The Pocket FT8 Revisited Transceiver

Jim Conrad, KQ7B

***Abstract*—Pocket FT8 Revisited is a derivative of Charles Hill's (W5BAA) Pocket FT8 Transceiver. This document describes how to operate the little rig plus a few hints for modifying it in your own application.**

***Index Terms*—Amateur Radio, Digital Signal Processing, DSP, FT8, Homebrew, SDR, Software Defined Radio, Teensy, Si4735, Si5351, Transceiver, WSJTX.**

I. INTRODUCTION

THE Pocket FT8 Revisited is a self-contained, single-band, FT8 amateur radio transceiver. Features include:

- 4.0 X 2.8", 4-layer board
- Single-band operation
- FT8 modulation (only)
- 275 mW RF output
- 480x320 TFT touchscreen display
- 600 mHz Teensy 4.1 MCU
- Auto-logging to SD ADIF file
- JSON configuration file
- "RoboOp" sequenced QSO
- TCXO clock
- GPS disciplined date, time and grid square
- Operation from a single 5V power brick

To use this radio, you will need an SD card, suitable antenna, feed-line, 5 V USB power brick, and USB power cable. You may prefer to use a stylus with the resistive touchscreen. No computer, tablet or phone are required to make contacts.

The Pocket FT8 Revisited is a homebrew project, not a kit. If you're looking for a Heathkit-like experience, consider one of the products from QRP Labs or the DXFT8. On the other hand, this rig, both the hardware and the firmware, are fully open source. You can use or improve the KiCad schematics and board design, or the PlatformIO firmware and tests with few restrictions beyond your project must too remain open source. That said, I sometimes have a limited supply of extra boards.

II. USER INTERFACE OVERVIEW

The user interface (UI) displays five panels and a row of menu buttons. The upper-left panel provides a waterfall display of received signal strengths, and is used to change the FT8 offset frequency. The upper-right panel reports the station's status including the current date and time, four character Maidenhead grid square locator, callsign, carrier frequency, and the rig's current activity (e.g. RECEIVE, TUNE, TRANSMIT…). The middle-left panel reports the first 6 messages decoded during the previous FT8 interval. The middle-right panel displays this station's QSO messages. The lower panel displays error messages and informative messages about the little rig's activities.

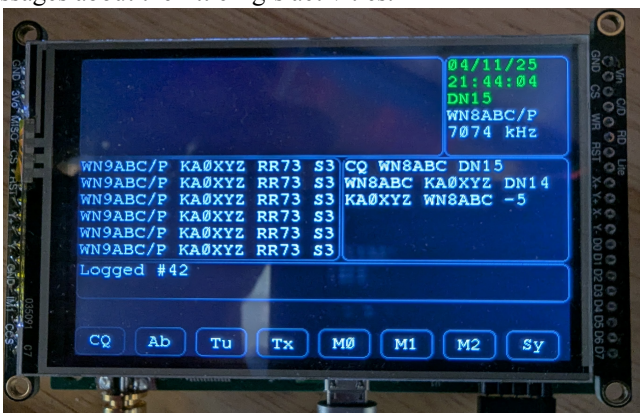

Fig. 1. Pocket FT8 Revisited's User Interface

The menu buttons appear in a bottom row and these control the functions of the FT8 rig.

A. Waterfall Panel

The waterfall provides an interactive display of FT8 activity. Brighter pixels indicate stronger signals while darker pixels reflect weaker signals. The vertical red line indicates your chosen transmitter offset frequency (from the carrier), and can be adjusted by touching the panel. You may wish to choose an offset in a quiet region of the band so your QRPP signals will not be buried beneath those of QRO stations.

B. Station Status Panel

When a GPS fix is available, the GPS-disciplined date and time are displayed in UTC with green text. When a GPS is not available, or when a satellite fix cannot be obtained, the date and time values are retrieved from the battery-backed Real Time Clock (RTC) and displayed with yellow text.

Likewise, when a GPS fix is available, the four-character Maidenhead grid square locator is displayed with green text. When a GPS fix is not available, the rig uses the locator provided by the SD configuration file, if that's available.

Note that the GPS, while highly desirable, is optional. The rig will "make the best of it" when a GPS fix cannot be obtained. In particular, if a GPS fix has been previously obtained since the firmware was loaded into the MCU, the

Jim Conrad, KQ7B, is retired from the computer industry with his wife on their little farm near Grangeville, Idaho (e-mail: conr2286@gmail.com). The many nearby SOTA and POTA sites were an inspiration for this project.

previously disciplined, battery-backed RTC will likely be reasonably accurate. Keep in mind that Teensy resets the clock to your computer's local time when you load new firmware into the MCU.

