Quick Beacons

Chuck MacCluer W8MQW

w8mqw@arrl.net

08/14/2019

Although it is trendy to establish digital propagation beacons transmitting MSK144 or WSPR, the traditional CW beacons still play an important role --- especially at VHF, UHF, and SHF --- where such beacons are used to verify frequency or antenna-pointing accuracy and changes to preamplifier noise figure or overall system noise. Such CW beacons require a signal source, a beacon keyer, and an amplifier, as in Figure 1.

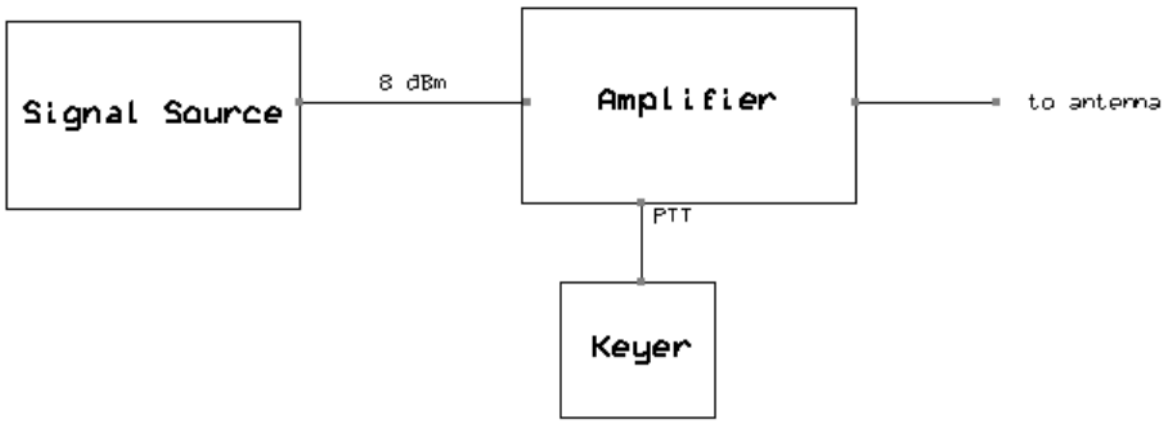

Figure 1. A typical CW beacon consisting of a signal source, a keyer, and an amplifier.

In the antediluvian past, a signal source at say 1296 MHz consisted of a 108 or 144 MHz crystal oscillator overdriving a MMIC with the 12th or 9th harmonic selected by a cascade of printed hairpin filters interspersed with MMIC amplifiers. But nowadays we have the Si570 from Silicon Labs and our signal source becomes simple indeed---see Figure 2.

