Pocket FT8: A Palm Size FT 8 Decoder / Encoder For SOTA - Teensy 4.1

Scope

This document is written to help you build your own self contained FT8 Rig. Listed below are the main features of the project.

- Small Size, 3.5” X 2.75” X 1.125”
- Mono Audio Input and Output for Transceiver Interface
- PTT Output To Control Transceiver
- SD Card Contact Logging
- 320 X 480 Resistive Color Touch Screen
- GPS Receiver For Setting Teensy Real Time Clock and Station Maidenhead Locator

Attributions

This project is based on the FT8 Decoding Library by Karlis Goba: https://github.com/kgoba/ft8_lib

DSP Audio Architecture

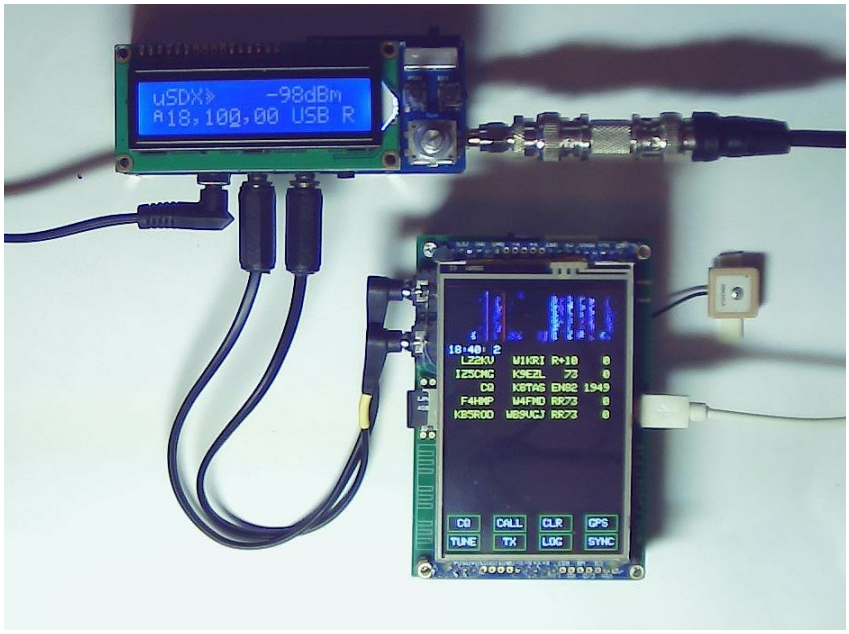
Decoding FT8 requires significant data storage and processing speed.

The Teensy Audio Library has been modified to allow Analog to Digital conversion to be run at the rate of 32,000 samples per second. The receive data is decimated by a factor of 5. This allows Receive Audio Data Processing to be done at 6400 Hz. Using the 6400 Hz audio processing rate along with a 2048 FFT to process the received audio yields a bin spacing of 3.125 Hz.

The algorithms developed by Karlis Goba use the 3.125 Hz spaced FFT bins to be screened in both frequency and time so that errors in symbol frequency and time reception can be overcome to provide really great FT8 decoding. The final spacing of the FT8 algorithms is 6.25 Hz as required by the FT8 protocol.

Target Transceiver

This unit has been designed to directly interface with the popular uSDX transceivers such as the uSDX Mono or the uSDX Tri-Band Pocket Radio.



The photo below shows the pocket_FT8 display on boot up.

The FT8_Log.txt is the filename used to store operating log data. The line “W5BAA:EM00 is the raw text stored in a file entitled “Station_Data.txt” which you can create to store your station details.
The lines “W5BAA” and “EM00” is the parsed station data.

The Synchronizing With RTC and FT8 Synched With World Lines are displayed at startup while the unit synchronizes the FT8 decode algorithm with the Teensy 4.1 Real Time Clock.

Finally, the last line in green shows the revision level of the software.

Decoded FT8 Signals

This photo shows decoded signals after the unit is synchronized. Up to ten FT8 Signals may be decoded and displayed per FT8 period.

Overall Architecture

The project consists of the following hardware modules:

Teensy 4.1
The Teensy 4.1 has internal Analog to Digital (ADC) inputs. Digital to Analog (DAC) outputs are provided via an external PT8211 DAC chip interfaced with the Teensy 4.1 via the I2S protocol.