The station's callsign and carrier frequency are obtained from the SD configuration file.

C. The Decoded Messages Panel

The receiver displays its decoded messages in the middle-left panel. Due to the constrained screen space and a conflicting desire to report as much as possible about each signal, the display differs from the familiar. The Received Signal Level (RSL) appears as S1-S9 rather than the FT8 decibel convention in order to fit the signal report into two characters rather than three. The MCU firmware refreshes the list of received messages following each FT8 time interval. What you see is always the most recent.

Clicking on a received message, e.g. a CQ, instructs RoboOp to engage the remote station in a QSO by transmitting a call in the next available FT8 timeslot (this is explained in more detail in the FT8 Essentials section below).

D. The Station Messages Panel

Signals received by and transmitted from the local station appear in the Station Messages panel. These messages scroll so the oldest message always appears at the top. Most messages appear in white text, but repeated messages appear in yellow rather than in multiple lines. The oldest message eventually times-out and scrolls off the panel.

E. The Application Messages Panel

The Application Messages panel displays error and informative messages about the rigs activities. Most are self-explanatory, but a few deserve further explanation:

- FATAL: You *must* copy AudioStream6400.h to .../teensy/hardware/avr/1.59.0/cores/teensy4/AudioStream.h: This occurs when firmware has been rebuilt with the wrong version of Teensy's AudioStream.h header file. You'll find the correct version in the Extras folder named, AudioStream6400.h, and this must be renamed and copied into the ...cores/teensy4 folder before the rebuilding the firmware.
- ERROR: Unable to access SD card: The SD storage disk containing your config.json file (and space for the ADIF log) is not accessible in the Teensy SD slot. Insert your SD card in the Teensy slot, not the Adafruit display board's socket. The rig will continue to boot up but many features are inoperable without your station's callsign.

E. The Menu Buttons

- CQ: Pressing the CQ button instructs the rig to begin calling CQ in the next available FT8 timeslot. If a response is received, RoboOp attempts to engage the responder in a sequenced FT8 QSO, and resumes listening when the QSO completes. If nothing is heard, the rig repeats the CQ message until RoboOp times-out. Pressing the CQ button while the rig is already transmitting CQs terminates CQ activity.
- AB: Abort whatever transmission is in progress.
- TU: Output a steady carrier for antenna tuning. Pressing the TU button during a tuning activity cancels the tuning activity.
- TX: Instructs RoboOp to respond to the first station it hears calling CQ. If successful, RoboOp engages in a standard, sequenced FT8 QSO and then returns to listening. RoboOp repeatedly attempts to contact the heard station until a time-out occurs. You can abort RoboOp's pursuit of a contact by pressing TX again, or by pressing the AB (abort) button. By default, RoboOp will not engage a station that's already in the log; this can be changed in the configuration file.
- M0: Not implemented.
- M1: Not implemented.
- M2: Not implemented.
- Sy: Not implemented.

III. FT8 Essentials

FT8 is a digital communication mode whose excellent weak signal performance imparts some compromises.

All transmissions are of a fixed length and require ~12.6 seconds. Every transmission occurs in one of four timeslots that arise at 0, 15, 30 or 45 seconds past the minute. There is, at most, a 2.4 second "dwell" time between the end of a received transmission and the beginning of the following timeslot, and that only if the transmission begins exactly on time and can be decoded concurrently as the bits arrive.

Consider the case of a remote station transmitting CQ messages during even-numbered timeslots and listening for responses during odd-numbered timeslots. At most, our receiving station has 2.4 seconds to decide to transmit a response during the odd-numbered timeslot while the remote is listening. That's not much time. What's worse is, if our station is too slow to respond in the first available odd-numbered slot, we will have to wait for the beginning of a future odd-numbered timeslot to avoid "doubling" with the remote station. If the remote station is rare DX, well… we likely missed our opportunity as well as our timeslot.

IV. RoboOp

Because the 2.4 second response time challenges human operators, FT8 software generally implements some form of automation to sequence a standard, contest-like, QSO without human intervention. Pocket FT8 Revisited automates FT8 QSOs with its "RoboOp" feature.

Clicking on a remote station's CQ message directs RoboOp to engage that station. RoboOp replies to their CQ and sequences a QSO with the standardized FT8 messages at the appropriate times. RoboOp retransmits its message when it doesn't hear an appropriate reply.

