Morsedecoder mit AVR

Volker G. Aurich, DK3PK

Mit dem Mikrokontroller ATmega16 von Atmel lässt sich ein billiger, aber leistungsfähiger Decoder für Morsezeichen verwirklichen. Dank des eingebauten Vorverstärkers werden nur sehr wenig zusätzliche Bauteile benötigt: eine LC-Anzeige, eine Mikrofonkapsel, ein paar Widerstände und Kondensatoren, zwei Taster und eventuell ein Quarz. Trotzdem ist auch bei schlechtem SNR noch ein Dekodierung möglich.

Als LC-Anzeige wird eine 2- oder 4-zeilige, 16-stellige Anzeige mit einem zu KS0070B oder HD44780- kompatiblen Kontroller verwendet. Weil sich 8 Zeichen frei programmieren lassen, kann man leicht kleine, vertikale Balken erzeugen, um ein Spektrum darzustellen. Solche Anzeigen gibt es recht billig zu kaufen.

Als Mikrofon wird eine einfache, kleine Elektretkapsel benutzt. Ihr Signal kann man direkt in den ATmega16 einspeisen, weil er einen eingebauten Vorverstärker besitzt. Allerdings muss man dafür einen Differenzeingang des ADC verwenden. Daher die etwas seltsame Beschaltung.

Als Taktgenerator kann der im ATmega16 eingebaute RC-Oszillator mit 8 MHz dienen. Bei schlechtem SNR wird die Decodierung jedoch ein wenig besser, wenn man den eingebauten Oszillator mit einem höherfrequenten Quarz zu betreibt. Es wurde ein Quarz mit 14,7456 MHz gewählt. Dementsprechend werden zwei Arten von Dateien zum Herunterladen bereitgestellt; davon noch je eine für 4-zeilige oder 2-zeilige Anzeigen. Und bitte nicht vergessen, dass auch die Fuses im ATmega16 entsprechend geändert werden müssen!

Der Stromverbrauch beträgt (ohne LCD-Beleuchtung) nur 20-30 mA. Daher werden als Stromversorgung 4 Batterien mit 1,5 Volt genommen. Eine Siliziumdiode in der Plusleitung erniedrigt die 6 Volt auf eine für den ATmega16 noch zulässige Spannung.

