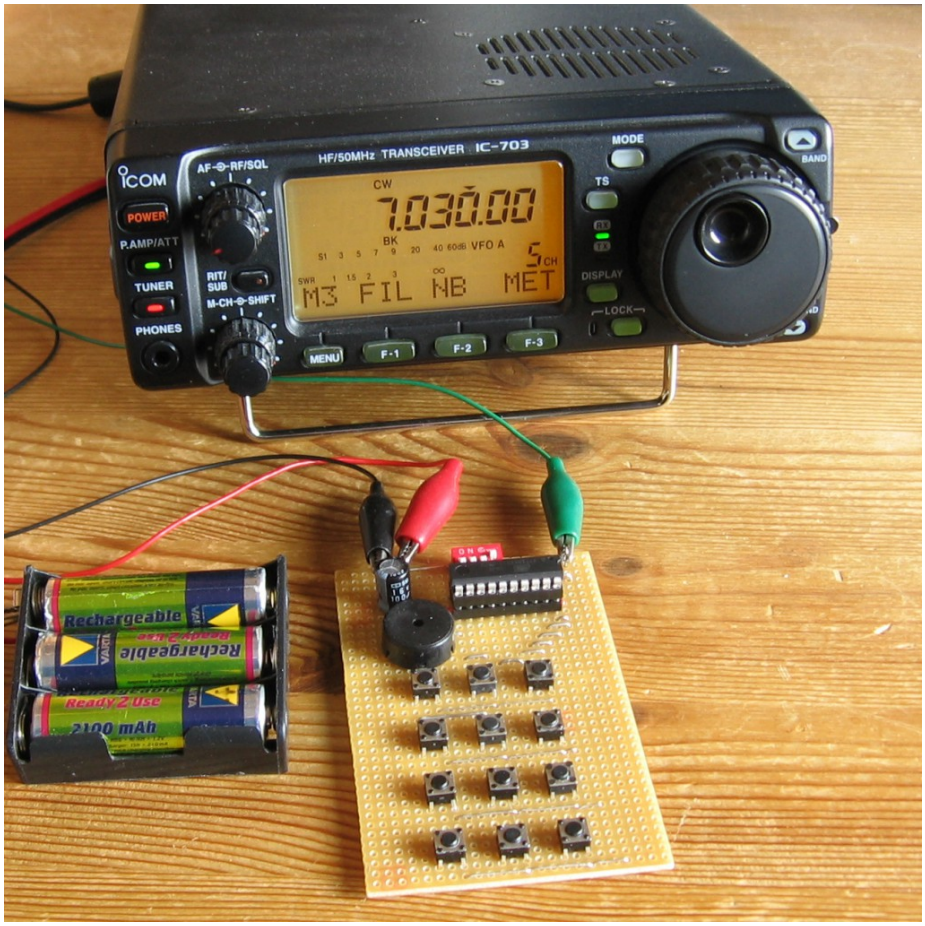
CI-V-QSY-Box

Ralf Beesner

9. November 2011



1 Einleitung

Hat einen der Mikrocontroller-Virus so richtig gepackt, wird nicht nur die Umsetzung zum Problem, sondern schon die Idee, was man denn bloß mal bauen

könnte. Da fallen einem dann nicht unbedingt neuartige Dinge ein, sondern Lösungen, die vor Jahren mal ein „cooles Gadget“ waren und die man damals nur passiv konsumieren konnte, die man aber heute dank BASCOM selbst bauen kann.

In diese Kategorie fällt der „QSYer“. Fast alle Amateurfunkgeräte weisen einen seriellen Eingang auf, über den man einige Funktionen des Geräts fernsteuern kann. Da nur wenige Geräte eine Zifferntastatur zur komfortablen Direkteingabe von Frequenzen aufweisen, brachte eine Firma vor vielen Jahren ein kleines Gerät mit 16er- Tastatur heraus, das die Fernsteuerdialekte einiger Funkgeräte beherrschte.

Damit konnte man Frequenzen per Direkteingabe aufrufen statt mit Bandschalter und Drehknopf mühsam dorthin zu kurbeln; bei einigen Geräten waren auch weitergehende Funktionen (z.B. Abruf der Memorykanäle des Funkgeräts) möglich.

Da ich nur zwei Geräte der Firma ICOM besitze (einen Transceiver IC 703 und einen Empfänger IC R8500), habe ich ein solches Gerät für ICOM- Funkgeräte entwickelt.

2 ICOM- CI-V- System

Die älteren Yaesu- Transceiver (z.B. FT757, FT747) hatten kein einheitliches Design ihrer „CAT- Schnittstelle“. Neuere Yaesu- Geräte (oder Geräte anderer Hersteller) kenne ich nicht näher.

