A DDS Based QRSS (and CW) Beacon

While experimenting with a DDS IC, the author added a PIC controller and QRP amplifier to create a really slow speed CW beacon.

Matteo Campanella, IZ2EEQ

Introduction

In early 2006 I became interested in the theory and applications of DDS (Direct Digital Synthesis), leading me to some experiments using Analog Devices ICs to better understand and evaluate this technology. Later on, during one of the usual Web-surfing sessions intended to find out something new to try out with the radio waves, I got in touch with the world of QRSS.

I knew little about these very narrow bandwidth modes, and I was convinced they were only used on the VLF bands. What I found on the Internet changed my vision about that, thanks to the contribution of a bunch of fellow hams belonging to a group called "The QRSS Knights."

These hams are active worldwide and they share a common interest: to break the micro-watt per mile record, renewing on a daily basis the challenge of being heard at the longest distance with the lowest power.

Most interestingly, a good part of the experimentation is done on the 30 m band, at about 10,140 kHz. This makes it accessible with the standard equipment and antennas an average ham usually owns (some traffic is done on 40 m, too, for an even wider audience).

What Is QRSS?

The term QRSS is derived from QRS — a CW abbreviation that means "You are sending too fast" or "Slow down." By extension, then, QRSS would imply a *very* slow sending speed. Another interpretation is that QRSS stands for "quasi-random signal source." This definition has to do with the very long dot and dash times.

There is a lot of information on the Internet about QRSS. One interesting and informative article is at **www.ussc.com/~turner/qrss1.html**.

Operating QRSS

Even though a commercial radio can be used to operate QRSS and related modes, usually the QRSS gear is home built, either on the basis of an existing project or starting from scratch, and the whole story works more or less like this: you build your QRSS/DFCW/FSKCW (more on these terms later) capable beacon, and put it on the air, possibly letting the fellow Knights know you're on the air. (There is a very active mailing list for this purpose: **mail.cnts.be/pipermail/knightsqrss_cnts.be**.)

The enthusiasm and participation on the list was so exciting that I decided to build my own beacon as well, and I couldn't think of a better way to put my recently acquired DDS knowledge to work.

Later on you will start receiving reports, but do not expect an RST report — a report is usually an e-mail or a posting onto the list, with a screen shot attached. This depends on the fact that, because of the very low power involved, the signal is not usually heard; it is buried in the noise. Given the very slow cadence of symbols, though, it is possible for some DSP enabled software to integrate over time and show the carrier on a frequency waterfall screen, much like the one we use with PSK31. One of the most-used programs for receiving QRSS is *Argo* (freely available on the Internet at **digilander.libero.it/i2phd/argo/index.html**). Even though this software has been developed with QRSS decoding in mind, it works perfectly with the other modes as well, as they are small variations on the main theme.

A Brief Guide to Narrow Bandwidth Modes

QRSS mode is 100% good old CW, only very, very slow; so slow that one dot is usually 3 to 120 seconds long! Considering that the dashes and spaces are in the usual relationship with the dot, you can easily imagine how long it takes to send a simple word! FSKCW and DFCW are instead 100% duty emission modes, as the transmitter never stops transmitting a carrier: FSKCW is quite similar to CW, the only difference being that keying is not off during the pauses but it's just shifted down a few hertz. Suppose you're transmitting on 10,140,080 Hz, in FSKCW. You have to shift your frequency to 10,140,070 Hz during the pauses. DFCW is quite different: dots and dashes in this mode have the same duration, and the difference is made by the frequency of the carrier. Referring to the previous example, you will transmit your dashes on 10,140,080 Hz and your dots on 10,140,070 Hz. This is the most efficient of the three, as it takes the smallest time to transmit a message. It's also a bit awkward to recognize at a glance, as we are used to seeing dashes always much longer than dots.

Graphically speaking, QRSS will show up on *Argo* or any equivalent software as a brighter line over the blue noise background; the line will be dashed, with the long marks corresponding to dashes and short ones corresponding to dots. The blanks will obviously be the pauses. FSKCW looks more like a square view, with the top line of the signal representing dots and dashes as in QRSS, and the bottom line the pauses. Finally, DFCW

SS Dei Giovi, 41/A
Binasco, Italy 20082
iz2eeq@arrl.net

looks like a square wave with 50% duty cycle, the dashes being on the top line of the signal and the dots on the bottom.

