A Simple and Inexpensive Auto-Calibrating Multifunction Project (QEX July/August 2020)

This is a menu driven multifunction project that includes a band switched two channel 110 kHz – 112.5 MHz VFO, band switched 6 – 2200 m WSPR source, 0 – 6.5 MHz frequency counter, and a clock that displays time, date, and temperature. Frequency and time accuracy are maintained by a highly accurate temperature compensated DS3231 real time clock (RTC) board.

Gene Marcus W3PM GM4YRE

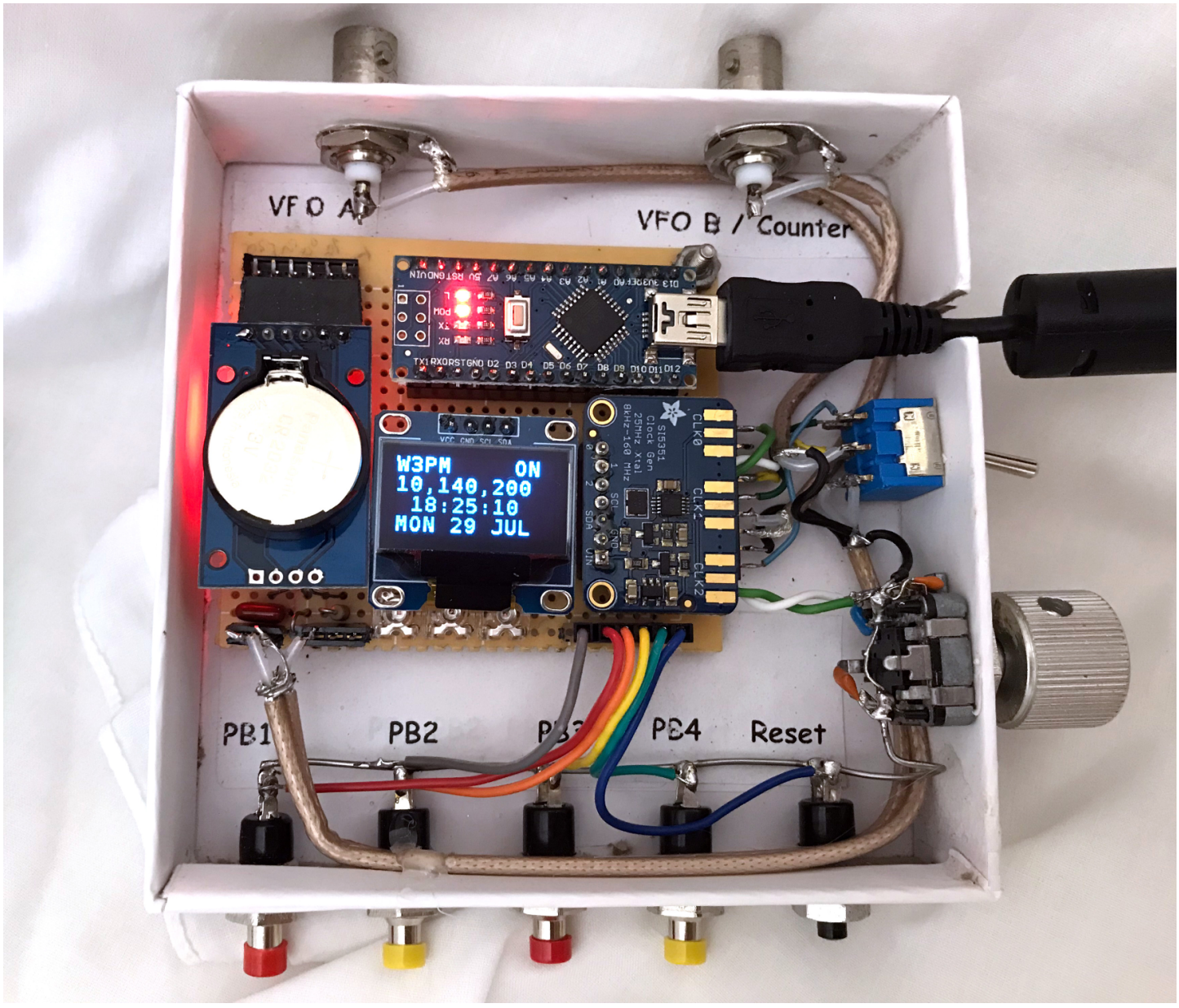

Figure_1 The completed unit with the WSPR function selected and band set to 30 meters. A discarded box from an Apple TV purchase was used to house the project. This image was taken with the clear plastic cover removed.

