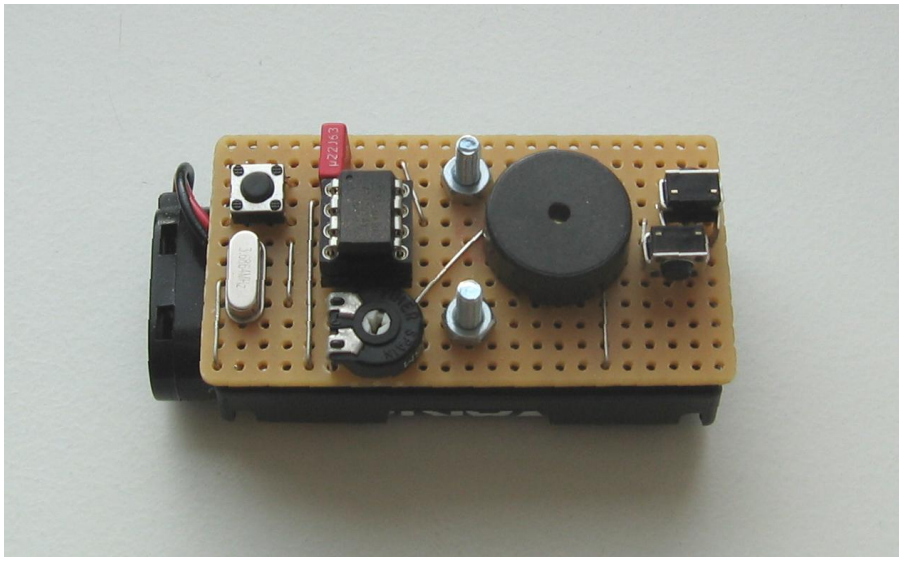
Morseuhr 3

Ralf Beesner

10.1.2011



1 Einleitung

Als mir die Idee einer Morseuhr in den Sinn kam, dachte ich eigentlich an eine „stilechte“ Lösung, die nicht nur Morse ausgibt, sondern die auch vollständig mit einer Morsetaste gesteuert wird. Das war mir zunächst zu aufwendig, und so entstanden die Morseuhren 1 und 2, die über DIP- Schalter vorkonfiguriert werden und die gewünschten Einstellungen nach einem Reset aus den Dipschalter-Stellungen übernehmen.

Die ursprüngliche Idee hat sich aber gehalten, und mit meinem Morseterminal hatte ich bereits die Subroutinen für die Morse- Eingabe beisammen. So war es nicht mehr

weit bis zur vollständigen Lösung der Ursprungsidee. Da noch Platz im Flash war, ist noch eine „fat version“ mit Weckalarm dazugekommen.

Außerdem zeigte sich, daß man nicht zwingend einen 32 kHz- Uhrenquarz und einen ATMega- Mikrocontroller mit speziellem Low- Power- Uhrenmodus einsetzen muß; auch mit einem Standardquarz 3,6864 MHz und einem ATtiny 45 läßt sich der Grund- Stromverbrauch im Idle- Modus auf Werte drücken, die für einen Batteriebetrieb mit Mignonzellen akzeptabel sind (etwa 0,2 mA, also 1,8 Ah / Jahr bei 3 V).

2 Hardware

Ein Quarz muß beim Attiny 45 an die Eingänge PB3 und PB4 angeschlossen werden. Der Buzzer liegt an PB0 ; für den Strich- und den Punktkontakt bleiben daher nur PB1 und PB2 . Außer dem Mikrocontroller, dem Quarz, dem Buzzer und den beiden Tastern sind nur noch ein Abblock- Kondensator für die Betriebsspannung und ein Lautstärketrimmer vorhanden. Der Reset wird hoffentlich selten benötigt; er ist auf dem Bild des Prototypen noch mit einem Mikrotaster beschaltet, im Schaltplan und Platinen- Vorschlag aber nur noch als Kontaktfläche ausgeführt, die bei Bedarf mit einem leitenden Gegenstand gebrückt wird.

Die Betriebspannung beträgt 3 Volt, sie wird aus 2 Mignonzellen gewonnen. Die Platine ist so bemessen, daß sie mit 2 Schrauben auf der Rückseite eines Zweier-Batteriehalters befestigt werden kann.

Der Quarz wird ohne die im Datenblatt empfohlenen Bürdekapazitäten (12 -22 pF) betrieben. Der Oszillator schwingt trotzdem sicher, die Frequenz ist lediglich ein paar hundert Hz zu hoch. Das ist aber durchaus erwünscht, denn damit läuft die Uhr stets etwas zu schnell; es ist einfacher, sie per Software durch Einlegen von ein paar μsec Wartezeit pro Sekunde oder durch einige Sekunden Wartezeit um Mitternacht etwas langsamer zu machen.

