W0EB/W2CTX
uBITX I2C/Digital I/O Modification and Firmware Instructions

This manual covers the latest uBITX modifications to allow I2C control of the display and accompanying software

by: Ron Pfeiffer, W2CTX and Jim Sheldon, W0EB

INTRO:

First, you MUST perform the hardware modifications to your uBITX Raduino card and LCD display or you will not be able to utilize this software as it no longer uses the A3, A6 and A7 lines for CW keying and SSB microphone push to talk. These functions have all been transferred to the digital pins D8, D9 and D10 lines vacated by the original LCD display that were made available by utilizing the I2C bus already on the Raduino card.

An additional hand key/external keyer jack should be installed as well as an additional pushbutton switch for the A/B vfo swap and newly added SPLIT function.

Second, you will need to unzip the ubitx_I2C_Vx_xx.zip file into your Arduino directory. This file contains ALL the necessary files to allow the program to be compiled and uploaded to the uBITX by the Arduino IDE program. Included in the provided zip archive is the LiquidCrystal_I2C library which must be installed in the Arduino IDE's "Libraries" directory if you do not already have it. The Arduino IDE MUST be upgraded to Version 1.8.5 or later in order to properly compile and upload this firmware.

In order to use the I2C firmware, you must modify your Raduino card by following these instructions.

I2C bus Pick-off Modification:

In order to use the 2 line (plus ground) I2C bus that uses the signals labeled "SDA" and SCL to communicate with addressable devices on the bus (the only one so far on the Raduino is the Silicon Labs Si5351A clock/VFO generator) several things have to happen – for safety to the 5351A we need to make sure the data signals do not exceed 3.3 volts as that is what the 5351 uses. We must tap into the bus on the Raduino card and no real provision was made to make the SDA and SCL signals available to devices external to the existing Raduino. However, since the Arduino Nano's header pins were soldered to the Raduino and left to protrude by a fair margin out of the back of the Raduino card, it's relatively easy to access those pins. We also need +5 volts and ground to power the I2C controller board and its attached display (more on the display and board later).

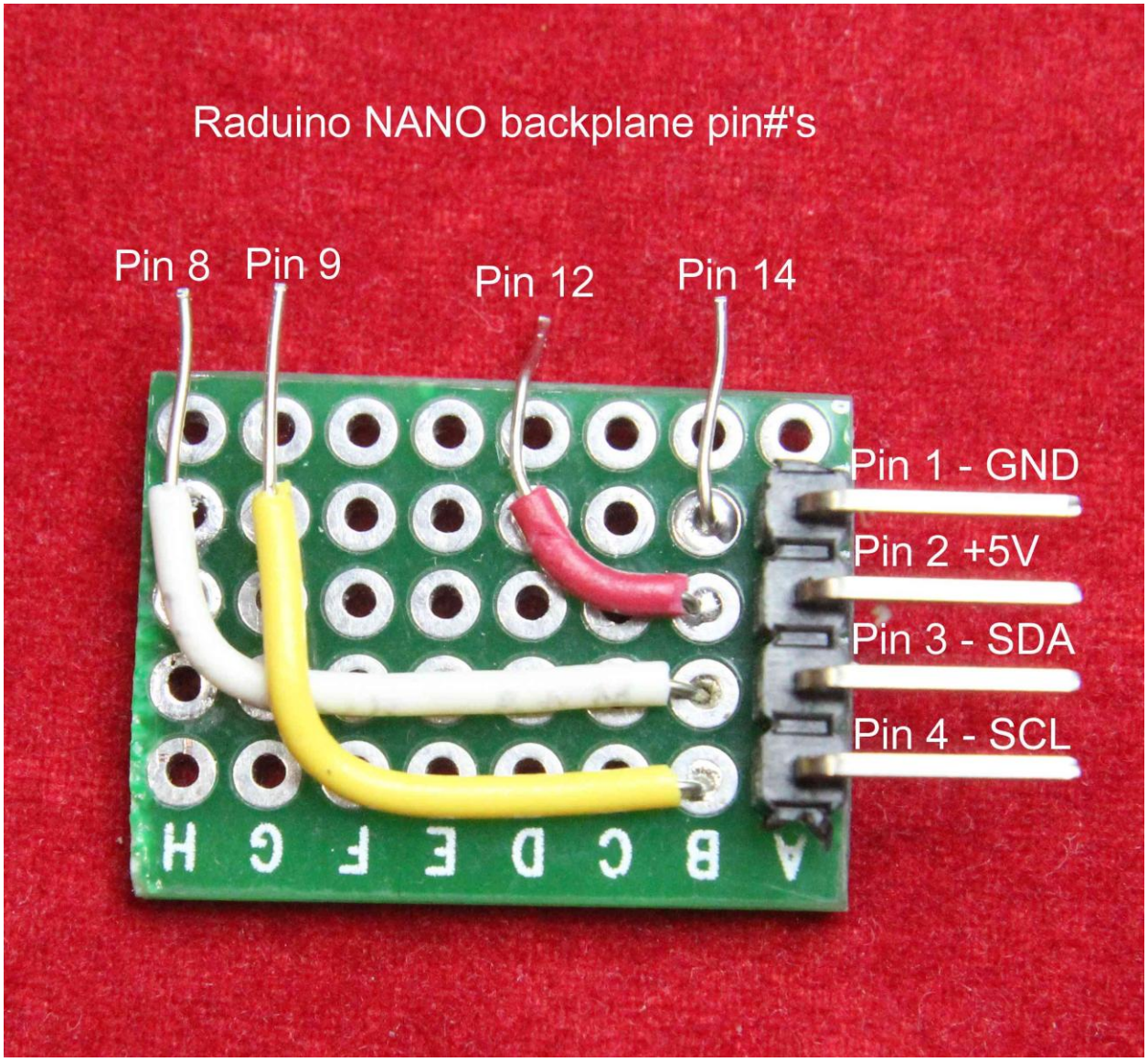
Modifying the Raduino:

