VU3GAO Easy BITX Firmware

User’s Manual 40m Band

October 2021

Nitin William, VU3GAO

User’s Manual

This is a work in progress and the software is designed to be modified. The Operation and LCD screen appearance may change.

This firmware was designed to work with the stock Easy BITX Version 2.0.

Since this software is designed to work with the stock hardware of Easy BITX V2, all the hardware operating concerns apply.

Before transmitting be sure that the SWR is low enough to prevent any damage to the final transistor. As the radio produces RF energy, practice RF safety.

User I/O

Inputs consists of the Rotary Encoder with Push Button, PTT, MIC, Power Supply and Mini USB Serial connector on Arduino Nano module behind the DDS Control board.

Output consists of the LCD Screen, the Speaker, Antenna connector and Mini USB Serial connector on Arduino Nano module behind the DDS Control board.

The software distinguishes between a short press and long (2 Second) press of Rotary Encoder button. If the encoder button is pressed with short click, it will change the tuning steps. A long press of this button will put the set in BFO tunning mode.

Initial Si5351 Calibration

The very first step after the kit has been assembled and wired as per Easy Bitx building manual is the calibration of Si5351. Note that it is assumed that the builder has fundamental knowledge of Arduino Nano as microcontroller, its basic programming and use of Arduino IDE along with knowledge of Serial Communication link.

Connect the Arduino Nano module of the Easy Bitx DDS Board with mini USB cable to your PC with Windows and upload the “si5351_calibration” program. Follow the given instructions within the program to determine the Si5351 Calibration value and note it.

Open the Easy Bitx firmware in the IDE and enter the Calibration value in the " #define ddsCal" within the code as shown below in yellow high highlight.

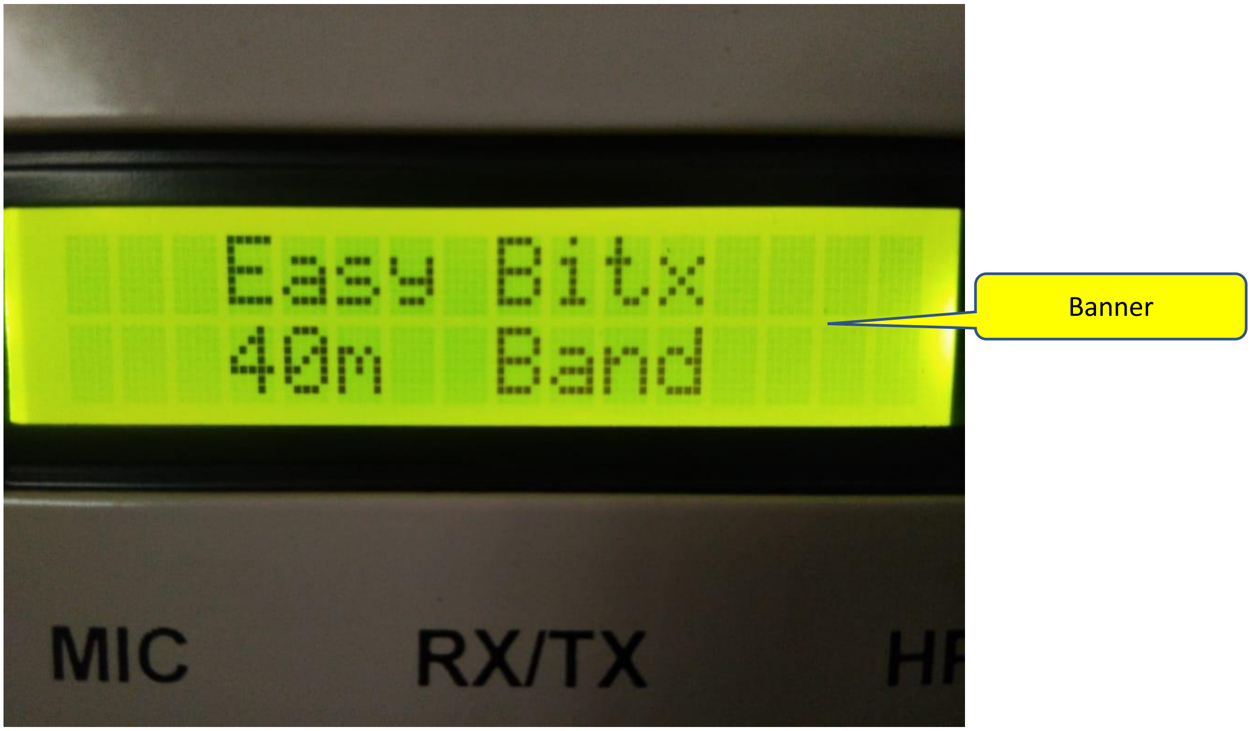
```
#include <Wire.h>               //IDE Standard
#include <Rotary.h>             //Ben Buxton https://github.com/brianlow/Rotary
#include <si5351.h>             //Etherkit https://github.com/etherkit/Si5351Arduino
#include <EasyButton.h>         //https://easybtn.earias.me/docs/introduction
#include <LiquidCrystal.h>

#define fMax  7300000UL
#define fMin  7000000UL
#define ddsCal 359000           //Enter the Calibration value here.
#define LCD_RS  5
#define LCD_E   6
#define LCD_D4  7
#define LCD_D5  8
#define LCD_D6  9
#define LCD_D7  10
#define encoderButtonPin 11
```