The Project

The most basic device to get on the air with QRSS is a crystal based oscillator followed by a buffer and a final amplifier. You want an output power that doesn't exceed half a watt most of the time (100 mW is the most common power). This allows for very simple circuit solutions and cheap components, but on the other hand there is a critical requirement — frequency stability. QRSS and similar modes are extremely narrow band, allowing many signals to be stacked close in the same 100 Hz frequency span (the range monitored by *Argo*). In such a condition it is very important to avoid as much drift as possible. This goal is normally achieved using heated crystals and thermally stable enclosures for the VFO module. Let's see how a DDS will behave.

The approach I have used with this project was to think modular, and to design each module so that it could be reused in a number of different contexts from the original project, as an elementary building block for something more complex. According to this model, I have identified three modules: the DDS module (responsible for signal synthesis), the CPU control module and the amplifier module.

Choosing Module Cores

When it comes to DDS, there's a huge choice on the market today. The most modern devices are out of reach for the casual experimenter anyway, mainly because of the package that makes them unsolderable by means of a simple iron. Luckily enough, Analog Devices makes a device that perfectly fits the needs at HF and low VHF bands — the AD9851. It's not exactly what you can consider easy soldering, having a pin to pin distance of 0.6 mm, but with some patience, a magnifying glass and a good low power iron it is possible to work it out.

This small wonder can be clocked at a maximum of 180 MHz, thus allowing synthesis of a sinusoidal signal up to 90 MHz for the Nyquist Theorem limit. (As a matter of fact you will want to limit synthesis to one third of the clock frequency, or 60 MHz, because of hardware limitations over theory.) The other interesting aspect of this chip is that it needs just one voltage (2.7 to 5.5 V) for operation, while some of the more powerful and modern solutions require different voltages for I/O