The Teensy 4.1 is supported by the feature rich Teensy Audio Library found here:
https://www.pjrc.com/teensy/td_libs_Audio.html

The original source for the Teensy 4.1 may be found here: https://www.pjrc.com/store/teensy41.html

Other sources include Sparkfun, Adafruit and Mouser.

Adafruit 320 X 480 TFT Touchscreen
https://learn.adafruit.com/adafruit-3-5-color-320x480-tft-touchscreen-breakout

Microchip MCP 3422 Analog to Digital IC
https://www.microchip.com/en-us/product/MCP3422

Princeton Technology PT8211 Digital to Analog IC
https://www.pjrc.com/store/pt8211_kit.html

u-blox NEO-6M GPS Module with Battery Backup
https://www.amazon.com/GY-GPS6MV2-NEO6MV2-Antenna-Arduino-Control/dp/B07QMTM5YM

The Teensy 4.1 communicates with the MCP 3244 A/D via an I2C Bus while the Touchscreen Display is interfaced via an SPI Bus.

The PT8211 communicates with the Teensy 4.1 via a simple I2S Bus.

The Microchip MCP 3422 Analog to Digital IC is used to interface the Touch Screen touch point coordinates to the Teensy processor. One may ask “Why not just use two of the Analog Pins on the Teensy?”. An attempt was made to do just this. However, the high rate of A to D sampling required by the Teensy Audio Library swamps out polling of the touch screen analog data. So, the MCP3422 is used so the touch point data may be fed to the Teensy 4.1 via the I2C Bus.

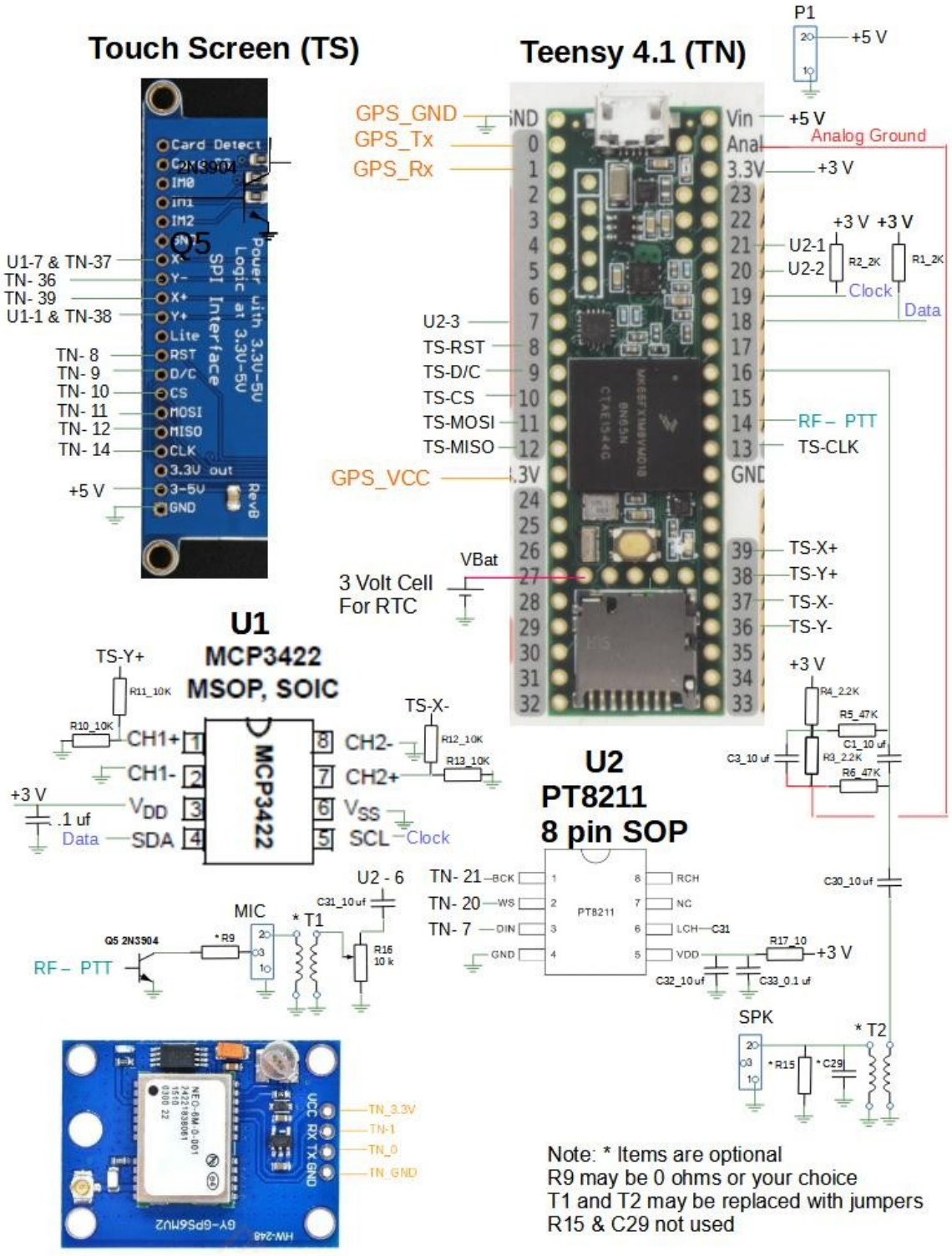
The Teensy communicates with the GPS receiver via a serial data port. The GPS data is interfaced using the NEMA protocol. The software uses the NEMA data to set the RTC and Station Maidenhead Locator.

