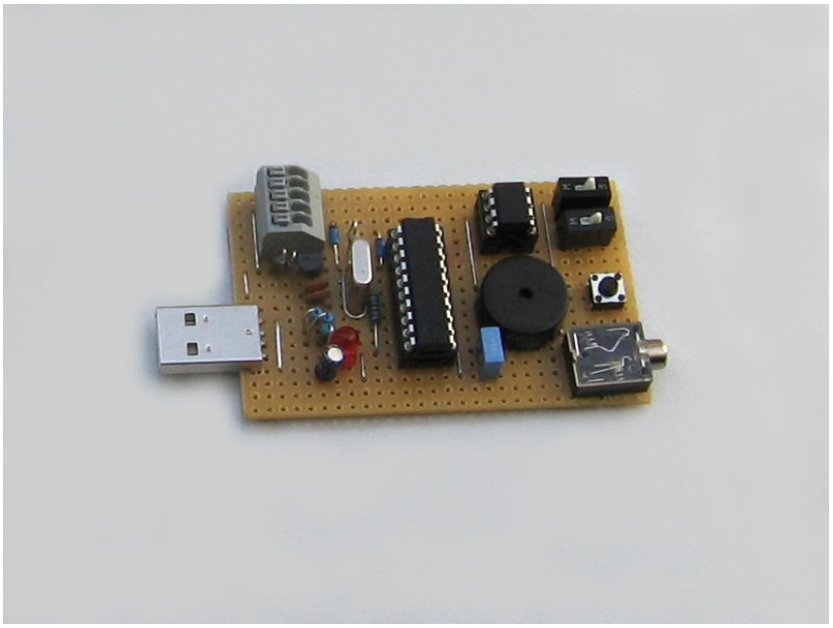
Ein inverses AVR- Morsekeyboard

Ralf Beesner

7. April 2012



1 Überblick

Vor ca. 20 Jahren kamen einige Morsekeyboards auf den Markt, mit denen man mittels einer "Schreibmaschinentastatur" Morsezeichen erzeugen konnte (ich konnte mich aber nie an die Dinger gewöhnen).

Unter Verwendung des "serial keyboards"
`http://hobby-electrons.sourceforge.net/projects/terminal-keyboard/`
habe ich mir nun die umgekehrte Lösung gebaut: eine Schaltung, die Morsezeichen in Tastaturanschläge umwandelt, so dass man per Morsetaste in Anwendungen, die auf dem PC laufen, hineinschreiben kann.

So seine Texte einzugeben ist zwar mühsam und weitgehend sinnfrei, aber ich finde, dass es dennoch eine coole Spielerei ist ;-)

Da ich mit der Programmiersprache "C" auf Kriegsfuß stehe, wandelt ein in Bascom programmierter AtTiny 13 die Morsezeichen zunächst in ein serielles Signal; dieses wird dann durch das mit einem AtTiny 2313 realisierte virtuelle HID- Keyboard in ein USB- Keyboard- Signal aufbereitet.

2 Hardware

Die Aufteilung auf zwei Mikrocontroller ist etwas unelegant, erfordert aber ausser dem zweiten Controller keine Bauteile, die nicht ohnehin erforderlich wären. Der AtTiny 13 übergibt das serielle Signal ohne Interface- Schaltung direkt an den seriellen Eingang des AtTiny 2313. Die übrigen Pins des Attiny 13 sind mit der Morsetaste, einem Buzzer für den Mithörton und zwei Schalterchen für die Wahl des (Morse-) Geschwindigkeitsbereiches und des Modus für die Leerzeichen beschaltet.

Die gesamte Schaltung wird mit ca. 3,3V betrieben, die mit einer in Durchlassrichtung betriebenen roten LED aus dem USB gewonnen werden.

Leider erfordert das "serial keyboard" einen Hardware- UART, deshalb muss man sich mit dem 20poligen AtTiny 2313 herumschlagen, obwohl die meisten Pins nicht benötigt werden.

Das Foto des Versuchsaufbaus weicht etwas von Schaltplan ab; an der grauen WAGO- Klemme liegen u.a. noch zwei Eingänge für ein RS232- Signal bzw. ein TTL- Signal, so dass sich auch andere serielle Signale mit dem "serial keyboard" umwandeln lassen.

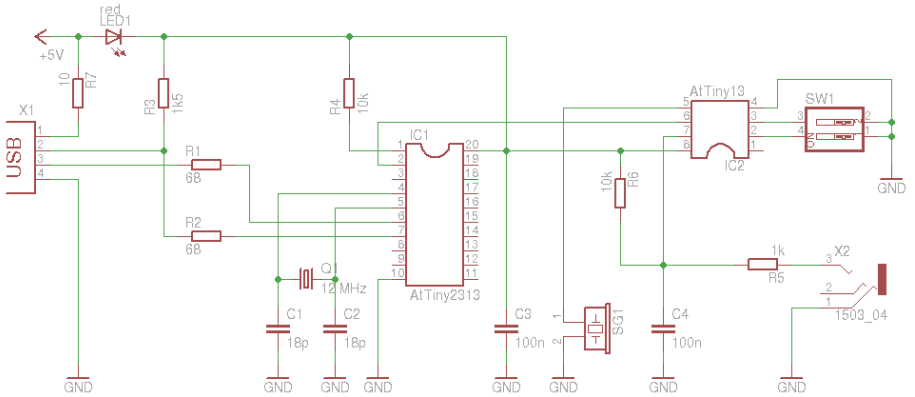


Abbildung 1: Schaltbild

3 Bascom- Software

Die Attiny 13- Software beruht auf morse.bas von Burkhard Kainka, das auf ein Atmel Butterfly mit dem AtMega169 zugeschnitten war und das ich vor einiger