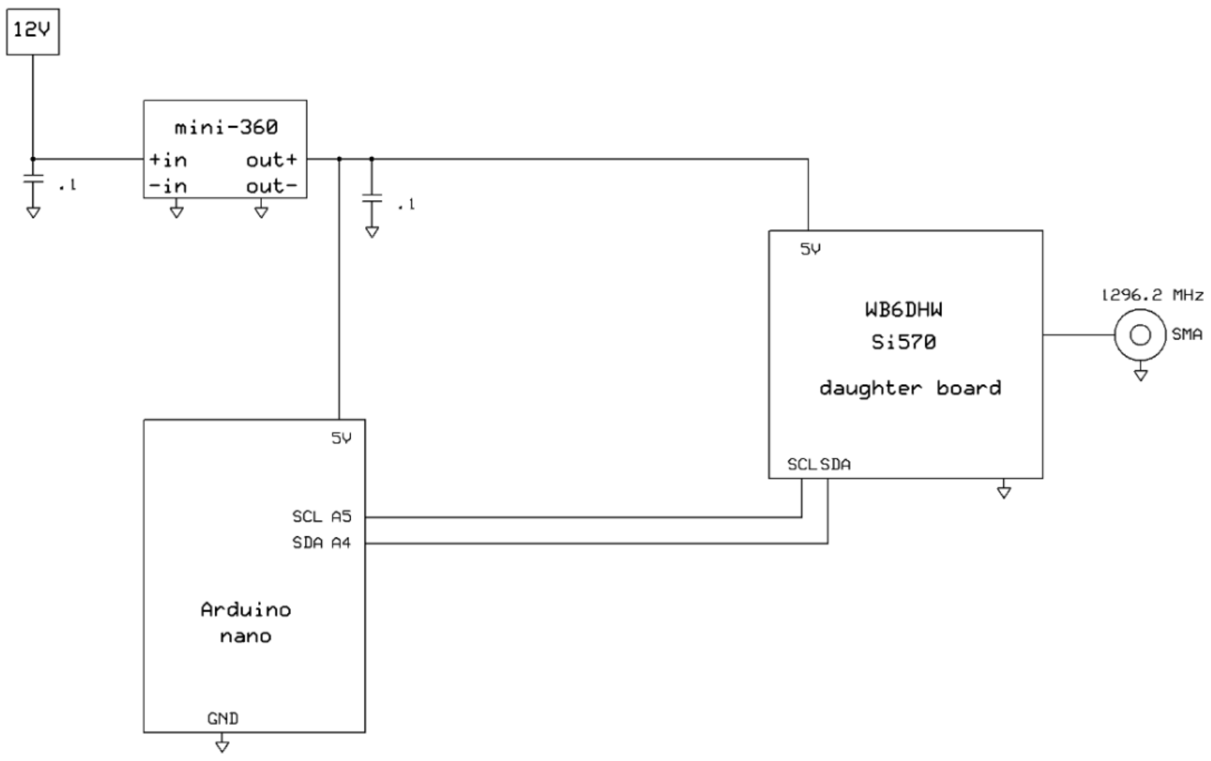

Figure 2. A simple beacon signal source using an Arduino (UNO, Nano, etc.) to write the frequency-determining registers of a Si570 programmable oscillator upon startup. The daughter board contains a Si570, a 3.3-V regulator, several pull-up resistors, and a level shifter to communicate I2C data from the 5-V Arduino to the 3.3-V Si570.

The Softrock Approach

For signal sources for beacons below 220 MHz there is an even simpler solution --- purchase a Softrock RX ensemble kit and use only the DDS portion as in Figure 3.