Modus 1: Anzeige des Spektrums von 300 Hz bis 1800 Hz in 100Hz-Schritten

Umschalten in Modus 2 durch Drücken auf Taster 1

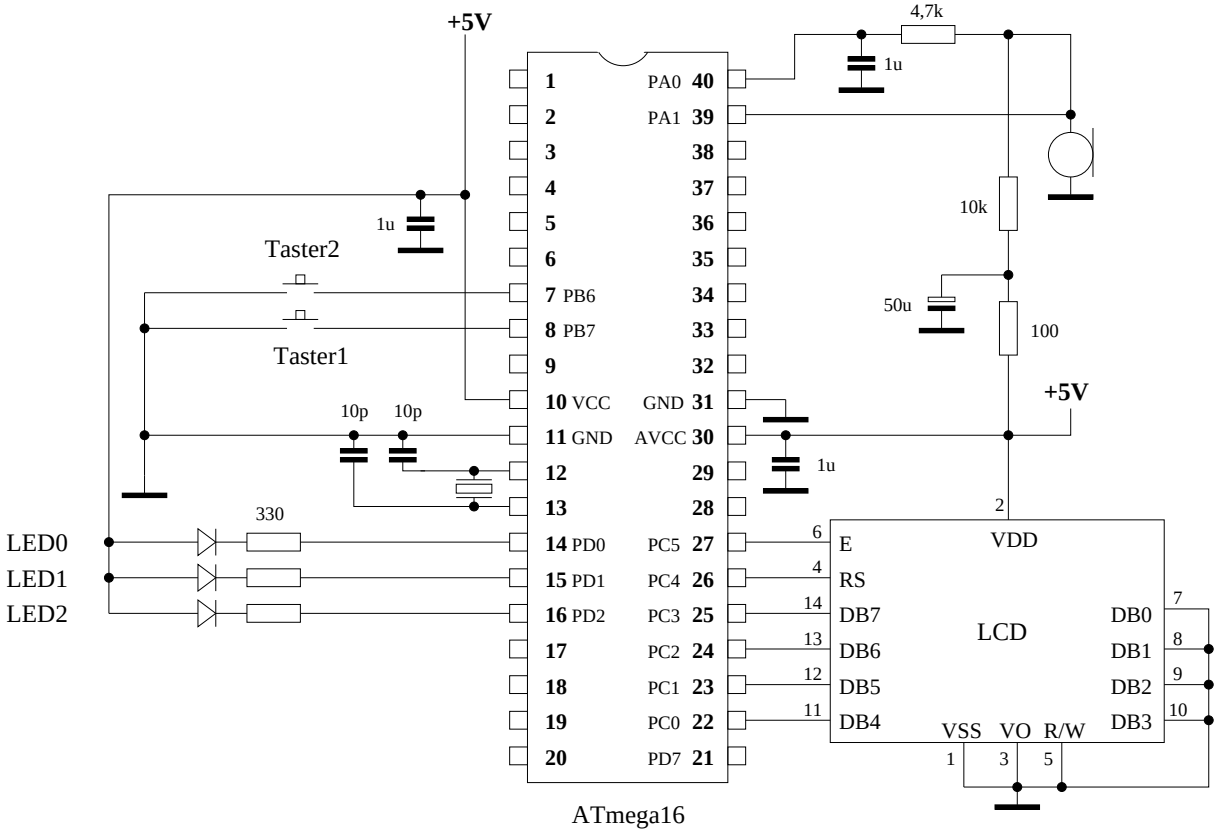
Modus 2: Dekodierung des 1000Hz-Signals

Umschalten in Modus 1 durch längeres Drücken auf Taster 1.

Die Tonhöhe am Empfänger so einstellen, dass die LED1 im Takt der Morsezeichen blinkt. LED0 zeigt die Amplitude des 900Hz-Anteils, LED2 die des 1100Hz-Anteils an. Eine automatische Anpassung an die Geschwindigkeit ist eingebaut. Es dauert ein paar Punkte und Striche, bis sie einrastet. Sie funktioniert meist recht gut. Man kann sie aber auch abschalten und manuell eingreifen:

- Langes Drücken auf Taster 2 schaltet die automatische Synchronisation ein bzw. aus.
- Kurzes Drücken auf Taster 2 setzt die Geschwindikeit herab, kurzes Drücken auf Taster 1 erhöht sie.

Das Mikrofon wird einfach in die Nähe des Lautsprechers des Empfängers gelegt; durch Verändern des Abstandes reguliert man die Lautstärke so, dass LED1 schön im Rhythmus der Zeichen blinkt. LED0 und LED1 sollen gar nicht oder weniger und symmetrisch aufleuchten. Eine Übersteuerung des ADC kommt dabei kaum vor, kann aber auf Wunsche durch eine LED an PD7 angezeigt werden.



Die Anschlüsse sind so gewählt, dass man die Schaltung auf dem STK500-Board erproben kann. Allerdings muss man dann den Jumper AREF entfernen, den Jumper OSCSEL neben die Zeichen OSC setzen und einen 14,7456MHz-Quarz in die Fassung des STK500 stecken.

Download

Auf STK500 mit 14.7456MHz-Quarz:

`morsetinyXX_14MHz_4rows.srec` oder `morsetinyXX_14MHz_2rows.srec`

Fuses: Highbyte D9, Lowbyte E0
Kommando für avrdude mit STK500 an serieller Schnittstelle:

```
avrdude -P /dev/ttyUSB0 -c stk500v2 -p m16 -U hfuse:w:0xD9:m -U lfuse:w:0xe0:m
```

Standalone mit 14.7456MHz-Quarz:

`morsetinyXX_14MHz_4rows.srec` oder `morsetinyXX_14MHz_2rows.srec`

Fuses: Highbyte C9, Lowbyte EF
Kommando für avrdude mit STK500 an serieller Schnittstelle:

```
avrdude -P /dev/ttyS0 -c stk500v2 -p m16 -U hfuse:w:0xC9:m -U lfuse:w:0xef:m
```

Mit internem 8MHz-RC-Oszillator:

`morsetinyXX_8MHz_4rows.srec` oder `morsetinyXX_8MHz_2rows.srec`

Fuses: Highbyte D9, Lowbyte E4
Kommando für avrdude mit STK500 an serieller Schnittstelle:

```
avrdude -P /dev/ttyS0 -c stk500v2 -p m16 -U hfuse:w:0xd9:m -U lfuse:w:0xe4:m
```