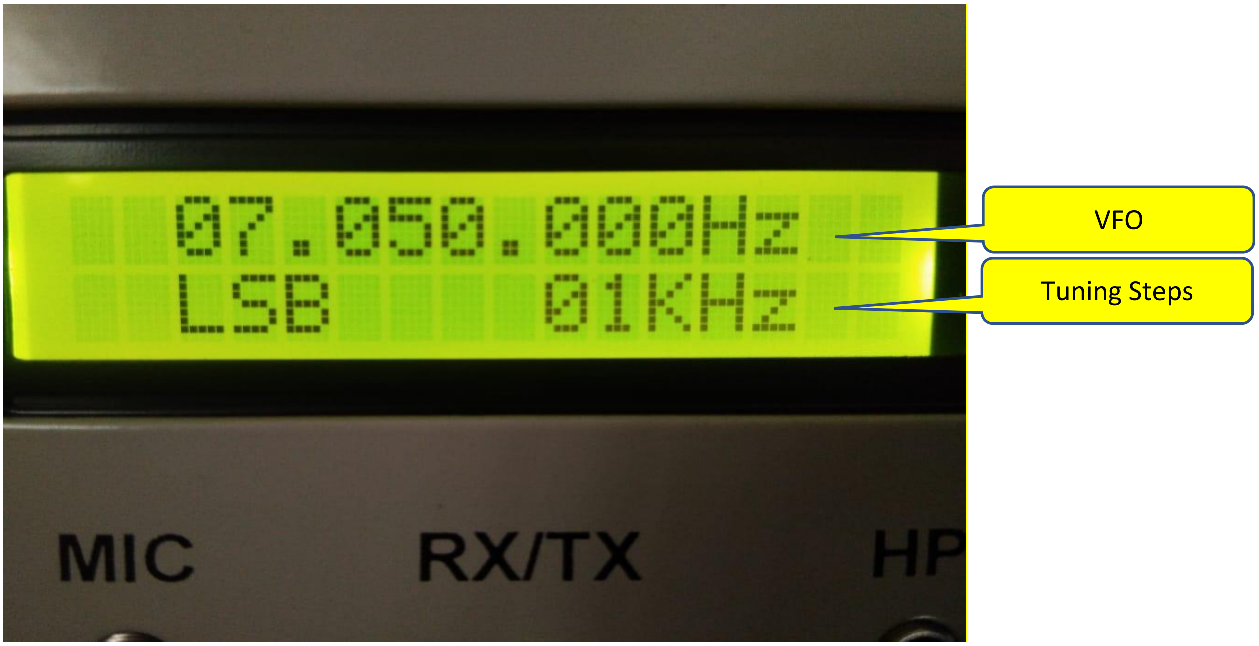
Ensure that you have already added the above libraries in your Arduino IDE as shown in grey high light.

First Power On

The set will switch on with the below banner on the LCD for 3 seconds.



