uBTIX Settings Editor User Manual

A Guide to getting the most out of your uBITX running the CEC Software

Mark Hatch (AJ6CU)
6-2-2023

Revision History

June 2: Updated manual to reflect changes from Beta1 to RC1. Installs have changed significantly.

March xx, 2023: Added note on SUDO for Linux, Corrected MacOS install recognizing that the package is now signed.

March 4, 2023: Initial Release

Preface

I had not originally planned to write the uBITX Settings Editor. My big plan was to first port the KD8CEC software from its lowly home on the Arduino Nano to faster and more modern processors. My hope was that by breaking free of the constraints of the Nano, I would open the door to not only a much-needed restructuring of the code, but also make it possible for more contributors to extend its functionality.

Well, that was a good goal…

But as I was in the middle of porting the KD8CEC software to the Arduino BLE (I already had it working on the Arduino IOT), I discovered that the original uBITX Memory Manager refused to talk to the uBITX when I had an Arduino BLE installed. Initially, I assumed it was my issue and kept going. But then I found that the uBITX Memory Manager didn't like the Arduino RP Connect (a Raspberry Pi Pico in an Arduino package) either. And then, I found it liked the Teensy... So, what was going on here?

Unfortunately, Dr. Lee never released the source code to the uBITX Memory Manager. So I couldn't fix the problem. And if I didn't have a working uBITX Memory Manager, then no one would want to use KD8CEC running on a Raspberry Pi Pico or any other processor. And there was a real possibility that as the KD8CEC software evolved the two pieces of software would increasingly become unwilling to communicate.

Consequently, I was left with no choice but to rewrite the uBITX Memory Manager.

Initially, I tried to constrain the problem. Within about a month of work, I had two programs that could read/write to the EEProm with a save format of a human readable/editable XML file. However, after making some stupid typo errors while I was editing my own XML files, I concluded that this was not a satisfactory solution.

So, the scope of the problem increased to embrace a more human friendly application with a graphical user interface. This increased the scope by multiple factors. Not only is designing a GUI hard (both in design and coding), it also means understanding what was really going on at a much deeper level than just retrieving and storing bytes. I had to understand what every byte meant to present it to the user.

So, 8 months later…

I am distributing a "Release Candidate" – RC1. I believe will be actual release of the uBITX Settings Editor. I believe it is a great leap forward. Not only is it open source, a better organized GUI, but it is also now available on macOS and Linux (as well as Windows).

I hope you like it too and I look forward to your feedback!

73

Mark (AJ6CU), June 2, 2023

Introduction

The uBITX is manufactured by HF Signals (https://www.hfsignals.com/) using an Arduino Nano V3 MCU. This Nano is the heart of the uBITX. It has 32kbytes of flash memory and 1024 bytes of Electrically Erasable Programmable Read-Only Memory (EEPROM). The flash memory holds the firmware or software that runs the uBITX and the EEPROM is used to store settings (e.g., calibration information, last used frequencies and modes, tuning rates, etc.).

Although functional, the original firmware supplied by HF Signals was basic and the plan all along was for other Hams to take the lead to extend it. As a result, Dr. Ian Lee (KD8CEC) developed an enhanced version of the firmware (we will refer to it as CEC throughout this manual to avoid confusing the software with the person (Dr. Lee KD8CEC)) that extended the original LCD offering and in later releases, provided a graphical user interface based on Nextion touchscreens. His efforts are documented in a blog format at www.hamskey.com.

The "eye candy" and enhanced functionality of the CEC software is what first catches the attention of most Hams. However, what many miss is that the CEC software is of limited value without a tool to easily tailor it to the user's needs. Sure, you can set the CW Keyer speed through the GUI (you will sometimes see me equivalently referring to this as the "UX" or User Experience), but suppose you need to:

- tune when your dot or dash is recognized as a dot or dash respectively.
- to use the automated keyer that is part of CEC, and you want to define the "canned message" (i.e., CQ CQ DE…).
- use the uBITX as a WSPR beacon.
- to tell your external linear amp the band being used so it can select the right bandpass filter.

Or you get the idea… You can't practically do this on the radio. That is where you need a tool to tailor the settings of the uBITX to meet your needs.

Originally, Dr. Lee wrote the uBITX Memory Manager (available at: https://github.COM/phdlee/ubitx) and kept it updated as he added new functionality. However, for reasons that I discussed in the Preface, that code, like all code does eventually, suffers from "virtual rot" and it was time for a rewrite.

This document is the User Manual for the rewrite of this software. I have attached the name "uBITX Settings Editor' (lets refer to it as the "*SE*" from now on) to it because that seemed to provide a clearer communication on its purpose.

The Plan for our Journey

The next section will give you a basic orientation to the software and suggest a process on how you should use it. The software makes heavy usage of tooltips and provides larger hints adjacent to key functionality. I suspect that after this basic overview, you will be able to jump in and just use the software.

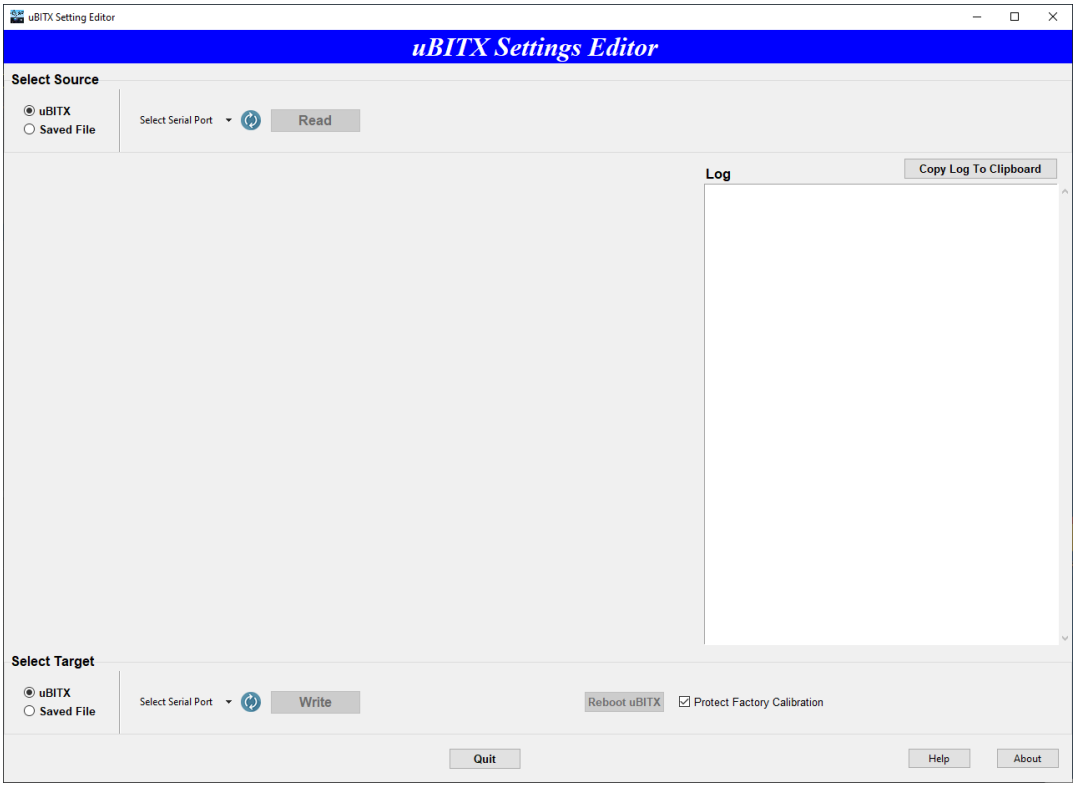
Upon completing the overview, we will take a slight detour and talk about how to install this software on Windows, macOS and Linux. Unlike the original Beta, the software packages for both Windows and macOS have been signed and verified by the respective OS provider. Although this reduces "false positives", you should be warned that many Virus checkers use the concept of "Reputation". In short,

software that has been downloaded and installed by only a small number of people is a suspect and often ends up being quarantined. This section will identify the more obvious points where this virus checker intervention happens and suggest work arounds.

The final section, and the actual bulk of this manual, will focus on each basic area of functionality in detail.

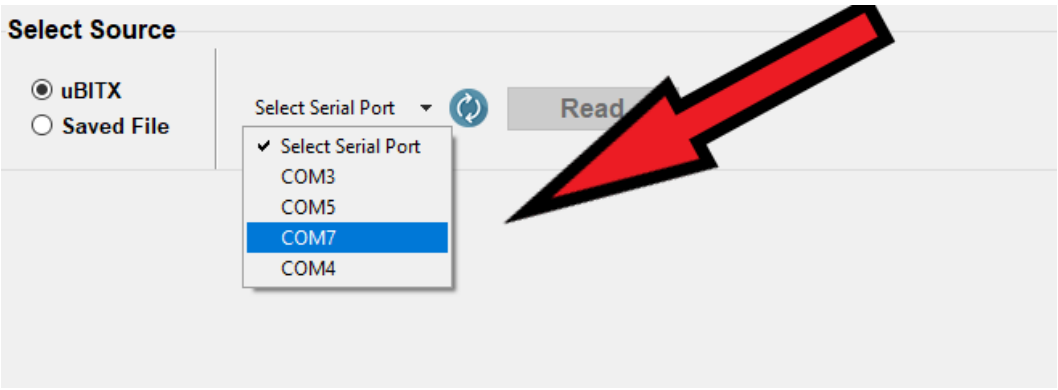
Quick Overview of the Settings Editor

After you start the SE, you will be greeted by a screen that looks like:



A lot of empty space, right? Lets look at the top left section.

This is where you select the source for all your uBITX settings. The most common choice is that you will read it from the uBITX itself. And that is the default as the uBITX “button” is selected. Assuming that is what you want to do, then you need to select the COM port that is linked to your radio like in the following screenshot:

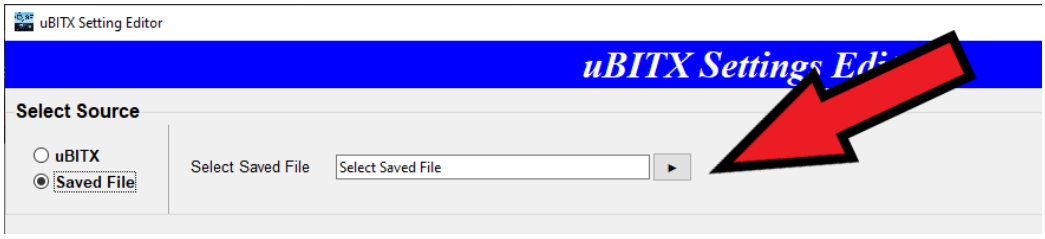


The (ugly) red arrow points out that the COM port listing box was clicked and that COM7 (in this case) was selected. Once the COM port is selected the READ button on the far right becomes active and you can read the settings into the *SE*.

I suspect the hardest part of this step is not the selection of the COM port, but trying to figure out **which COM port** corresponds to your radio. One approach is that you can start the SE *without* your Raduino plugged in. Note the existing COM ports in the drop down and then plug in your radio and hit the refresh button. The new port that appears is your radio!

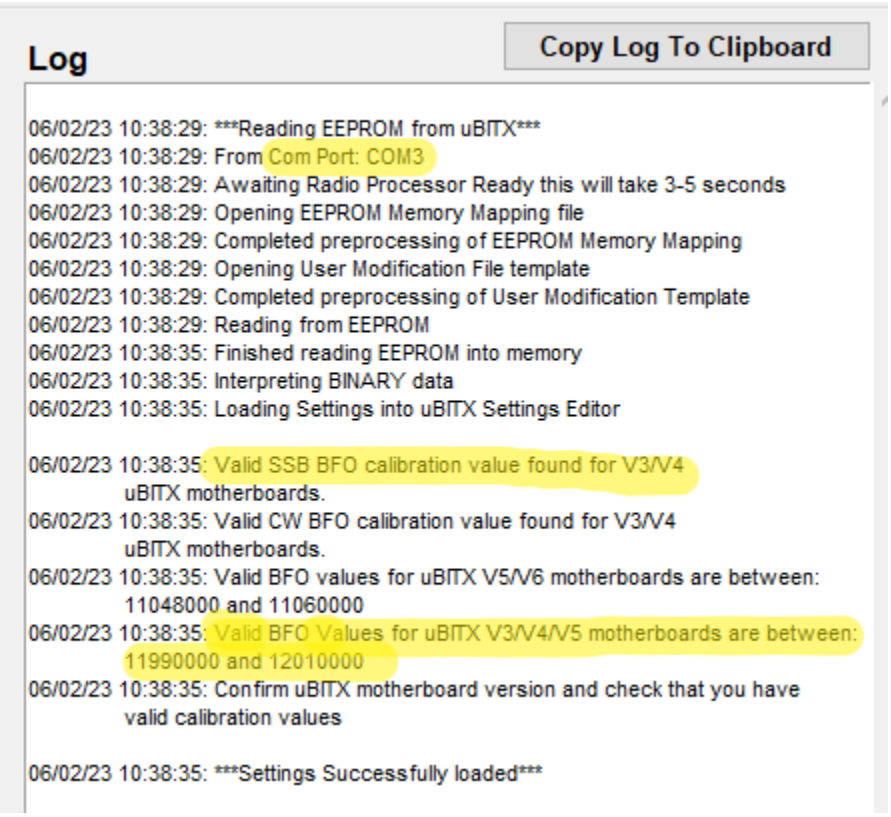
So why might you want to read your input from a file instead of the radio directly? Well, you really should backup your uBITX settings, right? If you do this, and in the future, you have to replace your processor (or upgrade to a new one) you can use this backup to restore your settings.

The screen below shows what happens when you select “Saved File”. The COM port selection portion of the application is replaced by a file selector box. Clicking on that small arrow (pointed to by the red arrow) brings up a standard file selector box that you can use to locate your backup file.



So again, like in the COM port section, selecting a file will allow you to click the READ button on the right.

As the *SE* reads in your settings, you will start to see action in the area called the “Log” that is on the right side of the *SE*. For example, the following Log snippet was the result of reading the settings from a uBITX:

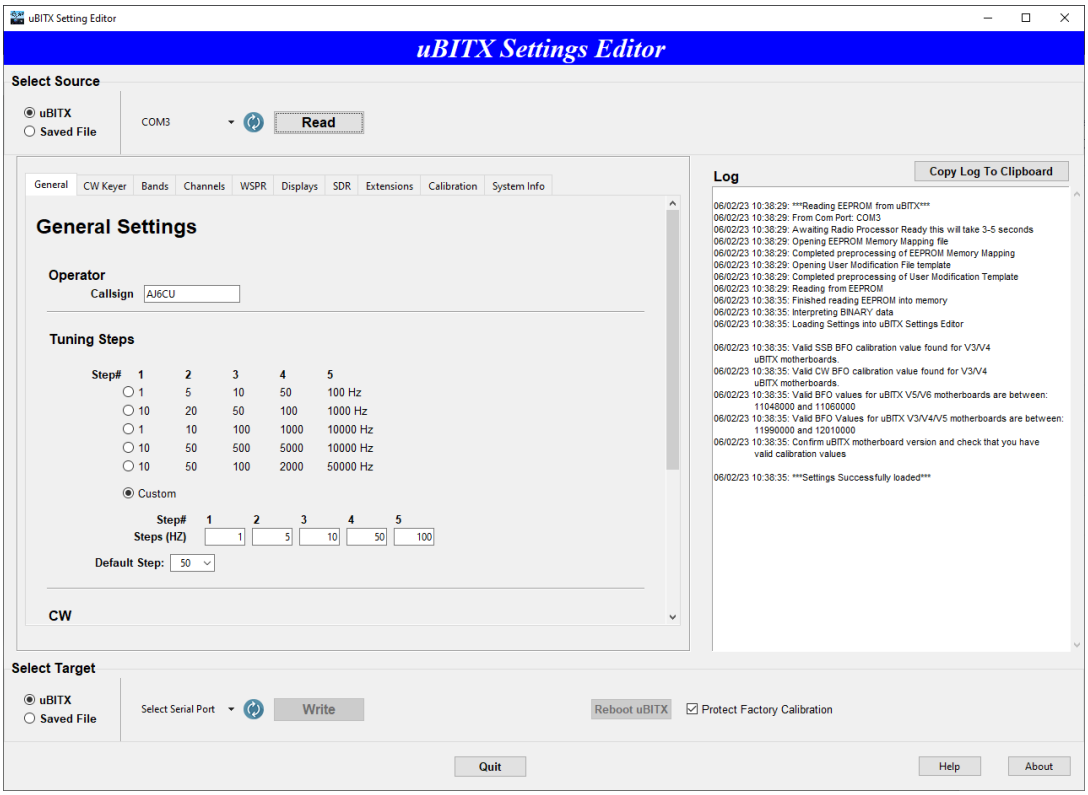


In this case, the second line of the log identifies the COM port where radio was connected (perhaps useful in the future). If you tried to enter data for a setting that was outside the legal bounds, a warning will appear here in red.

