Attiny13-Elbug Version 3

Ralf Beesner, DK5BU

18.8.2010

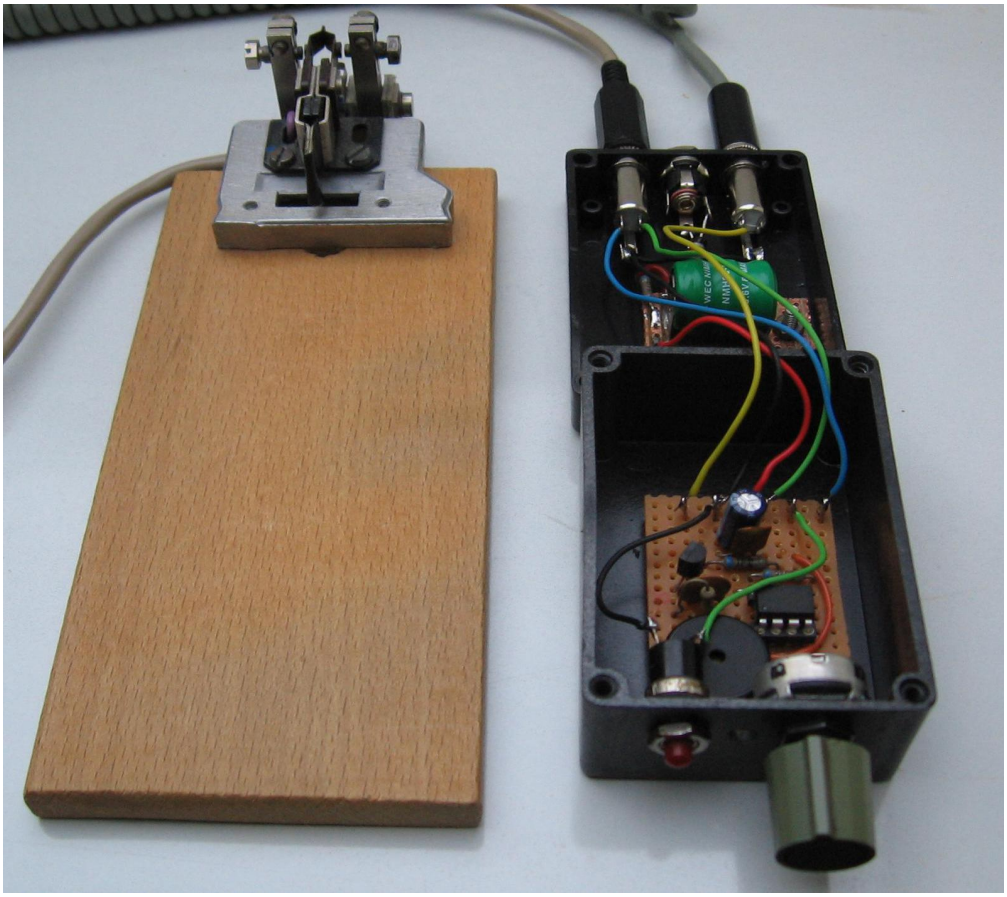

Abbildung 1: Gebemechanik und Elbug

1 Überblick

Mit einem kleinen 8-poligen Attiny13, der ca. 1 Euro kostet, wenigen zusätzlichen Bauteilen und etwas Basic (Bascom- AVR; es reicht die kostenlose Demo-Version) läßt sich recht einfach eine elektronische Morsetaste realisieren.

Sie erzeugt kurze und lange Morsezeichen automatisch (Punkte und Striche; ein Strich ist drei mal so lang wie ein Punkt). Die Zeichen werden mit einem Bedienhebel erzeugt, tippt man ihn rechts an, wird ein langes Zeichen plus normgerechter Pause erzeugt; tippt man links, ein kurzes Zeichen plus Pause. Es gibt auch „Squeeze"- Tasten, sie haben zwei Hebel; drückt man beide, wird eine alternierende Folge von kurzen und langen Zeichen erzeugt; so kann man komplexe Zeichen mit wenigen Tastbewegungen „zusammensetzen".

Auf dem Bild sieht man eine leichtgewichtige Portabel- Gebemechanik (ein Kontaktsatz aus einem Spezialrelais) und das aufgeklappte Gehäuse (in der oberen Halbschale die Elektronik, in der unteren ein 80mAh- Akku mit 3 Zellen und die Steckverbinder).

Auf Kurzwelle muß man manchmal lange herumrufen, um einen Funkpartner zu finden. Daher ist ein Speicher mit einem (fest programmierten) Ruftext hilfreich; er wird mit dem kleinen Taster neben dem Drehknopf gestartet und durch Antippen des Punkthebels gestoppt.