Carefully unplug the “Digital Cable” from the Raduino, unplug the Raduino from both the display and the uBITX main board and place it on your workbench. It would be prudent to use an anti-static mat and all the anti-static precautions you can to avoid damage to the sensitive parts on the Raduino and its attached Nano. Put the uBITX main board aside so you don’t accidentally splash solder into it while you are working on the modification.

Next, cut a small piece of perf-board (I use the Proto Board type as it has plated through holes and is very easy to solder to) about 8 holes wide and 5 holes high. Solder a 4 pin DuPont header into the bottom 4 holes on the right side of this Proto-Board being careful not to short 2 or more pins together.

Looking at the board with the header to the right and one row of holes open at the top, the header pins are numbered 1,2,3 and 4 with pin 1 being the top and 4 being the bottom. Carefully solder 4 pieces of small diameter solid #26 or #24 gauge insulated hookup wire to the 4 pins with black to pin 1, red to pin 2 white to pin 3 and yellow to pin 4 (you can use any colors you like, just make sure they connect to the right pins). Bend the wires on the top side of the board as shown in the picture on the next page and strip the ends near the top of the board about a quarter inch. The black wire is real short and since it will go to the ground pin, cut it to length and just remove the insulation completely or it will get melted anyway when you

solder the board's pins and hookup wires to the Raduino.



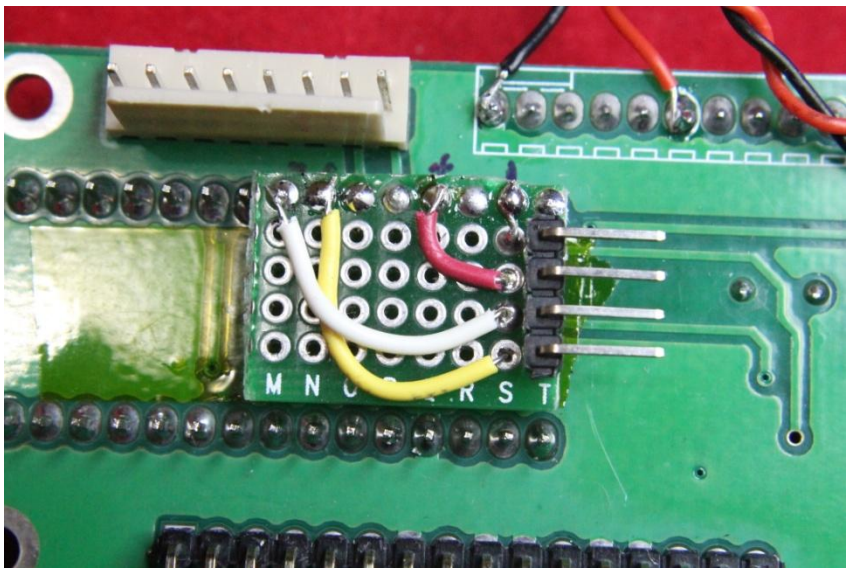
Now that you have your little interconnect board made, it's time to install it. Carefully set it over the Nano's pins that protrude quite far from the back side of the Raduino board with the Nano's pins inserted into the top row of holes (the ones the bare wire ends are located near) Pin 14 on the right is ground and the one just to the right of that is labeled Vin. Do not use Vin for your +5V connection because it could be connected to the +12 input on some versions of the Raduino. Therefore use Pin 12 for picking off the +5

volts. The only holes that need to be soldered are the ones for pins 8, 9 12 and 14 though if you are careful it doesn't hurt to solder a couple more just for stability of the board.

Before soldering, carefully check your wiring to insure that the header pins on the right side of the board go to the proper pins on the Nano (an ohm meter continuity check would be a good thing to do and you can check to make sure no pins are shorted together at the same time).

Make sure the pickoff board is raised above the Raduino slightly and that the connections to the DuPont header do NOT contact the Raduino's backplane. A small piece of the perf board placed underneath while soldering will keep it above the backplane. Just to be safe, you could put a piece of some kind of tape over the Raduino's back plane under the pickoff board before soldering it in place. Just about any tape will do but if you have some Kapton tape that would be ideal. I used abrasion resistant Kapton as I had a roll on hand.

Once at least the 4 required pins are soldered, it should be solid enough not to worry about.



Plug the Raduino back into the uBITX main board and hook it back up to the “Digital Plug”.