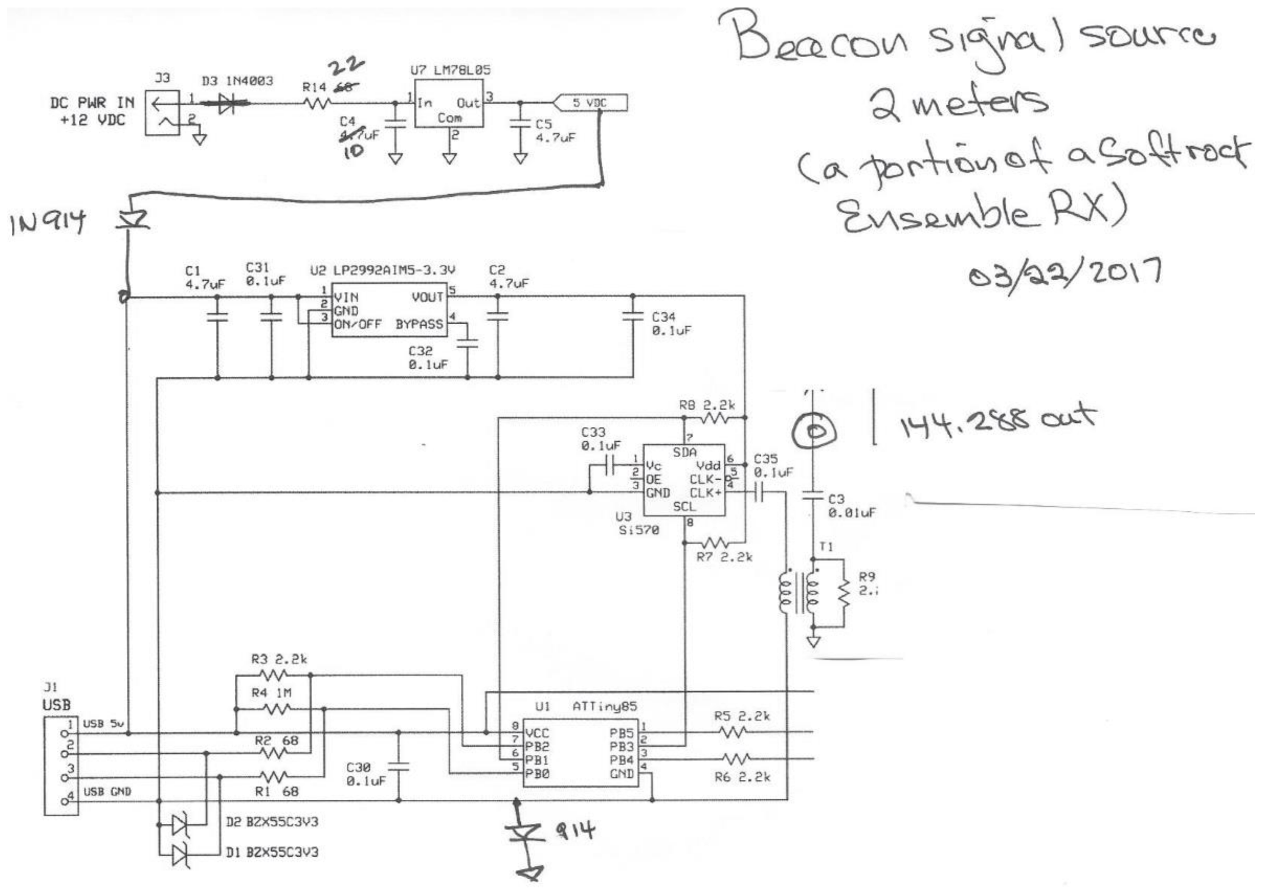

Figure 3. For 144 MHz (and below) beacons, merely employ the DDS portion of a Softrock RX Ensemble. Using the CFGSR utility, change the board's startup frequency to your desired beacon frequency.

Tony Parks of Five Dash occasionally has returned Softrock kits with functioning DDS portions that he will sell at a much-reduced price. More than 12 dBm is available at the output of T1 of Figure 3, at any desired frequency from 10—160 MHz. With the change of a single line in one tab of PE0FKO's CFGSR utility, the Softrock board can be set to start up at any frequency of your choosing. Once reprogrammed and USB power is removed, the board will wake up at the desired beacon frequency when powered up from 12 volts via the two added diodes shown in Figure 3. These boards also make excellent quiet LOs or lab frequency sources.

Unfortunately, the Softrock boards, with their CMOS Si570s, are unable to generate frequencies at UHF and SHF. Moreover, the firmware onboard the ATTiny85 controller locks out such programming choices. For these higher frequencies we must employ the Arduino/Daughter board of Figure 2.

Beacon sources at UHF and SHF

For these higher frequencies we replace the ATTiny85 controller of the Softrock with an Arduino (almost any Arduino will suffice) and employ a daughterboard populated by a faster LVPECL, LVDS, or CML version of the Si570. Again examine Figure 2. A typical daughter board (kit) is shown in Figure 4.