3 Bedienung

Die Uhr wird vollständig über Morse- Eingaben gesteuert. Nach Einlegen der Batterien gibt sie zunächst die Uhrzeit 0 Uhr aus. Das Viertelstunden- Schlagwerk (im folgenden als „Gong“ bezeichnet) ist eingeschaltet.

Folgende Befehle stehen zur Verfügung; sie bestehen aus einem Zeichen:

- ? : Auflistung der Befehle; Befehl ist jeweils der Anfangsbuchstabe des Schlüsselworts

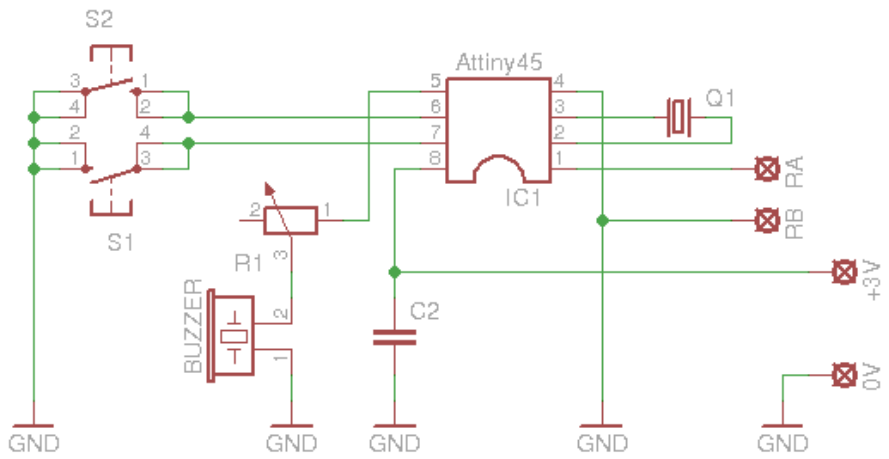


Abbildung 1: Schaltplan

- Z : Zeit setzen
- T : Zeit abfragen
- G : Gong (Schlagwerk) ein/aus
- C : Check; Ausgabe des Gongstatus
- M : Morsegeschwindigkeit setzen

In der „fat version“ kommen hinzu

- W : Weckzeit setzen
- A : Alarm ein/aus
- E : Alarm stop (es wird das Drücken des Punktkontakts ausgewertet)
- K : Korrektursekunden (1...9 sec) setzen

Die Zeitsetz- Befehle erwarten eine vierstellige Zahl (Eingabe ohne Zwischenraum und ohne Zwischenzeichen); die Ein- Ausschaltbefehle erwarten „0“ oder „1“, und die Morsegeschwindigkeit ist als 2-stellige Zahl einzugeben.

Ist die Eingabe der Zahl(en) vollständig, wird (bzw. werden) sie wiederholt.

Wurden nicht Ziffern, sondern andere Zeichen eingegeben, erfolgt sofort die Ausgabe „RPT“ (repeat), ist die Anzahl der Ziffern zu gering, erfolgt nach einer gewissen Wartezeit ebenfalls die Ausgabe „RPT“. Die Uhr fällt in beiden Fällen in den Idle-

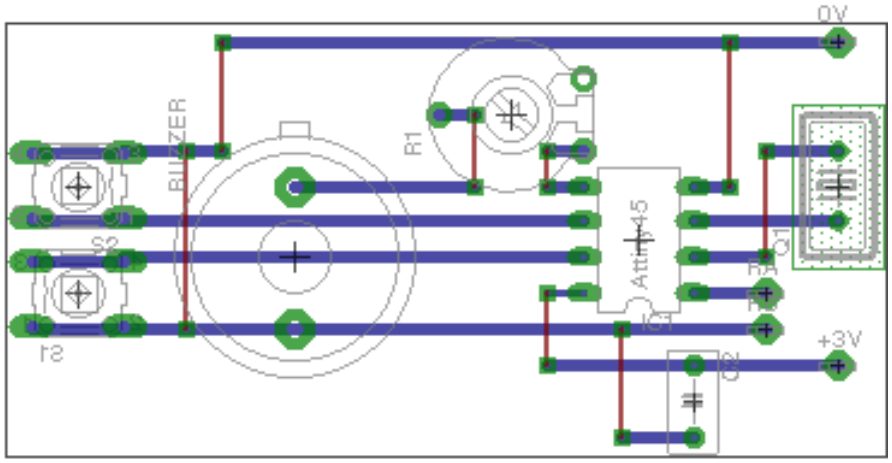


Abbildung 2: Layout

Modus zurück; das heißt, vor einer erneuten Eingabe der Zahl(en) muß zunächst das Kommando neu eingegeben werden.