Since we now have the I2C bus (that originally set only the Si5351A clock/VFO chip’s frequency) brought out to some easy to use connections, it’s time to make those connections control the LCD display.

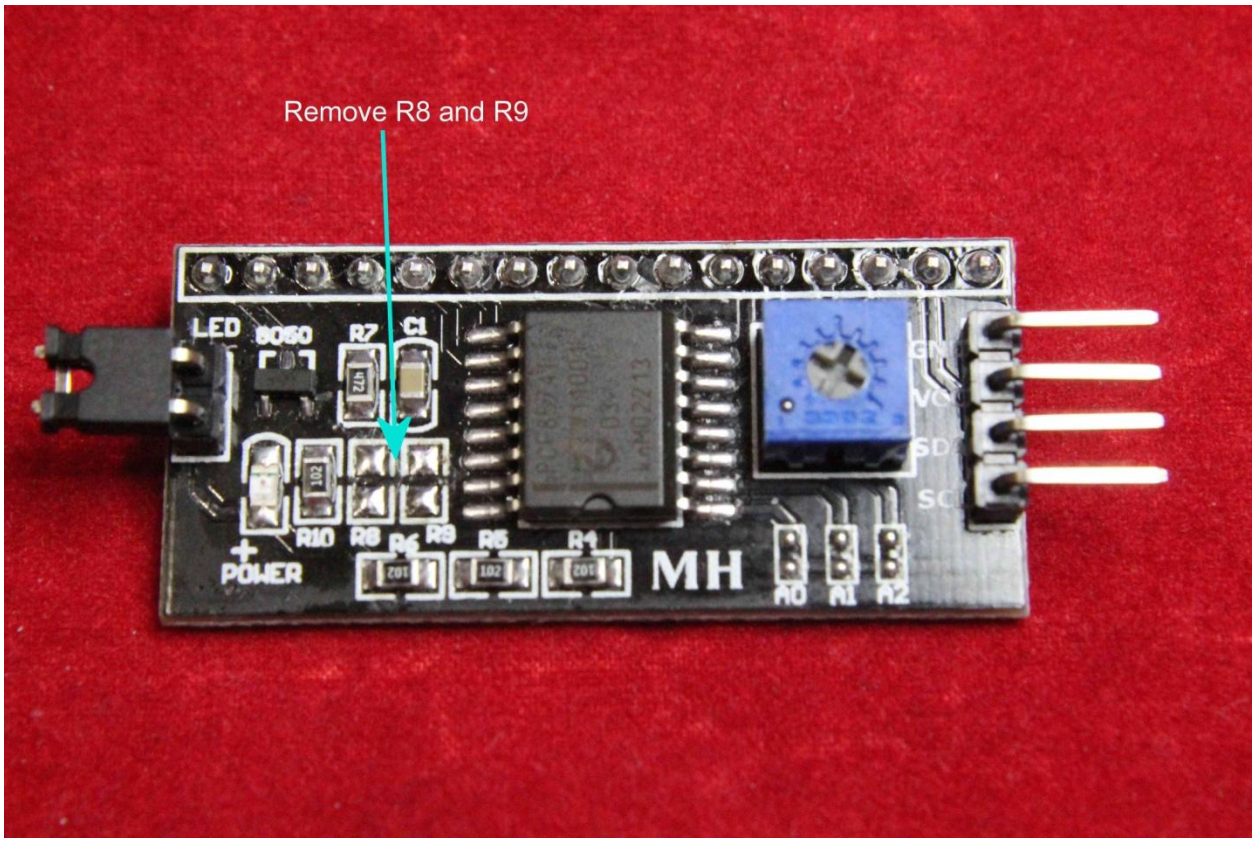
You have 2 choices here – you can remove the original display and carefully de-solder all 16 pins that were used to connect it to the Raduino or pick up another 1602 type 2 line X 16 character display as it will be necessary to install an I2C controller on the back of the display. If you want a blue rather than the provided yellow back lighted display, here would be a good place to change it out.

There are a number of I2C display controllers available and not all of them come from the same mould. The controller I used, came from Amazon.com and their listed seller was “IC Module”. It has 16 male pins already soldered to the back. The module can either be soldered to the back of a standard display or plugged into a female socket that has been soldered to the back of a standard display. Either way is acceptable.

We do have to be careful which type of controller we use though. The I2C bus as implemented on the Raduino to control the Si5351A chip is limited to the use of a 3.3 volt maximum control signal as that’s all the Si chip can safely handle. As received, the controller module from Amazon has 2 pull up resistors on it that have to be removed. They are 4.7K in value and pull the SDA and SCL lines up to 5 volts. If these resistors were left in place, the controller would put 5 volts on the SDA and SCL lines of the I2C bus and that would pretty quickly destroy the Si5351. Fortunately, these modules use the PCF8547T chip which has “Open Drain” outputs to talk to