The PT8211 is a standard output device in the Teensy Audio Library.

Module Connections

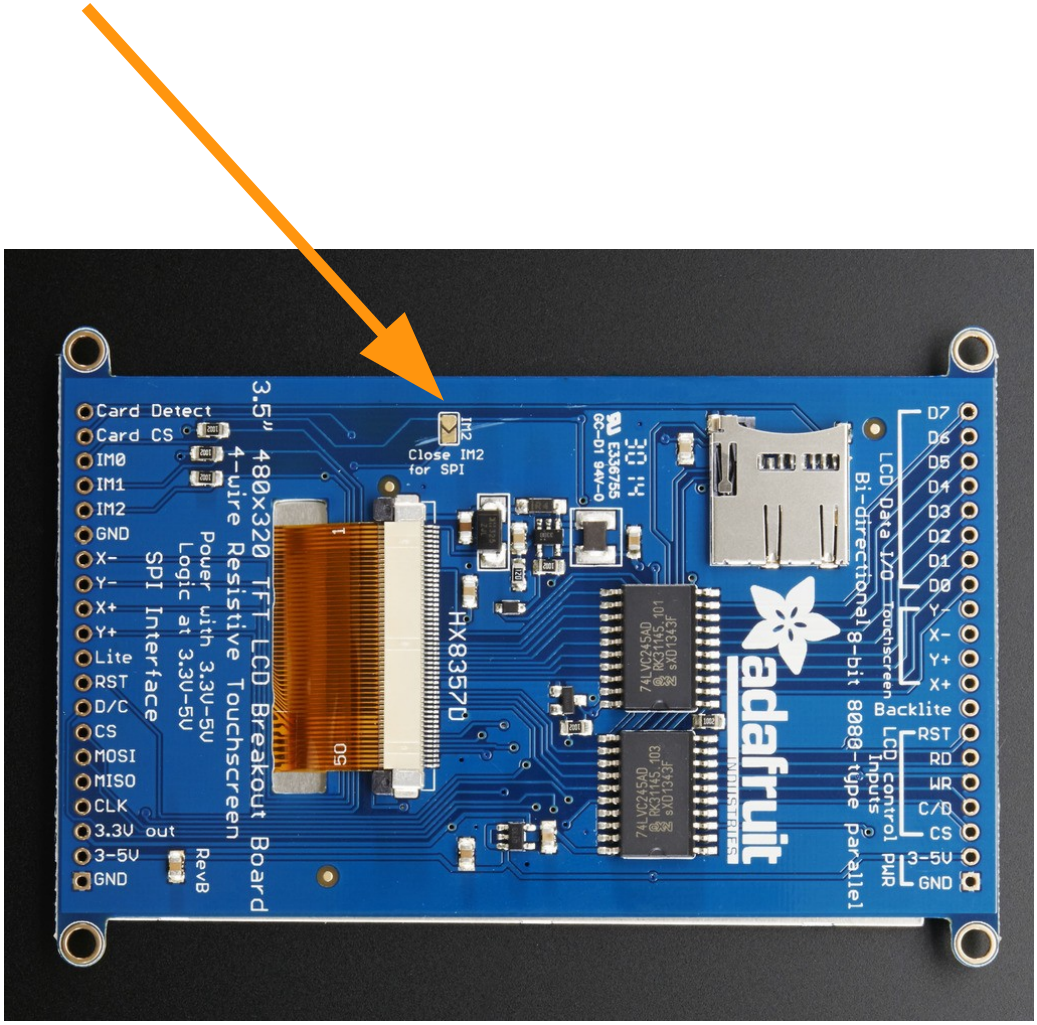
The project connections are shown on the drawing shown below.