In der Morsegeschwindigkeits- Subroutine wird zusätzlich geprüft, ob die Geschwindigkeit in einem sinnvollen Bereich liegt (zwischen 10 und 30 WPM). Ist dies nicht der Fall, wird „RPT“ ausgegeben und die Geschwindigkeit auf 20 WPM zurückgesetzt, damit die Uhr bedienbar bleibt.

Es erfolgt aber keine vollständige Prüfung auf Plausibilität; Zeiteingaben wie „1299“ sind möglich; die Zeichen werden ja nach der Eingabe wiederholt und der Nutzer muss prüfen, ob die Eingabe sinnvoll war. Lediglich Zeiteingaben, die größer als 23:59 Uhr sind, werden verworfen und es wird „RPT“ ausgegeben.

4 Einige Erläuterungen zur Software

Das wichtigste Unterprogramm ist die Interrupt- Routine, die ein mal pro Sekunde durch den Timer ausgelöst wird. Sie addiert die Sekunden und rechnet sie in (Tages-) Minuten um; ist ein Tag verstrichen (1440 min.), werden die Tagesminuten im Hauptprogramm wieder auf Null gesetzt.

Die Zeitberechnung ist in der Interrupt- Routine angesiedelt, damit auch dann, wenn der Mikrocontroller gerade ausgelastet ist (z.B. mit den relativ lange dauernden Morseausgaben), keine Minuten- oder Stundenwechsel verloren gehen können. Die Tagesminuten sind sehr schnell errechnet; die Interrupt- Routine ist daher sehr kurz, sie stellt aber dennoch eine eindeutige Zeitinformation bereit.

Das Hauptprogramm ruft nur kurz die Zeitberechnungen und die Tastaturabfragen auf und fällt dann bis zum nächsten Interrupt in den Idle- Mode.

Damit die Uhr trotzdem verzögerungsfrei auf Tastendrücke reagiert, sind für die beiden Tasten- Eingänge „Pin Change Interrupts“ aktiviert.

Kann man den Powerdown- Modus nutzen, werden fast alle Funktionsblöcke des Mikrocontrollers durch einen einzigen Registereintrag abgeschaltet. Im (hier wegen des Quarztaktes notwendigen) Idle-Modus bleiben die meisten Funktionsblöcke jedoch wach und verbrauchen Strom; man muss sie einzeln abschalten. Genutzt habe ich die Register PRR und DIDR0 - vielleicht gibt es noch weitere Möglichkeiten der Stromeinsparung.

Die meisten Unterprogramme habe ich aus älteren Projekten kopiert: die Subroutinen Zeitberechnung, Gong (früher: Schlagwerk), Zeitausgabe und Morse stammen aus der Morseuhr 2. Die Routine „Decodekeyer“ stammt aus dem Projekt Morse-Terminal. Sie sind dort näher erläutert.

Hinzugekommen sind:

Die Subroutine Menue. Eine Select - Case- Struktur vergleicht eingegebene Zeichen mit dem ASCII- Wert der Befehlszeichen „ ?,Z,G,C,M “ und verzweigt in weitere Unterprogramme. Andere Zeichen werden ignoriert.

Die Routinen Zeitsetzen und Speed erwarten die Eingabe mehrstelliger Zahlen. Diese werden im Unterprogramm Readnumbers erfaßt, auf Zulässigkeit geprüft und nach Eingabe der letzen erwarteten Zifffer zur Kontrolle und Bestätigung ausgegeben.

Der Befehl „G“ würde eigentlich mit einer Bitvariable (0 oder1) auskommen; aus Vereinfachungsgründen ist es jedoch ein Byte; alles andere als „1“ gilt als ”0“ (für: abgeschaltet).

Die zusätzlichen Funktionsblöcke der „fat version“ sind hoffentlich ausreichend durch die Kommentare im Quelltext erklärt.

5 Ausblick

Mit geringen Änderungen der Software und Umflashen von Fusebits kann man die Uhr prinzipiell mit einem 32 kHz- Uhrenquarz betreiben. Jedoch ließ sich der Mikrocontroller nach Umstellung auf den langsamen Takt nicht mehr flashen - weder ließ sich ein Programm aufspielen, noch ließen sich die Fusebits wieder auf den Betrieb mit einem Quarz oder mit dem internen RC- Oszillator zurücksetzen. Retten ließ sich der Mikrocontroller nur noch mit einem HV- Programmer.

Das schien mir für eine Bauanleitung dann doch zu „wacklig“. Schade, denn der Stromverbrauch in den Idle- Phasen betrug mit dem 32 kHz- Takt nur noch 0,08 mA.