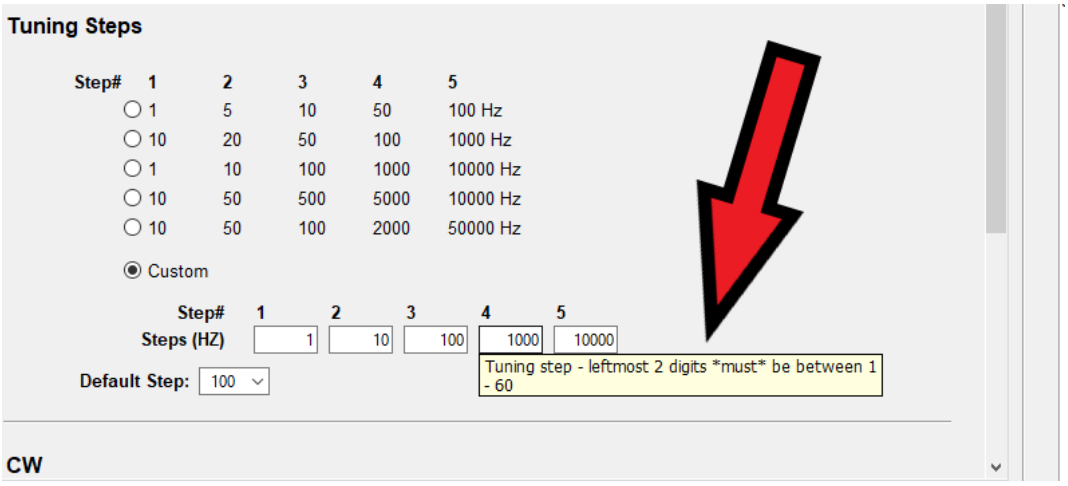
Note the highlighted messages in the middle and towards the bottom which are talking about your calibration values. This is a new sanity check that was added in this release to confirm that the calibration values for the SSB and CW BFO were within the valid range. In this case, the SE is reporting that the CW and SSB BFO's are within a valid range for a V3/V4 uBITX mother board. If you are running this on a V6, then these values are WRONG! This is a very common occurrence when a user makes a mistake and downloads the HEX file that does not match his uBITX motherboard.

Don't miss that button on the top right: "Copy Log To Clipboard". If there is ever a problem and you want to help me find the problem, I will need this log. Click the button and paste it into an email.

Now for the fun part, you have selected your input source, clicked READ, and now you have lots of information!



The large center blank area is now filled with a set of tabs that group similar settings together. In this case, the “General Settings” tab is active. Here you can set your callsign, adjust your default tuning step rate, and just turn off the screen, set your CW settings. (Remember the tooltips!)



So eventually, you have tweaked every setting, and you want to save your work. Saving is just a reverse of the reading process. The box at the bottom of the SE is your next destination to save your work.

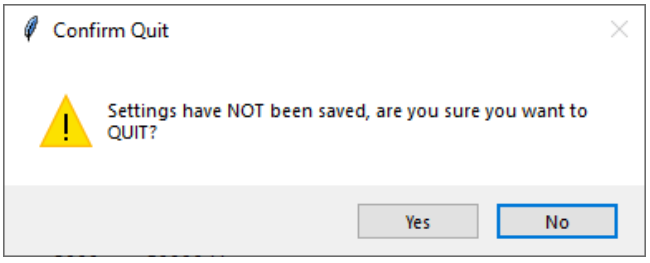
In the screenshot, the red arrow points to the selection where you would select your Com port and click Write. *Tip: If you want to write to the same Com port you read from, just click the "Refresh" button and the correct Com port will be selected!*

Generally, if you read your settings from a uBITX, you will just write them back to it. However, the separation of the input and output ports does allow you the possibility to read from one uBITX and write to another uBITX!

CAUTION: Note the highlighted section on the right. The default is to never overwrite the "Factory Settings" for the calibration values. This provides a way for you to do a "Factory Reset" in case you really mess up. Generally, you want to keep this box checked. However, if you were replacing a Nano with another processor, you might want to uncheck this box to allow the full contents of the EEPROM to be written to the new processor/board.

Tip: Notice that "Reboot uBITX" button to the left of the Protect Factory Calibration checkbox? This allows you to force your uBITX to reboot. You can then re-read the settings using the top panel and not have to unplug, power cycle, plug in your USB to get the radio working on the new settings.

And while you are at it, you really should write your settings to a file so they are available to you in case of a future problem. Just select "Saved File", click the arrow to bring up the file selection box, navigate where you want to save it. **Remember to CLICK WRITE TO ACTUALLY SAVE THE SETTING TO THE FILE.** If you forget to click WRITE, you will be greeted with the following message when you try to exit:



Hopefully, this overview has convinced you that this software isn't hard to use and that it might even be useful. Before taking a deep dive into customizing the settings of each tab, let's get it installed on your machine!

Installation

This software was written using Python 3 and the Pygubu Designer for the Tkinter GUI toolkit. Without these powerful tools, it would have taken much longer to complete. Furthermore, these tools made it trivial to support new platforms and operating environment. At this point, the following installation packages available:

- Windows 10 (x86 64 bit)
- macOS (Intel) (x86 64 bit)
- Linux (Raspberry Pi ARM and Intel)
- Source distribution for DYI builds

NOTE: THE WINDOWS AND MACOS RELEASES HAVE UNDERGONE A PROCESS KNOWN AS "SIGNING". THIS SHOULD SIMPLIFY THEIR INSTALLATION. HOWEVER, BROWSERS AND VIRUS CHECKERS MIGHT STILL ATTEMPT INTERVENTIONS AND QUARANTINE THE SOFTWARE JUST BECAUSE OF THE FEW TOTAL NUMBER OF DOWNLOADS. THIS MAY REQUIRE YOU TO OVERRIDE THEIR "HELP."

The download page for these software packages is at:

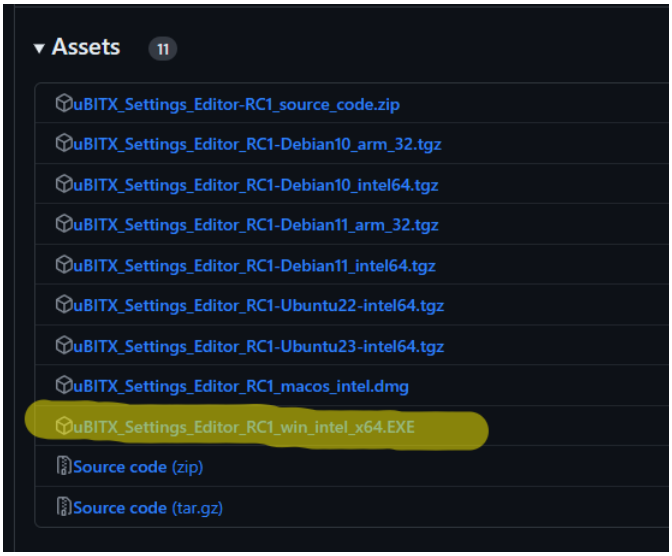
https://github.com/AJ6CU/uBITX-Settings-Editor/releases/tag/V2.0RC1

Scroll to the bottom of this page looking for the section labeled: "Assets"

The asset itemized as "uBITX_Settings_Editor_User_Manul.pdf" is this document. The names of the other files should be sufficiently descriptive to let you select the correct package.

Windows 10 (x86 64 bit)

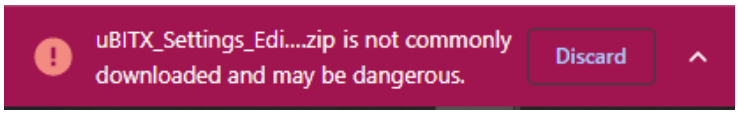
The Windows package .exe file contains an installer. Once you open it, you will be guided to save the SE application. I have a "bin" subdirectory in my Documents folder where I put such files. But you may want to just download it to your desktop to make it convenient. Once extracted, you will find a standalone, one file executable that you can store it anywhere and just double click it to start.



Installation process is:

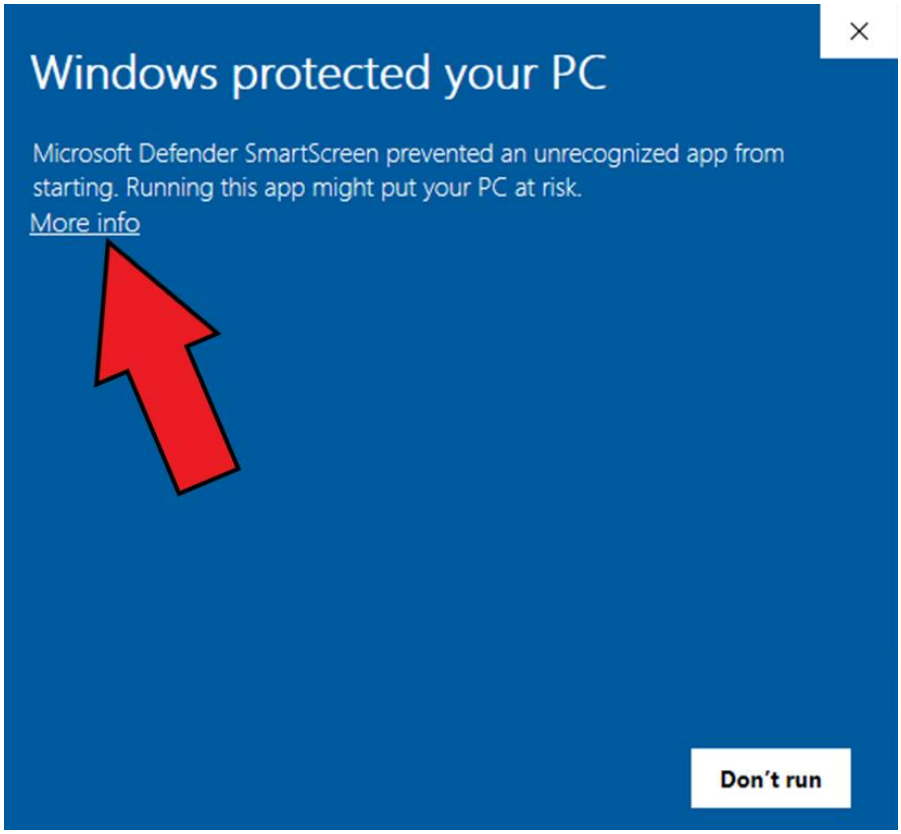
1. Download the highlighted file in the above screenshot.

   Your Browser may try to protect you. For example, Chrome provides the following message.



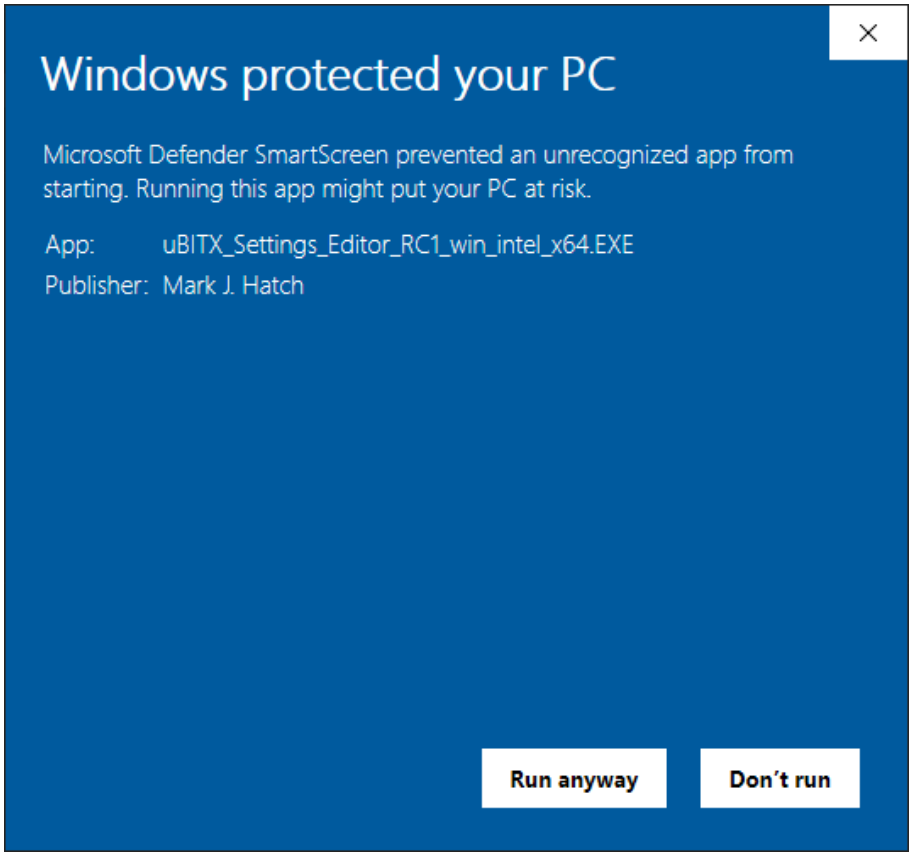
   Click on the little up arrow to the right of the “Discard” button and select the “Keep” option.

2. Navigate to where you downloaded the file and doubleclick it to start the installer. This dialog box may now appear:

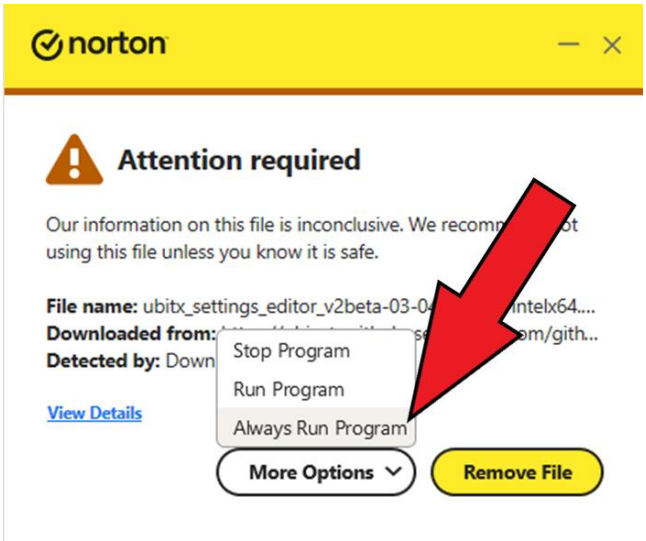


   Click the “More Info” link

3. Review the screenshot on the next page. The information should match the following. You can then click the Run Anyway button. (It is possible with time this warning will go away.)

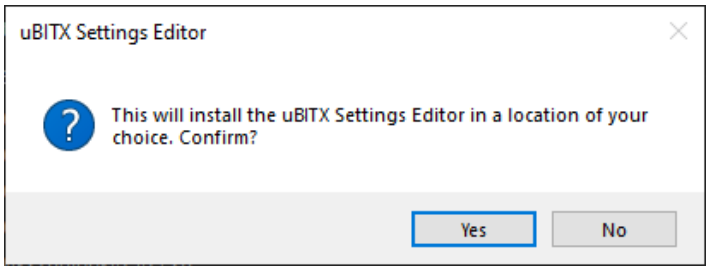


4. I run Norton as my antivirus software and I sometimes gat a popup like:

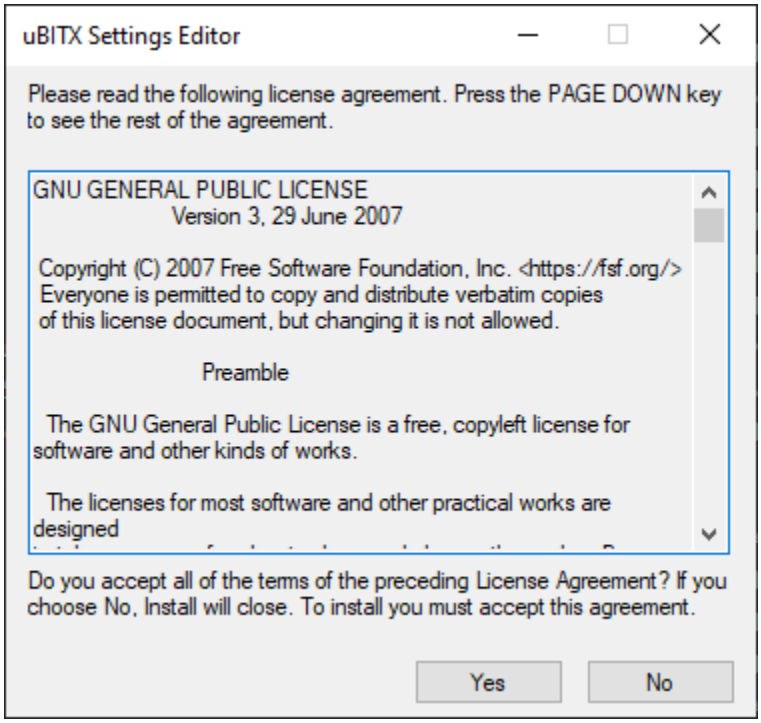


However, on my latest test install, I only got a "its safe" message on the download. So perhaps Norton is getting trained!I

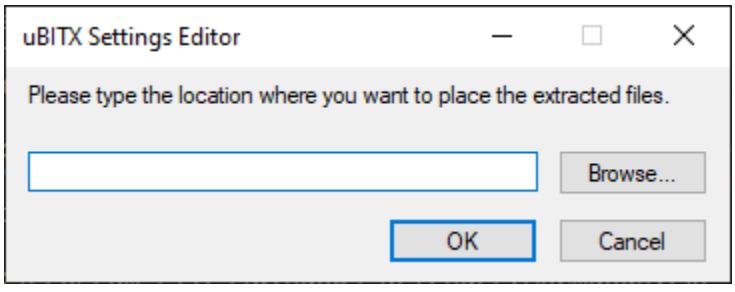
5. The Installer will start, and you will get the following dialog:



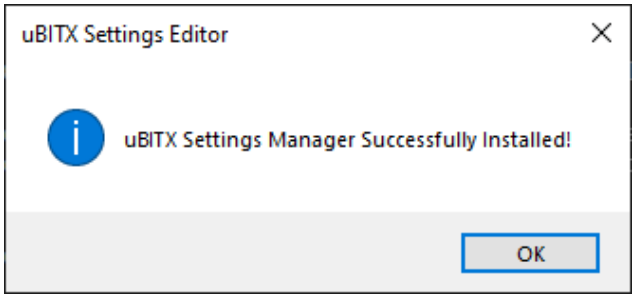
6. After clicking YES, you will get the License Agreement acceptance screen (it's the GPL V3 license)



7. Accept the License Agreement, then this very austere dialog box appears asking you where you want to install the SE:



8. After it finishes copying the executable, you will get the following success dialog:



9. If you did not put the file on your desktop, you might want to create a shortcut to it for your convenience.

macOS

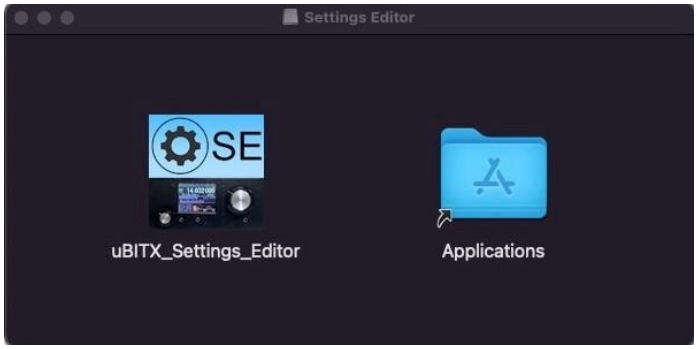
During this installation process, I was running macOS Monterey 12.4. If you are running a different release, your experience may vary. But the following steps worked for me:

Navigate to the release using either Chrome or Safari

https://github.com/AJ6CU/uBITX-Settings-Editor/releases/tag/V2.0RC1

1. Look for “Assets” at the bottom. Click on the highlighted link for the macOS version.

2. Navigate to the downloaded file and doubleclick the file to start the install.



3. The screenshot above shows the installer. If you are a macOS users, I suspect you know to drag the SE Icon to the Applications folder. And then click the Red ball on the upper right to close the installer window.

4. You will now find the Settings Editor in your Applications. Doubleclick on it to start. And the **FIRST TIME** you will see the final warning dialog:

Just click Open and the Settings Editor will start1

Linux

If you are using the Linux distribution, you will be happy to hear that things are much easier for you.

Navigate to the release using browser.

https://github.com/AJ6CU/uBITX-Settings-Editor/releases/tag/V2.0RC1

1. Look for “Assets” at the bottom. Click the distribution that matches your OS to download it.

2. Save the file (probably end up in your Downloads directory).

3. Open a folder to that location. Right click on the “.tgz” file you just downloaded, and then select either “Extract To” or “Extract Here”. Alternatively, you can directly use the tar command from a terminal window and save it where you want it.

4. In your target directory for your extracted folder, you should see a folder “ uBITX_Settings_EditorRC1….”. Open it.

5. Within the folder, you should see a file “uBITX_Settings_Editor”. Double click it to start. Depending on the distribution, you might get an annoying “what do you want to do” (with this file). Just open it.

6. You can add the Settings Editor to your favorite or create a desktop icon. This varies by distribution and desktop environment.

NOTE: BEFORE YOU CAN READ/WRITE TO YOUR RADIO, YOU MUST EXECUTE THE COMMAND BELOW AND REBOOT.

sudo usermod -a -G dialout <yourloginname>

sudo usermod -a -G tty <yourloginname>

where you replace <yourloginname> with the user name you use to login to your system.

Source Distribution for DYI Builds

One of the great things about using Python, is that it is very easy to take the source code to any machine and get the application working within minutes.

If you are a source builder, take the following steps.

1. Make sure you have Python3 installed. Open a cmd or terminal window and try typing “python” at the prompt. When it starts, it will provide a Version#. If it says Python 2, type exit() and when you get back to the terminal prompt, try typing “python3”. If you get a “not found” error message at this point, download and install the latest python for your distribution.

2. Make sure the following extensions to Python are installed by executing the following commands (this is done at the shell prompt and **not within** python interpreter:

```
pip3 install lxml
pip3 install pyserial
pip3 install bitarray
pip3 install pygubu
```

   Note: this installs it in a subdirectory of your HOME. And that is not on the PATH search variable. Best to either create a virtual environment or prepend “sudo” for each of the commands above to make them available system wide.

3. Download the source distribution (see the project assets) or clone the repository

4. Navigate into the uBITX-EEPROM-Manager /uBITX_Seetings_Editor and type:
   python3 uBITX_Setting_Editor.py

5. If you get an error when trying to open your radio like:

   failed to open port /dev/ttyxxxxx, trying to open the serial port,

   then try:

```
sudo usermod -a -G tty <yourusername>
Sudo usermod -a -G dialout <yourusername>
```

6. REBOOT (not just login and logout)

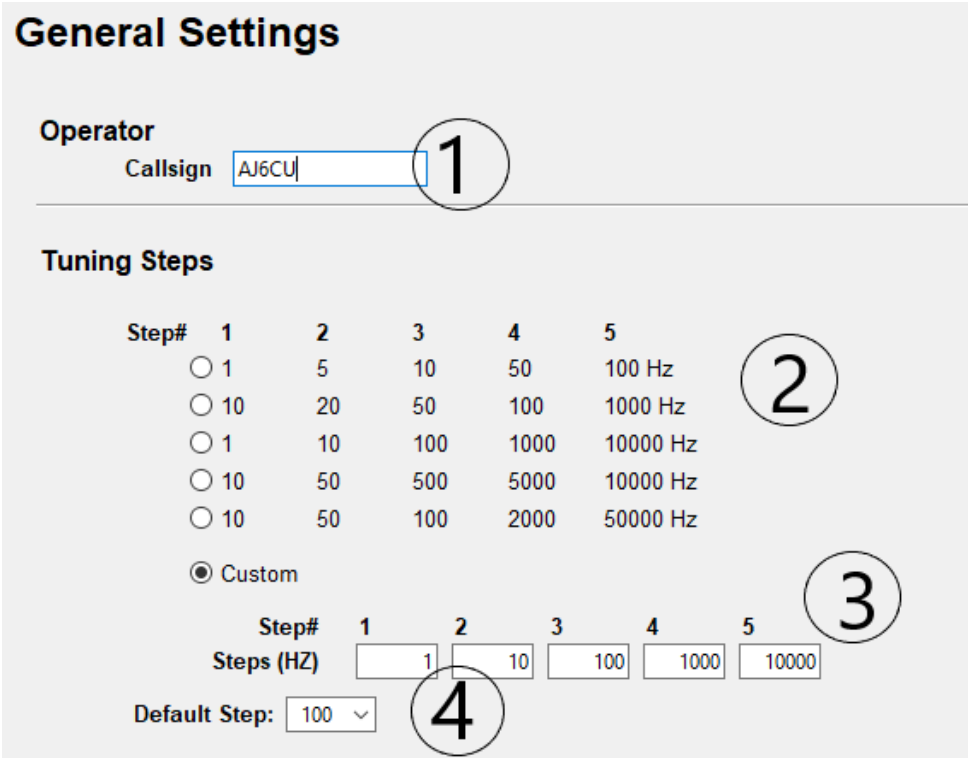
In Depth Review of the Functionality of the uBITX Settings Editor

The primary workspace for modifying the settings of your uBITX is a set of tabs -- much like dividers in a 3-ring binder -- that is in the middle of the application. Currently, there are tabs for:

- General – settings that most users will need to modify.
- CW Key – controls the automated CW keyer used contesting.
- Bands – defines the available frequency bands.
- Channels – allows the creation and management of frequently used frequencies.
- WSPR – builds messages and select frequencies for a WSPR beacon.
- Displays – manage LCD display and customize their presentation of information.
- SDR – configures the uBITX to use an SDR dongles.
- Extensions – supports the edition of function keys, customization of the LPF and IF
- Calibration – key hardware values that govern frequency accuracy, CW key interpretation and S-meters.
- System Info – documents installed firmware version, factory values and last used frequencies and modes.

General Tab

The General Tab collects the first settings that a user *should* want to change. Ideally, a user should tweak these settings, save them to the uBITX (and hopefully a backup file).



By the Numbers

The screenshot above is the *top half* of the General Settings tab. The numbers 1,2,3,4 are used to highlight key features. We will use a similar approach with all the other tabs and screenshots.

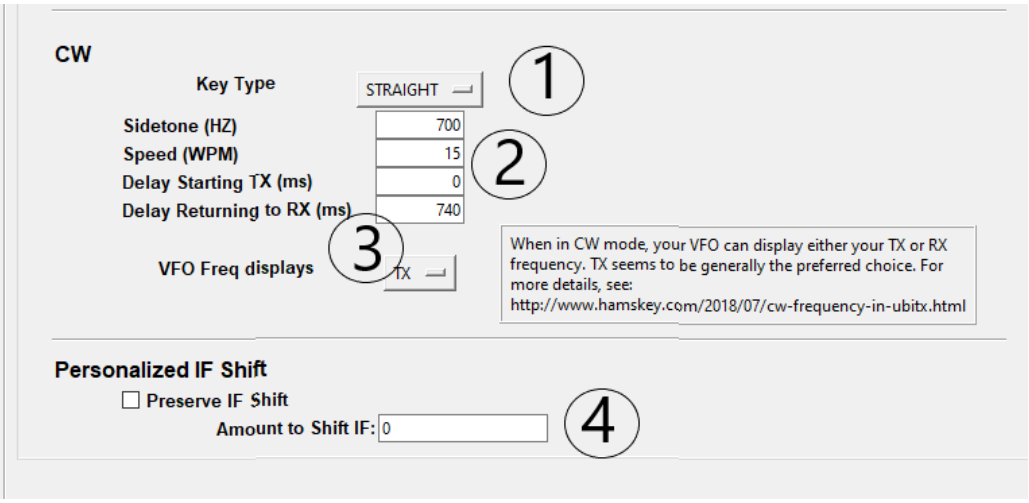
1. The first thing any Ham should do is to put his Callsign into his radio. You do this on the uBITX by entering your data in this field. As the tooltip indicates, you can enter up to 18 characters. This should leave enough room for callsigns that need to be qualifies (i.e., AJ6CU/MM if this radio was being used in a portable mode)

2. Most Hams spend a lot of time searching the dial. The "2" is next to the "standard" tuning rates. Click the circle next to the tuning rate that seems to best me your needs. You can switch between the rates either using the UX of the Nextion based systems or by pushing in the encoder knob for a "long press" and then turning it to the desired step when prompted.

3. CEC does allow you to adjust the tuning rate of your radio. But because of the (ugly) way that the tunings steps are saved in the EEPROM, there are limits to the types of rates you can input. The rule is that the first two digits (left most) for every tuning step, must be between 0 and 60. So for example, in the above screenshot, 50 would be a fine step for tuning step 3. And even 60. But 61 would be forbidden. Similarly, in tuning step#4, 600 would be fine, but 610 would not. This strange constraint was the result of a devious byte saving trick where one byte holds both the significant digits (0-60) and the multiplier (i.e., x1, x10, x100, x1000, etc.) Do yourself a favor and just select one of the standard step ranges. They will be fine.

4. The final field of interest is a drop-down box that allows you to specify the default tuning rate when the radio is turned on. In this case, turning the knob one encoder click will increase or decrease the frequency by 100hz.

The screenshot below is of the bottom half of the General Tab (if you cannot see it, use the scrollbar on the right of the General Tab to move it into view).



By the Numbers

1. The first choice you need to make is whether you will be using a Straight Key or a paddle. And if a paddle, are you using Iambica A or Iambica B keying modes.

2. The first two options in this area are clear: what should be the frequency of the CW sidetone and what speed do you want your keyer to be set at. The next two settings are delays. The first is the delay *before* starting transmission. Generally, you want this to be zero. On the other hand, perhaps you have some external equipment that need to be switched into position and you need to give them time. The other setting, Delay Returning to RX is probably more commonly used. If you are a little hesitant with your keying, you could find your uBITX going in and out of TX mode between words or characters. This allows you to add some delay to avoid the annoying bang of the TX relays.

3. This is an interesting and sometimes controversial setting. When you send CW, your actual TX signal is offset from the frequency of the radio by the amount of your sidetone (in this case 700). The debated question is whether the frequency shown on the VFO is the TX frequency or the RX frequency. The general consensus of contesters and DX folks seems to be that they want to know where they are transmitting, this drop down box allows you to control whether your VFO shows the TX frequency (RX +/- Sidetone depending on USB/LSB) or the RX frequency. There is an article called out in the note on this that you can read on Hamskey site. There is also a lot of discussion about this in the archives of BITX20 group and on ubitx.net

4. By adjusting the IF Shift, you can tune the radio to make things sound better, especially with SSB voice. If you find yourself playing with the IF Shift a lot on your radio, you might want to program the offset in and click the “Preserve IF Shift” box. This way when you turn on your radio, it is tailored to your preference.

CW Keyer

The CEC software provides a functional keyer that can be tailored for contesting. Its main limitation is that it does not provide anyway to generate and track sequence numbers that some contests require. It also does not interface with N1MM, which is favored by many contesters.

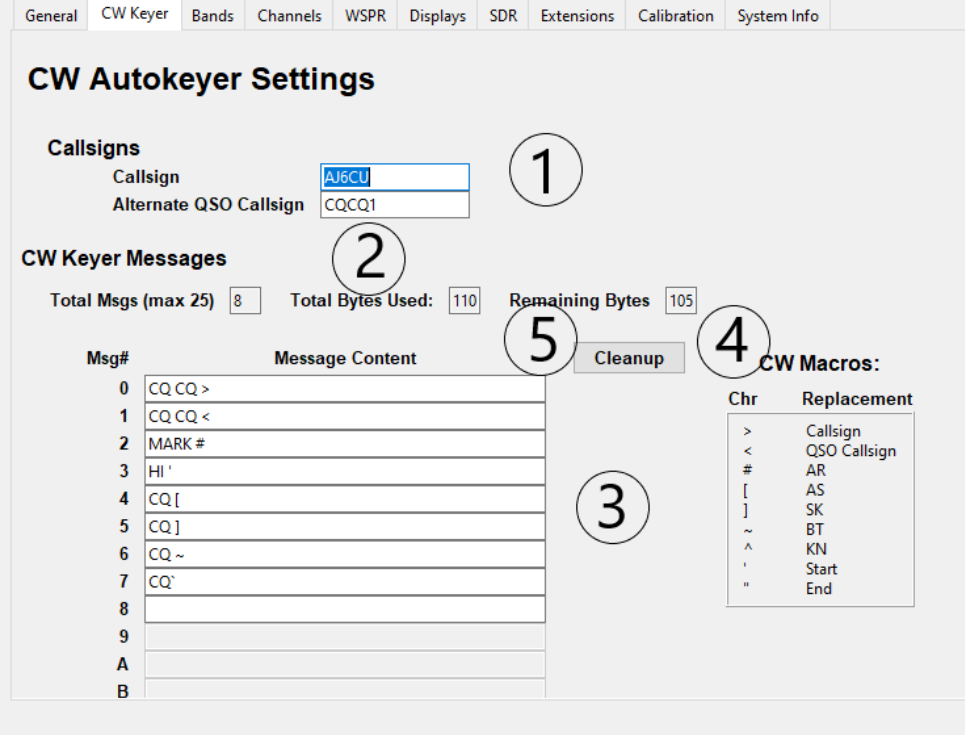
Although not as sophisticated as some, it does provide for character substitution and up to 25 messages! But from a practical viewpoint, 25 is probably far too many handle and the total memory available for keyer messages is on the order of 210 characters. So realistically, you will out of memory before you run out of available messages.

Tipe: There is a separate document that describes the use of the Keyer. Look for it in the Assets section of each CEC software 2.0 release.

By the Numbers

1. The callsign you entered in the General Settings is repeated up at the top. This tab also provides the option of an alternative callsign that you can use. Perhaps you arecontesting from a remote site and want to signify that with something like AJ6CU/5. Like your normal callsign it is limited to 18 characters.

2. Since EEPROM memory is limited, the line identified by “2” tracks 3 key aspects: 1. How many messages are active, 2. Total bytes used, and perhaps most importantly 3. Remaining bytes
3. These are the CW messages that have been saved. Everytime you enter the last one (for example we just entered the “CQ” in message #7, the next line (#8) becomes open. This was done this way to help you save memory since every active message has a 2-byte overhead.



4. This is the list of macros you can use in your CW message. Even though a “>” expands to your full callsign, you only use one byte (for the character “>”) in your message. Although these macros are convenient, they are especially useful for saving Keyer memory.