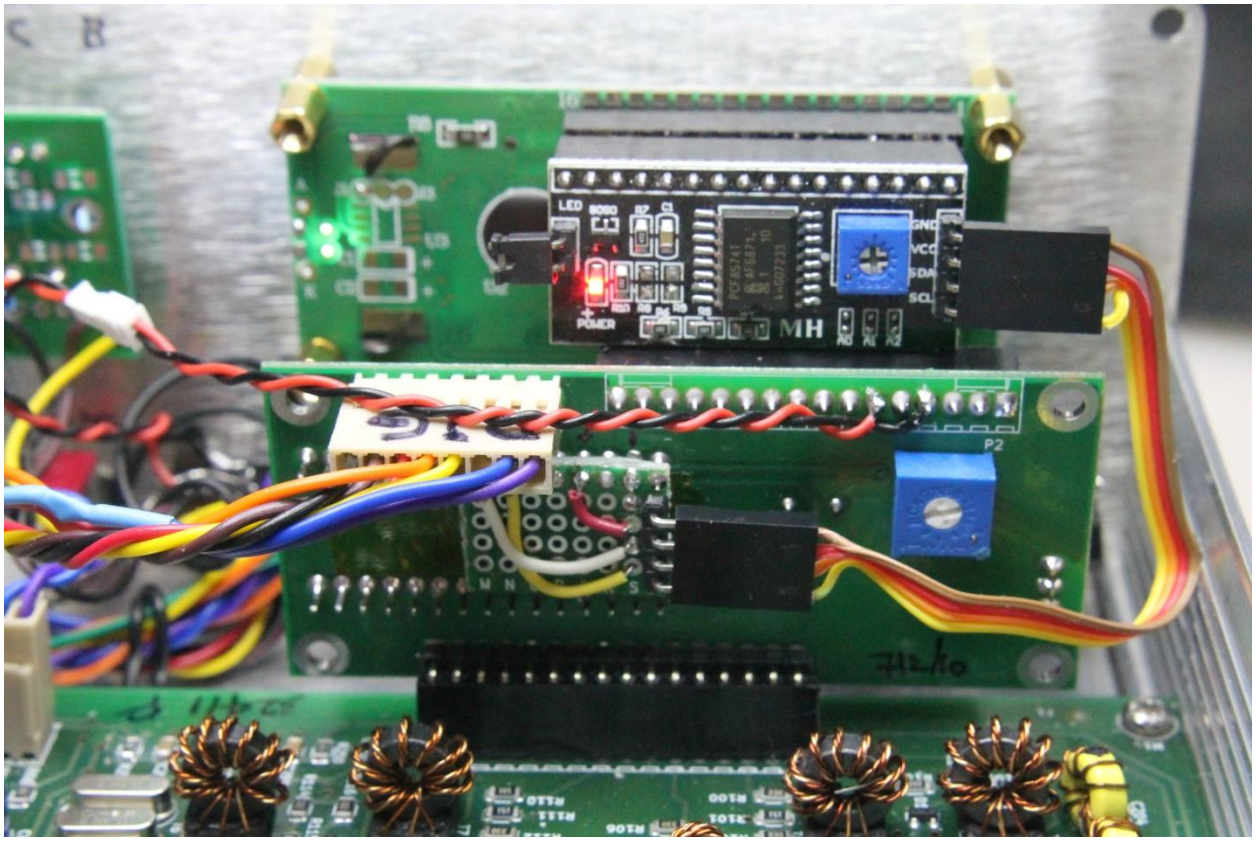
the I2C bus and we can safely remove the 4.7K pull up resistors between the 5 volts and the SDA/SCL line. This leaves the 3.3 volts placed on the bus by pull up resistors already mounted on the SDA/SCL lines close to the 5351 to be switched using the PCF8547T I2C controller chip's open drain outputs. No strain no pain to the 5351 and the lower switching voltage also helps to diminish any radiated noise from the cabling.

Here's a picture of the controller with the resistors removed (labeled R8 and R9 on my particular module).



Next we need to make up a 4 wire, female to female DuPont cable, with straight through connections, pin1 to pin1, 2-2 etc. This cable needs to be as short as possible but long enough to connect our little pickoff board on the back of the Raduino card and the 4 pins to the right of the controller shown above when it's plugged in or soldered on to our LCD display for the

uBITX. Shown below is a picture of how this is connected on one of my uBITX transceivers. (You can see the open pads for the 2 missing resistors just to the left of the main IC on the same plane as the lit LED.)



Finally we need a version of the uBITX operating software (Firmware if you will) that has been written to utilize this I2C connection.

The main advantage of using the I2C bus to run the display is it uses only 2 data lines instead of 6. We have now opened up a total of 6 digital input/output lines, D8, 9, 10, 11, 12 and D13 on the Raduino/Nano that can be used for other tasks like Keyer paddle inputs, separate Hand Key, Bug/External Keyer input and PTT input and it also frees up the A3 analog line which was originally used for PTT, and the A6 Line which was originally used for the Key/Paddle inputs. Due to the fact that D13 is a little tricky to

use for digital input (it controls an LED on-board the Nano), we'll leave that one alone for now.

Now that you have modified your uBITX to utilize the I2C bus for driving the LCD display and opened up the 6 digital I/O lines that once were utilized exclusively by the display, there is no reason to continue using the 3 analog lines A3, A6 and A7 for Push to Talk and the Keyer input functions. These analog lines can be better utilized to provide for an "S" meter and maybe a Power Output/SWR meter. However, the metering function programming will be left to others and is not in the scope of what we have done with the I2C and digital keyer/PTT input changes.