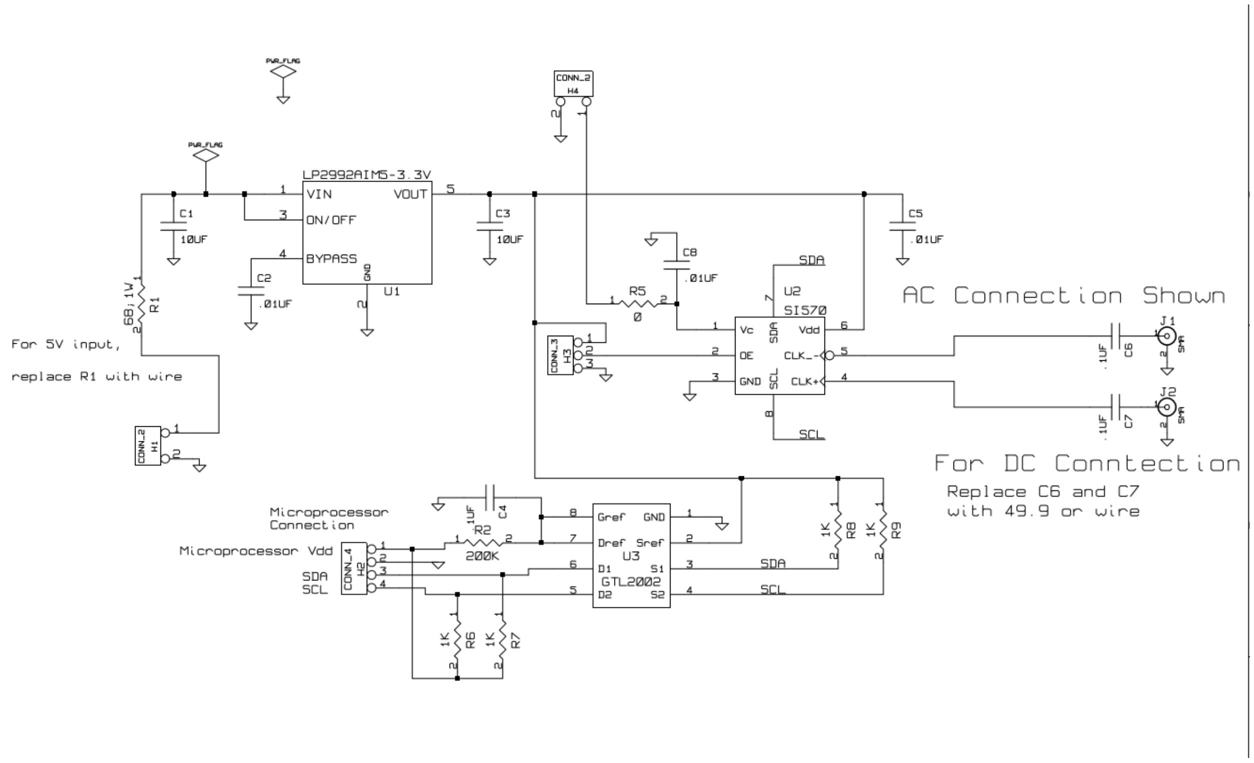

Figure 4. A Si570 daughter board from WB6DHW. If a 3.3-V Arduino is used, then the level shifter U3 would not be necessary.

It would not be difficult to self-fabricate a daughterboard, especially if a 3.3-volt Arduino were employed, since no I2C level shifter would be necessary and an off-board 3.3-volt buck regulator (e.g., a Mini-360) could supply both the Arduino and the Si570.

Keying the beacon

I have had great success with the United Microsystems XT-4 Beacon Keyer. It is well made, dependable, and trivial to program----merely send it your message once. If one of the Downeast Microwave 30-watt amplifiers (2M30PA, 222PA, 7025PA, 3340PA, or 2330PA) is used, the XT-4 can directly key the PTT line of the amplifier to produce excellent keying envelopes.

