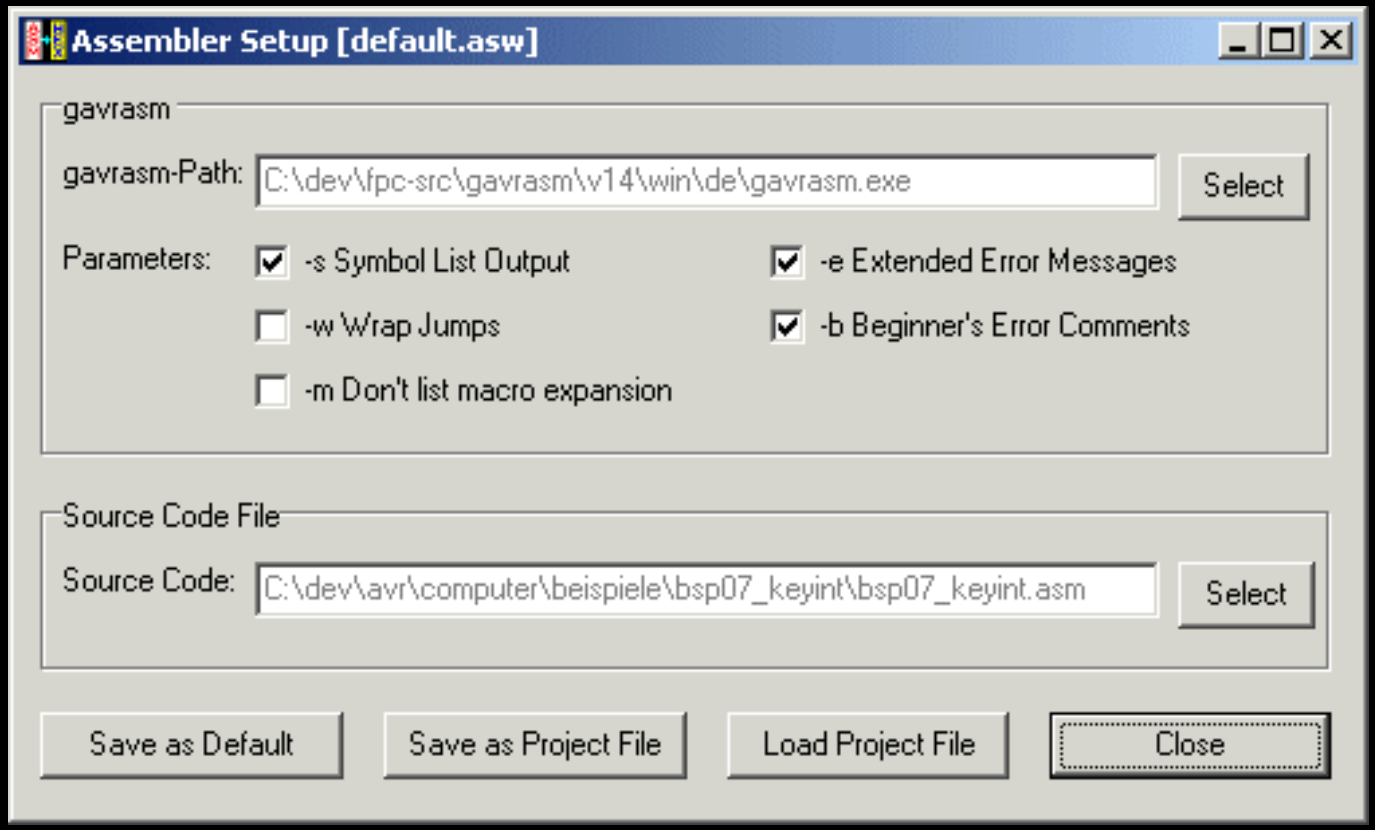
gavrasmw – Leichter fensterln



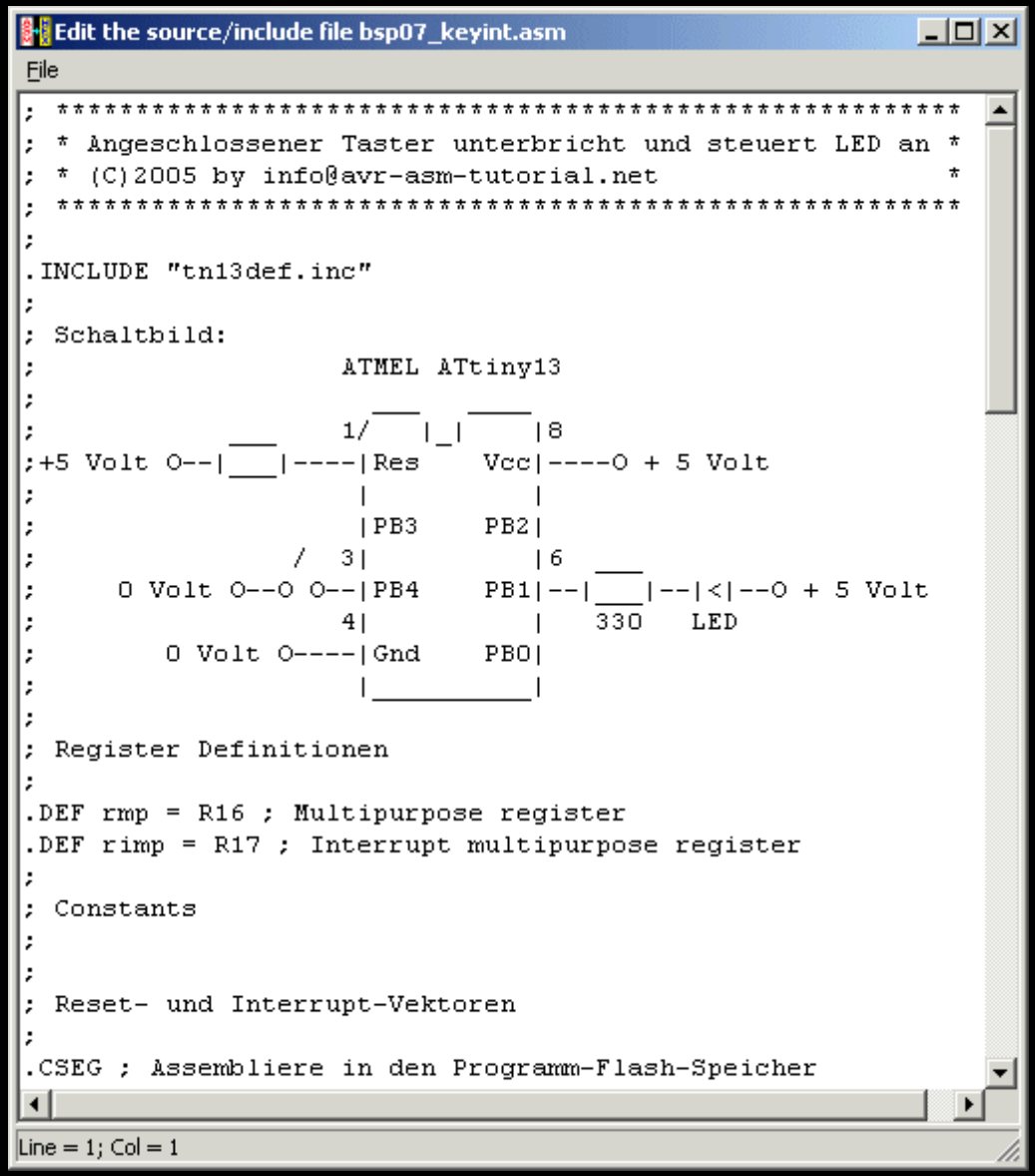


gavrasmw - Setup

Einmal einstellen: wo ist das Assembler-Programm gavrasm, wo ist meine Quellcode-Datei, welche Einstellungen beim Assemblieren hätten‘s denn gern?



gavrasmw – Kleiner Editor auf Knopfdruck



gavrasmw macht eine Batch-Datei

Das mühsame Eintippen von Pfaden wird von gavrasmw erledigt und in einer Batch-Datei abgelegt.