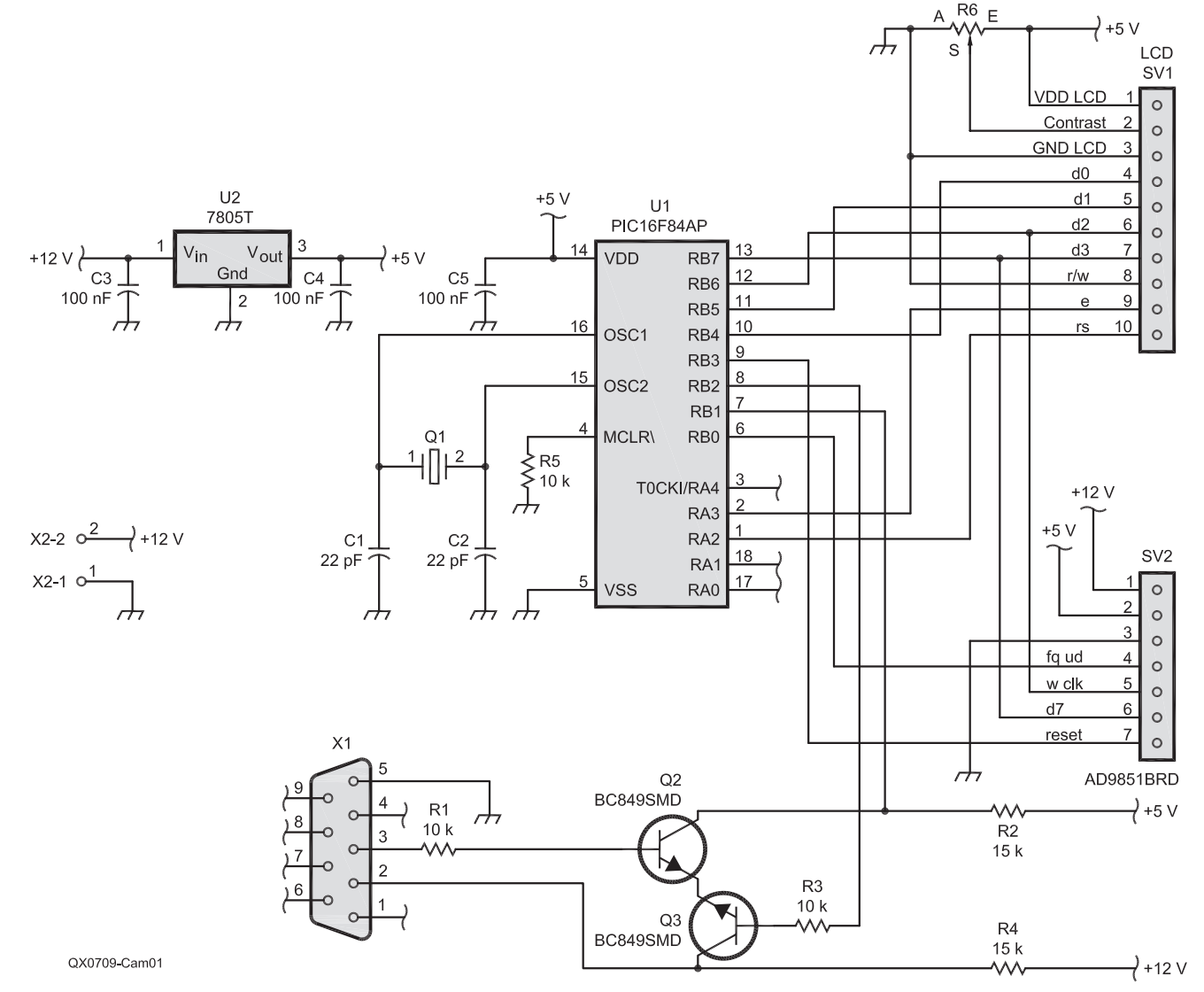
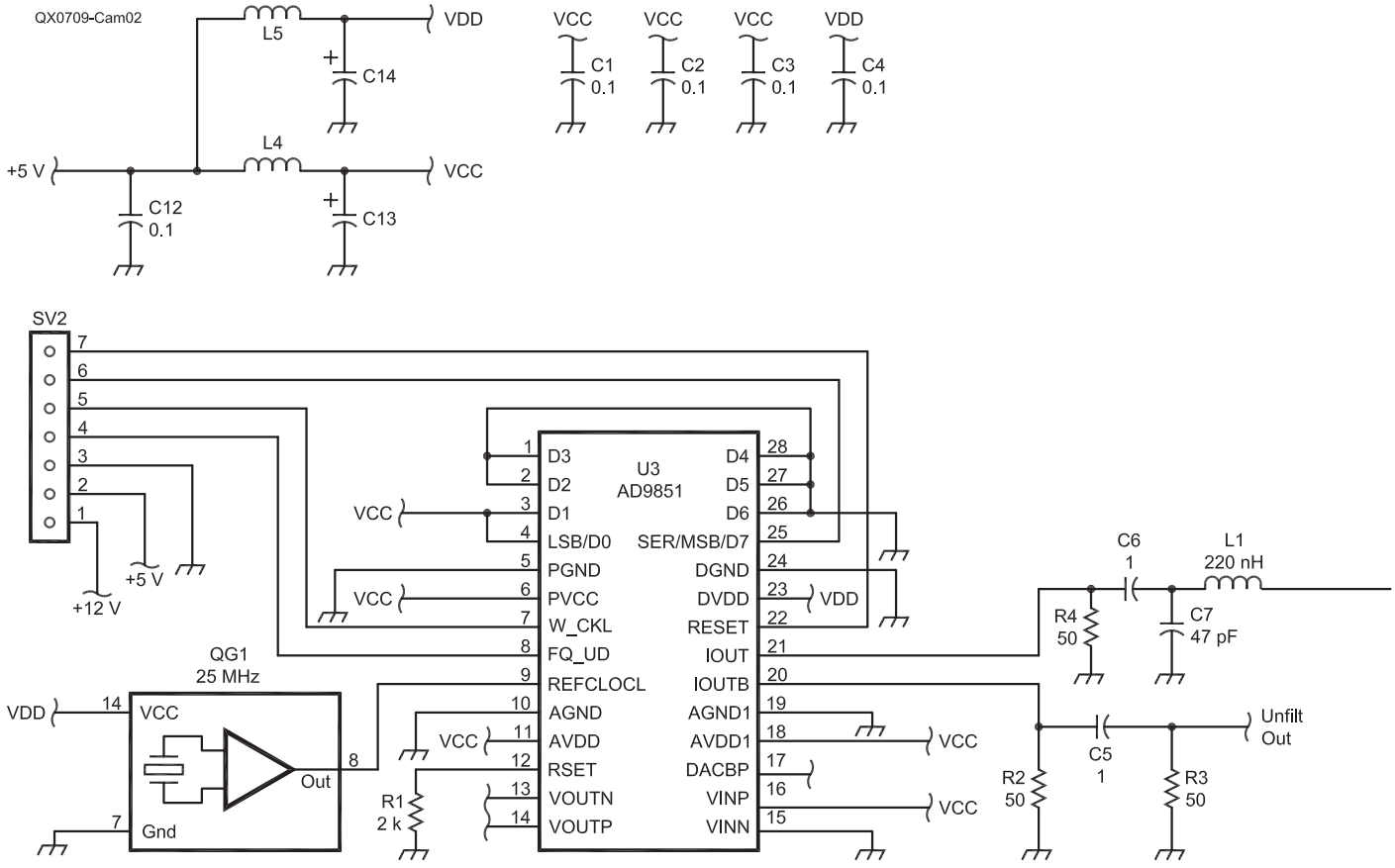


