Gene Marcus, W3PM/GM4YRE

113 Wickerberry Lane, Madison, AL 35756: **w3pm@arrl.net**

An Arduino Controlled GPS Corrected VFO

A VFO that provides 1 to 112.5 MHz signals on two independent outputs. Use it as a stand alone unit or with a GPS receiver to improve frequency accuracy. UTC and six digit grid square locations are also displayed in the GPS Mode

This project began with the purchase of an Si5351A clock generator breakout board for less than $8 from Adafruit Industries. Designed as a substitute for crystal oscillator clocks, it features three output ports for frequencies between 8 kHz and 160 MHz. Although the board is specified for a wider bandwidth, this project is limited to 1 through 112.5 MHz.

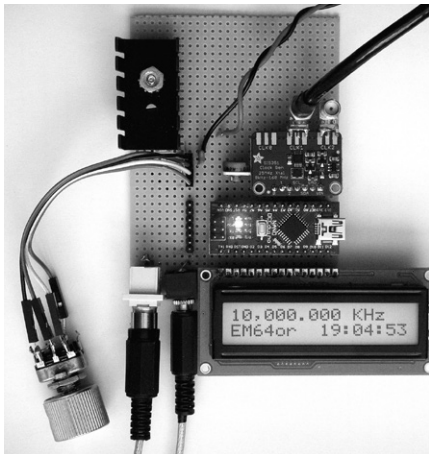

Figure 1 — I constructed the VFO on a piece of perfboard. The heatsink shown at the top left corner of the board is for the 7805 voltage regulator. The regulator is not required for the basic non-GPS configuration of the Si5351 VFO project, if used without the display backlight. The VFO output signals connect to the CLK1 and CLK2 connectors at the edge of the Si5351 board. The Arduino Nano is between the Si5351 board and the display board. At the bottom left of the perfboard are the GPS connections. Also note the rotary encoder to the left side of the perfboard. The pushbutton switches were not included on this version of the VFO.

Figure 1 shows my project, built on a piece of perfboard. The Si5351 board is the top board on the right side of the perfboard. Just below that is the Arduino Nano board I used to control the oscillator. This version uses a rotary encoder to set the operating frequency. You can see the encoder off the left side of the board. Figure 2 shows a completed unit, packaged in a plastic project box. The Resolution, Band Select, and Reset pushbuttons are on the right, just below the rotary encoder.

The Si5351A board does have limitations. Although it is a highly capable and stable board, the output is a square wave with odd harmonic frequencies present in the output. The square wave output does make a good source for some mixers. Phase noise is also higher than other popular programmable signal sources. A quick search of the Internet will yield a wealth of data concerning the performance of the Si5351A IC. Builders are urged to consider phase noise and crosstalk limitatinos before using this IC in their project.

A simplified version of the VFO can be built without the GPS module. Figure 3 shows the circuit for this configuration. Figure 4 shows the schematic diagram for the complete circuit, with GPS module, rotary encoder and pushbuttons.

Unlike a GPS disciplined oscillator (GDO) using a phased lock loop (PLL), this project uses a GPS 1 pulse per second (pps)

Figure 2 — Here is a completed VFO project, housed in a plastic project box. The Resolution, Band Select, and Reset pushbutton controls are located just below the rotary encoder.