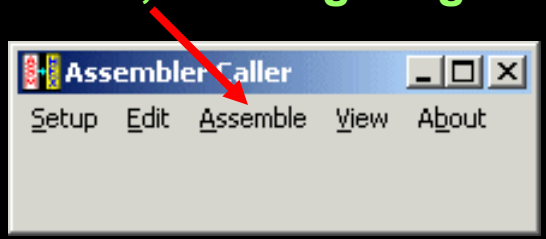
View the file bsp07_keyint.bat

File

```
REM Batchfile for bsp07_keyint.asm; Generated by gavrasmW!
C:
CD "C:\dev\avr\computer\beispiele\bsp07_keyint\"
C:\dev\fpc-src\gavrasm\v14\win\de\gavrasm.exe -seb bsp07_keyint.asm
PAUSE
```

Line = 1; Col = 1

Zum Assemblieren einfach die Batch-Datei ausführen lassen oder den Menuepunkt „Assemble“ drücken, schon geht gavrasm voll ab.



bsp08_morsekey.asm - Editor

Datei Bearbeiten Format ?

```
; ***************************************************
; * Angeschlossener Taster macht Toene              *
; * (C)2005 by info@avr-asm-tutorial.net            *
; ***************************************************
;
.INCLUDE "tn13def.inc"
;
; Schaltbild:
;                     ATMEL ATtiny13
;                          ___   ____
;                 ___     1/   |_|    |8
;+5 Volt O--|___|----|Res     Vcc|----O + 5 Volt
;                    |            |
;                    |PB3     PB2|
;                  /  3|          |6  ___
;      0 Volt O--O O--|PB4     PB1|---|_|--O + 5 Volt
;                    4|            |   LSP
;         0 Volt O----|Gnd     PB0|
;                    |____________|
;
; Register Definitionen
;
.DEF rmp = R16 ; Multipurpose register
.DEF rimp = R17 ; Interrupt multipurpose register
;
; Constants
;
;
; Reset- und Interrupt Vektoren
;
.CSEG ; Assembliere in den Programm-Flash-Speicher
.ORG $0000 ; beginne mit Adresse 0
;
; Sprungvektoren fuer Reset und Interrupts
;
        rjmp main ; Reset vector
        reti ; Int0 interrupt vector
        rjmp intpcint ; PCINT0 vector
        reti ; TC0 overflow vector
        reti ; Eeprom ready vector
        reti ; Analog comparator int vector
        reti ; TC0 CompA vector
        reti ; TC0 CompB vector
        reti ; WDT vector
        reti ; ADC conversion complete vector
;
; PCINT0 Service Routine
;   wird jedes Mal ausgefuehrt, wenn sich der Pegel am
;   Pin 3 (=PB4) aendert
;
intpcint:
        sbic PINB,4 ; Ueberspringe 1 Befehl wenn PB4 Null
        rjmp intpcint1 ; springe weil PB4 = Eins
        ldi rmp,0b00010010 ; CTC, Ausgang B bei Cmp Match a
        out TCCR0A,rmp
        reti ; Kehre vom Interrupt zurueck
intpcint1:
        ldi rmp,0b00110010 ; CTC, Ausgang B bei Cmp Match a
        out TCCR0A,rmp
        reti ; Kehre vom Interrupt zurueck
```

Beispiel 08: Taster macht Töne

- Morsetongenerator
- Kombiniert Beispiel 6 (NF-Erzeugung mit Timer) und Beispiel 7 (Interrupt-gesteuerte Tastenüberwachung) zu einem praktischen Gerät
- Minimaler Stromverbrauch wegen CPU-Schlafmodus und Timer-Automatik
- Minimale Außenbeschaltung (keine Kondensatoren, ein Widerstand)
- Genauso groß wie ein 555, aber viel flexibler (beliebige NF-Frequenz ohne Hardwareänderung und alleine durch Programmieren einstellbar)