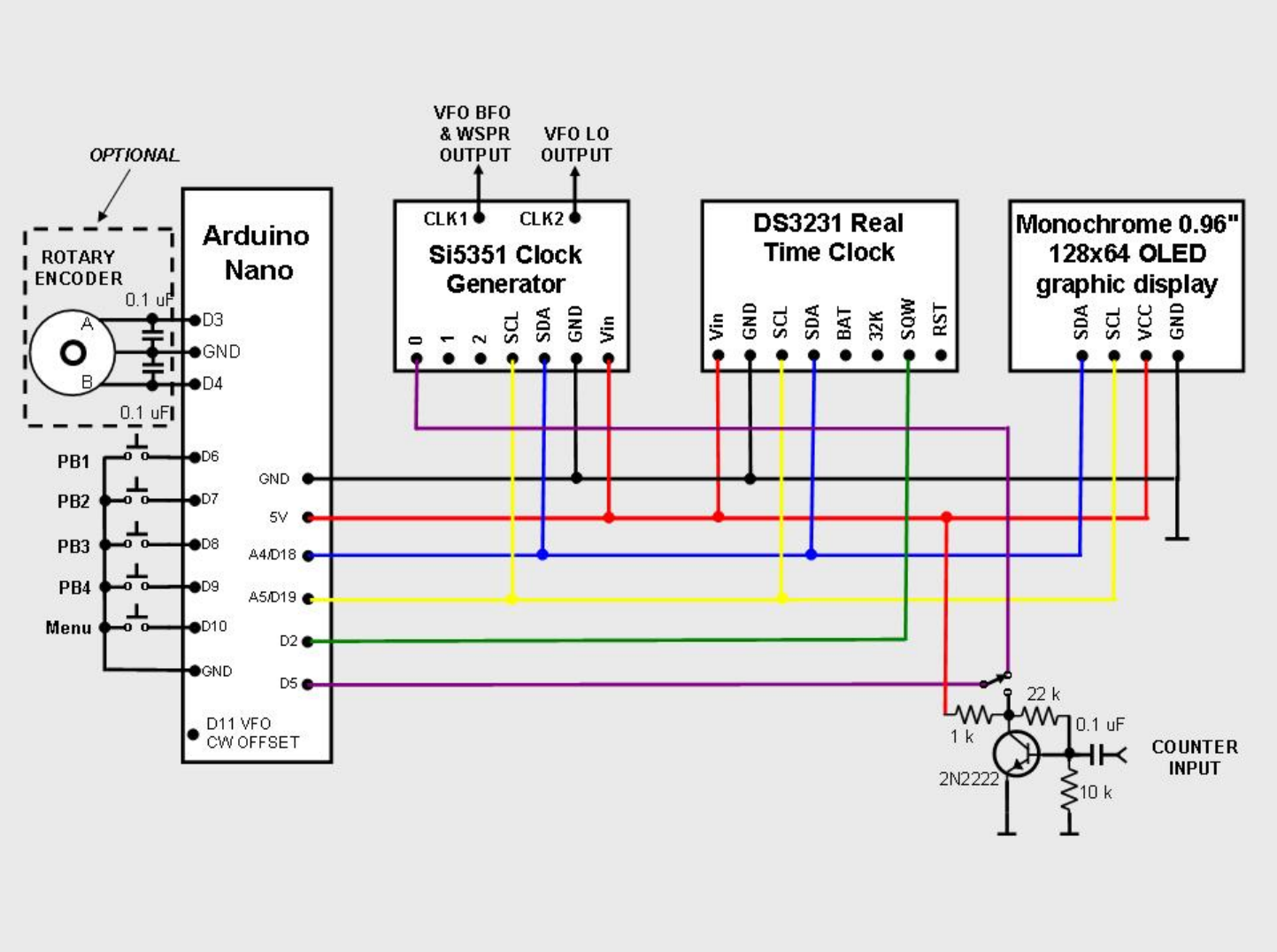

Figure_2 Schematic diagram showing optional rotary encoder and frequency counter preamplifier/buffer.