The necessary wiring changes aren't overly difficult either. Since we don't normally use PTT (push to talk) in CW mode, we can use the PTT input as the Hand Key/External Keyer input as well. In SSB mode it will be PTT and in CW mode it will be Hand Key. The best way to connect this is by adding a separate Hand Key/External Keyer jack either on the front or back panel of the uBITX. The SLEEVE connection of this jack hooks to ground and the TIP connection should connect to the PTT pin on the microphone connector. This wire then should connect to pin 13 on the old display connector. This is digital input pin D12 on the Nano.

The original key jack now becomes the CW Paddle jack. The TIP should be wired to Pin 4 on the old display connector (D8) and becomes the DOT input from the paddle. The RING should be connected to Pin 12 on the old display connector (D11) and becomes the DASH input from the paddle.

The extra pushbutton switch used to select the A/B VFO's and turn SPLIT on or off needs to be wired between ground and Pin 11 of the old Display connector (D10).

NOTE: There is a distinct advantage in using the same input for both microphone PTT and the hand key input. In an emergency situation where CW is the only mode that might be able convey a message under extreme propagation conditions and key paddles or a hand key are not available, you CAN use the push to talk switch on your microphone as a CW hand key. This might just save you some grief someday.

This completes the hardware modifications necessary to use the I2C display and the digital Inputs for the Paddles, Hand Key, PTT and the A/B selection switch.

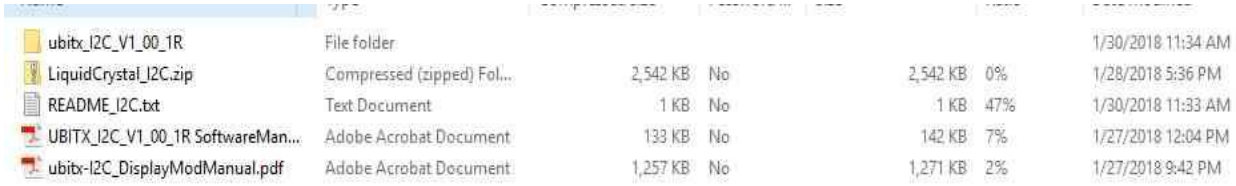
Of course these modifications require software designed to take advantage of them and the original software supplied with the uBITX will no longer function using these modifications.

Compiling, Uploading and Using The Software:

The software is supplied as an Arduino Nano “Sketch” which consists of several routines contained in a specific directory.

In our case, you must be using the Arduino IDE and the IDE must be version 1.8.5 or newer or you may get compile errors. You must also have the LCD IDE library called LiquidCrystal_I2C (provided with this software for convenience) installed in your Arduino IDE’s “Libraries” directory or the program will not compile.

Unzipped, the file should look similar to this (there will be only one PDF file not two and the file names/lengths will differ depending on the version).



ubitx_I2C_V1_00_1R	File folder					1/30/2018 11:34 AM
LiquidCrystal_I2C.zip	Compressed (zipped) Fol...	2,542 KB	No	2,542 KB	0%	1/28/2018 5:36 PM
README_I2C.txt	Text Document	1 KB	No	1 KB	47%	1/30/2018 11:33 AM
UBITX_I2C_V1_00_1R SoftwareMan...	Adobe Acrobat Document	133 KB	No	142 KB	7%	1/27/2018 12:04 PM
ubitx-I2C_DisplayModManual.pdf	Adobe Acrobat Document	1,257 KB	No	1,271 KB	2%	1/27/2018 9:42 PM

I2C devices have a distinct individual address which much be known for the software to address them to transfer data between the device and the microprocessor. On the controller modules received from Amazon, several default addresses have been observed. These are 0x3F and 0x27 (Hexidecimal 3F and 27 respectively). The default I2C address for your particular controller should be supplied by the seller. This address must be hard coded into the firmware in order for the program to control the display properly.

If the seller does not provide the address, look for an Arduino sketch called I2C Scanner. Compile this, load it into your modified Raduino WITH the I2C display controller connected (you don’t have to have the display on it if you haven’t done that yet.) run this program with the IDE’s Serial Monitor

(set to 9600 baud rate) running and it will report back all devices on the I2C bus. The Si5351A will always show up as address 0x60 and any other address will be your display controller. Write this down and don't forget it.

From wherever (desktop, or directory) you have unzipped the main file, move the entire ubitx_I2C_Vx_xx_x folder from there to your "Arduino" directory, wherever on your computer that is located.

Bring up the Arduino IDE, double click on the ubitx_I2C_Vx_xx_x (substitute the actual folder name) directory and once that is showing, double click on the xxxxxx.ino file to bring all the source modules into the IDE's compiler.