Figure 1 — This schematic diagram shows PIC controller module portion of the project.



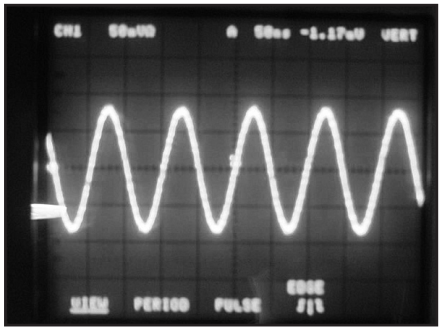


Figure 3 — The DDS output signal waveform is displayed on the oscilloscope CRT.

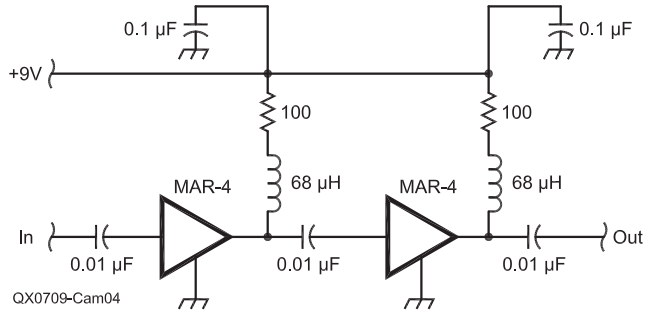


Figure 4 — This schematic diagram gives the class A MMIC amplifier circuit.

and core (as happens in computer processors). Ease of use, together with the advantages described above make this device very popular among Amateur Radio experimenters.

The choice of the microprocessor couldn't be easier: requirements are to control the device, drive some human readable interface (LCD) and manage input of commands. The PIC16F628 was chosen for this task, because it is small, versatile and abundant in my component box.[1]

[1]The PIC controller hex program file for this project is available for download from the *QEX* Web site at **www.arrl.org/qexfiles**. Look for the file **9x07_Campanella.zip**.

The CPU Module

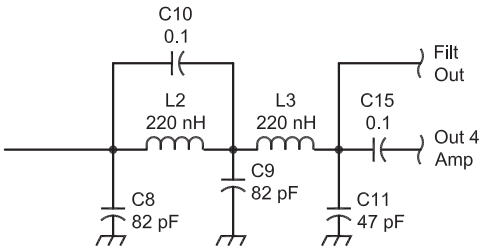
There's really nothing special about the CPU module, with the possible exception of the serial interface. In this case, rather than using an ad-hoc device for doing CMOS to RS232 level conversion, I have used two transistors and a bunch of resistors to obtain a similar result. Even though this way the transmitted signal is going to be in the range 0 to 12 V and not bipolar (–12 / +12 V) as RS232 specification dictates, I have never found a computer serial input that got confused and garbled the serial stream.

The crystal frequency for clocking the CPU is not critical at all; you can choose any frequency in the range of 4 to 20 MHz for the 16F628.

Of the 13 bits available as inputs or outputs on the CPU, six are used for LCD control (I used a Hitachi HD44780 compatible LCD module) and four for DDS control, while three ports are still free for custom uses. In order to achieve the minimum number of wires around, a four bit control mode has been chosen for LCD and serial control mode for the DDS; that way only seven wires are required to control the LCD and four for the DDS.

The schematic diagram of this module is shown in Figure 1.

Figure 2 — The direct digital synthesizer (DDS) module is given in this schematic diagram.



