This is a CQ machine, developed from my simple QRSS callsign keyer . It was inspired by email correspondence with Rick M0RCP and is based on an Atmega AVR programmed in C: the ATtiny13 with 1K of program space and 64 bytes of data memory. The keyer sends:

CQ CQ CQ DE G0UPL G0UPL G0UPL K

then waits for a certain period, and tries again. The idea is that if you hear someone reply, you switch off the keyer, and have a QSO. If nobody replies, then just let it continue, and continue, and continue some more. Actually, the project was inspired by correspondence with Rick M0RCP, but not exclusively. Other times ideas arise, but don't cause a project. The additional motivation here was caused by the G-QRP club's valve-QRP day. I tried, with my one-valve CW TX . This project, is what you definitely would like, if you send CQ for hours from a tiny urban apartment where you can only hang a short wire aerial, and where the urban QRN is so loud that even if someone did hear your signal, you probably wouldn't have heard theirs.

I will program this chip and send it to you with your own callsign. The price is £3.00 including UK postage. Add £0.50 for Europe, £1 for rest of world. Email me if interested.

The CQ machine has the following features:

+ Choice of speed: continuously variable from 7wpm to 180wpm.
+ Choice of delay: 0 to 45 seconds (or once-only operation)
+ 750Hz sidetone output
+ Positive-going keying output

Pinout

To the right, see the pinout of the chip. The basic specifications of the chip such as power consumption, supply voltage etc should be taken to be the same as the ATtiny13 datasheet. Supply voltage range is at least 3-6V.

Pin 1 of the IC (top left) is identified by a small dimple in the chip, but I have added a blob of white paint to aid identification. Pin 1 is the "Reset" input to the processor and can be connected to Vcc, or left unconnected.

Pin 2 controls the wait period between CQ transmisisons. The delay is proportional to the voltage applied to this pin, from 0 to 45 seconds (when the pin is connected to Vcc). When the pin is grounded, the IC operates in a once-only mode. It sends out one CQ message, and after the final "K" falls asleep. The only way to CQ again is to power it off and on again, or to take the reset pin (pin 1) low temporarily to reset the program.

Pin 7 controls the keying speed.

Keying speed in wpm = k * Vcc / Vpin

Where:

Vcc is the supply voltage
Vpin is the voltage applied to the speed control pin
k is a contant having value 7.03125

Saying just "7" for the value of the constant would be fine, given the inherent few % accuracy of the RC-based oscillator in the AVR which this is based on. It should also be noted that at Vpin = 0, the keying speed maxes out at 180 wpm, rather than becomes infinite :-D   The formula is a simplification, which doesn't capture this finer detail of behaviour, though I think that matters little for practical uses!

Rick M0RCP's CQ machine project

here is a schematic of the circuit I'm using. A few notes:

The potential divider formed using VR2 gives a range of speeds of approximately 12 wpm to 35 wpm

At 12 wpm the stored message takes approx 37 sec to send while at 35 wpm it takes approx 12sec.

SW2 in conjunction with the resistor values shown (picked because I had them to hand) selects between approx 44, 22 and 11 second pauses and single shot mode.

Since it is theoretically (and actually) possible to short circuit the power supply using this arrangement (and I'm using NiMH cells) I've included a fuse in the schematic.

S0, S1 and S2 inputs select the keying speed and should be connected to GND or Vcc as per the table below. On powering up the device, it will start keying your callsign and repeat indefinitely.

Keyer speed selection

The table below shows the available keyer speed. I have also produced a version A, which has 50wpm and 60wpm settings suitable for meteor scatter experiments.

S2	S1	S0	Keyer speed	Keyer speed (A)
0	0	0	12 wpm	12 wpm

0 0 1 6 wpm
50 wpm
0 1 0 QRSS 1
60 wpm
0 1 1 QRSS 3
QRSS 1
1 0 0
QRSS 6
QRSS 3
1 0
1
QRSS 10
QRSS 6
1 1 0
QRSS 15
QRSS 10
1 1 1 QRSS 20
QRSS 20

NOTE: the microcontroller is clocked using the internal 9.6MHz RC oscillator. Being RC based, this is inherently inprecise. This means the actual keying speeds and sidetone could differ by a few percent from the values shown in the table above.
Source code

Here is the source code, written in C. It isn't optimally elegant but it's all I've had time for so far.

Finally here's a picture showing my programmer board and test set-up. I power this off two AA batteries in series (3V). I have connected one earpiece (half a pair of poundland headphones) to pin 2 via a 100-ohm resistor, so that I can hear the sidetone and make sure I've programmed the message correctly.

Shipping Problems

All was not well with my attempts to provide QRSS keyers at low cost to encourage as much QRSS experimenting as possible. Initially I offered these chips at a not-for-profit price of £2, which is about what I estimated it would cost to buy the ATtiny13 chips in low volume, pay the shipping on them, keep a small stock, buy some envelopes and some UK 1st class stamps. Unfortunately, the Royal Mail would disrupt my attempts, and I'm not just talking about the postal strikes due to Pay & Conditions disputes.

The first round of QRSS keyer chips, I packaged in short cuts of the plastic chip tubs that integrated circuits are often supplied in, then put these into a normal white envelope. Unfortunately one recipient reported back that his envelope had arrived torn in half, with a humble letter of apology from the Royal Mail, but nevertheless no QRSS Keyer chip inside. Presumably the bulge in the middle of the envelope must have caused it to get caught in some piece of Royal Mail machinery in some sorting office somewhere. Another recipient reported that his was fine, except that he'd had to go to his local sorting office to collect it, and pay a postage surcharge because the applied 1'st class stamp was deemed insufficient. I should explain that in the UK, a standard letter is defined as no bigger than A5-sized, and no more than 5mm thick. Any bigger than that, and one must buy a more expensive "Large" stamp. Definitely the bulge in the middle of those envelopes was more than 5mm.

So the next round of QRSS keyer chips, I packaged differently. This time I wrapped them in foil and stuck them to a piece of paper, then put that into a more industrial style of stronger-looking brown envelope. Now unfortunately several people gave feedback about the Royal Mail having converted the chips into Surface Mount Devices! I learn something new every day: now I know that the Royal Mail put all their mail under a steam roller in the sorting offices. Some work with the pliers could unbend the chip pins, so I am told. See below for some photos of the result (Left: Dave G8XUL; Middle, Right: Charles G3OTH).

So sadly, now I have to increase the price to £2.50, so that I can afford to buy some nice padded Jiffy-bag envelopes, and use the chip tubes, and pay for the more expensive "Large" stamp. Sorry about that: I tried to beat the system, but I failed.
Last Updated on Sunday, 22 November 2009 00:53