Simple Arduino software

The principal barrier to the wider amateur use of the Si570 in homebrewing is the intimidating Arduino code (sketches) that have been available for controlling the Si570. Even 'bare bone' sketches are full of calibration routines, number base conversions, register stuffing, and branching lookup tables for automatically generating the three numbers necessary for setting the oscillator frequencies. In contrast, for these beacon projects I have viciously pared the simplest of the available sketches to a mere skeleton. I will supply one sketch per band, each set up to generate a signal just below the lower limit of the ARRL beacon band plan. You need only experimentally increase the least significant (HEX) digits of one number (RFREQ) that is defined in the preamble of the sketch in order to steer the signal source to the desired final frequency. These sketches can serve as a tutorial and provide (I believe) the clearest example of how one programs the Si570.

Readers who find the above cut-and-try steering to the desired final frequency objectionable may consult the technical procedure that I have attached as an appendix to calibrate for your particular Si570. Once calibrated, there is a systematic procedure to exactly spot the desired final beacon frequency --- see Appendix.

Models of the Si570

The 3.3-volt Si570 comes in 36 flavors:

- Outputs: LVPECL, LVDS, CMOS, CML (A,B,C,D)
- Temperature stability: 50, 20, 7 ppm (A,B.C)
- Speeds: 10—1417*, 10—810, 10—280 MHz (A,B,C)

So for example, a Si570 with part number 570 CAC xxx-xxx-xx is a CMOS, with temperature stability 50 ppm, and speed 10—280 MHz, which is the Si570 included in Softrock kits. The remaining numerical xxx-xxx code is for start-up frequencies, the trailing letters xx denote packaging.

For UHF or SHF beacons we need the fast LVPECL, LVDS, or CML version with frequency range to 1417 MHz, but can live with the modest temperature coefficient of 50 or 20 ppm. That translates to the model number 570 {A,B,D}{A,B}A. My sketches for 220 MHz and up assume the use of the fast chip. A signal source using the fast chip and sketch could be used on any band by merely changing to the suitable N1, HS_DIV, and RFREQ in the preamble of the sketch.

If you happen to acquire a 570 *C*, there is one last complication: the six (one byte) registers of the Si570 holding the programming data are different for these 7 ppm chips. For the less expensive 50 or 20 ppm version the frequency programming registers are 7 – 12, but for the 7 ppm chip they are the registers 13 – 18. I resisted the temptation to automate this register change in the code. You will have to modify the register numbers in the sketches to use the 7 ppm chip.

Construction details

The si570 daughterboard, Arduino, and 5-V regulator can all be mounted together on a perfboard (see Figure 5), then enclosed with the keyer in a cast aluminum box --- see Figure 6.