bsp08_morsekey.asm - Editor

Datei Bearbeiten Format ?

```
;
; Main program start
;
main:
;
; Stapelzeiger setzen fuer Rueckkehr-Adressen vom Interrupt
;
        ldi rmp,LOW(RAMEND) ; Stapelzeiger auf Ende SRAM
        out SPL,rmp
;
; Angeschlossene Hardware initiieren
;
        sbi DDRB,1 ; Lautsprecher-Ausgang als Ausgang definieren
        cbi DDRB,4 ; Taster-Eingang als Eingang definieren
        sbi PORTB,4 ; Internen Pull-Up-Widerstand einschalten
;
; Tonerzeugung mit Timer TC0
; 8-Bit-Timer mit 1,2 MHz Prozessortakt mit Vorteiler durch 8
; 1,2 MHz / 8 = 150 kHz, / 75 = 2000 Hz, /2 = 1000 Hz
;
        ldi rmp,75 ; Setze Compare A auf 147 (Ende Zaehler)
        out OCR0A,rmp
        ldi rmp,38 ; Setze Compare B auf halben Timer-Wert
        out OCR0B,rmp
        ldi rmp,0b11110010 ; CTC, Ausg. A + B bei Cmp Match auf 1
        out TCCR0A,rmp
        ldi rmp,0b00000010 ; Vorteiler durch 1024, Timer starten
        out TCCR0B,rmp
;
; Pin-Change-Interrupt fuer Taste aktivieren
;
        ldi rmp,0b00010000 ; Maskieren der aktiven Eingaenge
        out PCMSK,rmp
        ldi rmp,0b00100000 ; PCINT0-Interrupts ermoeglichen
        out GIMSK,rmp
;
; Interrupts generell einschalten
;
        sei ; Setze Interrupt Flagge
;
; Schlafmodus der CPU einstellen
;
        ldi rmp,0b00100000 ; Schlafen ermoeglichen, Modus Idle
        out MCUCR,rmp
;
; Loop mit Interrupt
;
loop:
        sleep ; Prozessor schlafen legen
        nop ; Tue nichts nach dem Aufwachen
        rjmp loop ; Prozessor wieder schlafen legen
;
; Ende Quellcode
;
```