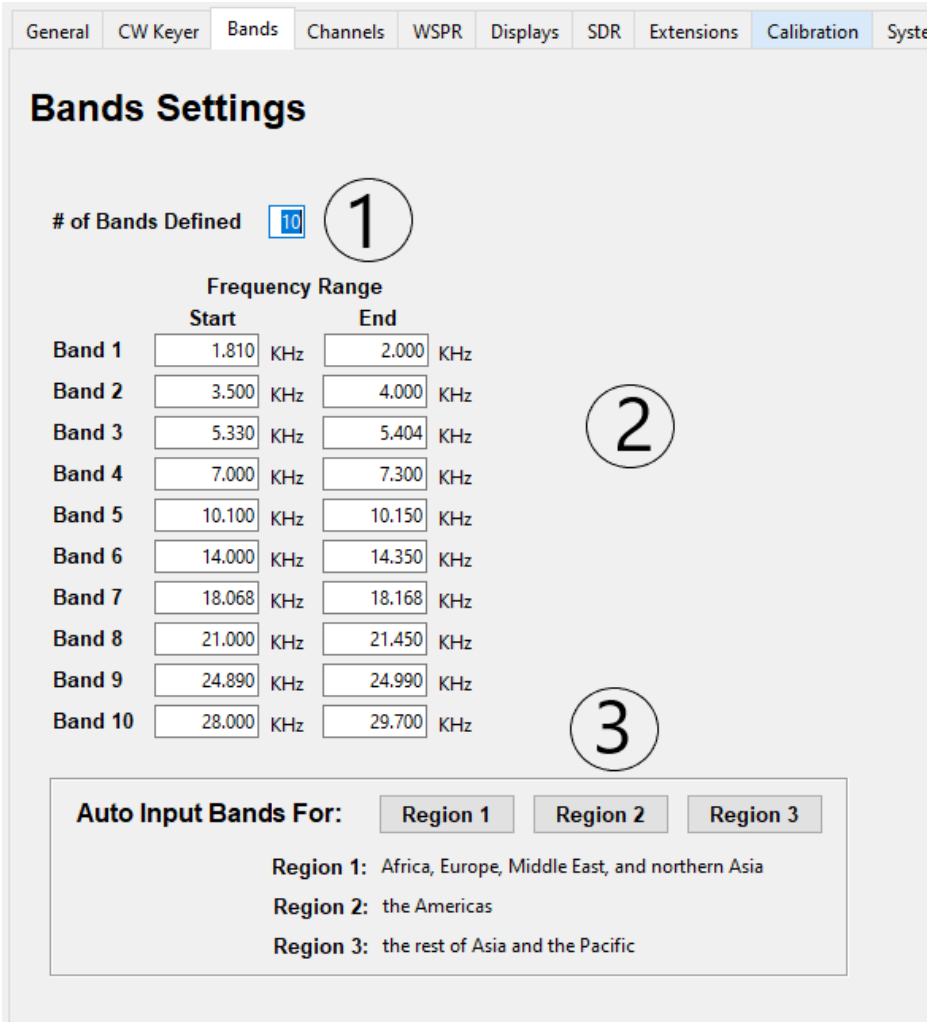
5. This button (CLEANUP) will eliminate empty messages and move them all up to the front. For example, suppose I decided to delete message 5 “CQ]”. This would leave an empty slot in message #5. Hitting the CLEANUP button, will move 6 into slot 5, 7 into slot 6, delete the blank line at 8, and recover the extra overhead bytes. You get a neater looking set of messages (especially important if you are keying via the rotary encoder), and you free up some bytes.

Bands

This tab is used to set the bands that are appropriate for your region. Since the uBITX is sold and assembled by amateurs worldwide, making sure that you are in the legal portion of the band in your part of the world is important. These settings directly impact the bands you see as you go +band or -band. Let’s do the numbers for the screenshot below.

By The Numbers

1. This is the number of active (defined) bands that your uBITX knows about. If you put 9 into this field, it will only know about the first 9 bands even though you might have a frequency in slot 10. And if the number is greater than the number of bands you have defined, you will have to cycle through those empty bands to get back to your favorite.

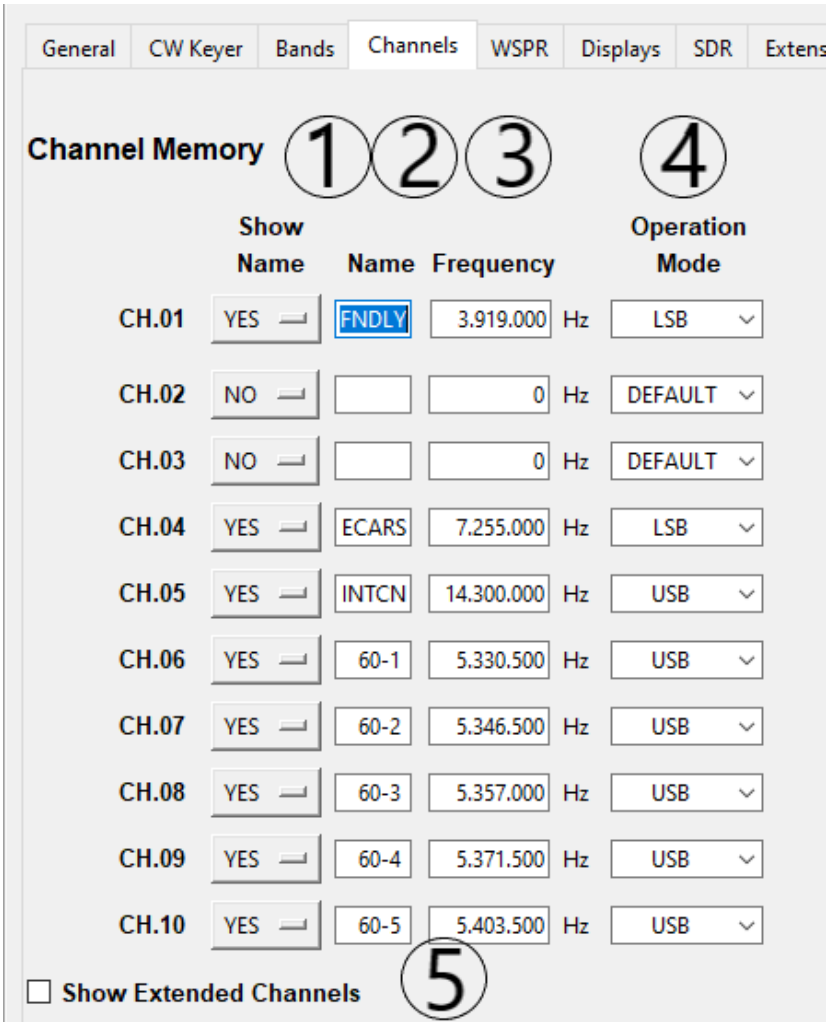


2. For each Band, you enter the beginning and ending frequency **in KHz (not MHz).**

3. It is a real pain, and error prone to find the definitive frequency map for your area and then enter 9 or 10 bands of numbers. So like in the original uBITX Memory Manager, this software will automatically fill this in at the push of the button.

Channels

CEC provides the useful concept of frequency channels. This allows you to identify and hop between up to 20 channels (10 with names, 10 just by numbers) and their respective mode (CWL, CWU, LSB, USB).



By the Numbers

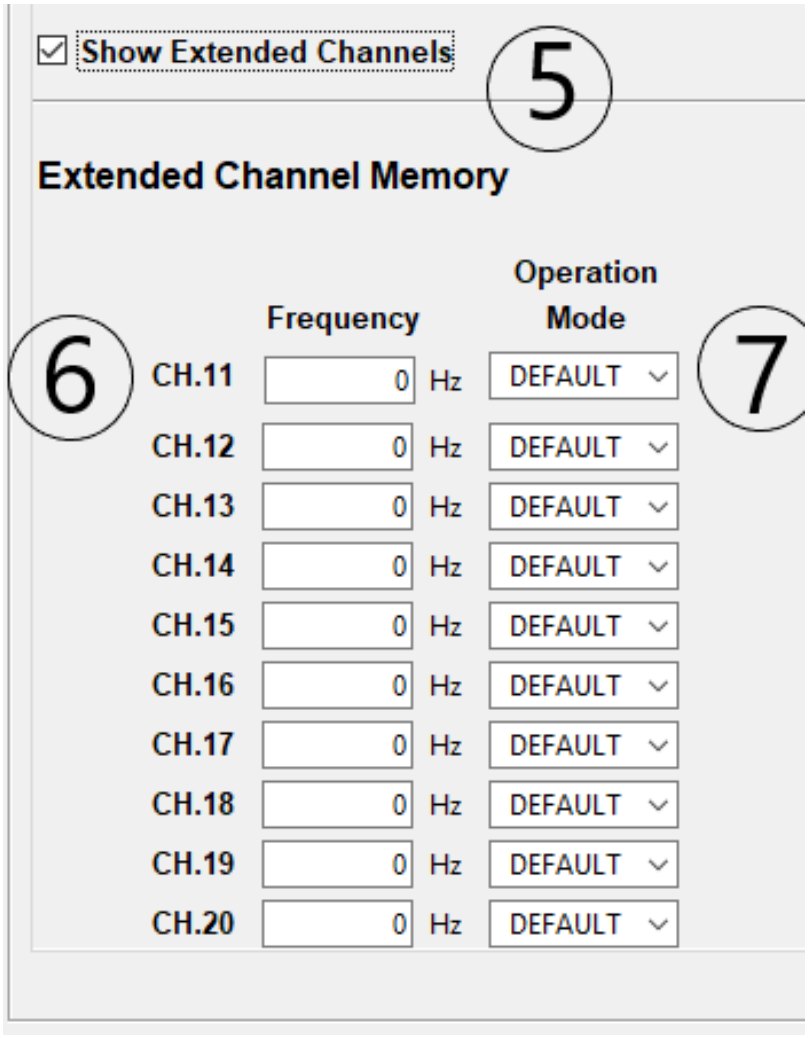
1. The first 10 channels can be assigned a 5-character name. But you might not need or want the name displayed on your screen (LCD or Nextion). This menu allows you to say **YES** show the name or **NO** just use the Channel# instead.

2. Enter a 5-character name here. Any characters are allowed.

3. This is where you would enter the frequency in HZ.

4. This is a drop-down menu that allows you to select LSB, USB, CWL, CWU, DEFAULT).

5. So where are the additional 10 channels? Click this box to see them and then use the scrollbar on the right to make them visible.



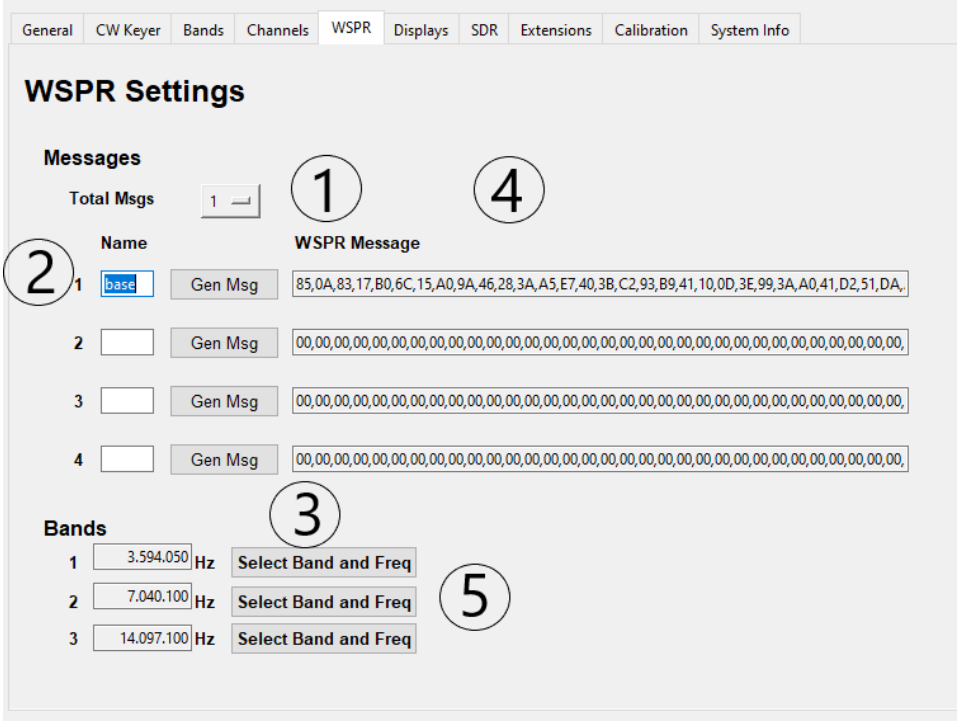
6. Here are the other 10 Channels! When you access them on your uBITX they are simply named CH11, CH12, CH13, etc. Enter the frequency of the Channel in Hz.

7. Then choose the operational mode. Same options as before: LSB, USB,CWL,CWU, DEFAULT.

WSPR

Going back through the archives of the BITX20 group, there's a lot of excitement about the addition of WSPR (Weak Signal Propagation Reporter) to the CEC software. The neat thing about this feature was that it is standalone and did not require that it was being driven from an attached PC. Basically, you can define up to 4 messages, and three frequencies and then transmit them in any combination.

The SE kept the UX format of the WSPR subsystem originally implemented in the uBITX Memory Manager. the same as it was presented originally. So existing users will make the transition easily.



By the Numbers

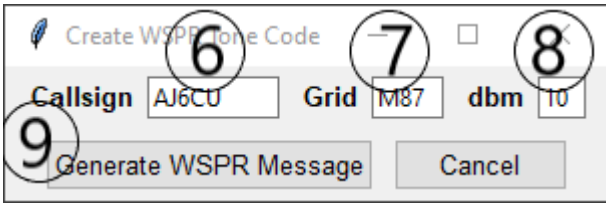
1. This drop down identifies how many messages will be made available on your uBITX. If you select 1 (as in this screenshot), you will only have one message available (even if you have more messages defined.)

2. You can name your WSPR message here using any 5 characters. That makes it easier to select the preferred message when you are operating.

3. Click the “Gen Msg” msg button to bring up a dialog box that will collect the information needed and generate the actual WSPR message.

4. This is an actual WSPR message in Hex format. The CEC software that runs on the uBITX will take this message and convert it to the 2-bit frequency shift that is used to transmit the WSPR message.

5. The WSPR frequencies that are going to be used are selected here. You can choose up to 3 different frequencies. Then after you choose which message to send, you choose the frequency. Click “Select Band and Freq” to bring up a dialog box that helps you select the band and specific frequency.

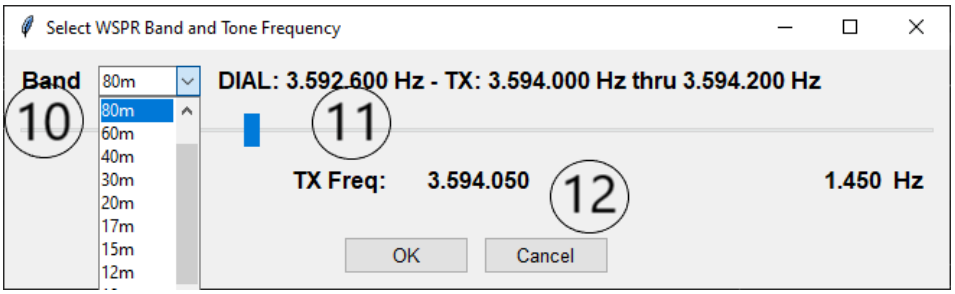


6. This dialog is used to create the WSPR message. The first thing to enter is your callsign. The WSPR protocol is very strict and there is a maximum of 6 characters allowed with numbers having to be in specific positions. Just enter your call sign here.

7. Enter the first 4 characters of your maidenhead grid square. For example, mine is CM87.

8. If you like, you can adjust the reported dbm to match your TX output.

9. Once all the information is entered, hit the "Generate WSPR Message" button to generate the message and close this dialog box.



10. This dialog pops up to help you select your TX frequency. The drop down on the left (#10) allows you to select the band you will transmit on. Do this first.

11. The slider allows you to adjust your transmit frequency to maximize its opportunity to be received by another station.

12. The actual TX frequency, Selected band + offset is reported here. Ideally, you should see your transmission reported very near this frequency. If not, you might want to consider tuning your Master calibration number.

Displays

This tab only addresses LCD displays. If you have a Nextion display, just keep moving on to the next section.