Look through the .ino source file until you find a line (around line 108) similar to the one below -

```
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

Notice the first numbers after lcd(are 0x27 which means hexadecimal number 27. This is the I2C bus unique address for your display controller module. You will need to know that address and if it is not 0x27, but something else such as 0x3F, you will need to change the 0x27 to 0x3F (or whatever your own device's unique address is. Whatever, do NOT repeat NOT use address 0x60 as that is reserved for the Si5351 clock generator on the Raduino.

Once you have gotten the correct display controller address into the source code, save it and then run the compiler to check the program for errors. If there are none, compile & upload the software to your Raduino. Put the card back into the computer, connect the I2C cable (make sure the polarity

is right – you don't want to put +5 Volts on the wrong pin of the controller it might be destroyed.

If you did everything right and the program loaded into your Raduino's Arduino Nano properly, you have the Raduino re-installed in the uBITX, you have the I2C display connected properly and your front panel jacks & switches wired like the previous instructions said to wire them, power up the uBITX and you should see a display similar to this: (Will depend on the software version but should be close to this.)

uBITX Operational Modes:

There are 3 operational modes available to the uBITX in this version of the software These modes are selected during power-up via the encoder push button.

The three modes are: Normal, EEPROM UPDATE and ALIGNMENT. (ALIGNMENT is the original "Factory Alignment" programming modified

slightly to add a “Tune” function to set the receiver to a usable standard frequency).

All mode selections occur ONLY at power-up. They are mutually exclusive and when you are finished using the EEPROM UPDATE and ALIGNMENT modes you must cycle the power (Turn the uBITX off and then back on again).

SELECTING MODES:

Normal Mode: General radio operations. Power up your uBITX and do nothing. The normal radio features will be defaulted. This is the mode you will use to operate the uBITX on the air.

EEPROM UPDATE Mode: This mode will accept EEPROM reads/writes via Ian Lee’s (KD8CEC) Windows EEPROM Manager. (In the future there will be a Linux EEPROM manager).

Some of our used EEPROM locations may not agree with those used by Ian Lee in his versions.

******* NOTE: CAT is NOT supported in this version.

In order to use Ian’s EEPROM Manager:

1. Power up your uBITX.
2. When the Version banner is displayed, push the encoder button.
3. Keep the encoder button pushed until “EEPROM” is displayed.
4. When “EEPROM” is displayed release the encoder button. You may now use Ian’s program (follow his directions) to read/write the EEPROM memory locations. (Not for the faint hearted, but it will allow you to change values in the various locations.)

To exit this function close down Ian's program and depress the encoder button until "EXIT" appears on the top line of the uBITX display. Tap the button to Exit. Power cycle the uBITX and it will come up in "Normal" mode with the changes you made applied.

ALIGNMENT Mode: Will present the menu for performing a factory alignment (Ashhar Farhan's original factory alignment code has not been changed other than to add a "Tune" function to allow tuning in WWV or some other standard frequency, so follow the original uBITX instructions for setting Calibration and BFO.)

1. Power up your uBITX.
2. When the Version banner is displayed push the encoder button.
3. Keep the encoder button pressed until "Alignment" is displayed.