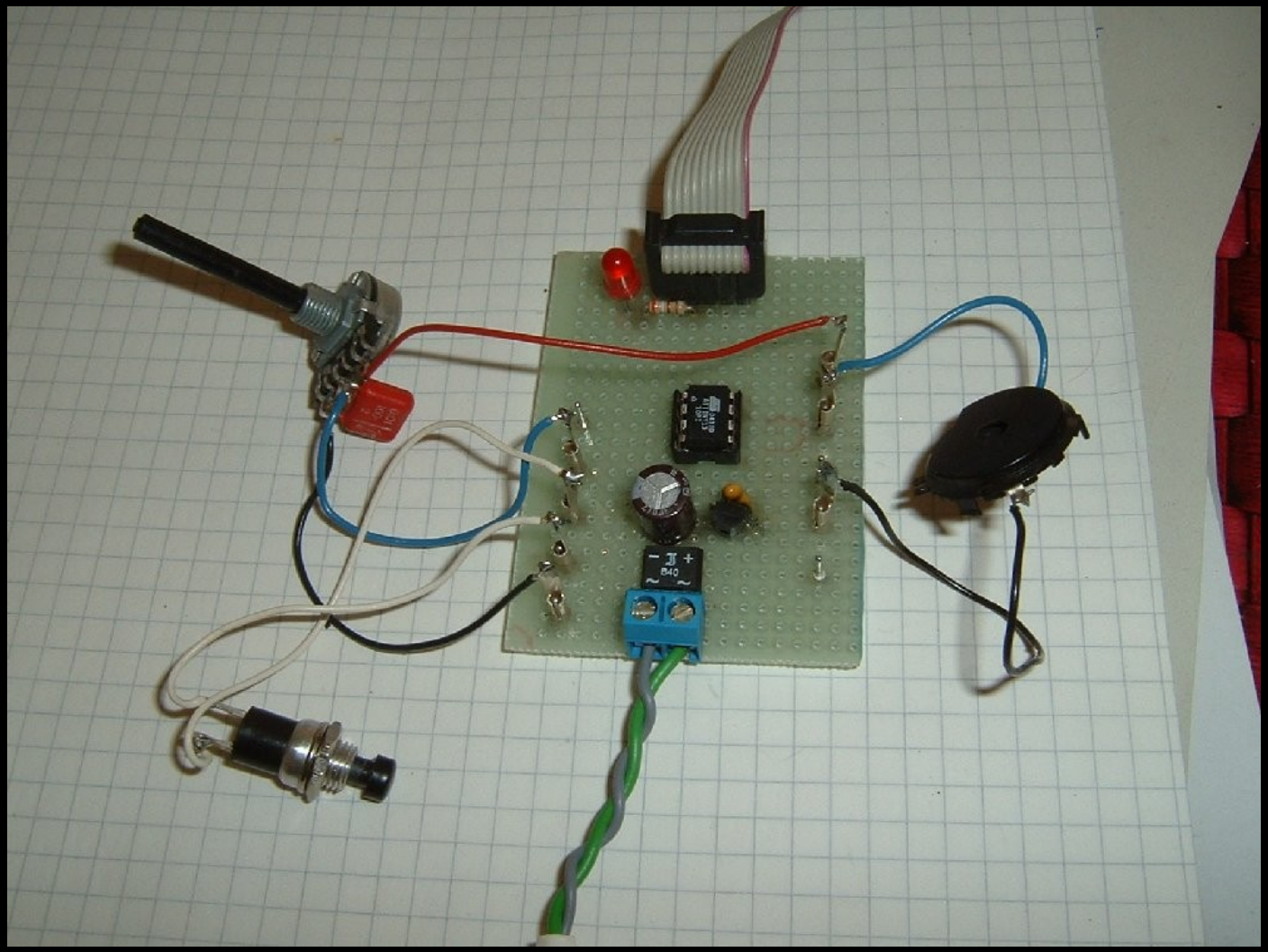
Beispiel 08: Taster macht Töne II

- Ändern der NF-Frequenz durch Ändern einer Zahl im Quellcode

Beispiel 09: Tonhöhen-Einstellung

- In diesem Beispiel wird die Tonhöhe der NF-Frequenz mit einem Potentiometer eingestellt.
- Die Poti-Stellung wird mit einem AD-Wandler von Analog (0..5 Volt) in eine digitale Zahl verwandelt (0..1023) und stellt die Timer-Auflösung durch Teilen durch vier auf Werte zwischen 0 und 255 ein.
- Der AD-Wandler wird einmalig gestartet, nach Ende der Umwandlung unterbricht der AD-Wandler die CPU, schreibt das Ergebnis in ein Register und startet sich automatisch wieder selbst.
- Die angeschlossene Taste unterbricht beim Schließen und Öffnen die CPU, schreibt den aktuellen Tonhöhenwert in den Timer und startet (Taster geschlossen) oder stoppt (Taster offen) die Ausgabe der NF über den Ausgangs-Pin.
- Die CPU wird schlafen gelegt und nur durch die beiden Interrupts aufgeweckt. Der Interrupt wird in der Interrupt-Service-Routine bearbeitet, anschließend wird die CPU wieder schlafen gelegt.
- Beide Interrupts sind voneinander völlig entkoppelt und können zu beliebigen Zeiten auftreten, ohne sich gegenseitig zu stören.

Beispiel 09: Hardware



Der Kondensator am Poti (10 nF) dient zum Abblocken von Wechselspannungs- und HF-Einstreu-ungen. ADC-Wandler sind da sehr empfindlich.

bsp09_adcmorsekey.asm - Editor

Datei Bearbeiten Format ?

```
;***************************************************************
;* Angeschlossener Taster macht Toene, Tonhoehe mit Poti *
;* (C)2005 by info@avr-asm-tutorial.net *
;***************************************************************
;
.INCLUDE "tn13def.inc"
;
; Schaltbild:
;                    ATMEL ATtiny13
;
;                  1/   |_|    |8
;+5 Volt o--|___|----|Res    Vcc|----o + 5 Volt
;                   |            |
; +0..5 Volt o------|ADC3   PB2  |
;                  / 3|            |6   ___
;     0 Volt o--o o--|PB4    PB1  |---|_|--o + 5 Volt
;                   4|            |    LSP
;        0 Volt o----|Gnd    PB0  |
;                   |____________|
;
; Register Definitionen
;
.DEF rTonA = R14 ; Tonhoehe = TC0 Compare Match A
.DEF rTonB = R15 ; halbe Tonhoehe = TC0 Compare Match B
.DEF rmp = R16 ; Multipurpose register
.DEF rimp = R17 ; Interrupt multipurpose register
;
; Reset- und Interrupt Vektoren
;
.CSEG ; Assembliere in den Programm-Flash-Speicher
.ORG $0000 ; beginne mit Adresse 0
;
; Sprungvektoren fuer Reset und Interrupts
;
        rjmp main ; Reset vector
        reti ; Int0 interrupt vector
        rjmp intpcint ; PCINT0 vector
        reti ; TC0 overflow vector
        reti ; Eeprom ready vector
        reti ; Analog comparator int vector
        reti ; TC0 CompA vector
        reti ; TC0 CompB vector
        reti ; WDT vector
        rjmp intadc ; ADC conversion complete vector
;
```

Beispiel 09: Quellcode

Im Schaltbild ist das Poti an Pin2 (ADC-Kanal 3) angeschlossen.