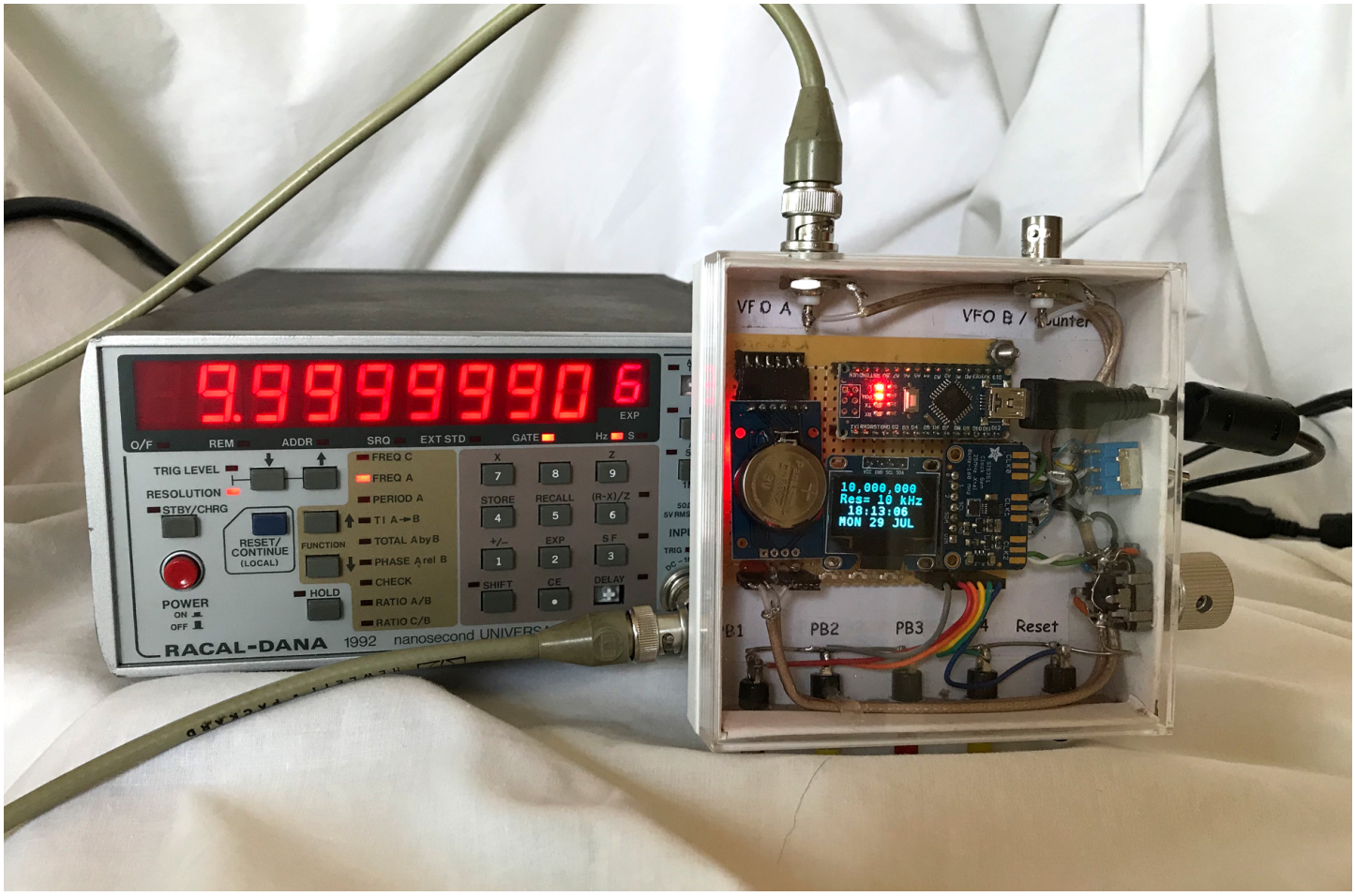

Figure_3 The completed unit with the VFO function selected and set to 10 MHz using 10 kHz resolution.