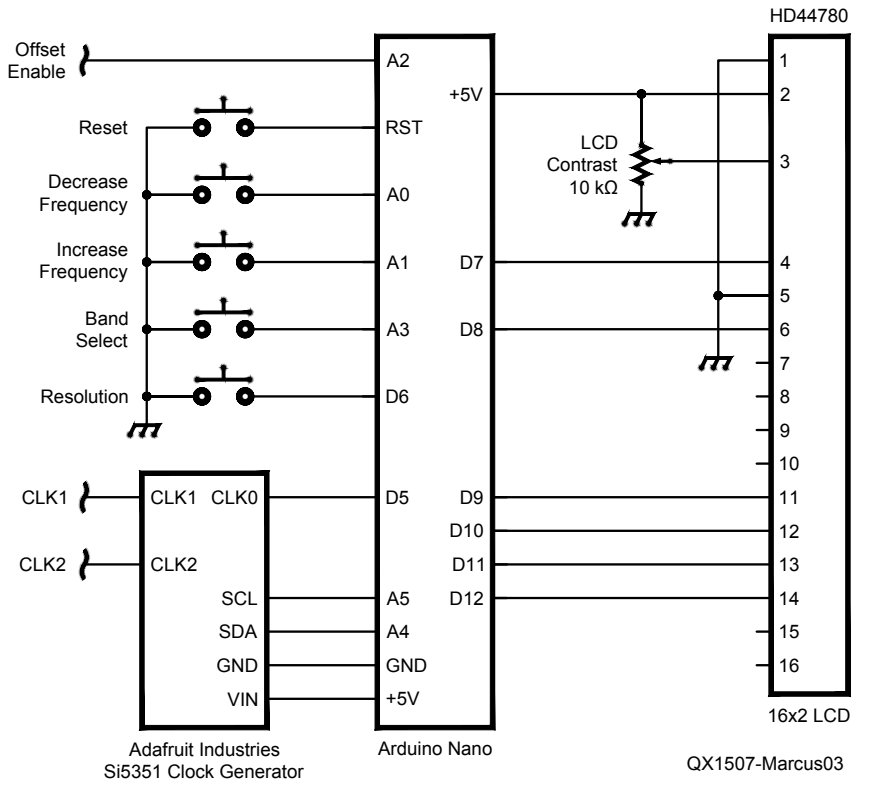

Figure 3 — This is the schematic diagram for the basic non-GPS configuration of the Si5351 VFO project. Either an Arduino Nano or Arduino Uno can be used.

output to act as a precision frequency counter gate. An Arduino Nano (or Uno) is used as a frequency counter to calculate a correction factor to use when programming the Si5351 board. Although not as accurate as a GDO, this simple method provides a variable signal source from 1 to 112.5 MHz with an uncertainty of better than 1 part in 10^7.

The Arduino provides the processing power required to calculate the frequency, control the Si5351A board, serve as a UTC timekeeper, and as an added bonus, calculate your 6 digit Maidenhead grid square locator.

Favorite operating frequencies are stored in software and are accessed by the ***Band Select*** pushbutton. Each band may be configured in a VFO only or VFO/LO combination. The Si5351A ***CLK1*** output port is the VFO and ***CLK2*** is used as an LO. The displayed frequency is arithmetically corrected when the VFO/LO configuration is used. Multiple bands can be configured in this manner. A programmable offset function allows you to use the unit for transceiver operation.

Theory of Operation

The Si5351A is based on a PLL/VCXO and high resolution MultiSynth fractional divider architecture. The Si5351A board can generate any frequency up to 150 MHz on each of its outputs. System short and long term frequency uncertainties are attributed to the onboard 25 MHz clock.

One of the three Si5351A outputs (CLKØ) is programmed to 2.5 MHz and routed to the Arduino's frequency counter input port (pin D5). The 1 pulse per second output from the GPS receiver is routed to the Arduino's interrupt 0 input port (pin D2) to act as a counter gate. The Arduino counts the 2.5 MHz input over a 40 second gate time, resulting in a 100 MHz total count. This count is used to recalculate the 25 MHz clock frequency. Total system uncertainty, including calculation resolution limitations and clock drift during counter gate time, is better than 1 part in 10 million. The VFO frequency is updated every 40 seconds, or when the frequency is changed. Typical frequency uncertainties versus time for GPS and non-GPS configurations can be seen in Figures 5 and 6.

When the unit is first turned on, the GPS processing routines are enabled to determine the correct UTC and 6 digit Maidenhead grid square location. When the software determines that valid data has been received, the GPS processing routines are disabled. At this point, frequency accuracy and time is maintained by the GPS 1 pps signal. GPS NMEA processing is turned off to eliminate processing conflicts and resultant frequency counter errors.

Excellent library routines are available on the internet to simplify Si5351A frequency programming. Instead, I chose to program the Si5351A PLL and MultiSynth functions directly without the use of library routines. The resultant code is very simple compared to other routines, but works quite nicely in this application.

I found I could easily program the Si5351A board up to 150 MHz using PLL divider techniques. Unfortunately, PLL divider techniques create glitches each time the frequency is changed. Fixed PLL frequencies using MultiSynth division provide glitch-free tuning, but the frequency range is limited to 112.5 MHz using this method.

Options

The unit may be built without the GPS receiver and used as a stand-alone VFO (refer to the software installation instructions). If used in the stand-alone mode, you can expect a drift rate of approximately 1.6 Hz / °F (2.8 Hz / °C). Operating the VFO in stand-alone mode also provides an excellent means to test the project prior to connecting the GPS related hardware.