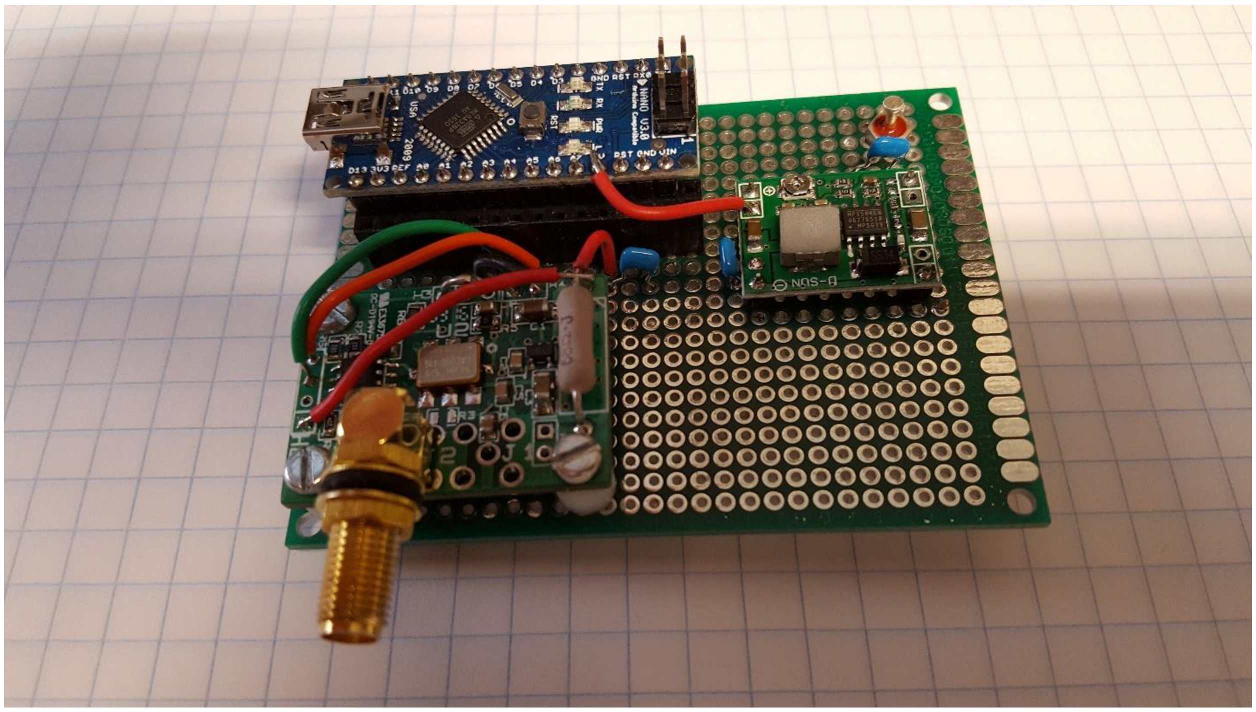

Figure 5. The si570 daughterboard, the Arduino, and 5-V regulator can be mounted together on a project board.

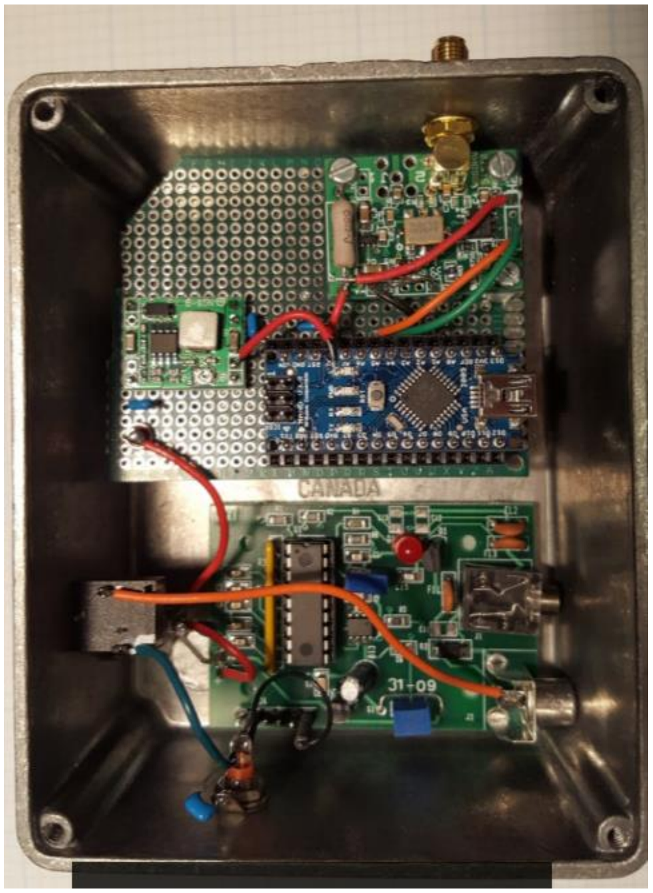

Figure 6. The signal source is enclosed with a KT-4 beacon keyer in a Hammond box with a 2.5 mm jack that brings in 12V, and a 3.5 mm stereo jack that brings in an amplifier sample voltage and the keying line. The RF is brought out via the daughterboard SMA and an RCA jack is used for external metering of the amplifier sample voltage.