The unit may be assembled using perf board and wire wrap wire to make connections between modules.



Adafruit 480 X 320 Display Gotcha!

The Adafruit Display requires that you fill in solder bridge IM2 in order to put the display in the SPI Interface mode. If you do not install this bridge you will only get a totally white screen.



GPS Receiver NMEA Sentences

The application software uses the TinyGPS.h library for parsing the NMEA sentences produced by the GPS receiver module. This library uses the NMEA GPRMC Sentence or the GPGGA Sentence to provide date-time data. So, please ensure that your GPS module produces one of these two sentences at 9600 Baud. Also, limit the NMEA sentences to only these two to prevent overloading of the serial data connection between the GPS receiver module and the Teensy 4.1. This will insure proper parsing of the NMEA sentences.

Construction Hint

In order to have a low vertical clearance between the project board and the Adafruit Display we build our units without using headers and pins between the Teensy 4.1 board and the project board.

Also, with the severe shortage of IC’s in these perilous times we have found that it is necessary to scavenge parts from previous projects. So, being able to easily remove a Teensy from a project board has been most helpful.

To this end, we install modules such as the Teensy 4.1 by first mounting the Teensy on the project board with double sided foam tape. Only small strips of foam tape near the ends of the board are necessary. When mounting the board carefully align the Teensy so that the pin holes on the Teensy line up with the holes in your project board.

Next, connect the Teensy pin holes to your project board pin holes by soldering wire wrap wire with the insulation removed in both mating holes. Only connect the pins which are used. This helps promote laziness and reduces the chance of killing your Teensy.

Installing Firmware on Teensy 4.1

The firmware is loaded by using the USB Serial boot loader provided with the Teensy 4.1.