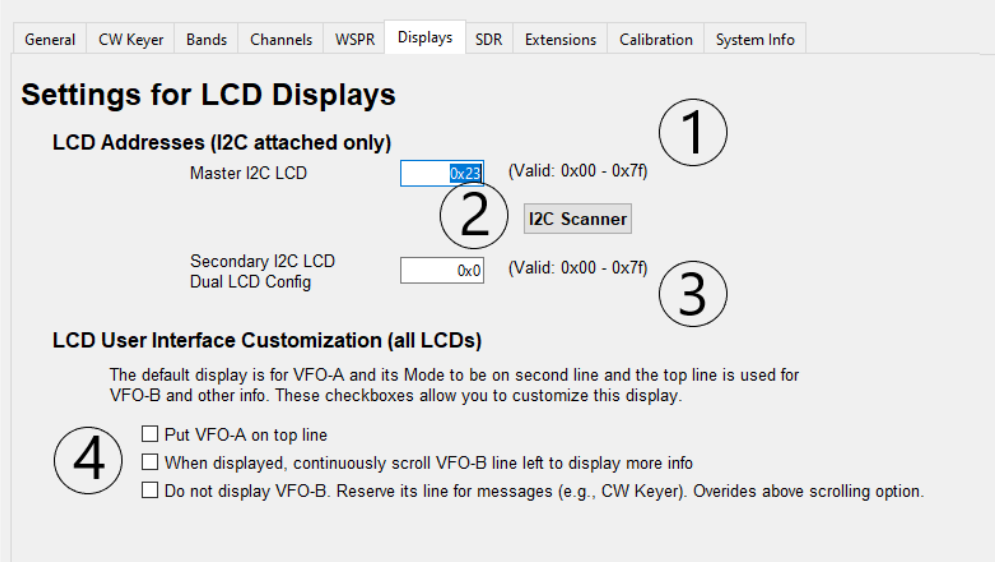
There are two types of connections you might have between your LCD and the Raduino (the daughterboard that holds the processor). The original connection was "parallel" where 6 digital lines (4

data and 2 controls) were connected to the LCD. This approach occupied a lot of pins, and the Nano didn't have extra pins. So early on, some folks went to the I2C based LCD's. This immediately freed up the 6 digital pins for other things. If you never did anything with your display, you have a parallel connected LCD. If you bought your uBITX second hand and you are not sure, then you can figure this out by seeing where the LCD is connected. If it is plugged directly into the Raduino board, it is parallel.

Knowledge of whether you have a parallel or I2C LCD is important for selecting the right values on this tab. If you have an I2C, you might need to set the I2C address (or addresses if you have two I2C LCD's). In which case, the upper part of the screenshot below is relevant.

By the Numbers

1. You can enter the address of your primary ("Master") LCD here. Unless you pulled out your soldering iron and changed some jumpers, the standard address is 0x23 as shown in the screenshot. This is true for both 16x2 as well as 20x4 (20 characters by 4 lines) LCD. BTW, unless you are going for small and light, the 20x4 screen and the extra info you get is really nice!



2. If you are not sure of the address, then you can always click the I2C Scanner button and peek at your I2C bus. More on this on the next screenshot analysis.

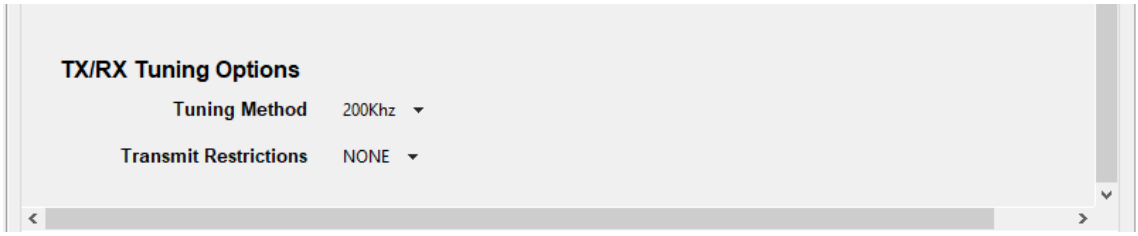
3. You can attach a second 16x2 LCD. The address of the LCD is here. (This would be as an alternative to a 20x4.) I don't recommend this solution as it is physically larger than a single 20x4 and it is just hard to mount two 16x2 screens.

4. This deals with customizing how the information on your LCD appears. The first option ("Put VFO-A on top line") moves VFO-A from its default bottom position to the top.

   Shouldn't be a surprise that a 16x2 LCD has limited amount of space to display information. You might check the scrolling option for VFO-B and see if this works for you. (I find the movement distracting and leave it unchecked.)

The final checkbox allows you to eliminate the display of VFO-B completely and let CE repurpose this space for other messages.

The bottom of the LCD display tab is two options to control the RX and TX of your uBITX.



The "Tuning Method" selection is currently set at 200KHz. This means that when you hit the Band+ or BAND- control via the encoder knob, the frequency will increase/decrease by 200KHz. The other setting, probably more common, is "BAND". This means that a Bands+ click will jump to the next higher band (as defined in the Bands tab. And of course, a Band- event means moving down one Band.

The second option is to set restrictions on where you can Transmit. In the screenshot, it is currently set to "BAND" which means that you can transmit on any frequency you can dial in. If it was set to "HAM", then you could only transmit on bands reserved for Hams. On the screenshot below, the

I2C Scanner

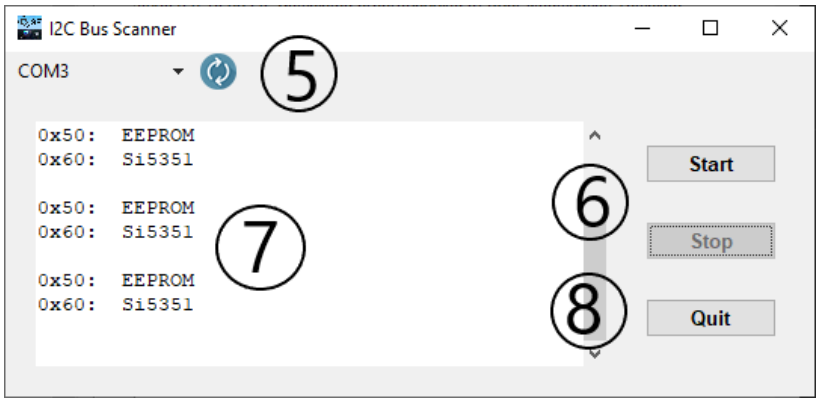
If you clicked the I2C Scanner button near the LCD addresses in the screenshot, the window below would pop up. This allows you to scan your I2C bus.

5. Make sure you are connected to the right COM port. Hit "Refresh Port List" if you don't see your uBITX.

6. Click the "Scan" button. A scan will run continuously until you hit the "Stop" button

7. The scanner will report the addresses of devices that it discovered. For common devices, it will also tell you what it is likely to be. For example, in the above example, the external EEPROM (many modern processors do not have an onboard EEPROM, so an external one is required) is reported on address 0x50. The Si5351 clock chip is reported to be attached to its standard address at 0x60. Notice that no LCD was found. That was because the LCD on this uBITX was attached via the parallel interface!

8. And click Done when you are finished.



SDR

Dr. Lee spent quite a bit of his time investigating alternatives for connecting a SDR receiver (e.g., RTL-SDR USB dongle) to a uBITX. You can get a listing of his blog entries on this subject by going to this link:

http://www.Hamskey.COM/search?q=sdr

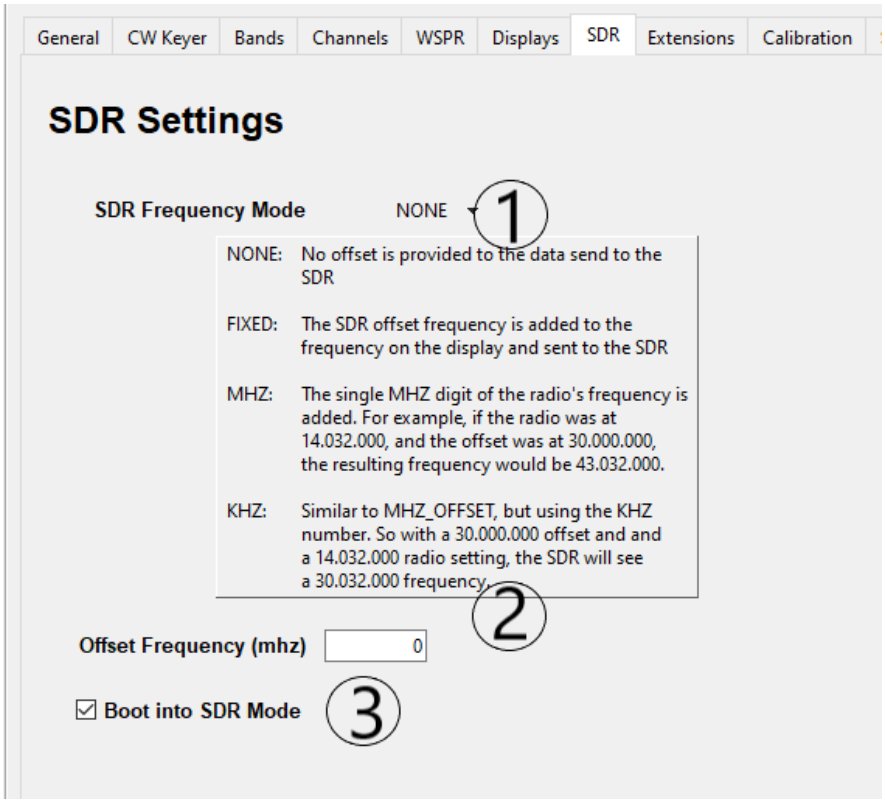
This tab deals with the various configuration parameters that need to be set to support his circuit design to support a SDR attachment to the uBITX. Since this is an involved hack (and there might be better and less invasive ways), most users will probably not need to set these variables. Indeed, most people's experience with the SDR feature of CEC is negative. Somehow they click it, (look for "SDR" on your display) and suddenly they have no sound output! You want it to say "SPK" for speaker and not "SDR"!

By the Numbers

1. This drop down allows you to set whether you have an SDR attached and what type of frequency offset the SDR uses. The text below "1" explains each option in detail. The MHZ and KHZ options are unusual, and I suspect that you will never need them.

2. This entry box is where you set the offset for the SDR. Each SDR has a standard offset, but often these are "nominal" and you need to adjust them for your particular SDR. Lots of tools out there for this.

3. Normally, you put your radio into SDR mode and when you reboot (or power cycle) it returns to non-SDR mode (i.e. "SPK"). You can change this behavior if you want by checking the box. Then when you reboot, it will start up automatically in SDR mode.



Extensions

In addition to the various extensions to the original stock firmware of the uBITX, Dr. Lee invested a lot of effort in several major extensions that seem to have been forgotten. This section deals with two of them, Extended Keys and Custom Lowpass filters.

Many commercial rigs have provided a facility to have a remote “pod” where a user can tune the VFO and push one of several buttons to control the radio. This small device (the Elecraft K-Pod is perhaps the star here) allows the bulky radio to remain on the shelf and its control is replaced by something that is approximately 150mm square.

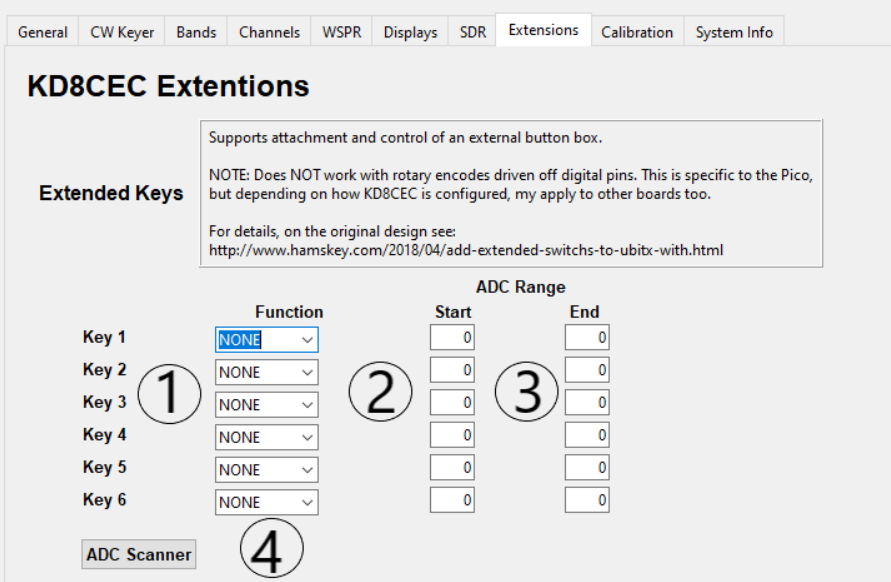
Extended Keys

The Extended Keys function was Dr. Lee’s effort to provide this functionality for the uBITX. The URL referenced in the text below will provide more technical details on how to implement it. The basics is that he re-used the encoder button interface (remember Nano pins are in short supply) and differentiate between the keys by the voltage level read at the analog pin used as input for the encoder button. For example, a voltage reading of close to zero is still assigned to a push of the encoder. However, if there was another button connected in parallel to the pullup that had a resister to ground, then a voltage reading somewhere less than 5 volts would indicate that the key was pressed.

So, to get this to work, you need to create a set of buttons with a resister network, different resistance for each button and your encoder switch (the “push action”) must be connected to an analog pin.

By the Numbers

1. This menu allows you to select what happens when the key press is detected. There is a whole host of common functions you might like to invoke including Mode (change), Band Up, Band Down, Change the Tune Step, Switch VFO's etc.



2. The starting ADC reading (0-1023) that will be trigger the recognition of this key.

3. The ending ADC value (0-1023, but greater than the start!) that will result in the key being triggered.

4. So how do you figure out what these ADC values are? Just click the ADC Scanner button and you can push buttons and see what the "safe" range is for a particular button. The explanation of the ADC Scanner will be reviewed in the next section "Calibration".

Custom Low Pass Filters

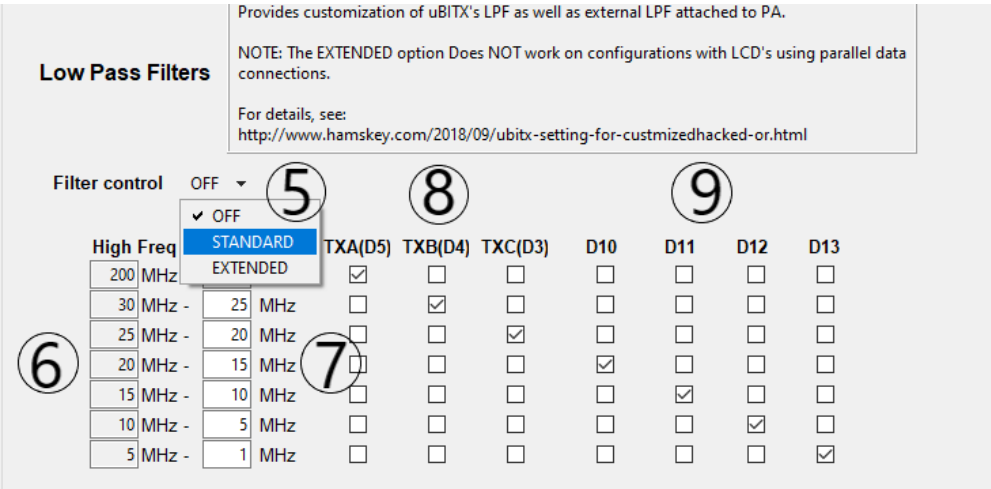
In the early uBITX v3/v4 days, there were significant concerns that the TX of the uBITX did not meet the standards set forward by the Federal Communications Commission (FCC) in the USA. There was a lot of experimenting in the community trying to figure out whether there was a combination of low pass filters that would make the uBITX compliant. I suspect that the original work by Dr. Lee in this area was primarily related to supporting this experimentation.

However, in solving this problem, Dr. Lee also provided a solution to signal other devices on the frequency bands the uBITX was using. These extra lines (repurposed from eliminating the parallel LCD) can be brought out and interpreted by external devices. I am not sure if that was his original purpose or not, but it can be a very useful feature, even if you just use the standard filters and frequencies of the stock LPF network of the uBITX. The next screenshot reviews how to set these filters.

It is important to note that this is a compile time configuration parameter. The standard HEX files that were originally generated by Dr. Lee did **NOT ENABLE** this feature.

By the Numbers

5. This drop down allows you to turn off this feature or select the “STANDARD” or “EXTENDED” set of lines. STANDARD lines are the ones already configured to the three relays built into the uBITX that control the LPF. EXTENDED adds to this the data lines that were reserved for a parallel LCD. These lines can be used to signal frequency settings to external devices as well as customization of a LPF.



6. Although you cannot directly enter this, these are the HIGH frequency limits. The values are just set to the start of the PRIOR band. So, for example, the third band down as a HIGH frequency of 25mh. This was the LOW frequency of the prior LPF band.

7. This column is where you enter the LOW frequency of each set of filters.

8. For each pair of frequencies, you can select to enable or disable TXA, TXB and TXC. The referenced article by Dr. Lee provides you how these three lines are set by default.

9. These are the additional data lines that you can use when an LPF band is enabled. For example, you could use these lines to select an antenna or perhaps an external LPF on a PA.

IF Adjustments

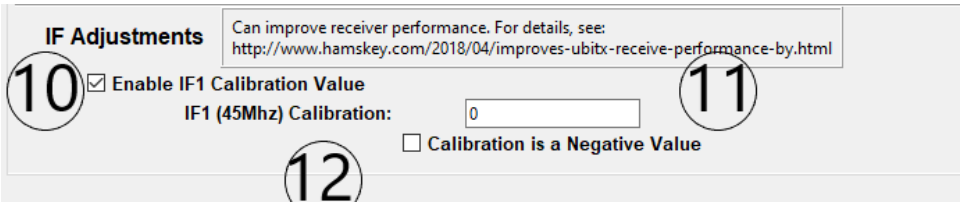
One of the early discoveries of Dr. Lee was that tweaking the IF slightly can improve the reception of your receiver. I personally suspect that the benefit of this tweak has long since been incorporated in both the uBITX hardware and firmware. But judge for yourself. There is a link provided in the *SE*.