After the banner has been displayed, the set will go into operation mode with initial frequency of 7050 Mhz as displayed on the LCD screen below.



VFO

Tuning Steps

Operation

The set tunes from 7000 to 7300 Mhz on the 40m Amateur Band. Tuning will be done by the rotary encode or the tuning knob.

Tuning steps can be changed by short press of the tunning knob. Frequency steps will change with each press of the button with the sequence as 1Hz, 10Hz, 100Hz,1Khz and 10Khz.
Note that the tuning knob and step button as also active during Transmission with the press of the PTT.

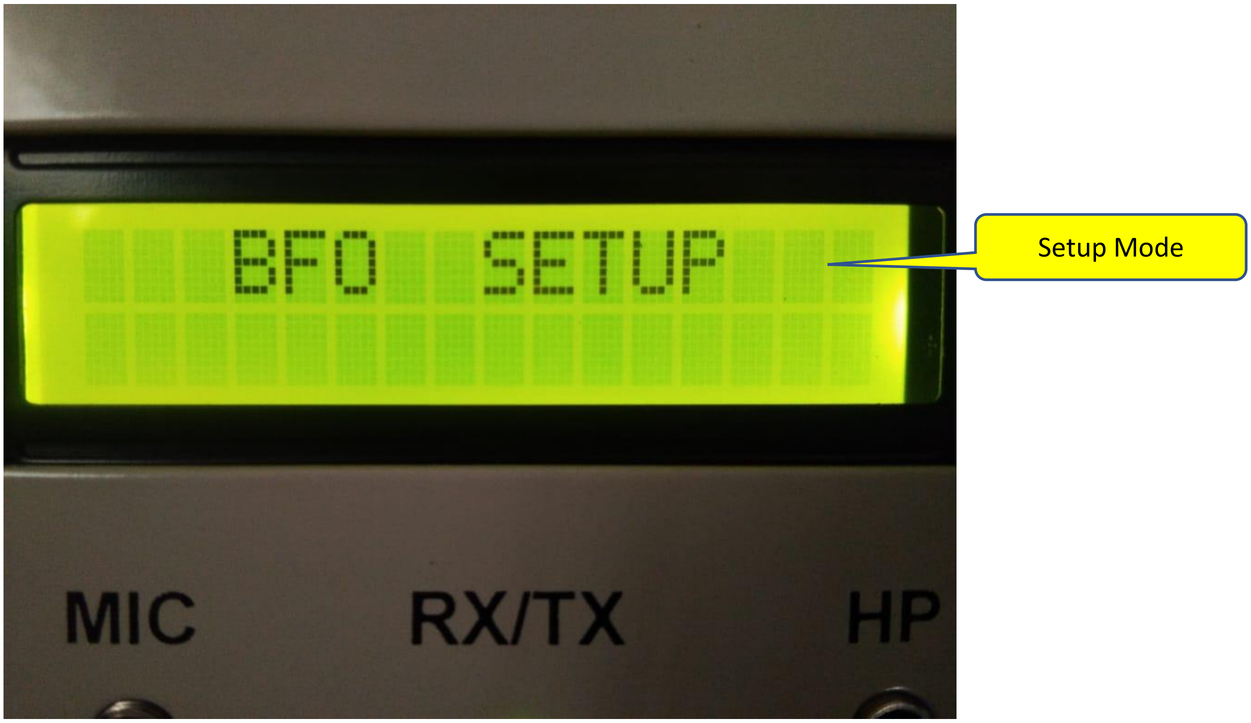
BFO tunning

The start up BFO value is hardcoded in the code as “9996800”. Every set would have a different pass band of the Xtal Filter in the circuit and would require **Critical tuning** of the BFO.