The frequency is controlled either by the rotary encoder or the frequency up/down pushbuttons. You may want to include either the encoder or pushbuttons, or both.

Construction

Construction of the VFO unit is not critical provided adequate RF layout techniques are used. Do not use long unshielded wires for RF and GPS 1 pps connections. I first built the system using a solderless breadboard without any problems. Later, I transferred the circuit to a RadioShack perfboard.

The Si5351A board and LCD may be directly powered from the 5 V pin of the Arduino. A separate 5 V DC source is required if you use the LCD backlight and/or GPS receiver. A 7805A voltage regulator with a small heat sink works nicely. LCD backlight current requirements vary by manufacturer, so circuit details are not included here.

GPS antenna location is critical. A solid GPS signal is necessary to ensure consistent system operation. You should try to locate the GPS antenna with a clear view to the sky, away from noise sources. The GPS receiver

requirements are 5 V operation, 1 pps output, and 4800 baud NMEA data output in either $GPGGA or $GPRMC format.

Software Installation and Setup

The Arduino download website (**http://arduino.cc/en/Main/Software**) outlines installation instructions for the first-time Arduino user. My Arduino Si5351A_vfo.ino file is located at: (**www.knology.net/~gmarcus/Si5351.html**). This file, current at publication time, is also available for download from the ARRL *QEX* files web page.[1]

[1]The software code for the Arduino Controlled GPS-Corrected Dual Output Si5351A VFO, current as of the publication date of this issue of *QEX*, is available for download from the ARRL *QEX* files web page. Go to **www.arrl.org/qexfiles** and look for the file **7x15_Marcus.zip**.

Important note: The serial input port (pin 0) is used for both GPS and USB serial data. Do not attempt to upload software into the Arduino Nano while receiving GPS data. Disconnect the GPS NMEA data line before uploading software.

The software will allow the displayed frequency to be arithmetically corrected when both output ports are used in the VFO/LO configuration. Multiple bands may be configured in this manner.