This project started with the purchase of a DS3231 real time clock (RTC) to use in a time critical project. As I became familiar with the functions of this device I realized that it had far more potential than only providing accurate time. I discovered that the internal temperature stabilized oscillator could be programmed to output a one pulse per second square wave with an uncertainty of about 2 parts per million from 0 – 40 degrees C (32 to 104 degrees F). This output could now be used as a frequency counter gate to automatically calibrate a Si5351 frequency source to provide accurate frequencies from 8 KHz to over 112 MHz. An auto-calibrated VFO was shortly in operation, then a WSPR source, followed by a very basic frequency counter, and of course a station clock with day-of-the-week, date, and temperature functions. All of these individual projects were then combined into one multi-function box controlled by single Arduino Nano (*or Uno*) using one Arduino sketch *(Arduino's term for the program or firmware*).

As shown in Figure 2 the building blocks for this project consist of four boards: an Arduino Nano (*or Uno*) microcontroller, a Si5351A clock generator breakout board, a DS3231 real time clock (RTC) board, and a 0.96 inch 128x64 serial I2C OLED display board.

The boards communicate with the Nano microcontroller over an I2C bus using only three wires. The boards were chosen because they are highly capable, inexpensive, well documented, and available over the internet from a wide variety of vendors. The completed unit draws little power and can be powered by the Nano microcontroller through the USB cable connected to a computer, a USB charger, or by some 5V USB battery backup chargers.

The Building Blocks

DS3231SN

Frequency and timing accuracy is provided by the DS3231SN RTC. Be sure your board uses the DS3231SN IC, some vendors substitute the DS3231M version which is not as accurate as the "N" version. The RTC is re-configured by the sketch for a 1 pulse per second output which is used as a counter gate to calibrate the Si5351A clock generator breakout board. The frequency calibration algorithm is a modified version of the one I used in my "An Arduino Controlled GPS Corrected VFO" article published in the July/August 2015 QEX magazine.

Si5351A

The Si5351A is a very popular clock generator used in many commercial and homebrew projects during the last few years. 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 7 dBm (5 milliwatt) 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.

Excellent library routines are available on the internet to simplify Si5351A frequency programming. Instead, I chose to program the Si5351A PL 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.

The Arduino sketch is developed for a Si5351A using a 25 MHz clock frequency. Minor sketch changes are required to accommodate other clock frequencies.

Arduino Nano (or Uno)

The Arduino Nano, as the name suggests is a compact, complete and user friendly microcontroller board. These boards are widely used in robotics, embedded systems, and electronic projects where automation is an essential part of the system. Based on the ATmega328P processor, it is quite powerful and easy to program using Arduino IDE software. All that is needed is a USB cable to transfer the program from your computer to the board.

OLED Display

The monochrome 0.96" 128x64 OLED graphic display is very small, but highly readable due to the high contrast OLED display. It uses no backlight and uses very little power. Although this is a graphic display, only text data is used in this application.

Options

An optional rotary encoder may also be connected to facilitate frequency changes for VFO operation. Simply wire the encoder as outlined in the schematic. Sketch changes are not required. I used a DigiKey P10859-ND mechanical rotor which is no longer available, however DigiKey does list PEC12R-4217F-N0024 as a substitute.

Construction

Construction of the unit is not critical provided adequate RF techniques are used. Do not use long unshielded wires for RF and 1pps connections. I first built the system using a solderless breadboard without any problems. The circuit was then transferred to a piece of perfboard and placed in a box with a clear plastic top that I happened to have on hand. Mounting the unit in a metal cabinet is the preferred way to house this project. To improve short and long term accuracy the unit should be kept at room temperature away from any drafts. Figure 1 shows my completed unit with the WSPR function selected and band set to 30 meters. A discarded box from

an Apple TV purchase was used to house the project. This image was taken with the clear plastic cover removed.

The Si5351A, DS3231SN, and the display are powered from the 5 volt pin of the Arduino. You can use the USB programming cable, a separate 5VDC source connected to the "+5V" pin, or 7-12VDC source connected to the Arduino Nano's "VIN" pin to power the unit.

Software Installation and Setup

The Arduino download website <http://arduino.cc/en/Main/Software> outlines installation instructions for the first-time Arduino user.

The Arduino sketch used with this project is located at: <www. knology.net/~gmarcus> and <https://github.com/w3pm/> . The file, current at publication time, is also available for download from the ARRLQEX files web page.