Bei den Versionen 1 und 2 des Attiny- Elbugs, die auf www.elo-web.de unter der Rubrik „Microcontroller" - „Bascom" zu finden sind, hatte ich das Aufwecken des Microcontrollers aus dem Tiefschlafmodus auf eher ungeschickte Weise gelöst. Beim Atmega8, mit dem ich die ersten Programmierversuche unternommen hatte, geht das nur über die Interrupt- Pins INT0 und INT1, neuere Prozessoren (auch der Attiny13) lassen sich eleganter mit Pin Change Interrupts (die für fast alle I-O-Pins verfügbar sind), aufwecken.

Version 3 nutzt daher Pin Change Interrupts; dadurch wird einer der knappen I-O-Pins frei, so daß nun Mithörton und Sender- Schalttransistor auf separate Pins gelegt werden konnten.

2 Erläuterungen zur Schaltbild:

Der Elbug soll ständig an der Batterie verbleiben und möglichst wenig Strom verbrauchen, wenn er nicht genutzt wird. Eine so geringe Energieaufnahme erfordert den Power-Down- Modus, in dem jedoch der Taktoszillator abgeschaltet wird und der Mikrocontroller nur durch den externen Interrupt INT0 und die Pin Change Interrupts geweckt werden kann.

Punkt- und Strich- Eingang („Dit" und „Dah") liegen an PB3 und PB2. Die Eingänge werden über interne Pullup- Widerstände des Microcontrollers auf Plus gezogen und

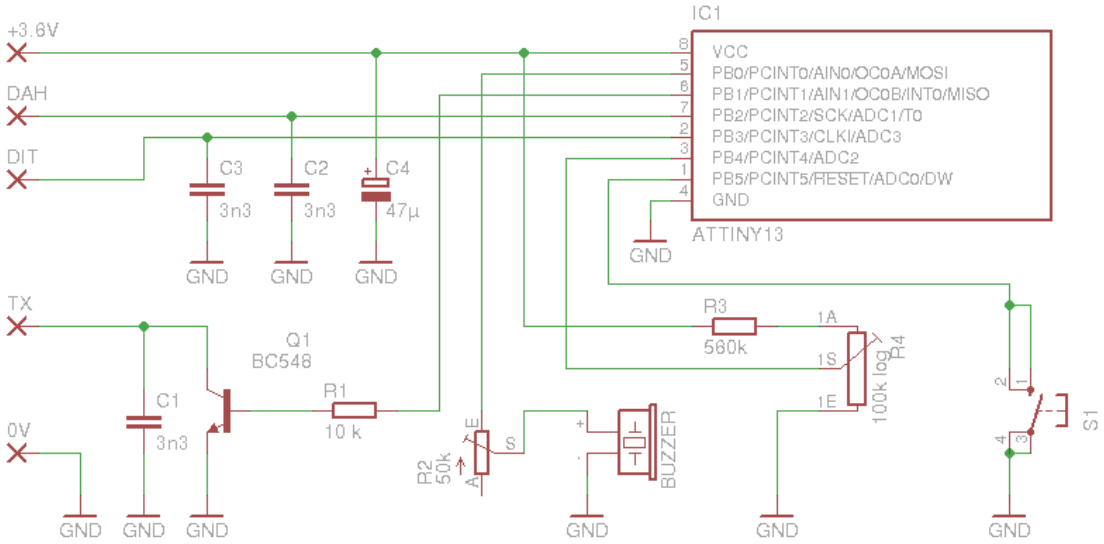

Abbildung 2: Schaltplan

sind gegen HF- Einstrahlung mit 3,3 nF abgeblockt. Punkt- und Strichtaster schalten gegen Masse und ziehen den jeweiligen Eingang auf „Low“, wenn sie gedrückt werden.

Die Morsegeschwindigkeit wird durch den AD- Wandler vorgegeben. Ein Potentiometer an PB4 bestimmt die Zeitkonstante. Man kann den Einstellbereich per Software oder per Hardware festlegen. Wegen des Stromverbrauchs ist ein Poti mit 470 kOhm oder 1 MOhm (log.) günstig; da ich nur ein Potentiometer mit 100 kOhm (log.) hatte, habe ich beides gemacht: in Reihe mit dem Poti habe ich einen Festwiderstand mit 560nkOhm gelegt und zum ausgelesenen Wert jeweils 32 addiert. Der AD- Wandler löst 10 bit auf (0...1023).