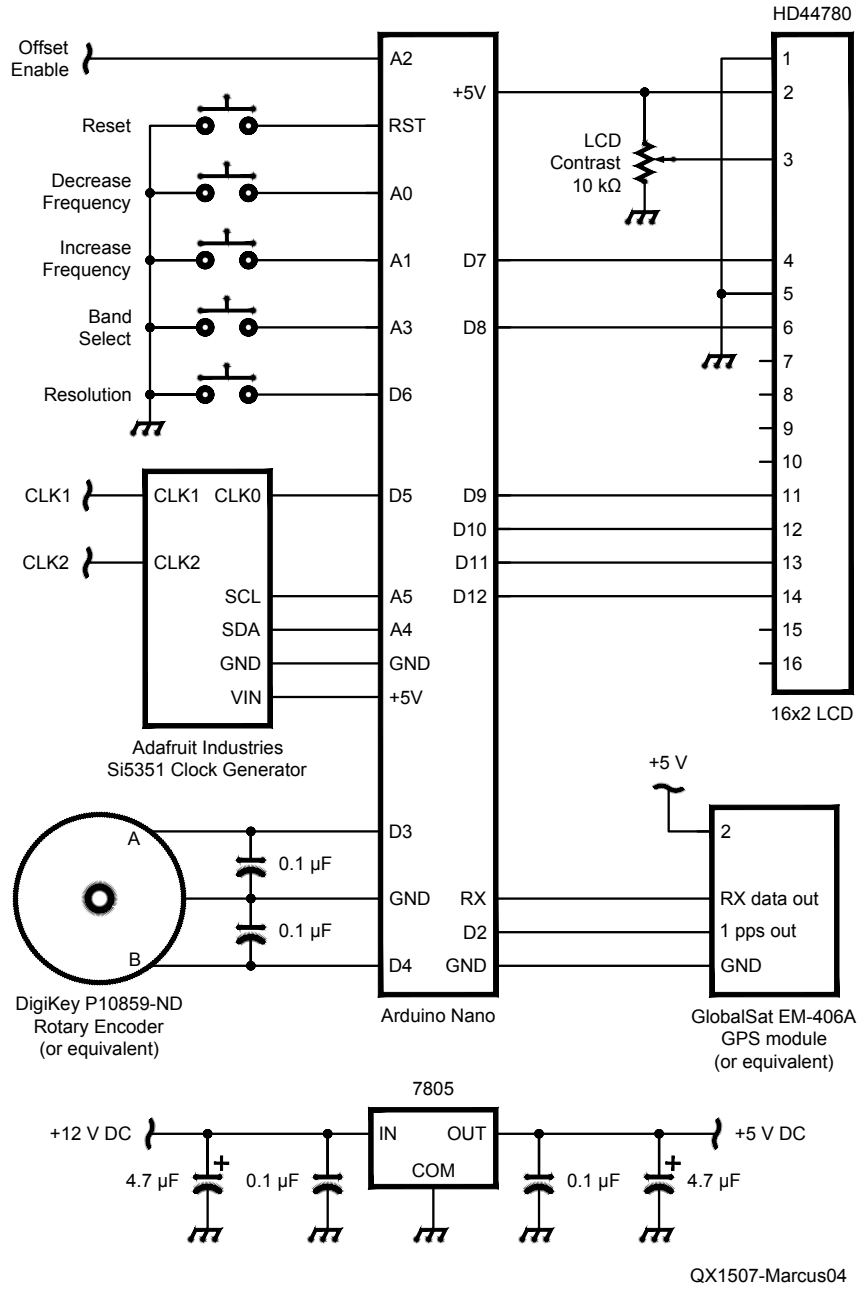

Figure 4 — The GPS corrected version of the Si5351 VFO project is shown in this schematic. The rotary encoder can be included to control the operating frequency. Either an Arduino Nano or Arduino Uno can be used.

```
const unsigned long Freq_array [] [3] = {
{ 7030000,0,0 },                    // CLK1=7.030 MHz, CLK2=0 MHz, Display=7,030.000 kHz
{ 1810000,0,0 },
{ 3560000,0,0 },
{ 7040000,0,0 },
{ 10106000,0,0 },
{ 14060000,0,0 },
{ 18096000,0,0 },
{ 21060000,0,0 },
{ 24906000,0,0 },
{ 28060000,0,0 },
{ 50060000,0,0 },
{ 5286500,8998500,1 },      // CLK1=5.2865 MHz, CLK2=8.9985 MHz, Display=14,285.000 kHz
{ 5016500,9001500,2 },      // CLK1=5.0165 MHz, CLK2=9.0015 MHz, Display=3,985.00 kHz
(0,0,0)

};
```

Near the beginning of the Arduino sketch you will find the following variable that defines the Band Select configuration:

Depressing the Band Select pushbutton will step through the programmed frequencies. Each entry follows the format "{CLK1 frequency, CLK2 Frequency, math command}." You may modify, delete, or add to the Band Select list. The last entry in the list must be (0, 0, 0). The rotary encoder or frequency up/down pushbuttons will only control the CLK1 frequency.

All frequency entries are in Hz. The math command controls how the frequency is displayed. A "1" will add CLK1 and CLK2, and a "2" will subtract CLK1 from CLK2. A "0" results in no change.

For example: {5286500,8998500,1}, will result in:

```
CLK1 output: 5.286500 MHz
CLK2 output: 8.998500 MHz
LCD display: 14.285000 MHz
```

The ***Offset*** variable controls the frequency offset. When the Arduino pin A2 is held low, the programmed offset in hertz is added or subtracted. Any positive or negative value of hertz can be used to program the frequency offset.

There are two ways to configure this project: either as a GPS corrected frequency source or as a stand-alone unit without GPS correction. Locate the variable ***GPSflag*** near the beginning of the Arduino sketch and set the variable to either "1" for GPS correction or "0" for non-GPS operation.

If you choose to use the project without GPS correction, you can enter a correction factor to allow for greater frequency accuracy. To calculate and enter this correction factor, perform the following steps:

1) Connect the VFO to a frequency counter

2) Set the VFO to 25 MHz

3) Note the measured frequency in Hz

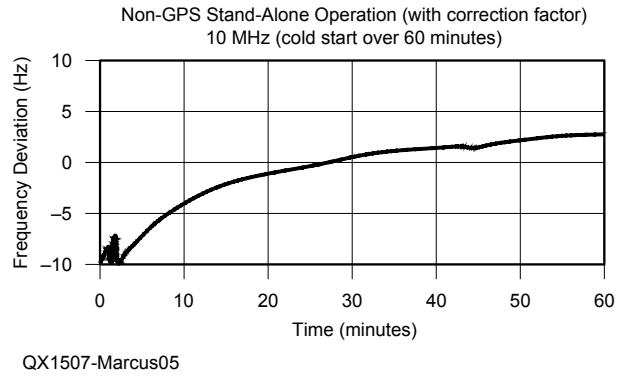

Figure 5 — This is a graph of the frequency versus time over the initial 60 minutes, for the output from the non-GPS version of the VFO project. A constant room temperature was maintained during the data collection. The drift rate is approximately 1.6 Hz / °F (2.8 Hz / °C).