The sketch requires two open source libraries; "SSD1306Ascii" by Bill Greiman and "PinChangeInterrupt" by NicoHood. Both libraries are found in the Arduino IDE at; Sketch > Include Library > Manage Libraries. (Windows users will find the menu bar on top of the active window. Linux and MAC users will find the menu bar at the top of the primary display screen.) I found that other more robust ASCII/Graphics libraries were not compatible with the functions and timing complexity of the multifunction sketch.

In the unlikely event of operational problems, I suggest the builder check the Arduino, OLED display, and RTC boards individually to ensure proper operation. The Arduino board can be checked using some of the example sketches provided with the open source Arduino software. The simple "blink" example sketch will confirm that a sketch can be loaded and the Arduino board is functioning. The OLED display may be checked by running one of the AvrI2C examples found with the SSD1306Ascii library. The DS3231SN RTC board may be checked by running a time initialization sketch. A search of the internet will provide a number of methods to initialize the board. Adafruit Industries provides a very good tutorial to confirm the board is operational. Upon initialization, the time will be a few seconds slow. This is because the time set is the time the sketch was compiled, not the current time. If WSPR is to be used, the time will have to be updated using the "Clock Set" push buttons after the project is completed. Provided a battery is used with the DS3231, the date will not require setting for a year or more.

Prior to using WSPR, your callsign, grid square locator, power level, transmit offset frequency, and transmit interval will require customization within the sketch. The sketch is fully documented and configuration instructions are found in the WSPR configuration data section near the beginning of the sketch.

There are a number of pre-set frequencies divided into bands within the VFO configuration data section of the sketch. The displayed frequency is arithmetically corrected when both output ports are used in the VFO/LO configuration. Multiple bands may be configured in this manner. Instructions are found in the sketch if changes are required.

Operation

When the unit is turned on, the auto-calibration algorithm begins to calculate the correction factor for the Si5351A's 25 MHz clock. This takes 40 seconds to complete. The internal DS3231 temperature compensation algorithm concurrently calculates its correction factor every 64 seconds. The correction factors are continuously updated, therefore you may see small frequency jumps for the first few minutes until the unit stabilizes.

When the unit is turned on, the function selection menu is first displayed beginning with the WSPR function. Depress pushbutton 3 to scroll through the other functions. Depress pushbutton 2 to select the desired function.

Once selected, all functions are controlled by pushbuttons. An outline of pushbutton operation follows:

	WSPR	**VFO**	**Counter**	**Clock**	**Set Clock**	**Set Date**
PB1	N/A	Decrease Frequency	N/A	N/A	Time sync / Set Hour	Set Day
PB2	ON/OFF	Increase Frequency	N/A	N/A	Set Minute	Set Month
PB3	Band Select	Band Select	N/A	N/A	N/A	Set Year
PB4	N/A	Resolution Select	N/A	N/A	Hold to change time	Hold to change date

Depress the **MENU** pushbutton to exit and return to the original function selection.

WSPR

The WSPR output is a square wave, therefore, an external low pass filter for the desired band of operation is required to meet FCC regulations for spurious emissions. The WSPR function is designed to be used as a WSPR source using a small external amplifier and a low pass filter. However, operation at the unit's 5 milliwatt level using an external low pass filter and a good antenna will result in a surprising number of WSPR spots from a large geographical area depending upon the band used and band conditions.

Prior to WSPR operation, ensure that your callsign, grid square locator, and power level have been loaded into the sketch.

Be sure the clock setting is accurate to within about 1 second before transmitting WSPR. Select the desired band using Pushbutton 3. Band switching is the only way to change the frequency while in WSPR mode. The specific WSPR frequency within the 200 Hz window of each band is determined by the user configured variable "*TXoffse*t" within the sketch. Use Pushbutton 2 to turn the transmitter on or off. When the display is showing that the WSPR transmitter is "ON" a WSPR transmission will begin on an even minute after a few minutes of stabilization time and then repeat based upon the "*TXinterval*" schedule set within the sketch.

WSPR frequency accuracy is dependent upon the uncertainty of the DS3231 RTC which is about 2 parts per million between 0 and 40 degrees C. This will keep transmissions within the 200 Hz WSPR window provided the sketch's "*TXoffset*" is set to 1500 Hz which is the middle of the WSPR window. If you want to use the optional in-band frequency hopping, or want to transmit nearer to the edge of the WSPR window, a calibration crystal aging offset register adjustment is documented near the end of the sketch.