Bei ICOM setzte man recht früh auf ein halbwegs konsistentes Gesamtsystem: einen seriellen Bus, mit dem man nicht nur mit einem PC mehrere Funkgeräte bedienen konnte, sondern auch Geräte untereinander vernetzen konnte, um sie z.B. im Frequenz- Gleichlauf zu betreiben.

Neben der Arbeitsfrequenz lassen sich weitere Parameter wie z.B. Schrittweite, Modulationsart, Eingangsabschwächer und dergleichen fernsteuern. Man kann über die Schnittstelle auch die Speicherkanäle auslesen und beschreiben.

Die Vernetzung erfolgt über eine Zweidrahtleitung. RxD und TxD sind zu einer gemeinsamen Leitung zusammengeführt (Pegel 0V/5V); die Übertragung erfolgt als asynchrones serielles Signal mit 8bit (8N1) wie zu den Zeiten, als man noch mit Telefonmodems und Mailboxen hantierte.

Um selektiv ansprechbar zu sein, weisen die ICOM- Geräte unterschiedliche Hexbyte- Geräteadressen auf. Die Adresse (und die Übertragungsgeschwindigkeit) lassen sich meist im Einstellmenü des Gerätes verändern.

Der PC (bzw. hier die externe QSY-Tastatur) hat stets die Adresse &HE0 .

Die Struktur der Datentelegramme ist in Bild 2 dargestellt:

Die Telegramme beginnen mit zwei Synchronisationsbytes &HFE, dann kommt die Geräteadresse, es folgt die PC-(bzw. Tastatur-) Adresse. Danach kommt das Steuerkommando für direkten Frequenzwechsel und anschliessend die in 5 Bytes

umgeschlüsselte Frequenz (den Aufbau der Frequenzbytes erkennt man am besten an dem ersten Beispiel 1.234.567,890 kHz). Den Abschluss des Telegramms bildet das Byte &HFD.

Frequenzeinstellung bei ICOM- Geräten

Aufbau der CI-V- Datentelegramme

	Startbyte 1	Startbyte 2	Geräteadresse (hier R8500)	PC- Adresse (immer E0)	Steuerkommando Frequenz	Frequenzbyte 1Hz/10Hz	Frequenzbyte 100Hz/1kHz	Frequenzbyte 10kHz/100kHz	Frequenzbyte 1MHz/10MHz	Frequenzbyte 100Mz/1GHz	Endbyte
	FE	FE	A4	E0	05	1	2	3	4	5	FD
Die Frequenzbytes sind BCD-codiert (also HEX- Zahlen; 2 Stellen pro Byte)											
Beipiele für die Frequenzbytes:											
1.234.567,890 kHz						90	78	56	34	12	
4.567 kHz						00	70	56	04	00	
936 kHz						00	60	93	00	00	
Komplettes Telegramm für 936 kHz:	FE	FE	A4	E0	05	00	60	93	00	00	FD

Abbildung 1: CI-V-Protokoll

3 Hardware

Das Programm entstand zwar ursprünglich auf einem AtMega8- Board; die Aufgabe passt aber besser zu einem AtTiny 2313, da nur etwa 1,6 kB Flash und 14 IOs benötigt werden. Der AtTiny 2313 hat einen Hardware- UART und im Gegensatz zum AtMega8 sogar PinChange Interrupts.

Die Tastatur besteht aus einer 3*4- Matrix. Oben links liegt die „1“, in der unteren Reihe liegen die Tasten * , 0 und #.

Man kann sie wie ich aus DIP- Tastern selbst bauen, es gibt aber im Versand-

handel (z.B. bei Conrad) schönere Tastaturen (man muss aber darauf achten, dass die Tasten in einer Matrix verdrahtet sind; es gibt auch Versionen, deren Tasten einzeln herausgeführt werden).

Eine solche Tastatur lässt sich sehr einfach mit dem BASCOM- Befehl `Getkbd` abfragen; er setzt einen 8 bit breiten Port voraus, der voll belegt wird. Die unteren 4 IOs liegen an den Spalten der Tastaturmatrix (der 4. IO bleibt bei einer 12er- Tastatur unbeschaltet), die oberen 4 IOs liegen an den Zeilen der Tastaturmatrix.

Die Tastatur liegt an Port B des AtTiny 2313. PD1 ist der UART- Ausgang, an PD2 bis PD5 liegt ein „Mäuseklavier“. Ein Buzzer für Quittungstöne ist an PD6 angeschaltet.