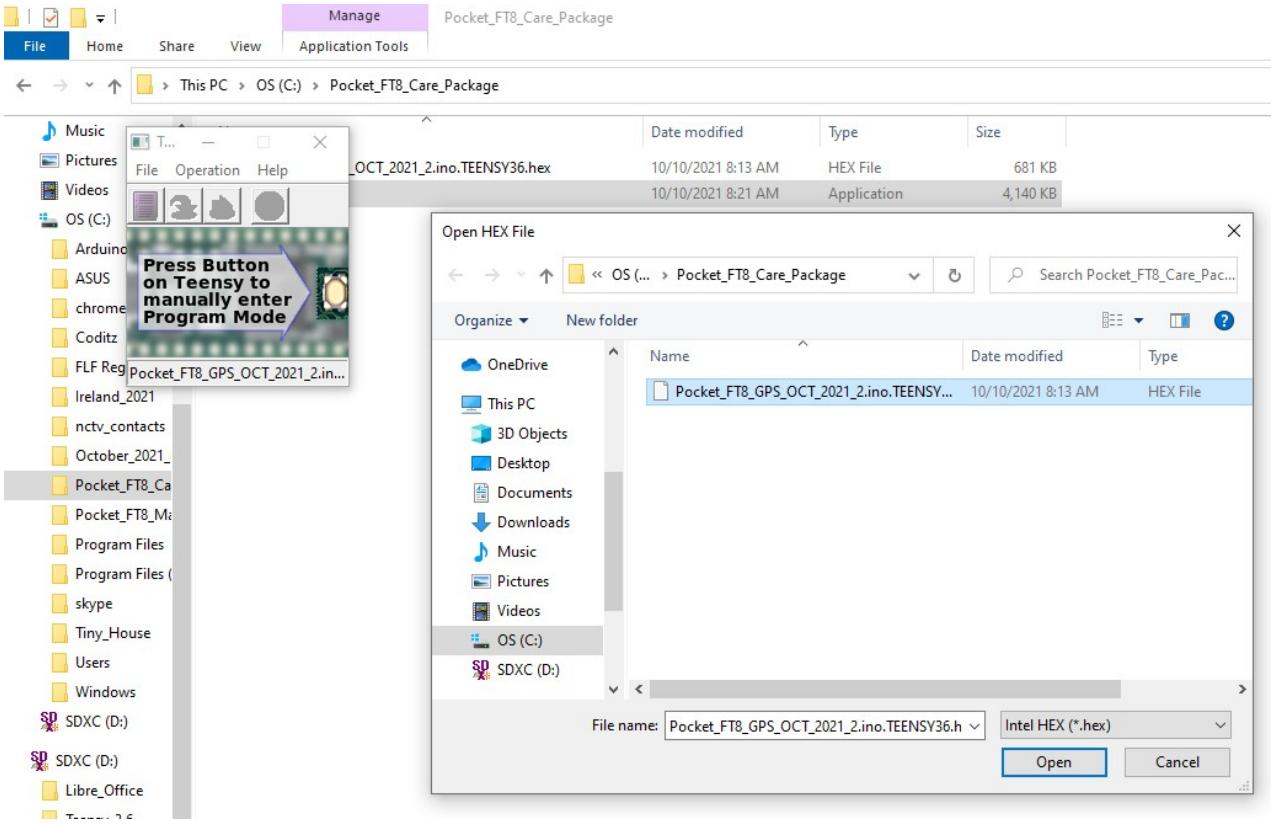
When you first connect the Teensy to your computer using a USB be patient and allow the Teensy and Computer go thru the handshaking ritual to establish the serial connection for downloading firmware.

The firmware is supplied as a hex file which can be loaded with the TEENSY.EXE application. You can download TEENSY.EXE from this site: https://www.pjrc.com/teensy/loader_win10.html

This website has complete instructions on using TEENSY.EXE.

The Pocket_FT8 firmware is provided as a file such as :
Pocket_FT8_GPS_OCT_2021_2.ino.TEENSY41.hex

A screen shot of the firmware being loaded is shown below:



Real Time Clock

The Teensy Real Time Clock (RTC) is used to display relative time and to mark log entries with date – time groups.

The software may be manually synchronized using the touch screen button marked “SYNC”. This technique is copied from the work done by Karlis Goba and is a very effective way to manually sync the Pocket_FT8 rig with the current worldwide FT8 traffic.

A 3 volt coin cell battery is required to be connected to the Vbat pin on the Teensy 4.1 to make the RTC work.

Setting Your Call and Location

Your call and location data are stored on an SD Card as simple text file. Use Notepad to create a text file called “Station_Data.txt”.

The content of this file should be a single line in the form of Your_Call:Madienhead_Locator.

For example here is the line I use: W5BAA:EM00.

Building Your Own Version

The Teensy 4.1 sketch source code is provided as part of the project. Setting up your IDE and getting the source to build is your responsibility.

To adjust the sample rate in the Teensy Audio Library from 44.1 kbps to 32 kbps for the Pocket-FT8 Project please do the following:

1) Please find in the project software care package folder a file named “AudioStream_32k.h”.

2). Using your file explorer, drill down to the Teensy Cores Folder on your Arduino IDE installation to find the folder which is similar to the path below:

C:\Arduino_2021\Arduino\hardware\teensy\avr\cores\teensy4

3). In this folder you should see a file named “AudioStream.h”.

Place a copy of “AudioStream_32k.h” in this folder.