By The Numbers

10. This is just a checkbox to enable or disable this feature.

11. The article discusses how to figure out what should be entered here for the calibration figure.

12. If the calibration figure is negative, you need to check this box and enter the POSITIVE number in #11 above.



Calibration

This is the largest set of numbers of any tab. However, for most users, these are the settings you DO NOT want to touch unless you are sure of what you are doing.

There are three general areas of calibrations:

- Radio frequency/BFO
- ADC associated with the press of a CW key
- S-Meter

Radio Frequency/BFO

Don't touch these except when you have no choice. Changing these values can result in not only your uBITX RX/TX on a vastly different frequency (Master Cal) than you think but can also result in your radio being "deaf" to signals (BFO).

If you ever need to rediscover or tune these numbers, HF Signals has provided a nice video on YouTube:

https://www.youtube.COM/watch?v=t6LGXhS4_O8&ab_channel=AshharFarhan

that uses a tuning aid hosted on their website:

https://www.hfsignals.COM/index.php/bfo-tuning-aid/

These resources are just for the V6. However, I have used them successful with my older uBITX.

However, for radios running CEC V2.0 or better, there is a new Calibration Wizard that should simplify this process. Sell later in this document for detil usage guide.

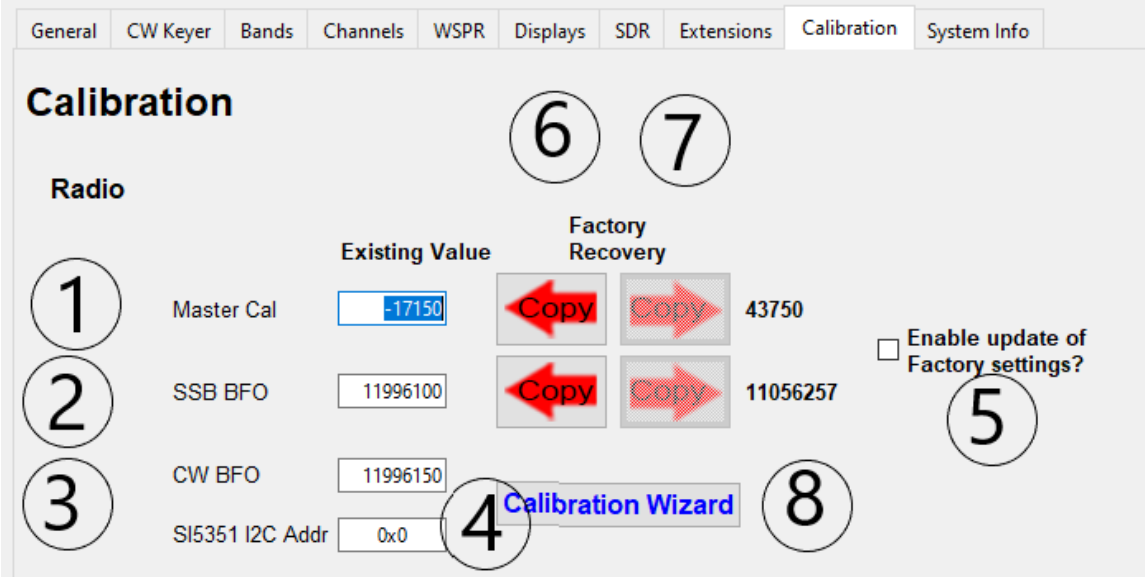
When you originally install the CEC firmware, it automatically copies the original settings for the Master Cal and SSB BFO to a special location in the EEPROM. This provides you with a "factory reset" in case you mess up these numbers and forgot what they were. The screenshot below shows the value for these factory numbers and, as will be discussed, allows you to restore or update these numbers.

By the Numbers

1. This is the Master Calibration number. Part of the mystery of this number is that it isn't just an "add this to the base frequency" number. Nor is it a percentage (we are off by .01%), Dr Lee has a special blog entry on just calculating this number. You can read it here:

   http://www.Hamskey.COM/2018/05/frequency-adjustment-of-ubitx-si5351.html

Dr. Lee recommends calculating the new calibration and setting it using the uBITX Memory Manager (of course you can use the *SE* for that too). Reset the uBITX and check it again. I tend to favor the old school approach which is as done in the video above. Using the encoder, you can quickly change this setting and then check your results. (BTW: If you have a Nextion, this encoder-based calibration is not turned on. So, you must follow Dr. Lee's guidance.)



2. This is known as "THE BFO". Dr. Lee created confusion by referring to this as the "USB Calibration" number. Since everybody seems to refer to this as the BFO, that is the terminology adopted here.

3. Interestingly, CEC added the concept of a separate BFO for CW that did not exist in the original firmware. It is not clear whether this is essential differentiation as the (SSB) BFO and the CW BFO are generally very close. Although I am no expert on calibration, I ended up with both BFO numbers within about 30 points of each other.

4. This is the address of the Si5351 chip on the I2C bus (we saw it appear earlier in our discussion on the I2C Scanner app.) Like the rest of the calibration numbers, don't touch unless you know what you are doing!

5. The "Factory" settings for the Master Cal and the BFO should not be changed unless you really know what you are doing. If you REALLY DO KNOW what you are doing, you can check this box and now you can change the Factory settings too.

6. Clicking these left directional arrows will copy the Factory settings to the working settings of your radio.

7. These arrows are only made clickable if you have checked the checkbox mentioned in #5 above. Once you do, you can copy your existing settings over the Factory Settings. This means you Factory Reset will now be your current values.

8. This launches the new Calibration Wizard provided in the SE. We will go into this in more detail later.

CW Key Calibration

Calibrating your CW key is not as scarry as changing your Master Cal/BFO. But getting it right can still be a little tricky because the design of the CW key uses just one pin! For a straight key, the uBITX is ready to go. Hook it up and use the ADC scanner (see #12) to determine the ADC value when pressed vs not pressed. For example, on my rig, with a Nano, A6 reads 1023 without the key pressed. When pressed, A6 goes down to the low 300. So I would set the Start range to 0 and the End Range to 500.

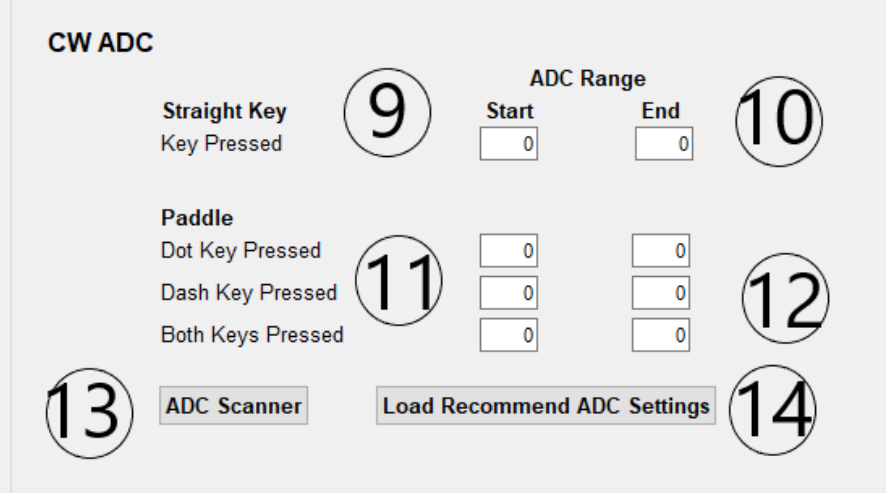
If you are using a paddle, you may want to put together a small perf board with a resister network. This creates a create a different analog voltage on A6 when the Dot key or Dash Key or Both Keys are pressed. This article does a good job of expanding and describing this circuit:

https://ubitx.net/wiring-up-a-paddle-and-straight-key/

The *SE* (and the uBITX Memory Manager before it) allows you to set the bounds of the ADC for each key press as well as simultaneous presses of both keys.

By The Numbers

9. This is the beginning number for the ADC value when a straight key is pressed. Obviously, zero is the lowest it can be (but impossible to reach because of the 4.7k resister that is in series with A6.



10. This is the high-end ADC value for the key being pressed. On my stock V6 with a 4.7k resister in series, this number is around 325. You could enter 500 here and be pretty safe.

11. Assuming you have the resister network in place, you can check the value of ADC on each of the combinations of the keys pressed and see what the values are.

12. Same as 10 but you are looking for the value that is just higher than the ADC you see with the ADC scanner.

13. Click this to start the ADC scanner. This tool will help you a lot as you try to tune these ADC values.

14. You can also try loading the “Recommended Values” and then see which keys don’t work as expected. Then focus on tuning just those keys.

ADC Scanner

The ADC scanner can help you see the ADC values that processor is seeing when a key is pressed or the amplitude of the signal in the case of the S-Meter. It is important to note that this scan is for a classic Nano. In the beta, pins that are not assigned (not available or used by the processor, will show a “0”. This limitation will be addressed before final release.

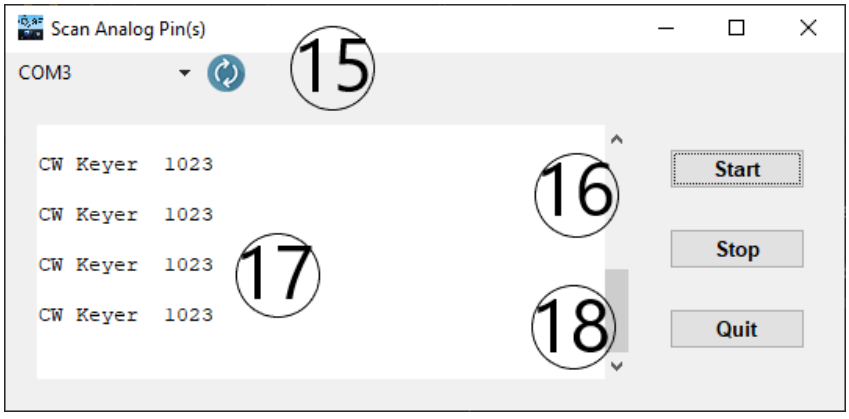
By the Numbers

15. As usual, make sure you are talking to the correct uBITX by checking on the selected COM port.

16. Click the Scan button to start a scan. Click the Stop button when you have enough data.

17. The symbolic name and value of the pin we are monitoring. For example, you can put key down=, see values, then release it and see values and get a good estimate of the probable ranges.

18. Click the Quit button when you have the information you need to return to the primary UX of the SE.



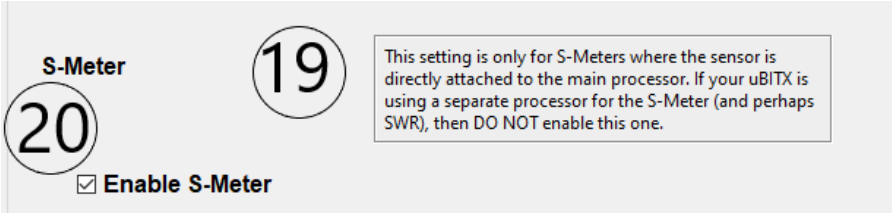
Configuring the S-Meter

You might not have to do anything here… These configuration parameters are for an S-Meter where the signal strength is directly being measured at the primary processor by a small (simple) sensor. See the following for the description of this sensor:

http://www.Hamskey.COM/2018/06/creating-simple-s-meter-sensor-for.html

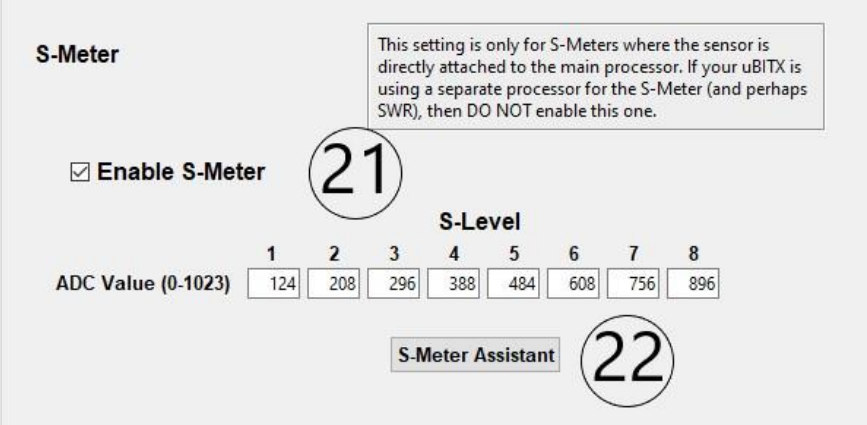
By the Numbers

19. If you are measuring signal strength using a separate Nano, or you don't have a sensor at all, just skip this section.
20. If you are using a sensor like described in the link above (this is on board in my design), then click this checkbox.



21. With the Checkbox clicked, you will have the option to individually set the ADC value for each S-level. For example, S-Level 7 is up through the ADC reading of 756.

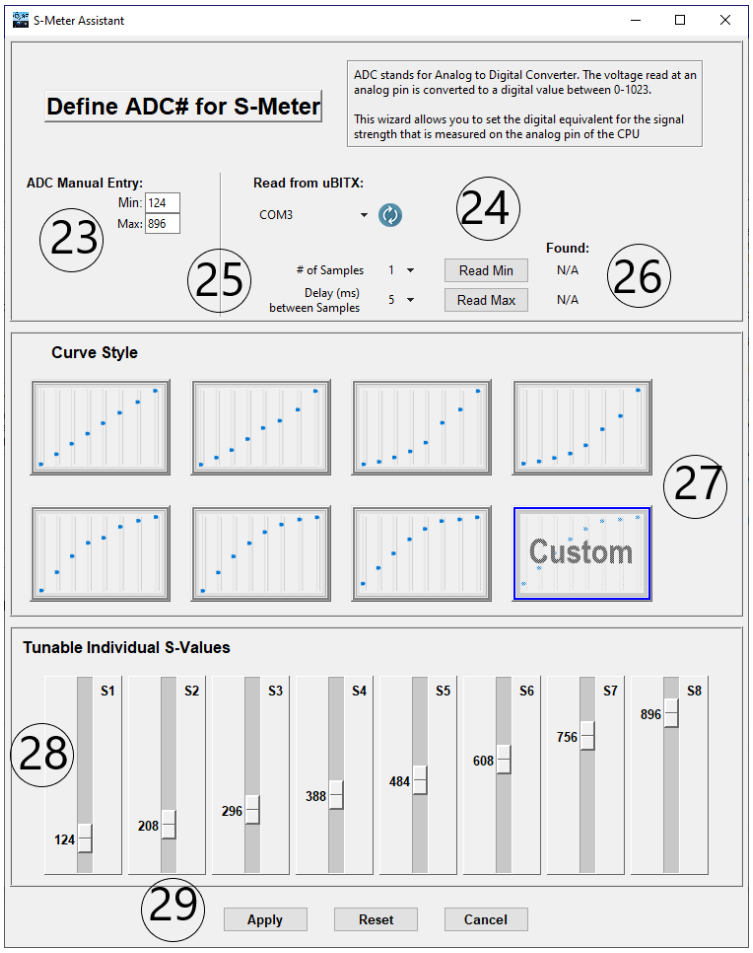
22. Need some help deciding on the S-Levels? Then click the S-Meter Assistant.



S-Meter Assistant

When you click the S-Meter Assistant button, the S-Meter Wizard appears. The goal of this wizard is to help you make a good initial choice for the ADC reading at each S-Level. Although you could try to

exactly calibrate this, it is really a subjective feeling. One reasonable goal would be to try to get the uBITX S-Meter to be like the S-Meter of other radios in your shack.



The process of wizard is to designate a Low and High value for the ADC. Once you have that, you can then just select one of the sample curves provided and the rest of the S-Levels are adjusted accordingly. Then if you want to, you can tweak the individual S-Levels a little using the sliders near the bottom of the dialog.

By The Numbers

23. If you already have a good feeling for the lowest and highest ADC values your radio is hearing, then you can just enter these values hear to kick things off.

24. If you are not sure, you can scan the ADC to determine the minimum and maximum values. First, make sure you are communicating with the desired uBITX by choosing the proper COM port here.

25. The two drop down menus allow you to decide how many samples you want to take (top menu) and the time between each sample (bottom menu). Although this may be useful, you need to be aware that zero values do happen a lot, so the "min" value will likely always be "0". High end is a better, but still will tend to be too low because of the result of min and max values being returned.

26. This area just reports the results of your scan. Note that these values are not automatically used. If you want them to be your new minimum and maximum, you need to type them into the boxes discussed in #23.

27. This section shows that given a minimum and maximum, how do you want the middle points distributed. There are 7 different distributions (by definition, what you have currently is a "Custom" distribution.) Click the icon of the graph you want your points to look like and the various intermediary points are distributed according to the graph.

28. You can tweak individual S-Levels using the scrollbars provided in this section.

29. Finally, click Apply if you like these values, Reset if you want to go back to the original values, and Cancel if you just want to quit with no changes.

Calibration Wizard

The Calibration Wizard is a major new functionality added to this release and I hope that users will find it useful. I decided to add this function after repeatedly seeing posts on the BITX20 group of other Hams struggling with calibrating their radios.