Der Mikrocontroller läuft mit dem internen RC- Taktoszillator. Er ist ausreichend stabil für den Betrieb der seriellen Schnittstelle mit 19200 bit/s; allerdings nur, wenn der Takt nicht auf 1 MHz (dem Auslieferungszustand) belassen wird, sondern auf 8 MHz hochgesetzt wird.

Der Controller muss jedoch nicht umgefust werden; die Änderung der Taktrate von 1 MHz auf 8 MHz erfolgt zur Laufzeit des Programmes.

Da die Funkgeräte TTL- Pegel erwarten, kann man den seriellen Ausgang des AtTiny 2313 über einen Schutzwiderstand mit dem CI-V- Anschluss des Funkgerätes verbinden. Bei meinen Geräten durfte er maximal 12 kOhm betragen. Zwei Dioden sorgen für die Ableitung von Überspannungen.

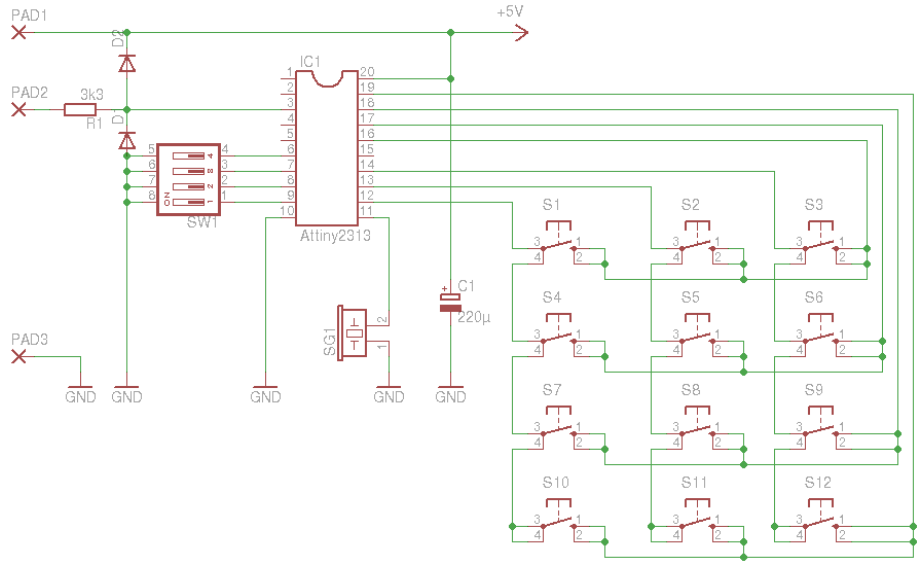


Abbildung 2: Schaltplan

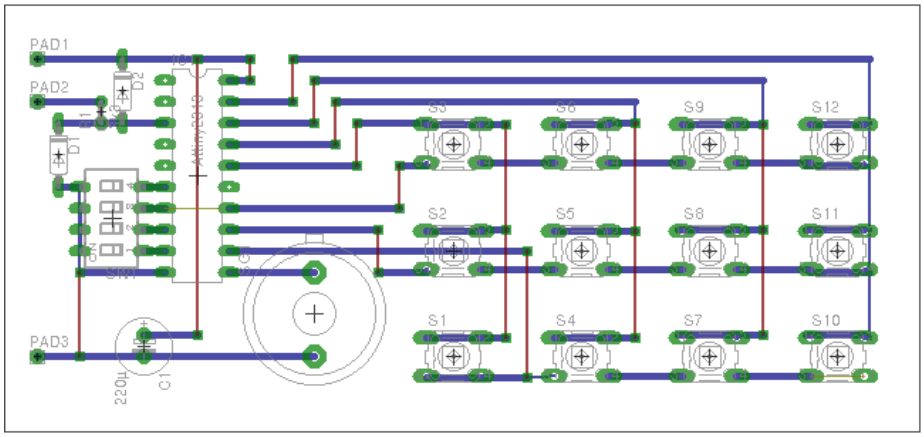


Abbildung 3: Layout

4 Software

Die Software besteht aus folgenden Blöcken:

- einem Initialisierungsblock, der nur nach einem Reset durchlaufen wird
- der Mainloop, die Frequenzeingaben entgegennimmt, umwandelt und an das Gerät sendet
- der Subroutine für die Tastaturabfrage
- zwei Subroutinen für kurze und lange Quittungstöne

Im Initialisierungsblock wird das „Mäuseklavier“ abgefragt; der wichtigste Parameter, der hier eingestellt werden kann, ist die Geräteadresse des zu steuernden Funkgerätes bzw. Receivers. Daneben lässt sich noch eine abweichende Baudrate einstellen und in einen 4-Byte- Modus für die Frequenzbytes umschalten (dem Manual des R8500 entnahm ich, dass zumindest der IC735 nur 4 Frequenzbytes nutzt).