4. Once "ALIGNMENT" is displayed, release the encoder button and "Exit" will appear. Turn the encoder knob to the right to select the desired alignment function, either Tune, Calibration or Set BFO. After these functions are complete, turn the encoder knob to the left until "Exit" appears and press the button to exit. Even though the uBITX will appear to come up in "Normal" mode on exit, you will still need to power cycle the radio to make sure the alignment changes are applied.

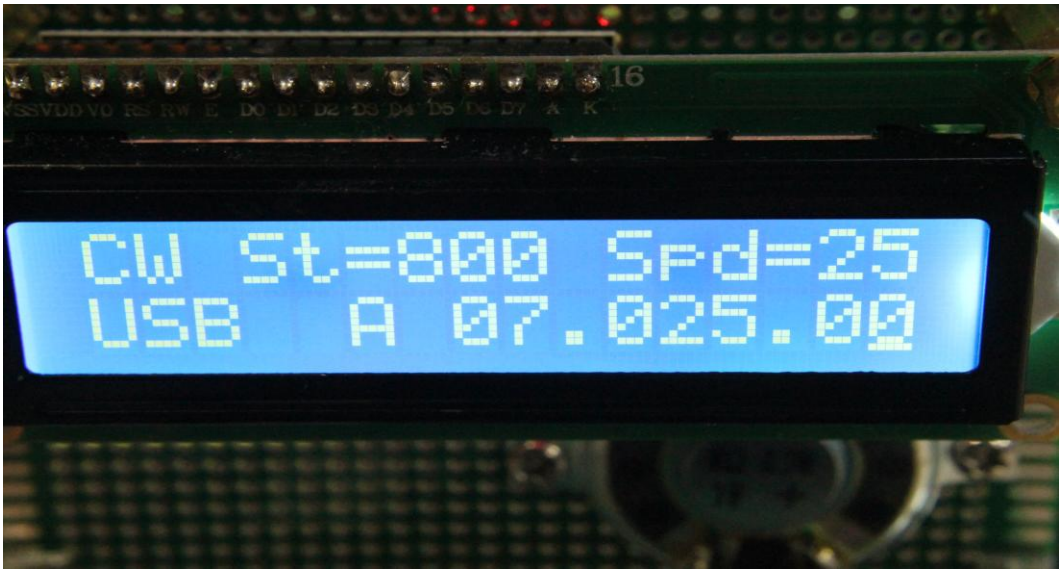
Turning the knob to the right after "Exit" appears will get you to "Tune": short press the encoder button to go into tune. Use encoder to select the frequency you wish to use for calibration. Use the same short press of the encoder button to change the cursor under the digits for frequency increments. A Longer press will exit the "Tune" portion of the alignment menu, leaving the uBITX tuned to that frequency and you may then select

the calibration or BFO adjustment procedures as normal. After adjusting the Calibration or BFO you press PTT (NOT the encoder/function button) to save the item. Pressing PTT exits the Calibration or Set BFO procedure back to the Alignment Menu. Turn the encoder knob to the left until “Exit” appears and press the encoder button.
“Power Cycle
Radio”
Will appear. Turn the uBITX off and then back on to make sure the changes you just made are saved to EEPROM.

NORMAL MODE FEATURES:

Many people expressed the desire for a different method of changing bands and tuning speed as the original method was sometimes difficult to control.

We removed the accelerated tuning feature as the original speed-up timing caused the accelerated speed change to happen far too quickly for many people and it was hard to set an exact frequency if you happened to slip. The frequency would sometimes jump by 2-300 KHz and then you had to crank forever to get back to where you were. Instead, we created a function where a short press (tap) of the encoder button would change the position of an underscore cursor and allow you to select the digit of the frequency you wished to change. This allows for more precise quick setting of the frequency.



The cursor under the digit can be moved by tapping the encoder button.

MENUS:

Now for the menu I mentioned near the beginning of this manual.

To enter the Menu, power up your uBITX and wait for it to boot up into the default which is “Normal” mode.

Press the encoder button until “Menu” is displayed. Immediately release the encoder button and a display full of changeable items will display.

Initial Menu Display

The picture shows the display just after entering Menu mode. The cursor is under the C in CW. A short press of the encoder button will change the mode from CW to SSB and exit the menu.

The CW, Sidetone Frequency and Keyer Speed items won't show in SSB mode as, of course they are not used for SSB.

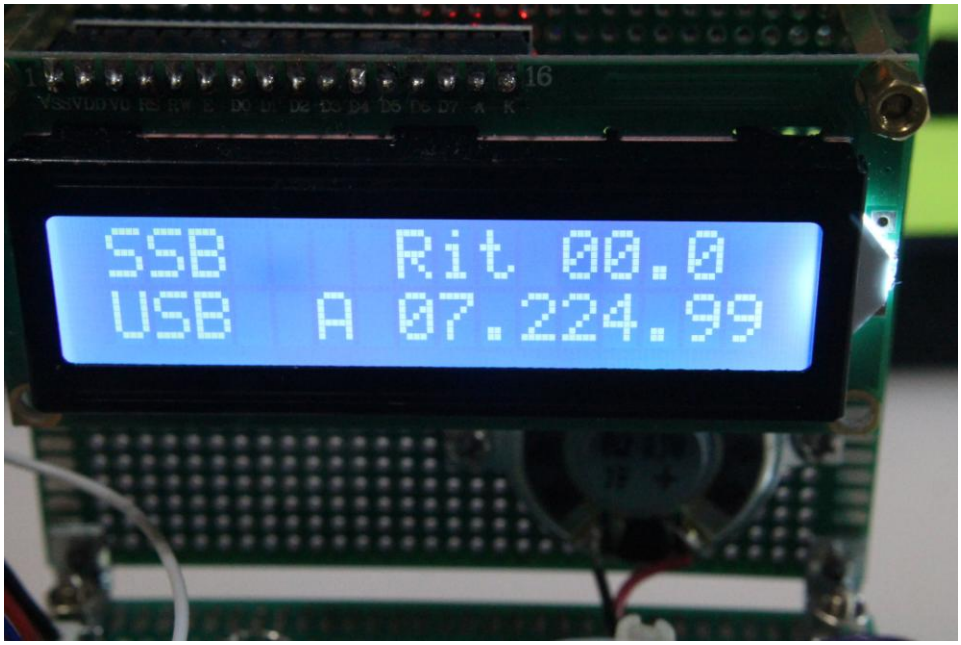
Entering the menu again, the cursor will appear under the S in SSB. A short press of the encoder button will change the mode back to CW and again exit the menu.

Anytime you enter the menu, the underscore cursor will appear under the item that will be changed by a short press of the encoder button. The cursor can be moved either right or left by turning the encoder knob. Once you have the cursor under the item you wish to change, a short press of the button will carry out the change.

When changing the sidetone frequency, once the item is selected, the current tone frequency will be displayed, the current tone will be sounded in the speaker/earphones and turning the encoder to the right or left will increase the tone frequency in 10 Hz increments. Once you have the tone you like, press the encoder button. The tone value will be saved and the menu exited.

Keyer speed works the same way – put the cursor under the current speed value and select it with a short press of the button. Turning the encoder knob right or left will increase or decrease the keyer speed in 1 WPM increments. Press the button again to save and exit.