The YouTube video by Ashhar Farhan, https://www.youtube.com/watch?v=t6LGXhS4_O8 is an invaluable resource and the steps of this wizard are the same as he suggested. The wizard actually provides links to this video as well as the HF Signals BFO Tuning aid, https://www.hfsignals.com/index.php/bfo-tuning-aid/ , via buttons that copy these addresses to the clipboard.

The frequency adjustment can (as suggested in Mr. Fahan's video), be done by zero beating against a known frequency AM station or WWV. If you are only going to calibrate one radio one time, this is the obvious choice. However, I have many uBITX radios and as I am always swapping Raduino boards (each with a different SiS5351 and crystal) I often find myself re-calibrating a radio. If you think you might need to calibrate your radio(s) more than once, or your antenna is compromised and it is hard to get a solid signal for the zero-beating process, I highly recommend this device:

https://www.ebay.com/itm/314264235972

It is accurate, can run off a 9v battery, and you can connect it to the antenna jack of your uBITX (probably should put a 3 or 6db attenuator between them, but I have not done so). It provides a zero-beat bandwidth of approximately 60 Hz. You can just find the high /low limits of that band and use the midpoint as the "zero beat" point. I have often used this and been pleasantly surprised that when I connect the radio to an antenna and turn it on, 10MHz WWV is crystal clear.

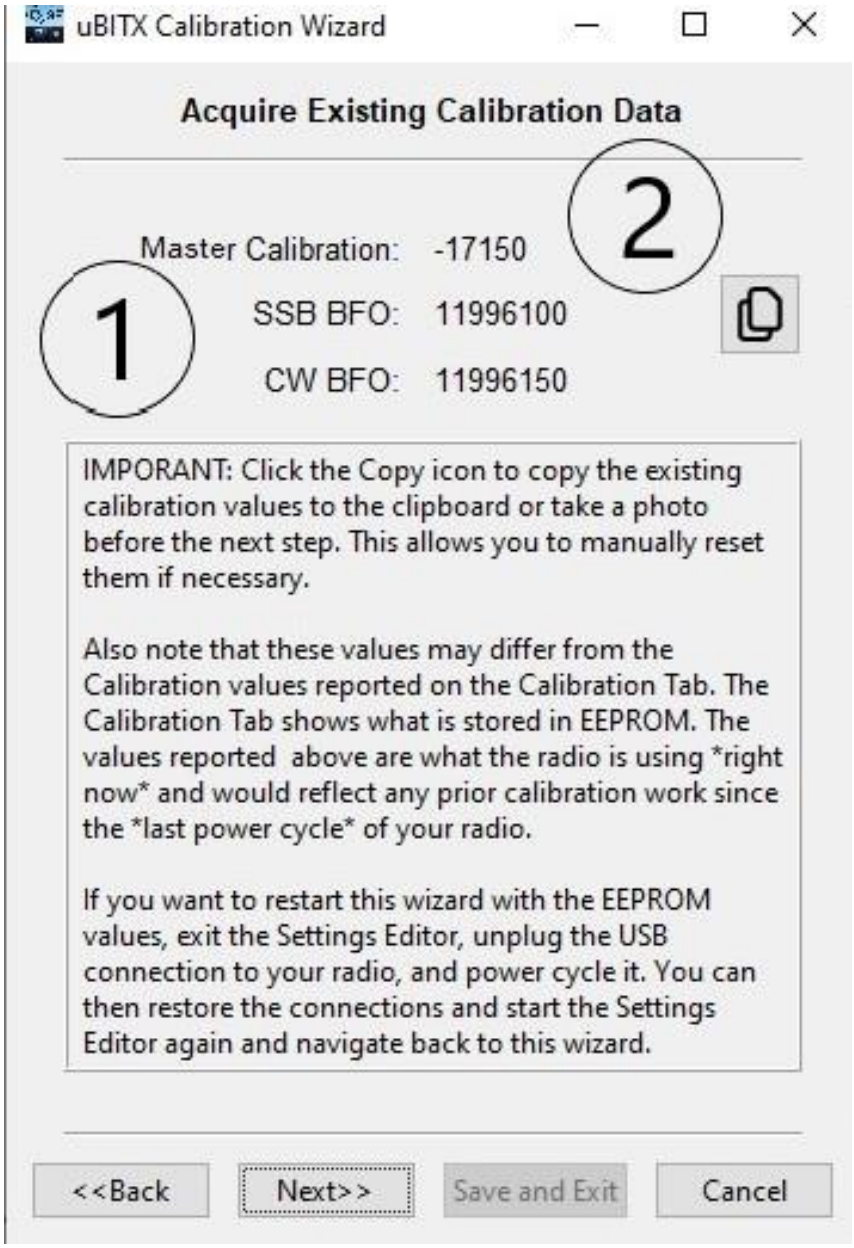
IMPORTANT: This wizard requires a radio to be running CEC V2.0 (or greater) because additional functions were added to allow interactive calibration. Upon startup, the wizard checks the CEC version number and exits if does not support these features.

The Calibration Wizard Welcome screen is shown below. It basically previews the process and tells the user that the original calibration values will be stored, and at any time, you can hit the Cancel button and they will be restored and your uBITX rebooted to use the original calibration values.

Clicking the Next button will result in the dialog below being displayed. At this point, the wizard has already connected to the radio and requested the existing calibration numbers. There could be a couple seconds delay between the introductory screen above and the display of this screen.

By The Numbers

1. These are the existing calibration values that the wizard has found in your radios's EEPROM.
2. By clicking the little "copy" icon, these calibration values are copied to your clipboard. It is strongly recommended that you paste these values in a document and save them in case you need to manually restore them later.

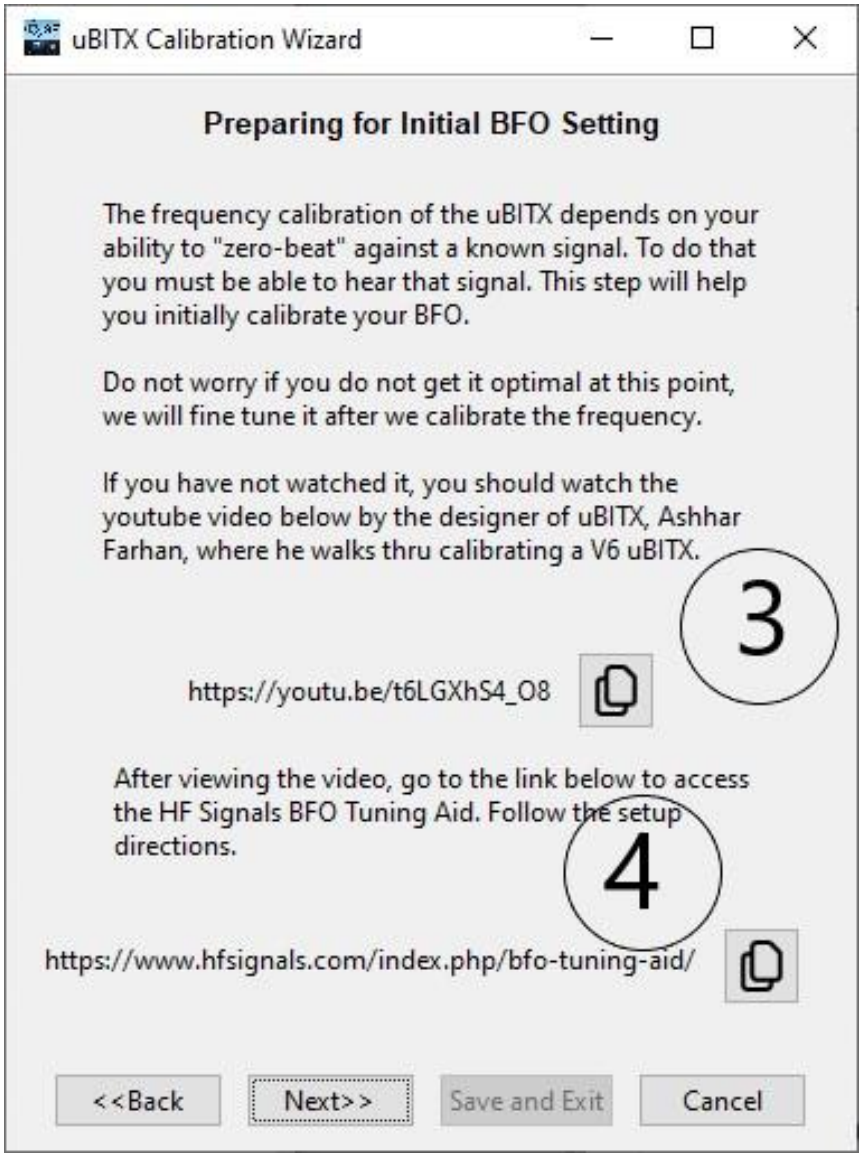


Clicking the Next button will result in the dialog below being displayed. This describes the process that is about to be used to roughly calibrate the BFO. This step is necessary to ensure that in the following step -- frequency calibration -- you can hear and zero beat a station. Note that the Master Calibration number is negative. This is generally unusual as almost all the numbers I have seen are positive numbers. However, the value is stored internally as a signed number, and it makes sense that the frequency of the radio can be either higher or lower than the calibration standard.

By The Numbers

3. Clicking the clipboard button will copy the URL for the Ashhar Farhan calibration video to the clipboard. The user is invited to review the video before contining.

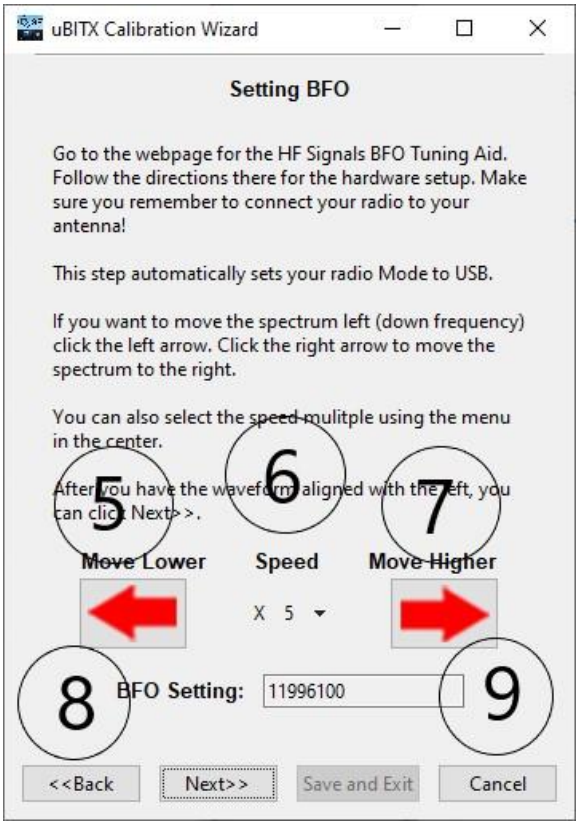
4. Clicking this clipboard link will copy the URL for the HF Signals BFO Tuning aid to the clipboard. Before proceeding, it is important to open this link and check that your microphone is picking the background hiss of an empty frequency.

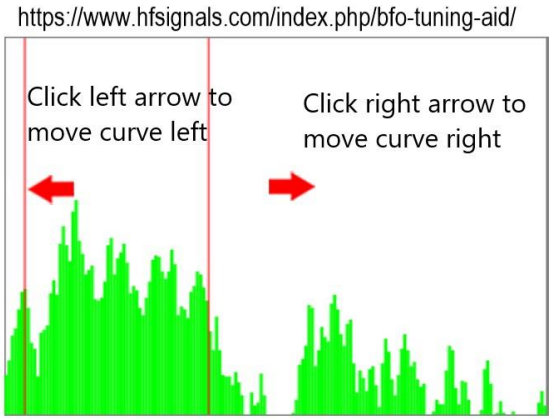


Clicking the Next button brings you to the next screen where finally you can start adjusting the BFO. At this point, it is very important that you have the HF Signals Tuning Aid (or some other audio analyzer) running in one window so that you can monitor the frequency response. The goal is to place most of the sound between the red lines on the Tuning Aid. Note: This step automatically puts your radio into USB mode (in case you forgot to do so!)

By The Numbers

5. Moves the audio spectrum to the left.
6. Sets the speed of movement. Although this defaults to “5X”, unless you are close, you might try bumping it up to 25X
7. Moves the audio spectrum to the right
8. The Current BFO setting is displayed in this field. It is read only and will only be updated based on the movement of the arrows.
9. Now is a good time to remember that the Cancel button will back you out of any errors and reset the radio to original values.





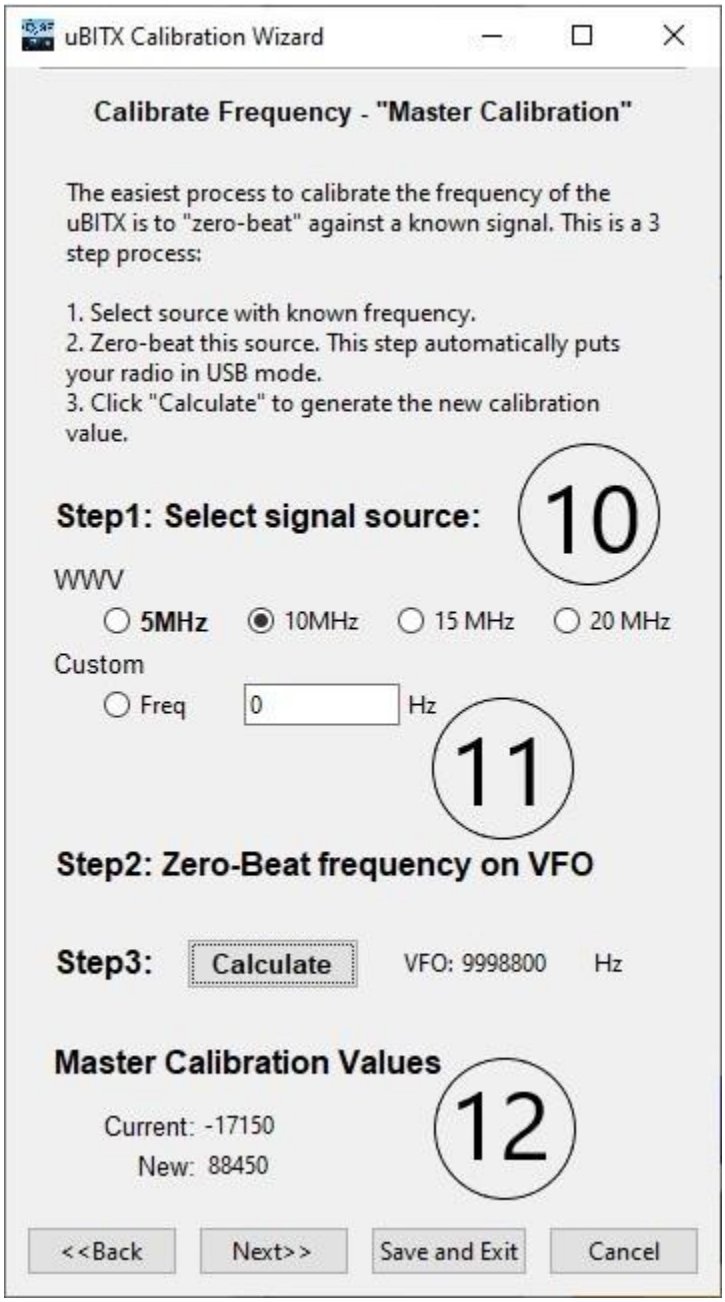
Clicking the Next button brings you to the next screen where we will calibrate the frequency of the radio. This is also called the “Master Calibration” value in some places.

By The Numbers

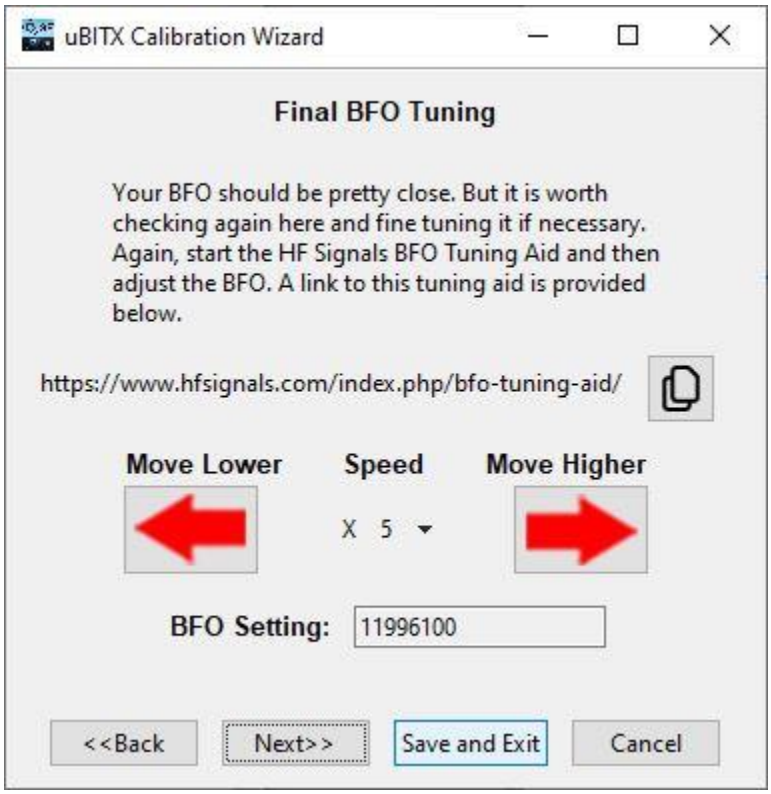
10. The first step is to select the frequency that you will use as your calibration source. Although I have included the typical WWV frequencies, you can click the Custom button and enter your own frequency.