Die Mainloop liest erst mal die Geräteadresse aus dem EEPROM. Dann wird in das Unterprogramm `read_kbd` verzweigt. Hier wird zunächst auf einen Tastendruck gewartet (`Getkbd` liefert „16“ zurück, wenn keine Taste gedrückt ist), die gedrückte Taste ermittelt und gewartet, bis die Taste losgelassen wurde (`Getkbd` liefert dann wieder „16“ zurück). Der Wert wird in das Array „Puffer“ geschrieben und ein Quittungston ausgegeben.

Ist der Quittungston nicht hoch und kurz, sondern tief und lang, wurden zu viele Zeichen eingegeben.

Wurde die Taste # gedrückt, ist die Frequenzeingabe abgeschlossen; der Quittungston ist dann hoch und lang.

Der `Getkbd`-Wert für # (12) wird durch 0 ersetzt.

Danach geht es dann wieder in die Mainloop.

Die eingegebenen Zeichen stehen nun im Array „Puffer“.

Damit man nicht alle Frequenzen 10-stellig eingeben muss, wurde die Taste * als Trennzeichen zwischen MHz und kHz definiert. Die Lage dieses Trenners im Array „Puffer“ wird daher zunächst gesucht.

Anschliessend werden die Ziffern in das 10- stellige Array „Freq“ umsortiert. In dem Array stehen zunächst Nullen, und ausgehend von dem Trenner * werden die eingegebenen Ziffern so ins Array „Freq“ kopiert, dass das unterste Element Freq(0) die GHz- Stelle enthält und das oberste die Hz- Stelle (sofern für die Stelle keine Ziffer eingegeben wurde, enthält das Byte eine „0“).

Das 10 Byte-Array „Freq“ muss jedoch noch auf 5 Byte eingedampft werden; die Bytes werden als Halbbytes (Nibbles) in das 5-Byte- Array „Freqbyte“ umsortiert (gerade Stellen ins untere Halbbyte; ungerade ins obere).

Danach wird das komplette Telegramm gesendet: zwei Synchronisationsbytes, Geräteadresse, Absende- Adresse, Befehlsbyte „Frequenzwechsel“, 5 (bzw. 4) Frequenzbytes und das Abschluss- Byte.

Da der ursprünglich verwendete Befehl `Printbyte freqbyte(n)` sonderbare Sachen machte (es wurde nicht nur das indizierte Byte, sondern das gesamte Array ausgesendet), wurde die umständlichere Variante `Print Chr(freqbyte(n));` verwendet.

Zum Schluss geht der Attiny2313 in den Sleep-Modus. Da bei meiner Bascom-Version der Befehl „Powerdown“ nicht wie erwartet funktionierte, ist das über direkte Registerbefehle und den Maschinenbefehl „!Sleep“ gelöst.

Mir war nicht so recht klar, ob Pin Change Interrupts bei einer Keyboard-Matrix funktionieren, da alle Anschlüsse des Keyboards an IOs des Controllers angeschlossen sind und daher möglicherweise nicht auf einem definierten Potential liegen, aber im Test klappte es einwandfrei.

Die Pin Change Interrupts für Port B werden über die Registerbefehle

```
PCMSK = 255
GIMSK.5 = 1
SREG.7 = 1
```

ausgewählt und aktiviert.

Kurz-Bedienungsanleitung

CI-V-QSY-Box an den CI-V- Anschluss des Funkgerätes anschalten, Mäuseklavier-Schalter „4“ auf „On“ stellen, dann die Betriebsspannung (5V aus 3 Mignonzellen oder aus einem Spannungsregler 12V/5V) anlegen.

Die Geräteadresse 4-stellig eingeben und mit # abschließen, z.B.

```
Geräteadresse 68 -> 0608#
Geräteadresse 4A -> 0410#
Geräteadresse FE -> 1514#
```

Die CI-V-Box geht dann in Powerdown und verbraucht praktisch keinen Strom.

Die Dipschalter 1 - 3 haben folgende Funktion:

Schalter „1“ auf „On“: 4800 bit/s (Standard ist 19200 bits/s)
Schalter „2“ auf „On“: 9600 bit/s
Schalter „3“ auf „On“: 4bit-Modus (IC735)

Bei der Eingabe einer Frequenz werden die MHz- Stellen eingegeben, dann ein *, anschließend die kHz- Stellen und # als Abschluss. Kurzeingaben sind möglich, z.B.

```
7*03# -> 7,030 MHz
7*#   -> 7 MHz
936#  -> 0,936 MHz
```