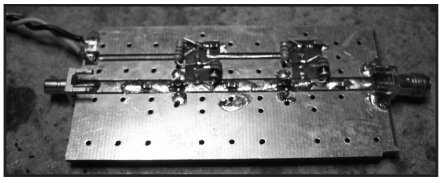


Figure 5 — Here is a photo of the MMIC amplifier construction.

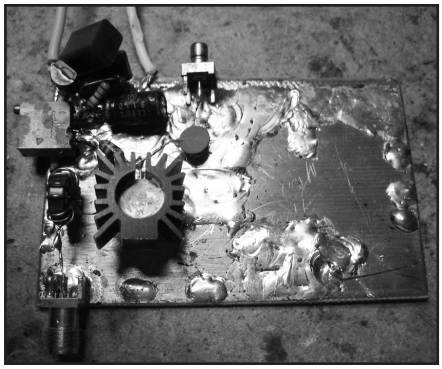


Figure 6 — This photo shows the 2N2259 transistor amplifier.

The DDS Module

This is the "trickier" module of the project, because it has a mixed nature of both analog and digital circuits. The recommendations about the correct use and placement of decoupling capacitors near the power supply pins are very precise and peremptory on the DDS datasheet. In order to achieve the best spectral purity of the synthesized signal, there are separate pins for analog and digital power supplies for the internal circuitry. C12, L4, L5, C14 and C13 are aimed at this purpose, and there is a dedicated 0.10 μF capacitor for every analog power supply pin (C1 to C4). See Figure 2.

The AD9851 can be clocked either by providing the right final clock frequency or by using the internal 6× multiplier. The latter choice is to be preferred for non critical applications, as it dramatically reduces the cost, considering that a 30 MHz computer grade oscillator costs much less than a 180 MHz one. (Actually, surplus computer cards are good sources of 24 MHz oscillators.) I am personally using a 25 MHz oscillator, thus limiting my theoretical maximum output frequency to 50 MHz (rather than the maximum obtainable 60 MHz).

The output of the DDS chip is balanced, and it is a current output; this means that I_{OUT} is "pushing" current out of the chip while I_{OUTB} is pulling current into the chip. Balanced outputs are really useful for cancelling common mode noise, supposing that the noise is equally induced on the two wires — as the difference between the two signals is considered — the same-sign components simply cancel out. Given the balanced output, two solutions are possible: one uses a center tapped 1:1 transformer to couple the I_{OUT} and I_{OUTB} outputs into a single output, while the other terminates I_{OUT} and I_{OUTB} to the same resistive load, capacitively coupling I_{OUT} to the output connector.

The first has the advantage of pushing out some more power and being quite immune to common mode noise; the second is less expensive and gives a flatter response over the whole device frequency range, at the expense of less output power. This is due to the fact that, no matter how broadband a transformer is, it will be difficult to obtain a frequency response that is as flat as that of a chip capacitor, especially at low frequencies, where the transformer efficiency degrades dramatically. I tested both solutions, and in the end I chose the capacitive coupling because I plan to use the same module for experiments at the IF as well, not to mention the reduced costs implied.

In order to eliminate aliases from the output signal, the last stage is a low pass filter with a cut-off frequency of about 50 MHz. Some applications are based on the ability of a DDS to produce aliases, so an unfiltered output could come in handy sooner or later — that's what you get from the I_{OUTB} circuitry. The important thing to remember is that the unused output *must* be terminated in 50 Ω for a correct operation of the device. The schematics show the DDS with the unfiltered output terminated, supposing you're going to use the filtered output to connect the power amplifier.

The output current, and therefore the output power, is dependent on the value of R1, because, according to the datasheet, it is given by 39.93 divided by R1, and it cannot be bigger than 20 mA. The given value of R1, 2 kΩ, is set for maximum output current. It could be a good idea to place a variable resistor in series with R1, so that the output power can be fine tuned to give the final amplifier the correct input power. R1 alone gives about 1 Vpp on a 50 Ω load, or 4 dBm. A good practice is to keep the maximum output current near 10 mA, rather than to the rated maximum. This value is obtained using an R1 of 4 kΩ, and it gives an output power of –2 dBm on 50 Ω.

Figure 3 shows the output of the DDS modules tuned at 10,140,080 Hz.