Verwendet man ein 1 MOhm- Poti, muß man sich den Einstellbereich per Software „zurechtbiegen“

PB0 ist als Ton- Ausgang konfiguriert. Bei richtiger Wahl der Timer- Parameter ergibt sich ein hörbarer Ton, den man durch Umprogrammierung des Datenrichtungs-Registerbits DDRB.0 ein- und ausschalten kann. Die NF treibt einen Piezo- Summer (Buzzer); die Lautstärke ist mit dem Trimmer R2 einstellbar.

Das Schaltsignal für den Sender liegt an PB1, ein BC548 fungiert als (invertierender) Schalttransistor. An „TX“ liegt also das Schaltsignal für den Telegrafieeingang eines Senders.

Da für den Abruf des Speichertextes kein Pin mehr frei ist, muß der Reset-Pin herhalten; durch Drücken von S1 wird ein Reset ausgelöst.

Als Energieversorgung ist eine Lithium- Primärzelle im Mignon- Format vorgesehen (erhältlich z.B. bei Reichelt; ca. 4 EURO). Sie liefert 3,6 Volt und sollte den Ruhestrom

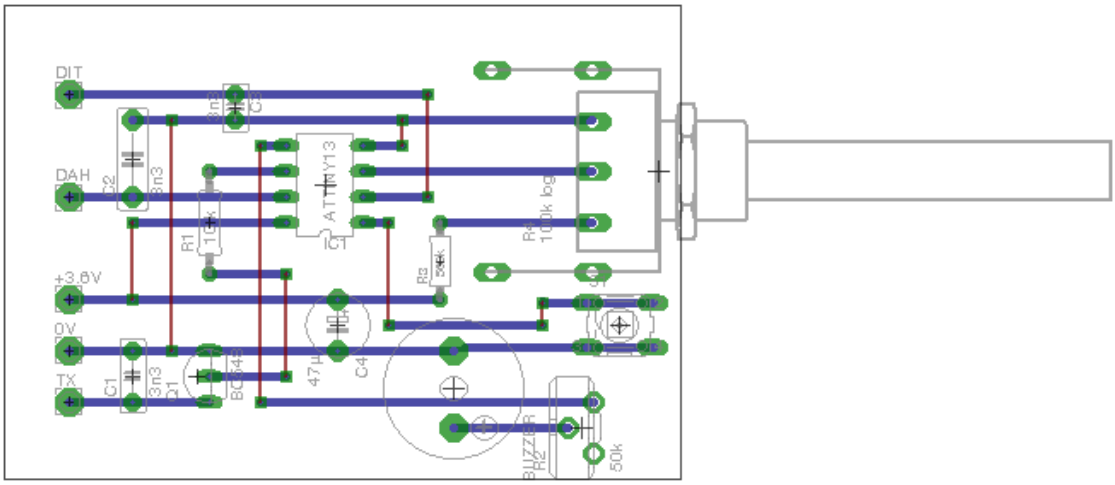

Abbildung 3: schematische Anordnung auf Streifenleitungsplatine

für die Schaltung (ca. 25 μA) jahrelang liefern können. Man kann auch 3 Mignonzellen oder einen kleinen 3- Zellen- NiMH-Akku 3,6V / 80 mAh verwenden (bei Reichelt unter 3 EURO). Sicherung nicht vergessen!

Da die Schaltung nur aus wenigen Bauteilen besteht, kann man sie leicht auf einem Stück Streifenleitungsplatine aufbauen, sie ist so klein, daß man sie im Gehäuse per Einlochbefestigung des Potis fixieren kann. Lediglich die Batterie ist separat unterzubringen.

3 Erläuterungen zur Software

Die Software ist in Basic geschrieben (Bascom-AVR - www.mcselec.com).

Zunächst werden die Variablen definiert und die Hardware- Register des Attiny konfiguriert.

Da der Reset- Pin als Speicher-Abruf- Taste fungiert, wird nach einem Reset als erstes der Speichertext ausgegeben. Unterbrechen kann man ihn durch Antippen der Punkt-Taste, dann verzweigt das Programm in den Keyer- Betrieb (Unterprogramm Keyer). Der Speichertext „steckt“ in den Data- Zeilen am Ende des Programms. Die Codierung der Zeichen ist im Kommentarbereich des Quelltextes genauer beschrieben.

Ist keine Taste gedrückt, durchläuft das Programm einmal die Do- Schleife bis zum Befehl „Powerdown“. Er stoppt fast die gesamte Hardware. Der Controller reagiert nur noch auf Potentialwechsel an den beiden zuvor definierten PCINT- Eingängen PB3 und PB2, an die der Punkt- und der Strich- Taster angeschlossen sind.