The TX menu button directs RoboOp to contact the first remote station it hears sending CQ messages. If successful, RoboOp will sequence the entire QSO thereafter.

The CQ menu button directs RoboOp to transmit CQ messages and engage the first responder it hears in a QSO.

RoboOp completes only a single QSO; it will not run continuously logging QSO after QSO while you enjoy a sandwich and cold beverage from the icebox.

RoboOp employs a configurable timer determining how long it persists retransmitting a message without receiving a satisfactory response from the remote station.

By default, RoboOp will not respond to CQ messages from a known station already in the log. However, this is configurable as this is exactly what you do when testing your latest enhancement to the hardware or firmware.

V. Logging

The Pocket FT8 Revisited records an ADIF log file on the Teensy MCU's SD card. By default, the file is called, "LOGFILE.ADIF" but the actual name is configurable.

In Version 2.* of the firmware, you will have to remove the SD card to extract the log file. This may change in the future.

VI. Building the Rig

Pocket FT8 Revisited is a github project at, *https://github.com/conr2286/PocketFT8Xcvr*. The firmware is located in the PocketFT8Xcvr folder. The design requires a change to the Teensy 4.1 AudioStream.h file, and a copy of the modified file you need resides in the Extras folder named AudioStream6400.h. You will have to locate Teensy 4.1's AudioStream.h file and replace it with the renamed AudioStream6400.h file. No, you can't fudge this. If you try, the firmware will tell on you (you might wonder how I would know that;).

In pursuit of improved testing, most of the V2.* firmware has been moved from the Arduino 2.0 IDE to PlatformIO, a more capable IDE for projects of this size. The platformio.ini file is located in the PocketFT8XcvrFW folder. The easiest way to get started is to install the Arduino 2.0 IDE first (no, you aren't going to actually use it, but PlatformIO will use the tools and libraries it installs), Arduino's support for the Teensy 4.1 board, the Adafruit GPS Library V1.7.5, the PU2CLR SI4735 V2.1.8 library, the bblanchon ArduinoJson V7.3.0 library, and the Adafruit GFX V1.12.0 library. Some of the libraries build with compiler warnings; PocketFT8XcvrFW is building clean as of April 23, 2025.

There are a number of folders within BenchTests containing little (mostly Arduino 2.0 IDE) test projects for the hardware. The folders are numbered in a suggested order, and the code they contain was developed for testing new boards.

The Investigations folder contains folders and files used to prove (or disprove!) some of the concepts developed for the firmware and hardware.

The RFFilters folder contains a variety of filter designs, simulations and boards.

The PocketFT8XcvrHW folder contains the KiCAD 8.0 files for the schematic and PCB artwork. The V2.00 BOM is in the Mfg folder. I had PCBWay build and assemble most SMT components on my boards. The only SMT components not in the BOM (like… the antenna jack, the RTC's battery holder…) are really easy to solder. I installed low-profile headers on the Teensy 4.1 and THT soldered these to the PCB.

To build a complete rig, you need an assembled PCB, a Teensy 4.1, an Adafruit 2050 480x320 resistive touchscreen display, a holder for the RTC battery, an SMA antenna jack, tall headers for the display, low-profile headers for the Teensy, a (highly recommended) Adafruit Ultimate GPS and cables, a low-pass filter for your chosen operating frequency, an SD card, and a 5V power brick. I built my low pass filters on the daughter board whose KiCAD files reside in the 5Pole40MLPFilter folder.

Acknowledgments

Pocket FT8 Revisited is a derivative of Charles Hill's (W5BAA) Pocket FT8 project with important contributions from Ricardo Caritti (the Si4735 library), Karlis Goba (YL3JG, the FT8 library, without which we'd be pondering Fortran), Barb (WB2CBA, the SN74ACT244 PA), and many other widely available Adafruit, Arduino and PJRC libraries. Charles and Barb continue their work with the DXFT8 project, another Pocket FT8 derivative. I am indebted to tips from now-forgotten mentors for getting the best phase noise performance from the Si5351 clock, optimizing the sensitivity of the Si4735 receiver, combined RF/AF/digital PCB design, and getting the most out of the Teensy 4.1 MCU.

Back in the 60s, the ARRL Handbook published an amazing receiver, the Junior Miser's Dream, that accomplished so much (for its day) with so little (if only I could have afforded that Eddystone dial;). Its focused design guided my engineering career, ever drawing me away from unwarranted complexity. The Pocket FT8 and its derivatives reflect that goal of doing so much with so little in our modern world. May you too stand on shoulders of giants.

References

[1] I. B. Author "Wonderland," *Lost in Space*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959.