Short term frequency stability during WSPR transmissions is particularly important on bands above 20 meters. Short term drift stems from the Si5351 25 MHz oscillator and can be minimized by connecting a 50 ohm resistor or dummy load to the unused CLK2/VFO LO output during WSPR operation. Also, keep the unit enclosed and away from drafts.

VFO

The VFO frequency may be changed by using pushbutton 3 to select the band and/or by using the pushbuttons 1 or 2 to decrease and increase the frequency. An optional rotary encoder may also be used. The resolution selection button (pushbutton 4) is used to change the frequency resolution in steps of 1 Hz, 10 Hz, 100 Hz, 1 kHz, 10 kHz, 100 kHz, and 1 MHz steps. Both BFO and LO outputs are available in addition to a CW offset. Instructions to implement these functions are documented within the sketch. The VFO power output is approximately 7 dBm (5 milliwatts) depending upon the selected band. The frequency accuracy is dependent upon the uncertainty of the DS3231 RTC which is about 2 parts per million between 0 and 40 degrees C. The accuracy can be improved by adjusting the crystal aging offset of the DS3231 RTC. This is explained in the calibration documentation found near the end of the sketch.

Frequency Counter

After selecting the frequency counter function, change the position of the switch in Figure 2 to connect the output of the counter amplifier to pin 5 of the Arduino Nano. Be sure to return the switch to its normal position after using the frequency counter function.

The frequency counter is very basic with a bandwidth of approximately 6.5 MHz and an accuracy of about 3-4 parts per million. The uncertainty is less than the VFO and WSPR because the unit uses a one second gate time derived from the DS3231 RTC that is subject to some jitter. The gate time may be extended by changing the *gateTime* variable found in the sketch. A basic 2N2222 NPN transistor amplifier is used for the counter input. The input sensitivity is about -20 dBm (0.63 V p-p) throughout the frequency range. The maximum input voltage is about 5 V p-p.

Clock

The clock displays hour, minutes and seconds and callsign. Every 10 seconds the display will alternate between the date (day of the week, date, and month) and temperature (degrees F and C). The DS3231 measures its internal temperature which will always be higher than the ambient temperature. A correction factor is used by the sketch to account for the difference, but do not expect the temperature readings to be very accurate.

Set Clock

Setting the correct time within +/- 1 second is critical for WSPR operation. The time is set by holding down pushbutton 4 while depressing pushbutton 1 to set hours or pushbutton 2 to set minutes. At the top of the minute release pushbutton 4.

Provided the displayed time is correct to within +/- 30 seconds a shortcut to synchronize the time is available. Simply depress pushbutton 1 at the top of the minute to synchronize the time.

Set Date

The date is set by holding down pushbutton 4 while depressing pushbutton 1 to set the date, pushbutton 2 to set the month, or pushbutton 3 to set the year. Release pushbutton 4 to save the changes.

DS3231 Aging Offset Adjustment

While the default RTC is already very accurate, its accuracy can be pushed even higher by adjusting its aging offset register. The documentation located at the end of the sketch provides methods to determine the correction factor and upload it to the aging offset register. Figure 3 shows the unit set to 10 MHz and connected to a frequency counter after the aging offset was adjusted.

For normal operation this adjustment is not required. The default 2 parts per million accuracy of the RTC will result in an uncertainty of less than +/- 30 Hz on 20 meters.

If WSPR is used, the time will require synchronization every 7 – 10 days without offset adjustment. The re-synchronization timeframe can be stretched to a month or more with proper adjustment.

Experimentation

This is an open source project. You are invited to experiment and improve upon system operation. Relatively simple system improvements can be made. For example, you can substitute a 1 pulse per second (pps) output

from a GPS unit for the 1 pps output from the DS3231 RTC board to further increase frequency (but not clock) accuracy.

Acknowledgement

I wish to thank Ronald Taylor, WA7GIL, for his suggestions to improve this project.

Gene Marcus W3PM GM4YRE
113 Wickerberry Lane
Madison, AL 35756
w3pm(at)arrl(dot)net