Drückt man einen der beiden, wacht der Controller auf, springt den auf „Powerdown“ folgenden Befehl an (“Loop“) und beginnt die Do- Schleife aufs neue. Da jedoch PB2 und/oder PB3 auf „Low“ gezogen wurden, verzweigt das Programm in den IF- EndIf-Schleifen; es laufen dann die Wait-Befehle und Umschaltbefehle für das Datenrichtungsregister ab, die den Ton durchschalten oder blocken und den TX-Pin umschalten.

Die If- EndIf- Schleife für die Morsestriche ist umfangreicher, weil hier auch abgefragt wird, ob zusätzlich auch der Morsepunkt- Hebel gedrückt wurde; dann wird eine Folge von abwechselnden Strich- Punkt- Signalen erzeugt (Sqeeze- Modus) bzw. ein rudimentärer Punktspeicher bei Einhebel- Tasten realisiert.

Im Quelltext sind weitere Kommentare zu den einzelnen Befehlen angefügt.

4 Flashen des Attiny13

Zunächst muß der Speichertext auf die eigenen Verhältnisse angepaßt und mit Bascom-AVR von MCS Electronis (www.mcselec.com) kompiliert werden. Das entstandene HEX- File kann mit einem beliebigen Programmer in den Chip übertragen werden.

Da die Morsetaste aus dem Franzis- “Lernpaket Microcontroller“ enstanden ist, bietet sich jedoch die Hardware und Software des Lernpakets an.

Das Lernpaket enthält ein sehr einfach gehaltenes Programm (LPMicroISP.exe), mit dem man Hexfiles über die serielle Schnittstelle eines PC in den Attiny13 flashen und die Taktrate des Attiny13 ändern kann. Es funktioniert auch mit USB- Seriell-Wandlern; die Übertragung ist dann allerdings sehr langsam.

LPMicroISP.exe ist ein reiner Programmer, der den Code im robusten ISP- Modus in den Microcontroller überträgt.

Auf der Website des Autors gibt es ein Update des Programms unter „www.b-kainka.de/LPmicrosUpdate.zip“. Es enthält zwei exe- Files; das zweite Programm LPMicros.exe ist auf die Experimente des Lernpakets zugeschnitten und für das reine Programmieren eines Attiny13 weniger geeignet.

Man kann zum Programmieren die Original- Hardware verwenden oder eine weiter vereinfachte Schaltung mit nur 8 Bauteilen, die sich ebenfalls problemlos auf einem Stück Streifenleitungsplatine aufbauen läßt.

Ab Werk werden die Attiny13 mit einem Takt von 9,6 MHz und Vorteilerfaktor 8 ausgeliefert, also 1,2 MHz. Die Fusebits des Attiny13 brauchen daher nicht umprogrammiert zu werden.

Möchte man die Taktrate ändern (z.B. um den Stromverbrauch noch weiter zu

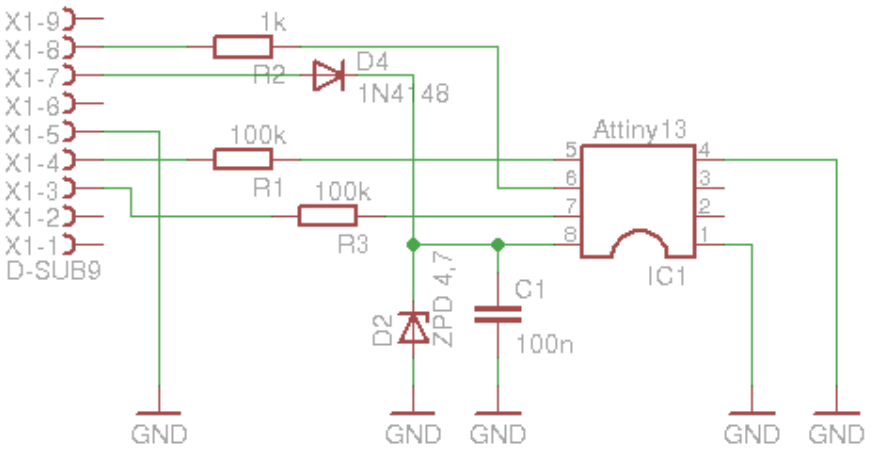

Abbildung 4: einfacher Programmieradapter

drosseln), muß man die Variable `$crystal` und die Timerkonstanten (Tonhöhe) anpassen.