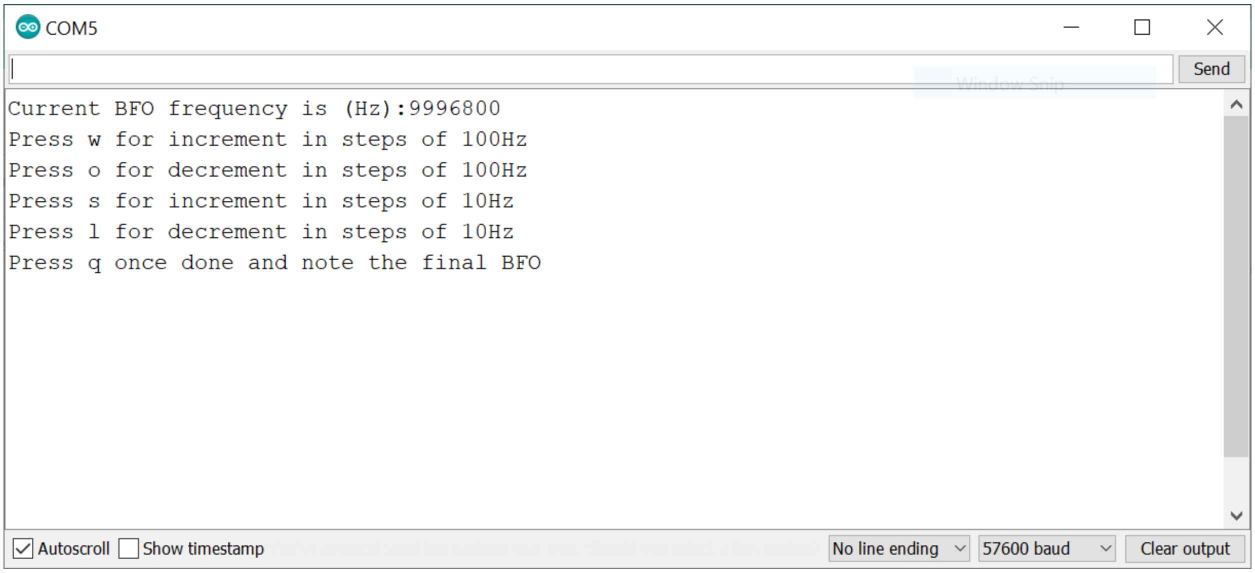
The firmware works with high side LO injection because of which the BFO frequency is chosen as USB side on the BFO/Productor detector side. As the VFO will be operating at 17Mhz (high side), the difference of VFO – RF will provide IF with inverted side band as LSB.

To get the BFO tuning right for your Xtal filter, we first need to connect the Arduino Nano module in the set via mini-USB cable to the PC. Launch the Arduino IDE and open the Serial monitor. Make sure the set is powered on.

Now put the set in “BFO Setup” mode by pressing and holding the tunning knob for 2 seconds. The display on the LCD will change to BFO Setup mode as below.



The set is now operating with last tunned VFO and current BFO value and the Serial monitor of Arduino IDE with show below setup screen.



```
Current BFO frequency is (Hz):9996800
Press w for increment in steps of 100Hz
Press o for decrement in steps of 100Hz
Press s for increment in steps of 10Hz
Press l for decrement in steps of 10Hz
Press q once done and note the final BFO
```