In den Registern rTonA und rTonB legt der AD-Wandler sein Ergebnis ab.

Die Tastatur löst wieder den PCINT0 Interrupt aus.

Der Interrupt-Vektor „ADC conversion complete“ wird angesteuert, wenn der ADC mit einer Wandlung fertig ist.

Beispiel 09: Quellcode II

bsp09_adcmorsekey.asm - Editor

Datei Bearbeiten Format ?

```
;
; PCINT0 Service Routine
;   wird jedes Mal ausgefuehrt, wenn sich der Pegel am
;   Pin 3 (=PB4) aendert
;
intpcint:
        sbic PINB,4 ; Ueberspringe Befehl wenn PB4=Null
        rjmp intpcint1 ; springe weil PB4 = Eins
        out OCR0A,rTonA ; Tonhoehe einstellen
        out OCR0B,rTonB
        ldi rimp,0b01010010 ; CTC, Ausg. A und B bei
        out TCCR0A,rimp ; compare match auf toggle (NF an)
        ldi rimp,0b00000010 ; Vorteiler 1024, Timer start
        out TCCR0B,rimp
        reti ; Kehre vom Interrupt zurueck
intpcint1:
        ldi rimp,0b11110010 ; CTC, Ausgaenge A und B bei
        out TCCR0A,rmp ;  Compare Match auf 1 stellen (NF aus)
        reti ; Kehre vom Interrupt zurueck
;
; AD-Wandler hat einen Wert gewandelt, abholen und speichern
;
intadc:
        in rTonA,ADCL ; Lese ADC-Wert, niedriges Byte
        in rTonB,ADCH ; Lese ADC-Wert, hohes Byte
        lsr rTonB ; durch 2 teilen
        ror rTonA
        lsr rTonB ; durch 4 teilen
        ror rTonA
        mov rTonB,rTonA ; Ergebnis kopieren
        lsr rTonB ; durch zwei teilen
        reti ; fertig und zureck
```

Der Tasten-Interrupt stellt fest, ob die Taste gedrückt ist.

Die Tonhöhe aus dem Register in den Timer übernehmen.

Bei gedrückter Taste NF an OC0A und OC0B einschalten.

Bei losgelassener Taste NF ausschalten.

Der ADC-Interrupt liest das Umwandlungsergebnis.

Teilt den Wert durch vier und schreibt den Wert für CompA in rTonA.

Teilt diesen Wert durch 2 und schreibt ihn in rTonB.

Beispiel 09: Quellcode III

Der Rest des Quellcodes ist ähnlich wie bei anderen Beispielen (Stapel einrichten, Einstellen und starten des Timers, Interrupts ermöglichen, Schlafmodus einstellen Schlaf-Loop.

Nur der Start des AD-Wandlers zu Beginn im Hauptprogramm ist neu. Zuerst wird der Eingangstreiber von PB3 abgeschaltet, beim Betrieb als AD-Kanal wird der nicht gebraucht (Strom und Noise ein sparen).

Dann wird die Referenzspannung (Betriebsspannung) und der zu messende Kanal ADC3 ausgewählt.

Im Kontrollport ADCSRB wird als Triggerquelle der Freilauf gewählt.

Im Kontrollport ADCSRA wird der ADC gestartet, die Triggerung eingeschaltet, der Interrupt ermöglicht und der Teiler für die Wandlung aus 128 eingestellt (langsam).

bsp09_adcmorsekey.asm - Editor

Datei Bearbeiten Format ?

```
;
; AD-Wandler starten
;
        ldi rmp,0b00001000 ; PB3 Treiber aus, spart Strom
        out DIDR0,rmp
        ldi rmp,0b00000011 ; Referenzspannung UB, ADC3 waehlen
        out ADMUX,rmp
        ldi rmp,0b00000000 ; Freilauf waehlen (startet selbst)
        out ADCSRB,rmp
        ldi rmp,0b11101111 ; ADC starten, Autostart,
        out ADCSRA,rmp ;  Int Enable, Teiler auf 128
;
```