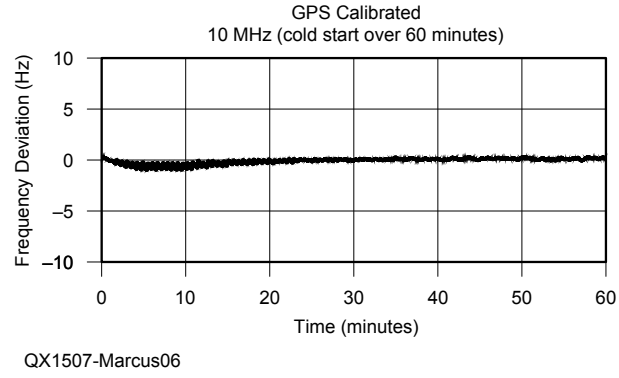

Figure 6 — Here is the graph of the frequency versus time over the initial 60 minutes for the GPS corrected VFO. A constant room temperature was maintained during this data collection. Frequency corrections to bring the frequency back to normal are shown during the early minutes of warm-up.

4) Subtract 25 MHz from the counter reading

5) Note the difference in Hz (such as –245)

6) Locate the variable ***CalFactor*** in the Arduino sketch and enter the value.

Frequency uncertainty without GPS calibration and without the calculated correction factor is normally less than 1 kHz.

Operation

When first turned on, you will see "Waiting for GPS" displayed on the LCD. See Figure 7. (If GPS correction is not used, the second line of the LCD will continually display the frequency step resolution, as shown in Figure 8.) It may take a few minutes for the GPS receiver to lock and obtain valid NMEA data. After the GPS receiver obtains valid data, the "Waiting for GPS" display will be replaced with the (UTC) time and your 6 digit grid square locator. The system is now ready for operation. See Figure 9.

Depress the resolution pushbutton to select the frequency step. When the button is depressed the selected resolution will be displayed for a few seconds on the second line of the LCD. See Figure 10. The frequency may be changed either by using the frequency up/down buttons or the rotary encoder. Depressing the Band Select pushbutton will step through the programmed frequencies.

Experimentation

This is an open source project. You are encouraged to experiment and improve upon the system operation. Simple system improvements such as using a 4 × 20 LCD to display additional data can be easily implemented. Variables containing latitude, longitude, and the number of satellites in view exist in the ***GPSprocess()*** subroutine. More complex updates such as varying the CLK2 LO frequency may also be incorporated.

Gene Marcus, W3PM/GM4YRE, was first licensed in 1963 as KN3YVP, and has held an Amateur Extra Class license since 1968. He was first licensed in Scotland as GM5AQM in 1969. He received a First Class FCC Radiotelephone license in 1977.

Gene completed an ASEE degree program at Penn State University in 1968. After a four year tour as a Cryptologic Technician with the US Navy, he began a 32 year career in the field of precision measurement equipment calibration. In retirement he enjoys experimenting with various RF and microprocessor projects, and enjoys world travel with his wife, Phyllis.

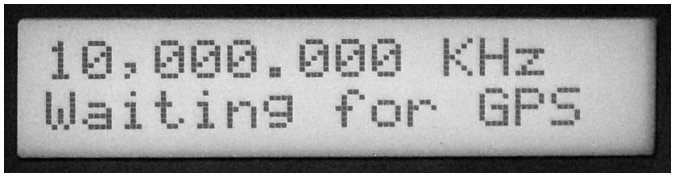

Figure 7 — When you first turn on the GPS corrected VFO, you will see the start-up frequency and the Waiting for GPS message on the display.

Figure 8 — If you build the non-GPS version, the initial display will show the start-up frequency and the initial frequency step size.

Figure 9 — When valid NEMA GPS data is received by the Arduino, the second line of the display will show your six-digit Maidenhead Grid Locator and the time, in UTC.

Figure 10 — When you change the frequency step size on the GPS version, the Grid Locator will be replaced by the step size on the display.