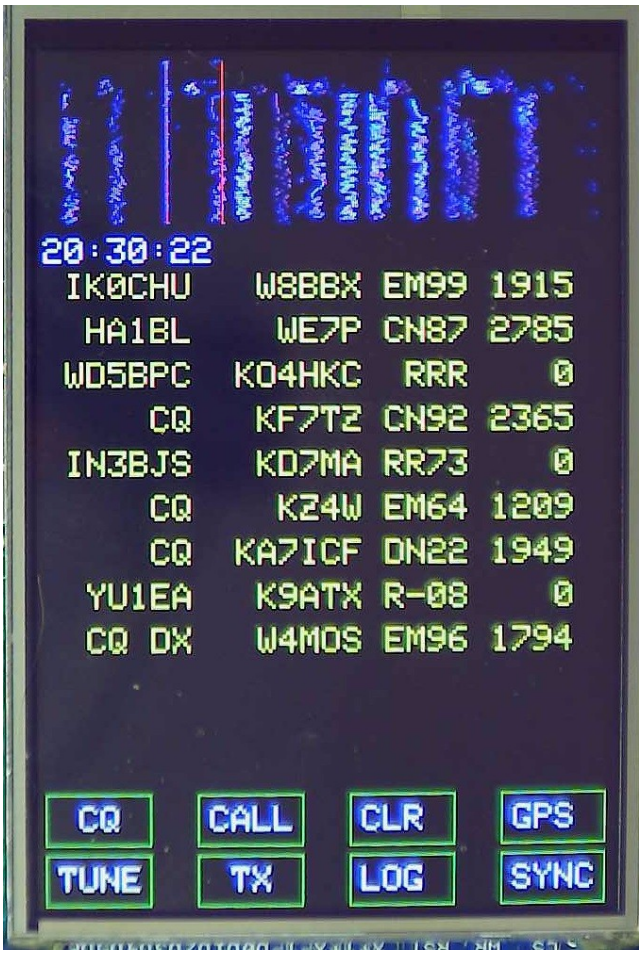
4). Rename the “AudioStream.h” file to “AudioStream.h_original.h”

5). Rename the “AudioStream_32k.h” file to “AudioStream.h”.

6). Restart your IDE.

Hope for the best!

Pocket_FT8 User Operation



The unit uses the Portrait Mode of display and the hardware has been designed so that the Pocket_FT8 may be operated in your hand. The unit shown above is my prototype.

The user is presented with a number of data displays, touch buttons and touch areas as shown above.

Display Items
Starting at the top, the received audio / FT8 spectrum / waterfall is shown. The vertical red line on the spectrum display shows the FT8 transmit base audio frequency.

Immediately below the waterfall the RTC Time is displayed as the default option. If the GPS mode of setting your Maidenhead location is activated, the GPS Time is displayed to the right of the RTC time.

In the middle pane the received FT8 traffic is shown. Up to ten FT8 signals may be decoded and displayed for each FT8 period.

Touch Areas

Two touch areas are provided.

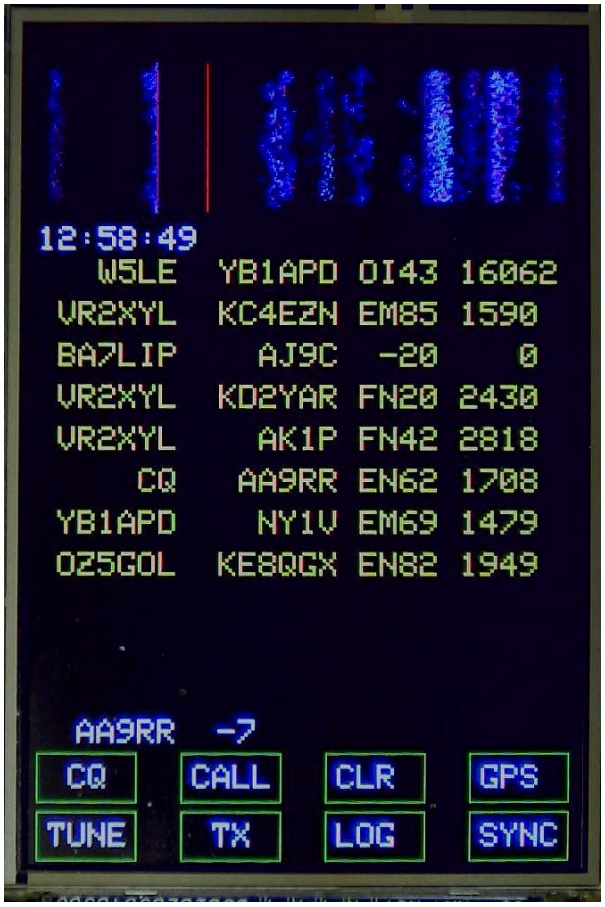
One is the FT8 waterfall. Touching this area moves the FT8 transmit audio frequency. The red vertical line moves to show the selected FT8 transmit base audio frequency.