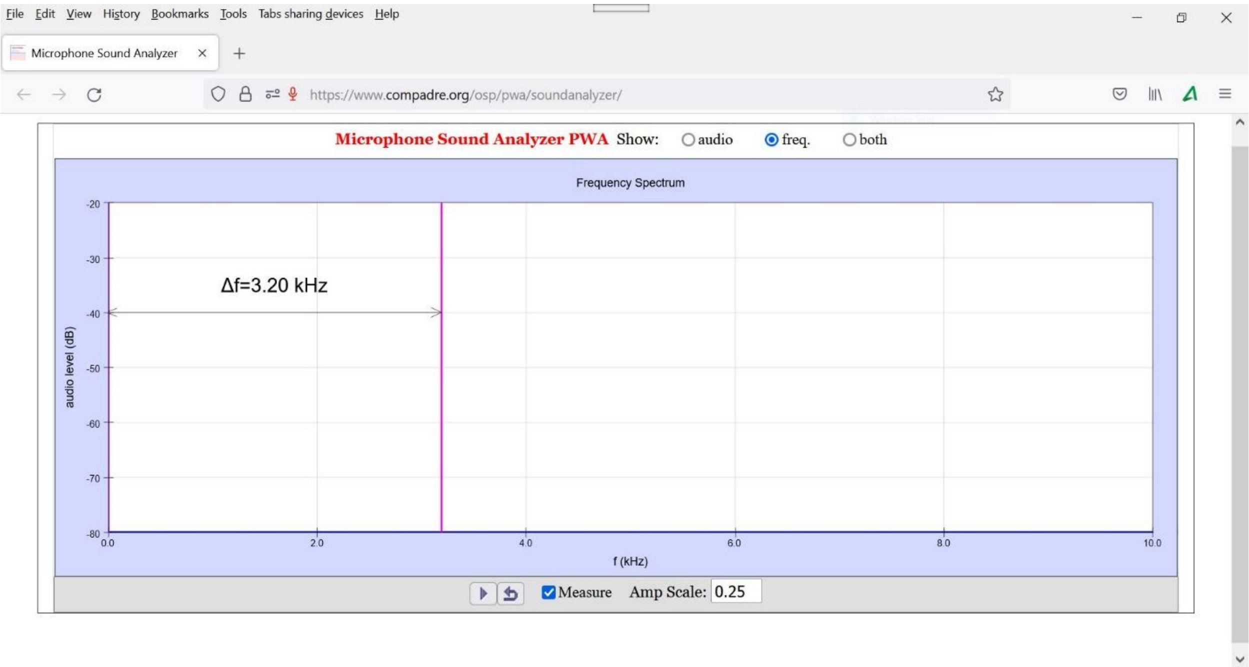
☑ Autoscroll □ Show timestamp No line ending 57600 baud

Follow the instructions till you get the right BFO frequency for your set. You may use any Audio spectrum analyzer to ensure the pass band noise of the set is within 300Hz to 3200Hz.

I suggest use of Audio Spectrum Analyzer. On your Laptop/Computer, open below link.

https://www.compadre.org/osp/pwa/soundanalyzer/

A page as shown below will open.