The Amplifier Module

The amplifier is the most "independent" part of this project, because you can use any power amplifier to get on the air, just taking into account that it has to accept an input signal at –2 dBm, and be able to provide 10 to 27 dBm (10 mW to 500 mW) output for QRSS activity. I built three different amplifier modules. The first one was based on MMIC amplifiers, the second on a 2N2259 transistor and the third on a bunch of 2N2222 transistors.

The MMIC amplifier is a two stage class A amplifier, based on a MiniCircuits MAR-4, and provides a power gain of 16 dB (each device is capable of 8 dB power gain). The third harmonic is well below 35 dB, and at these output levels a low pass filter after the amplifier is not really needed. Figure 4 is the schematic diagram of this amplifier. Figure 5 shows my circuit board. There is a limitation with this solution though: the maximum output of a MAR-4 device is 12.5 dBm, and this limits the maximum possible input to –3.5 dBm. This is not really a problem if you adopted the variable resistor in series with R1, as mentioned earlier.

The 2N2259 solution is a very simple power amplifier in class AB configuration, using a multiturn variable resistor to provide the correct bias to the base-emitter junction of the transistor. A low pass filter at the output of this circuit is mandatory because of the working class. An output power of about 100 mW is easily obtained with this solution. The transistor can reach quite high temperatures, so you will have to use a heat sink. Figure 6 shows my construction for this amplifier version. I haven't provided a schematic diagram because there was nothing special about the construction or operation of this amplifier.

The last amplifier I tried is based on a design that I found on the Internet, and is part of an all 2N2222 transceiver that K8IQY designed to attend (and win) the 1997 NorCal QRP Club building contest. It does not need any active devices except a couple of diodes

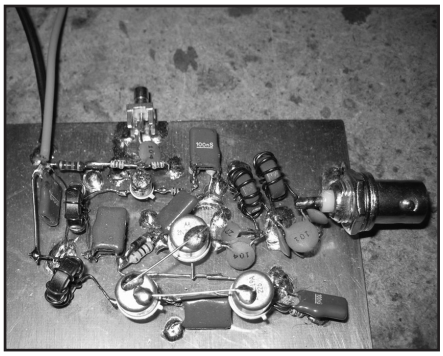


Figure 7 — This amplifier uses 2N2222 transistors to produce about 2 W of output power on the 40 m band.

and a bunch of 2N2222 transistors. (Actually, I used a 2N2219 transistor for the final stage. That is a somewhat more rugged version of the 2N2222.) It produces 2 W at 7 MHz. The only thing I had to redesign was the output low pass filter, because I chose to use this amplifier on 30 m. The details of this interesting project are at **www.k8iqy.com/qrprigs/2n240/2n240page.html** and the schematics are available at **www.k8iqy.com/qrprigs/2n240/2n2sche.html**. Figure 7 is a picture of the amplifier I built from this design.

As I mentioned earlier, any power amplifier that is able to take a –2 dBm input signal is perfectly suitable.

Figure 8 shows the prototype assembly of the DDS board, Control board and LCD.

The Firmware

The firmware of the beacon implements three narrow bandwidth modes (QRSS, DFCW, FSKCW) plus standard CW, serial communication to a computer for calibration, mode/speed selection and a way to enter beacon text. The mode and frequency are displayed on a Hitachi HD44780 compatible LCD. The text shown on the LCD, visible in Figure 9, indicates the mode (FSKCW in the example), the speed (03), and the frequency in hertz (10,140,080).

The serial port is always checked by the PIC processor during operation, so that commands can be sent to the beacon all the time; port setup is 57600,n,8,1. Valid commands are letters, as indicated in the following list.

1) q — switches to QRSS mode;
2) d — switches to DFCW mode;
3) f — switches to FSKCW mode;
4) c — switches to CW mode (fixed 16 wpm);
5) 1 — switches to speed 10;
6) 3 — switches to speed 3;
7) 6 — switches to speed 6;
8) a — increases frequency offset by 1 Hz;
9) z — decreases frequency offset by 1 Hz;
10) s — increases carrier frequency by 10 Hz;
11) x — decreases carrier frequency by 10 Hz;
12) h — increases calibration factor by 1;
13) n — decreases calibration factor by 1;
14) m — enables beacon message input;
15) r — resets device.

An echo is sent on the serial line for any key pressed, in order to give the user some sort of feedback.