RIT is enabled by a longer press on the encoder button. When you get the message "RIT ON" on the top line of the screen, release the encoder button and the following display will appear (In CW mode it would display CW instead of SSB but RIT works the same way.)



Even though no cursor is shown, turning the encoder knob changes the RIT increment on the top line of the display. Turning the knob to the right will increase the receiver frequency in 100Hz increments above that shown on the bottom line of the display and the + sign will appear in between the R and the first KHz digit. Turning the knob to the left will decrease the frequency and if the frequency is below that displayed on the bottom line, a minus sign will appear between the R and first digit. When RIT is on, the transmit frequency remains fixed on the frequency shown on the bottom line of the display. RIT is limited to plus or minus 5 KHz from the main displayed frequency.

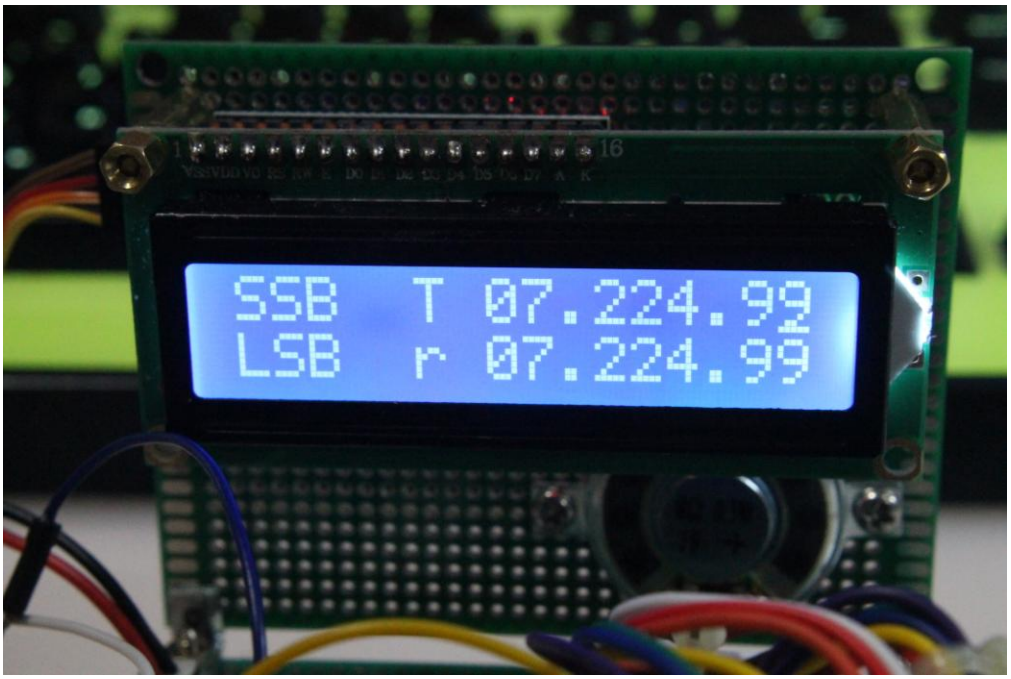
To turn RIT off, press and hold the encoder button until RIT OFF appears and immediately release the encoder button.

CW is normally received using upper sideband by current convention but this can be changed by using the USB/LSB change function in the menu as well.

Changing bands can be accomplished in a couple of ways. The easiest doesn't even require entering the menu. Simply place the cursor under the MHz digit on the display and turning the encoder right or left until the desired MHz frequency of the desired band is reached. Using the short press (tap) feature of the encoder button select and set the other digits to your desired frequency.

The second way is to enter the menu, move the cursor under the MHz digit of the frequency and short press the button. This will bring up the band select menu and you can use the encoder to select from the 80, 40, 30, 20, 15, 17, 12 or 10 meter bands. Pressing the encoder button will enter that band and set the frequency to the QRP calling frequency (dependent on CW or SSB modes) for that band/mode and exit the menu. This menu item does not have an exit function other than selecting a band, so if you get here by mistake you will have to select one of the 8 available bands in order to exit.

A/B VFO select and SPLIT mode. This is operated using the extra pushbutton switch we installed and connected to the D10 digital I/O pin on the Raduino. A short press (tap) on the A/B – SPLIT switch will toggle between the A and B VFO frequencies. A LONG press will turn SPLIT on. While in SPLIT mode, tapping the A/B button will switch between the transmit (top line of the display) and receive (bottom line) frequencies. The selected VFO will be indicated with a capital R or T and the non-selected VFO will be shown by a lower case r or t. To turn SPLIT off, simply do another long press of the A/B – SPLIT button and it will be turned off returning the display "normal mode" again.



This is what the display looks like when in SPLIT mode.

The last feature of the encoder button is something we recently added as a convenience feature so you could save your operating frequency before shutdown if you wanted to come back to it later instead of having the uBITX go to the default frequency on power up. It's called "Save State" and it's activated by a very long press of the encoder button (approximately 5 seconds) and "Save State" will appear on the display. Releasing the button saves the frequency as the "default" in EEPROM so the next time the uBITX is turned on it will come up on that frequency. All other items such as mode, VFO, tone, speed and USB/LSB are already saved in EEPROM when they are changed we just save the frequency in the Save State function.

Questions, comments (positive/negative, we're pretty thick skinned) or complaints should be directed to the authors at our respective email addresses listed below:

Jim Sheldon – W0EB – w0eb@cox.net

Ron Pfeiffer – W2CTX – w2ctx@yahoo.com