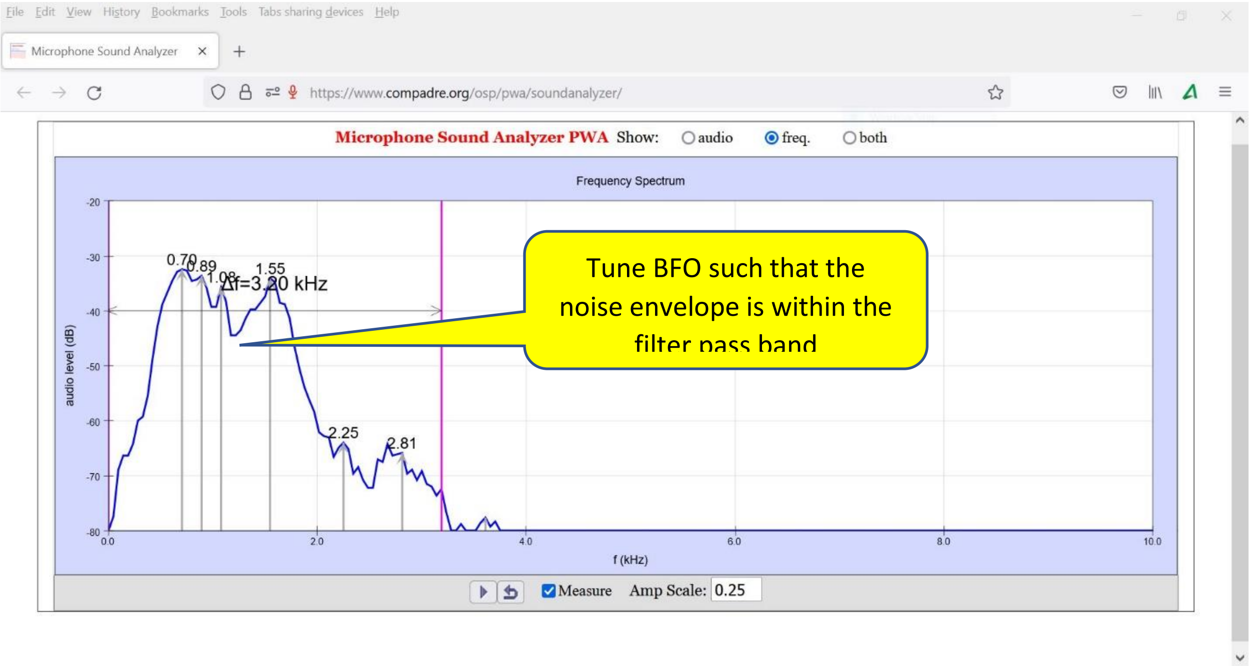

File Edit View History Bookmarks Tools Tabs sharing devices Help
Microphone Sound Analyzer
https://www.compadre.org/osp/pwa/soundanalyzer/
Microphone Sound Analyzer PWA
Frequency Spectrum
audio level (dB)
-20
-30
-40
-50
-60
-70
-80
0.0
2.0
4.0
6.0
8.0
10.0
f (kHz)
Δf=3.20 kHz

Show: ○ audio ● freq. ○ both

☑ Measure Amp Scale: 0.25

Select “freq” radio button in this window and click the “Measure” check box. Now click the play button.

Switch on your Radio set and connect the antenna. Tune the set to a silent frequency and place the computer’s microphone close to the speaker of the Radio set. The analyzer will pick up the noise and will show the spectrum on the screen.


File Edit View History Bookmarks Tools Tabs sharing devices Help
Microphone Sound Analyzer
https://www.compadre.org/osp/pwa/soundanalyzer/
Microphone Sound Analyzer PWA
Frequency Spectrum
audio level (dB)
-20
-30
-40
-50
-60
-70
-80
0.0
2.0
4.0
6.0
8.0
10.0
f (kHz)
0.70
0.89
1.08
1.55
Δf=3.20 kHz
2.25
2.81

Show: ○ audio ● freq. ○ both

☑ Measure Amp Scale: 0.25

Tune BFO such that the noise envelope is within the filter pass band

Tune the BFO such that the max audio amplitude falls within 300Hz – 3200Hz filter band width.

Please note the final BFO frequency given in the Serial Monitor screen after you have pressed the “q” in the Serial monitor screen of the calibration program.
Enter this frequency value of BFO in the code as below for hard coding. Note that for we have not used the EEPROM in Nano to burn this value for simplicity reasons.

```
--------------------------------------------------------------------------------------------------------------------------------------
#define LCD_D7  10
#define encoderButtonPin 11

unsigned long vfo = 7050000UL;            //Enter your initial frequency at startup, ex: 7000000 = 7MHz,
10000000 = 10MHz
unsigned long bfo = 9996800UL;            //BFO frequency for 10Mhz filter Bw=
unsigned long opsfreq = 0UL;              //Operating freq to be generated by Si5351
unsigned long offset = 3200UL;            //IF Offset
```

Save the firmware with this value, compile and upload the code to the Arduino Nano.

Power cycle the set with the mini-USB cable disconnected to avoid PC powering the control board.

Enjoy!

73s,
Nitin William

Download the firmware from the documents section of the Amateur Radio Kits Web site.

https://amateurradiokits.in/manuals-and-docs/

Note:
The firmware can be adapted for other bands for which below parameters in the code needs to be modified.
For example, if you want to use this firmware for 20m Band USB, update following highlighted code in the main EasyBitx_v1.x.ino file:

```
--------------------------------------------------------------------------------------------------------------------------------------
//Libraries
#include <Wire.h>              //IDE Standard
#include <Rotary.h>             //Ben Buxton https://github.com/brianlow/Rotary
#include <si5351.h>             //Etherkit https://github.com/etherkit/Si5351Arduino
#include <EasyButton.h>          //https://easybtn.earias.me/docs/introduction
#include <LiquidCrystal.h>

#define fMax  14350000UL
#define fMin  14000000UL

--------------------------------------------------------------------------------------------------------------------------------------
```

And change the following for Low side injection for USB 14Mhz as shown below,

----

```
void tunegen()
{
  opsfreq =  vfo - bfo;                              //High side LO injection, Side band inversion
  si5351.set_freq(opsfreq * 100,SI5351_CLK0);         //Update operating frequency
}
```