The Downeast Microwave series of 30-watt amplifiers (2M30PA, 222PA, 7025PA, 3340PA, or 2330PA) make an ideal match for these signal sources, requiring only 8—9 dBm drive. These amplifiers include a PTT line out for the beacon keyer and an RF sampling line for a 1 mA meter. However, the CMOS Si570s crank out more than 12 dBm and overdrive the Downeast amplifiers, so that a series 5-dB pad will be required.

Concluding remarks

I urge everyone with a bit of tower space to do their civic duty by throwing up a quick beacon. The PAR omnis are perfect antennas for VHF and UHF beacons. However, 23 cm omni antennas are a problem. I would like to hear from you if you have solved this conundrum.

An unexpected annoyance arises from the use of switching 12-V power supplies to power these beacons: in synchrony with the CW keying, the power supply emits birdies on random HF frequencies; a linear power supply may be warranted.

I will be delighted to supply the Arduino sketch for any band you ask. Email me at w8mqw@arrl.net

Footnotes and suppliers

*10—945, 970—1134, 1213—1417.5 MHz

Unified Microsystems: http://www.unifiedmicro.com/beacon.html

http://wb6dhw.com/Si570/Si570.html

https://www.downeastmicrowave.com/Default.asp

APPENDIX

Refining the Arduino band sketches to place beacons exactly on frequency

This post is for readers who find cut-and-try steering to the desired final frequency objectionable. Here are the means of adjusting for the slight variations in the crystal frequency found in individual chips, so that you can directly program to your exact desired beacon frequency.

The frequency of the crystal onboard the Si570 is nominally XTAL = 114.285 MHz. Upload the sketch for your band into the Arduino, power up the signal source, and measure the actual output frequency. Then your actual onboard crystal frequency XTAL is found by multiplying the nominal crystal frequency by the ratio of the actual output over the expected. For example, using the 144 MHz sketch, if the actual output frequency is 144.274 instead of the expected 144.275, then your actual onboard crystal is oscillating lower at

$$114.285 * (144.274/144.275) = 114.2842079 \text{ MHz.}$$

Use this corrected value henceforth for your value of XTAL.

Next, choose your desired beacon frequency Fout. Then using the decimal values of N1 and HS_DIV in the sketch preamble, compute the decimal value of RFREQ by the rule

RFREQ = Fout * N1 * HS_DIV / XTAL.

Last, convert this decimal value into hexadecimal with the leading two hex digits representing the integral part, while the next seven hex digits represent the fractional part.

For example, in the 144 MHz sketch, N1 = 4 and HS_DIV = 9. Then supposing we want to place the beacon at 144.288, we see that

RFRFEQ = 144.288*4 * 9 /114.2842079 = 45.45131909.

The integer part 45 = 32 + 13, or hex 2D, which is written as 0x2D. The fractional part is transformed into a 7 place hexadecimal by using any online decimal-to-hex converter to transform the integral part of

0.45131909 * 2^28 = 121,150,046, (rounding up)

resulting in the hex value 0x7989A5E. The two hex numbers are concatenated to become the 9 digit

RFRFEQ = 0x2D7989A5E.

Replace the nominal RFREQ in the sketch with this new hex value and the oscillator will be spot on your chosen beacon frequency.

These computational details are explained quite well on page 19 of the Silicon Labs Si570/571 Application Note. As you might expect, when these details are incorporated into the sketch instead of done offline as above, the code begins to become overwhelming.

The Si570 is programmed by writing these three parameters N1, HS_DIV, and RFREQ into 6 registers in a rather idiosyncratic way --- examine my sketches and see the Application Notes. For small changes in frequency, only RFREQ need be changed. For large frequency jumps, the digitally controlled oscillator must first be frozen, then restuffed with a new triple (N1, HS_DIV, RFREQ), set, then unfrozen. All this is explained in some places clearly, others places not so clearly, in the Application Notes.