The other touch area is immediately below the RTC Date Time or GPS time line. You may touch one of the displayed FT8 messages to capture the call sign of the transmitting station. The call of the selected station is displayed above the two rows of touch buttons displayed at the bottom of the screen.

In the example shown below on the left, the message “CQ AA9RR EN62 1708” has been touched to capture the call sign “AA9RR”.

Further, the “CALL” button has been touched to compose the message “ AA9RR W5BAA EM00” as shown below on the right.

The value “1708” shown in the decoded FT8 message is the distance between Maidenhead Locator EN62 and Maidenhead Locator EM00, my station Locator.



Touch Buttons

Eight touch buttons are displayed at the bottom of the screen. The buttons will be displayed with a Green or Red outline as explained below:

CQ places the unit in the beacon (CQing) mode. Red means the CQ mode is selected.

Call places the unit in the QSO mode. Red means the QSO mode is selected.

CLR clears the out going message from the message buffer and the out going message display.

GPS is used to synchronize the Teensy RTC with the GPS Time generated by the GPS Receiver Module. When this box is Red, the GPS Time is displayed. If the value shown in 00: 00: 00: 00: the GPS Receiver has not found a valid Time fix. When a valid GPS Time is found, the Teensy RTC is set to the GPS Time. At this point you may touch the GPS button again to turn off the GPS Time display and RTC synchronization. When the unit is turned off and then turned back on the FT8 decoding algorithm will be set to the RTC at 0, 15, 30 or 45 seconds. So, manual synchronization will not be required.

In addition, the Station Maidenhead Locator will be set using the GPS Coordinates data.

Tune puts the transceiver in transmit mode with with a steady tone at the frequency shown by the vertical red line in the spectrum display. And, the PTT output will be driven to ground to key the associated transceiver. Further the TX box will be shown in Red. **Please be careful with this interface. The 2N3904 transistor is the only thing standing between the 5 volt input on the uSDX and the 3.3 volt input of the Teensy 4.1. Don't ask me how I figured this out.**

Tx indicates that the unit is either in the receive or transmit mode. When green, the unit is receiving. When Red the unit is transmitting. Touching this button has no effect. It is simply an indicator.

Log is used to toggles the traffic logging on to the SD card on or off. Red means logging is on.

Sync allows the user to manually synchronize the FT8 receiver with the rest of the FT8 world. The Sync button has no effect on the Teensy RTC or Station Maidenhead Locator.

CQ (Beaconing) Operation Mode

In the CQ mode the user simply touches the CQ button and the button turns red and then the user may observe the traffic. As stations respond to your CQ call, their FT8 messages are displayed in green above the last station call selected. In the case below, Station “VE3XN” at Locator EN93 has responded to my CQ message.

Further, if the LOG button is touched the button turns red and the traffic logging to SD card is turned on.

Please note that when the TX button outline turns red the composed message shown in white is being transmitted. In the example above, the message CQ W5BBA EM00 is being transmitted at the time of 19: 38: 20:.

In the CQ mode four messages may be generated and transmitted as shown below:

CQ W5BAA EM00

VE3XN W5BAA EM00

VE3XN W5BAA -5

VE3XN W5BAA RR73

A traffic manager algorithm is called after each FT8 period is decoded. The algorithm searches the decoded messages for stations calling your station call, in this example “W5BAA”.

If no calling stations are recognized, the algorithm will set the CQ message to be sent on the next FT8 period.

When the algorithm recognizes a calling station the other three standard messages listed above are sent succession. After sending “RR73” the algorithm sends the CQ message on every other FT8 period.