11. After you have “Zero beat” against your calibration source, click the Generate button. The Wizard will read the current frequency on the VFO (it will be displayed to the right of the button) and perform the calculation for the new Master Calibration.

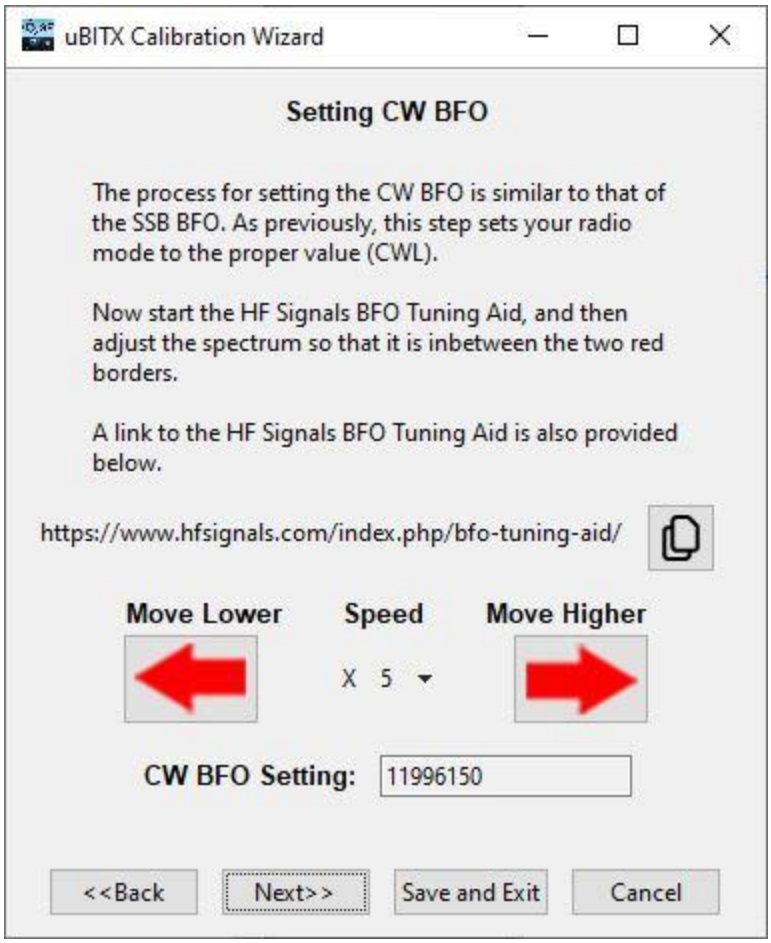
12. The old and new numbers are displayed in this section. Just as all the subsequent steps used the BFO number you set, the new value of the Master Calibration is used from this point forward too.



The Next button moves you to a screen that allows you to fine tune the BFO. The process is similar to the prior tuning of the BFO, so we will not repeat the instructions here.



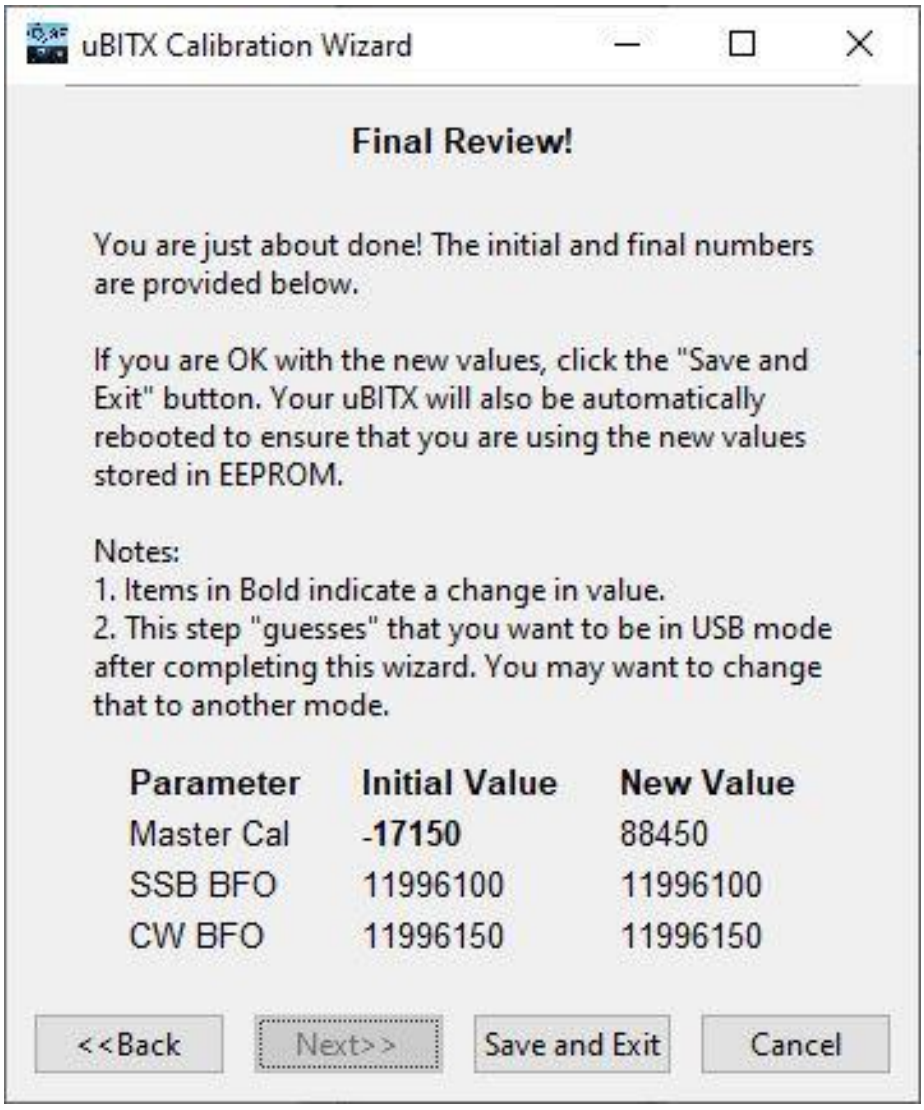
The Next button progresses you to a screen where you can tune the CW BFO. Although not strictly necessary if you are not a CW user, it is simple enough to set it here using the same process as was used for the BFO. Entering this step automatically puts you in CWL mode. Generally, the CW BFO is close to the SSB BFO. If you see it varying significantly, you might want to go back and check your SSB BFO before proceeding.



And we are just about done! Clicking next brings you to this screen where you can review the “Old” and “New” numbers side by side. The “New” number is bolded to highlight any changes from the “Old” value.

At this point, if you are happy with the results, just clock the Save and Exit button. Messed things up a lot? Click the Cancel button to backout of the mess. Want to fine tune things? Click the Back button to go back to the step that needs more work.

The Save and Exit button will save your new values to EEPROM, and force the uBITX to reboot so that the code uses the new values. You should hear a noticeable ship intone as the radio reboots. However, you need to remember that at this point, it would be good to hit the “Read” button in the SE input section to ensure that all the values of the EEPROM are reflected in the SE fields.



System Info

The final tab provides various information about your radio. Radios running CEC V1.2 will not have any useful information in this tab -- except for Factory calibration numbers and last used frequencies most of the values will be “N/A” because the firmware did not store this information. A Nano running CEC V2.0 on an OEM Raduino will be identified as a CEC V2.0, however the rest of the information will also mostly be “N/A”

The real data becomes available with a Raduino with an external EEPROM of 2kb or larger that is running CEC V2.0 The screens below demonstrate the type of data that is available in that configuration.

By The Numbers

1. This section provides details on the firmware you are running. In this case, we are running a pre-V2.0 release that was built on May 19th. It includes all the encoder menu items (“All Functions”).

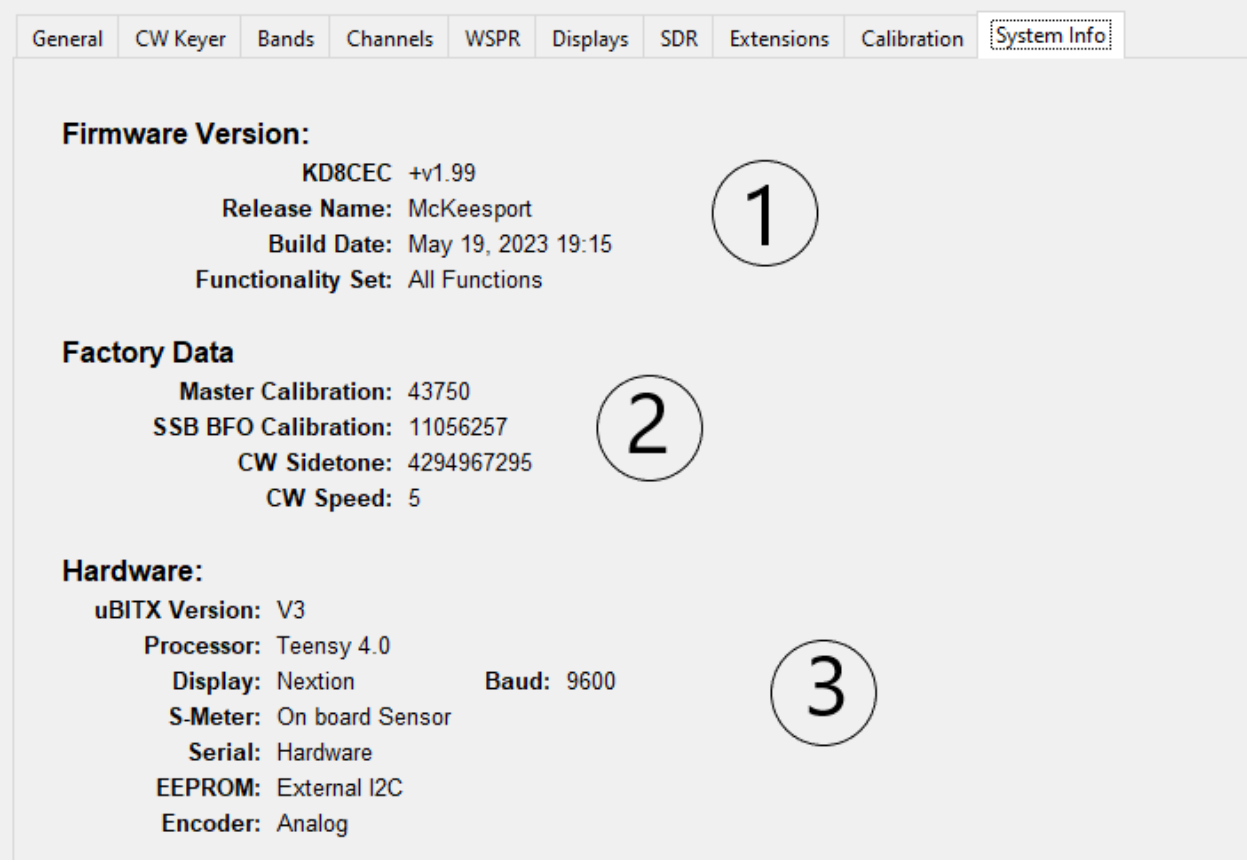
The release name is “McKeesport”.

2. In addition to the Master Cal and BFO Factory Data, the Factory Data also records the original CW sidetone frequency and keyer speed. This information is shared here. Note that the CW Sidetone is invalid. This is a common problem with radios that are first converted to CEC. This should be set to a valid value at the first opportunity using this SE.

3. This final section summarizes the hardware options that were selected in the build. Note, this is what the software thinks, and it is very possible that this particular uBITX radio is actually a V6 and not the V3 that the software thinks it is.

   The firmware was built for a Teensy 4.0 with a Nextion display. The Baud rate for talking to the Nextion is 9600. The software is also expecting the simple S-Meter sensor and not an external daughtercard with an additional Nano.

   The connection between the Nextion and the processor is via the Hardware serial port and *not* the software serial that was the only choice in prior releases of CEC. The firmware is also configured for an EEPROM that is interfaced to the processor via the I2C bus. Finally, the encoder uses the traditional Analog pins to monitor its state change.



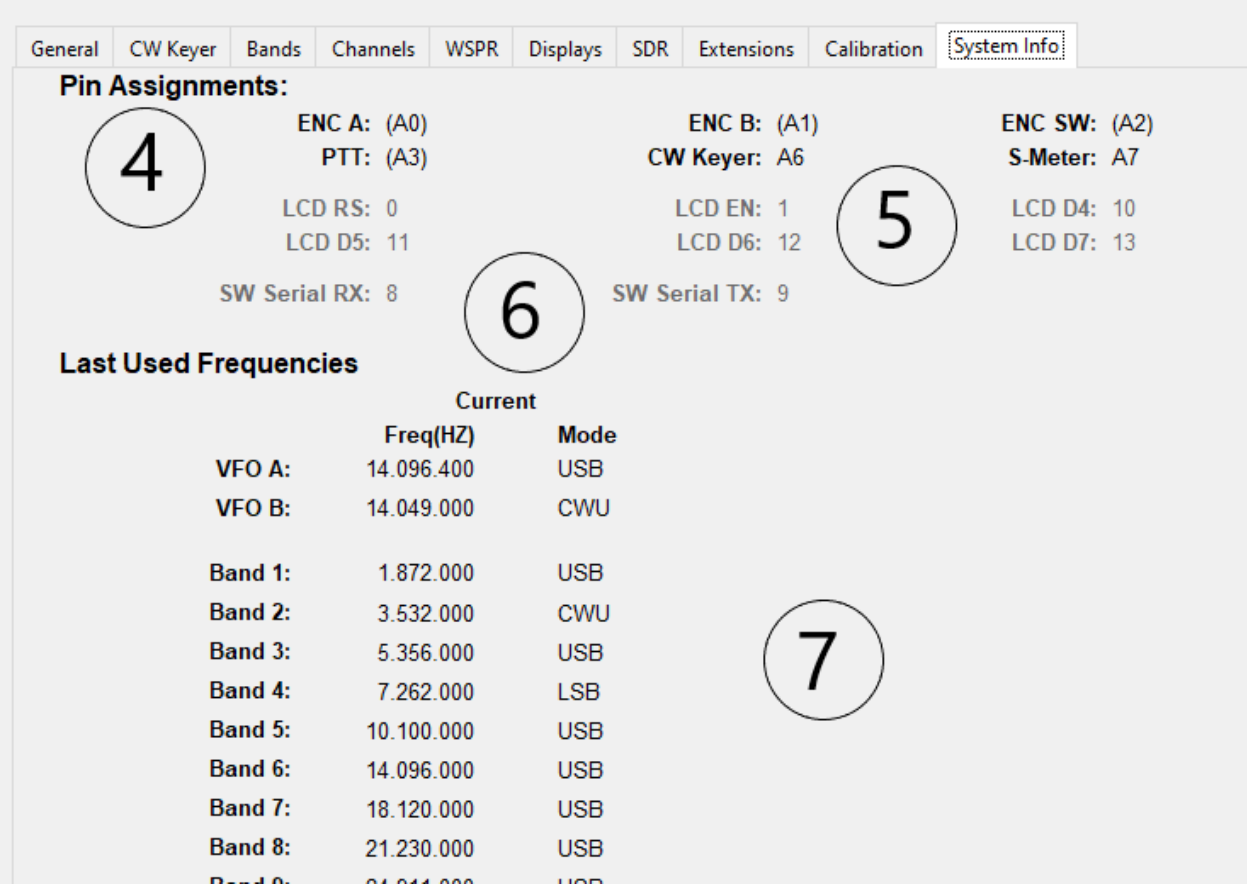
The next screen shows the bottom half of the System Information tab.

4. This section lists the pins assigned to the Encoder. As shown, Encoder-A is assigned to AO, Encoder B to A1, Encoder push switch to Ad2, etc. This is the classic pin assignment for the CEC software.

5. Below the encoder et. al. pin assignment, are the pins reserved for an LCD display. Notice that they are “grayed put”. This means that they are assigned but not used by the firmware (because we have a Nextion screen in this case.)

6. Similarly, we see that the Software Serial Pins (labeled SW Serial…) are also grayed out. That is because this version of the firmware is using hardware serial for communication.

7. CEC stores the last frequencies and mode for the VFOs (A and B) as well as the last frequencies used on each Ham band.



Final Thoughts

This *SE* will probably always be a work in progress. As new features are added within the CEC firmware, the tuning of these features will require changes and enhancements to the *SE*. I am hopeful that the current “tabbed” UX will make it possible for the SE to keep pace with CEC innovation!

For a piece of software I never intended to write, it has certainly taken on a life of itself!

I look forward to your comments as you use the uBITX Settings Editor.

73

Mark
AJ6CU