Beispiel 10: Morsebake

- In dieser Anwendung wird von einem ATtiny13 auf Knopfdruck ein voreingestellter Morsetext ausgegeben.
- Text, Gebegeschwindigkeit und Tonhöhe werden vor dem Programmieren im Quelltext eingestellt.
- Die Tastenüberwachung und die NF-Ausgabe erfolgt rein Timer- und Interrupt-gesteuert.
- Bei der Timer-Ausgabe von Tönen (NF an) und Pausen (NF aus) wird die Anzahl der Timer-Durchläufe gezählt (Anzahl der NF-Halbwellen). Dadurch wird die Dauer des Tons bzw. der Pause überwacht und nach Ablauf des Zählers die nächstfolgende Aktion eingeleitet.

Beispiel 10: Einstellung der Parameter

bsp10_morsebake.asm - Editor

Datei Bearbeiten Format ?

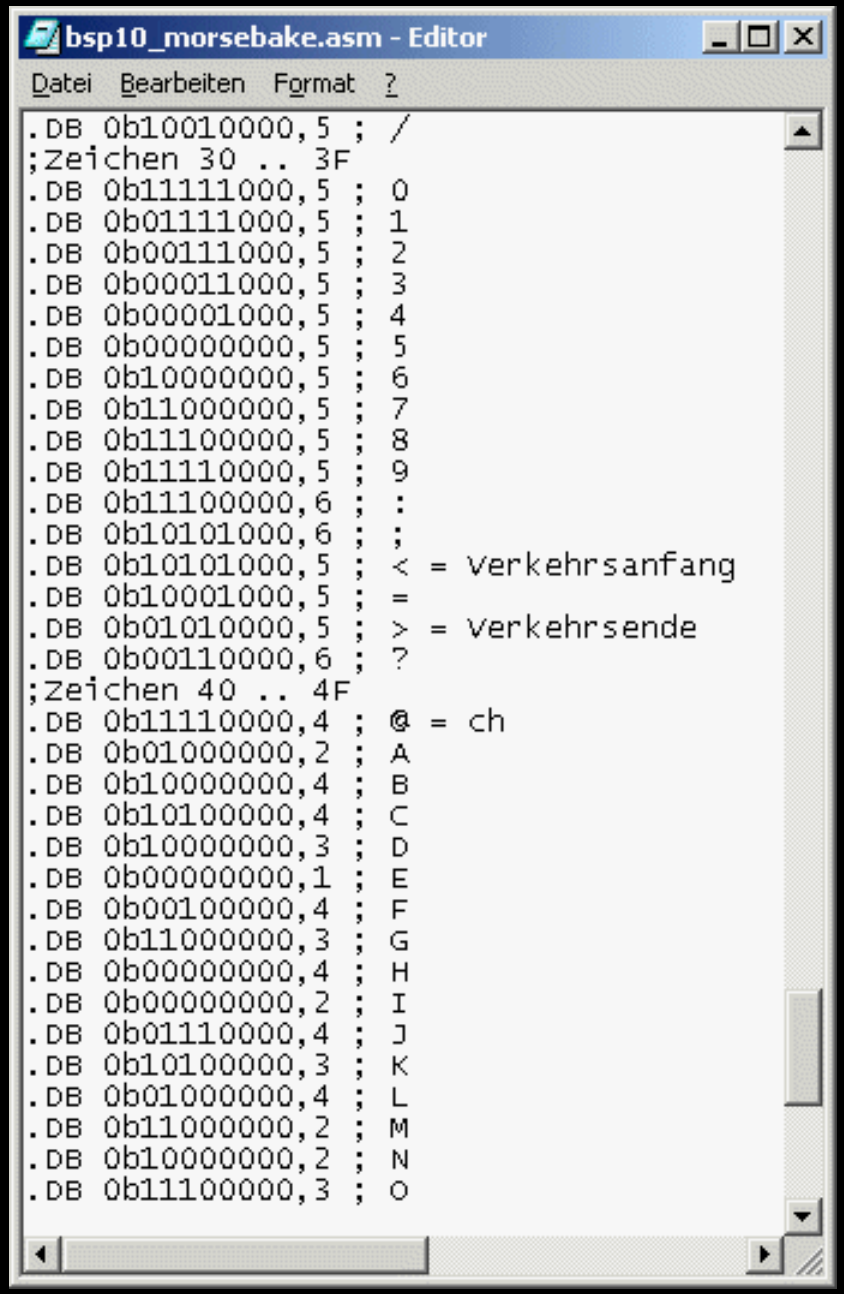
```
; ***************************************************************************
; * Angeschlossener Taster startet Morseausgabe eines Texts                 *
; * (C)2005 by info@avr-asm-tutorial.net                                    *
; ***************************************************************************
;
.INCLUDE "tn13def.inc"
;
; Schaltbild:
;                     ATMEL ATtiny13
;                          ___   ____
;              ___        1/   |_|    |8
;+5 Volt o--|___|-----|Res     Vcc|----o + 5 Volt
;                     |            |  ___
;                     |PB3     PB2|--|___|--|<|--o + 5 Volt
;                  /  3|            |6    _       LED
;      0 Volt o--o o--|PB4     PB1|---|_|--o + 5 Volt
;                     4|            |    /_\
;         0 Volt o----|Gnd     PB0|    ---
;                     |____________|    LSP
;
; Konstanten
;
.EQU cSpeed = 60 ; Gebegeschwindigkeit in bpm, 5..500 bpm
.EQU cTone = 1000 ; Tonhoehe in Hz, 300..8000 Hz
;
; Berechnete Konstanten
;
.EQU fClock = 1200000
.EQU cTc0Div = 8
.EQU cTc0CmpA = fClock/cTc0Div/cTone/2
.EQU cTc0CmpB = cTc0CmpA/2
.EQU cTc0N = cTone*16/cSpeed
.EQU cTc0N2 = 2*cTc0N ; zweifach lange Zeit
.EQU cTc0N3 = 3*cTc0N ; dreifach lange Zeit
;
; Ueberpruefen der eingestellten Werte
;
.IF cTc0CmpA > 255
        .ERROR "Ton zu niedrig!"
        .ENDIF
.IF cTc0CmpA < 8
        .ERROR "Ton zu hoch!"
        .ENDIF
.IF cTc0N3 > 65535
        .ERROR "Frequenz zu niedrig!"
        .ENDIF
.IF cTc0N < 8
        .ERROR "Frequenz zu hoch!"
        .ENDIF
```