Logged Message Example for CQ (Beaconing) Operation Mode

Shown below is a log of messages received and sent while the unit was in the CQ mode.

19/ 5/2022 19:16:50 W5BAA VE3XN EN93 2146
19/ 5/2022 19:16:50 VE3XN W5BAA EM00
19/ 5/2022 19:56: 5 W5BAA KN8DMK EM89 1763
19/ 5/2022 19:56:50 W5BAA VA3CK FN03 2279
19/ 5/2022 19:56:50 VA3CK W5BAA EM00
19/ 5/2022 19:57:20 W5BAA VA3CK -05 0
19/ 5/2022 19:57:20 VA3CK W5BAA RR73
19/ 5/2022 20: 6:34 W5BAA KF7TZ CN92 2365
19/ 5/2022 20: 6:35 KF7TZ W5BAA RR73
19/ 5/2022 20:13:49 W5BAA WA3GDB EM95 1765
19/ 5/2022 20:13:50 WA3GDB W5BAA RR73
19/ 5/2022 20:14:50 W5BAA WA3GDB EM95 1765
19/ 5/2022 20:14:51 WA3GDB W5BAA EM00
19/ 5/2022 20:15:20 W5BAA WA3GDB EM95 1765
19/ 5/2022 20:15:21 WA3GDB W5BAA RR73
19/ 5/2022 21:27:20 W5BAA K7OGW -02 0

CALL (QSO) Operation Mode

In the Call Mode of operation you will first touch the Call button. By touching the CALL button the QSO mode is activated

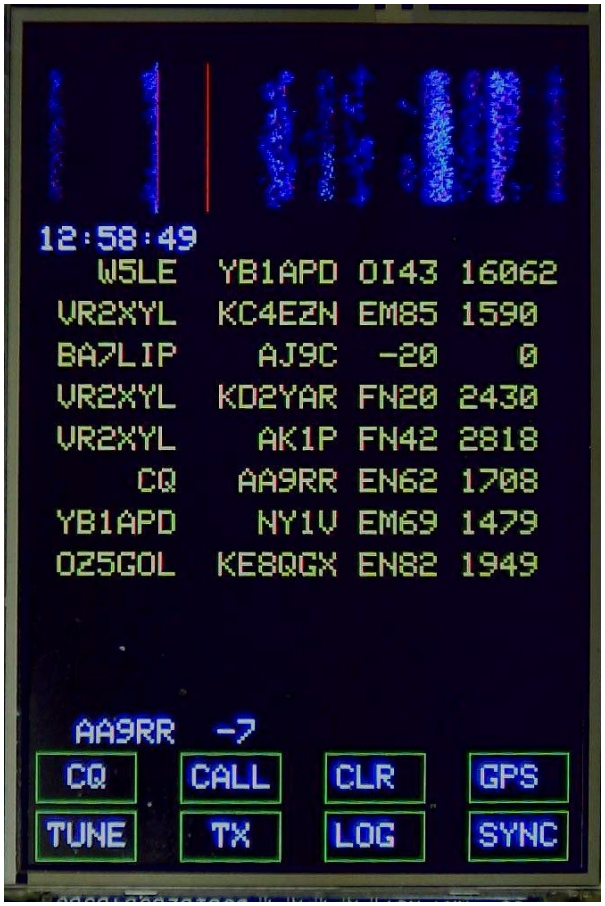
Then select a message from the station you wish to call. In the example shown below the message “CQ AA9RR 1708” is touched to capture the call sign “AA9RR” shown in white text immediately above the CQ and Call touch buttons. The first message to be sent to AA9RR is composed as “AA9RR W5BAA EM00”. Please note that the CQ and Call touch buttons are interlocked so that only one mode of operation may be activated at a time. This is indicated by the two touch buttons changing color.

Once a station call is selected, the QSO algorithm will call the selected station up to five times. After each call to the station the algorithm will listen for a reply message from the station called. If no reply is received after five listening periods, no further calls will be made.

When a reply is received, the algorithm will proceed with composing and sending a signal strength message and/or a RR73 message such as:

AA9RR W5BAA -5

AA9RR W5BAA RR73



Actual Pocket-FT8 Activity Report

I think this is a nice illustration of the “Grey Line”, what do you think?