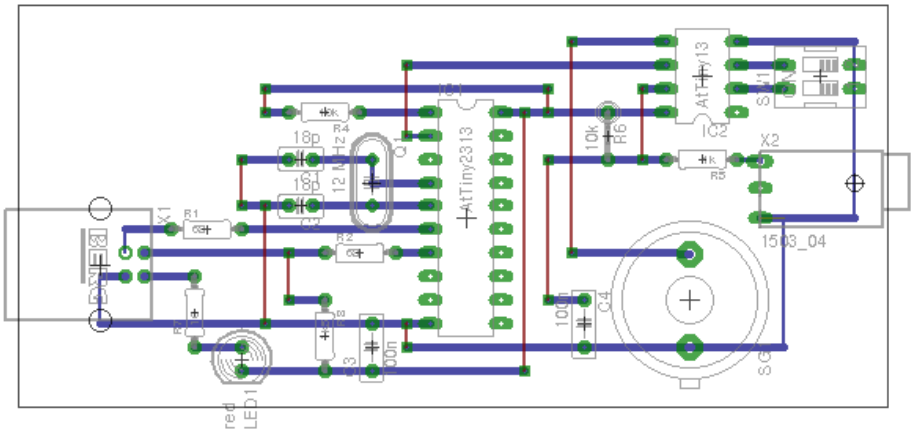


Abbildung 2: Streifenraster- Layout

Zeit auf den AtMega 8 portiert hatte -siehe `http://www.elektroniklabor.de/AVR/Morsedecoder.html` .

Neben der Umstellung von der LCD- Ausgabe auf die serielle Ausgabe war es erforderlich, den Zugriff auf die Morsezeichentabelle RAM- sparender zu lösen, da der AtMega 8 immerhin 1024 Byte SRAM hat, der AtTiny 13 jedoch nur 64.

Eine weitere Änderung betrifft die Leerzeichenerzeugung; es gibt keinen speziellen Morsecode dafür, ein Leerzeichen wird lediglich durch eine kleine Gebepause übermittelt. Zunächst hatte ich Leerzeichen nur aufgrund dieser Gebepausen definiert, aber in der Praxis führt das dann dazu, dass häufig Leerzeichen an den falschen Stellen eingefügt werden - ein PC ist ja nicht so semantisch begabt, dass er das wie ein menschlicher Operator "zurechtinterpretieren" kann.

Die Pause- Leerzeichenerkennung ist daher abschaltbar, und als Code für ein Leerzeichen habe ich das Verkehrszeichen <as> definiert. Da es auch kein Morsezeichen für <carriage return> gibt, musste dafür das Verkehrszeichen <kn> herhalten (Verkehrzeichen bestehen meist aus zwei zusammengezogenen Morsezeichen und kennzeichnen Spruchbeginn, Spruchende und dergleichen).

Das Verkehrszeichen "Irrung" (eigentlich 8 Morsepunkte, hier nur 6 oder 7 Morsepunkte, denn die Zeichenlänge ist auf maximal 7 Morsepunkte begrenzt) erzeugt ein <backspace>; der Cursor wird um einen Schritt zurückgesetzt und das darunterliegende Zeichen gelöscht. So kann man Falscheingaben korrigieren.

Es findet keine gleitende Anpassung an die Morsegeschwindigkeit statt; mit einem weiteren Schalter kann man lediglich zwischen 60 und 90 Zeichen pro Minute wählen. Der "Fangbereich" der Software reicht aber von 40 bis 90 Zeichen pro Minute bzw. 60 bis 120 Zeichen pro Minute, so dass die zwei Bereiche die üblicherweise mit "straight keys" erzielbaren Gebegeschwindigkeiten abdecken.

4 C-Software

Die "eingedeutschte Version" des "serial keyboards" hatte ich bereits in einem anderen Beitrag beschrieben; das Softwarepaket ist hier beigefügt.

5 Ausblick

Mit besseren C- Kenntnissen ließe sich die Bascom- Software nach C portieren und mit dem VUSB- Treiber zusammenfassen. Da dann die Notwendigkeit eines Hardware- UART entfiele, dürfte ein AtTiny 45 reichen, so dass die Hardware (selbst bei Verwendung eines AtTiny im DIP- Gehäuse) auf die Größe eines USB- Sticks schrumpft - damit wäre sie dann noch etwas cooler...