Schaltung: Die LED diesmal an PB2, der Lautsprecher an PB1 und die Taste wieder an PB4.

Die Gebegeschwindigkeit und die Tonhöhe werden als Konstanten vorgegeben.

Daraus abgeleitete Konstanten errechnet, die bei der Ablaufsteuerung eingesetzt werden.

Die sich ergebenden Werte werden überprüft (Überläufe, Unterläufe), damit beim Assemblieren schon erkannt wird, ob etwas aus dem Ruder läuft.



Beispiel 10: Auszug Morsetabelle

Die Morsetabelle ist im Programmspeicher in Binärform abgelegt. Hier die ASCII-Zeichen von 30 bis 4F hex.

Jedes Morsezeichen benötigt zwei Byte: im ersten ist die Abfolge von kurzen und langen Tönen kodiert, (bei einer „0“: 0b11111000 = 5 mal lang), im zweiten die Anzahl kurzer und langer Töne zusammen (bei einer „0“: 5).

Verkehrszeichen sind auf ASCII-Zeichen gelegt, für die es kein Morse-Äquivalent gibt (z.B. Verkehrsanfang ist ASCII-Zeichen <).

Beispiel 10: Textablage

Der auszugebende Text ist auch im Programmspeicher abgelegt. Er wird aus dem Programmspeicher ab dem Label MorseText: ausgelesen, in Morsezeichen übersetzt und ausgegeben. Wenn das Zeichen 0 erreicht wird, ist der Text zu Ende (Null-terminierter String) und die Ausgabe wird beendet.

bsp10_morsebake.asm - Editor

Datei Bearbeiten Format ?

```
;
; Text fuer Morseausgabe
;
MorseText:
.DB "<HALLO! HIER IST EIN ATMEL TINY13 BEI DER ARBEIT!>",0,0
; .DB "paris paris paris paris paris paris paris paris paris paris ",0,0
;
; Ende Quellcode
;
```

Aufnahme abspielen: hier klicken =>