Upon switching the unit on, if any digit is typed on the keyboard of a computer connected to the device, it enters the frequency programming mode, and expects a total of 8 digits (including the first one) that are the digits of the new frequency you want your beacon to operate on. The useful time frame during which the first digit is accepted is shown both on the serial line and LCD display by a ">."

Any variations to the settings are saved in the PIC flash memory, so the device will keep settings through resets or on/off cycles.

A calibration factor is used to fine tune the output frequency, in order to compensate for small errors in the clock nominal frequency. Since calibration is saved on flash memory, calibration is a once in a while operation.

I will offer programmed PICs for $15 US, plus shipping charges. I can also make the PIC source code available for $20 US. I will accept payment by PayPal.

Conclusions

Even if not aimed at the assembly of a QRSS beacon, the solutions presented in this article are well suited for reuse wherever a direct radio frequency synthesizer is needed. Considering the spectral purity of the generated signal, as well as the incredibly small tuning step DDS are capable of, only your imagination will limit the possible applications. One future extension of this project will be to connect a DAC to the output current calibration pin, in order to make the output current vary according to some input signal. We're talking about amplitude modulation here! If you're rather interested in discrete frequency modulation (such as FSK), no additional hardware is needed. It will just take some CPU reprogramming.

Matteo Campanella has been a radio amateur since March 1999, when he earned a Tech Class license with the call IW2NGE. He upgraded to Extra in June 2001 as IZ2EEQ. He has been interested in electronics since he was 13, when he built his first FM radio. He received the Laurea degree in Electronic Engineering in 1994 from the University of Pisa. Since then his work has mostly involved software related activities. He presently is Senior Consultant on Java Enterprise Applications and Open Source Solutions. For now, electronics has become a spare time activity, when he enjoys digital signal processing and embedded systems programming.

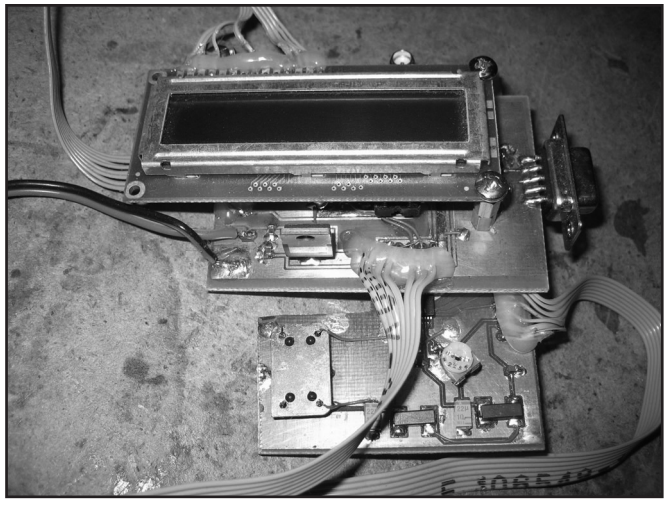


Figure 8 — Here is the combined PIC control module with LCD and the DDS module ready for testing.

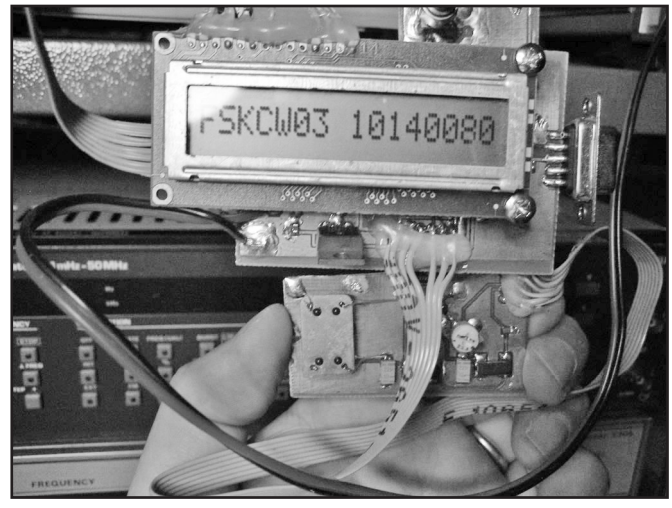


Figure 9 — With power applied, the LCD shows the operating mode (FSK CW), sending speed ((03) and the operating frequency (10,140,080 Hz).