Project Motivation, Attribution and Other Thoughts December 2019

For the past ten years after retiring I have had a lot of fun building ham radio projects which utilize microprocessors to perform DSP for SDR radios. Past projects include the SDR2GO, STM32_SDR and STM32F746_SDR. As part of the STM32F746_SDR project I included a Beaconing feature that uses a GPS receiver to periodically send a PSK call along with location latitude and longitude. Over time I noticed that the number of stations monitoring PSK traffic using PSK Reporter has significantly dwindled due to rise in popularity of FT8 operation.

So, during the Summer of 2019 I started a quest to do FT8 on a STM746 Disco board. This did not turn out well due to my code using more RAM than is easily available on the STM746 Disco Board. I gave up several times in frustration.

In one of my restarts of this project I came across the work done by Karlis Goba which you may review on this website: https://github.com/kgoba/ft8_lib . Karlis is a ham, YL3JG.

And, I decided to port my previous FT8 work over to the STM769 Disco board which has a lot more RAM than the STM746 Board. ShaZam I finally got the FT8 stuff to work! Not only does iit work, it works well due to the great work done by Karlis. I have had several email exchanges with Karlis and he has been most helpful and supportive in my effort to produce another DSP project.

At, present the operating mode of the application is quite limited. However, as I learn more about FT8 operations and experiment with the code I will probably add additional operation modes and features.

Time Synchronization

One of the main items that I learned from Karlis is that it is quire easy to synchronize the application in time without relying upon a GPS clock. To do this, we set up an internal clock in software with millisecond resolution to create what I call FT8 Relative Time. This clock is synchronized by simply watching the waterfall and depressing the Blue User Button on the STM769 board during the lull in FT8 traffic. My experience is that FT8 application will remain synchronized for hours using this method.

Simple Operating Mode

So, when the Becn Touch Button is red the application monitors the FT8 Relative Time clock and will issue an FT8 CQ that includes your Call and Maidenhead Locator when the clock reports 00 seconds. The application then monitors the FT8 traffic and searches for your Call at the beginning of each FT8 decode. When if finds the first decode with your Call, it composes an FT8 reply that includes the responding station Call, your Call and your Maidenhead Locator. This reply is then transmitted when the next 00 seconds is reported. At the end of this FT8 transmission, the Becn is reset to issue another CQ on the next 00 second interval.

STM32F769 FT8 Transceiver Project

The STM32F769 FT8 Project (F7_FT8) uses an STM32F769 Disco Board to provide audio IQ Processing and control for the Baseband SDR Radio Boards listed below so that the user may communicate with other radio amateurs using the FT8 mode of text message transmission and reception.

- HobbyPCB RS-HFIQ SDR Transceiver Board
- SoftRock Tranceivers Using an SI570 Clock
- UHFSDR
- Other Baseband SDR Boards Using an SI5351 Clock

In addition, the F7_FT8 may be used with any SSB tranceiver which provides mono audio input, mono audio ouput and PPT control at 5 volt signal level. Although I have not tested F7_FT8 with a uBITX I am confident that the combination will work fine.

The Disco Board includes a 3.5 inch color touch display with a resolution of 800X 480 pixels. The F7_FT8 project includes two rotary encoders for user adjustment of operating parameters and frequency.

The interface between the F7_FT8 and the RS-HFIQ board for frequency control and PTT control is via a serial UART connection. This connection uses the paddle inputs on the RS-HFIQ which are modified as an additional serial port. This change requires that the code for the Nano board be replaced with code modified by the author of the F7_FT8 project. Source code and/or a hex file is provided for reprogramming the Nano board.

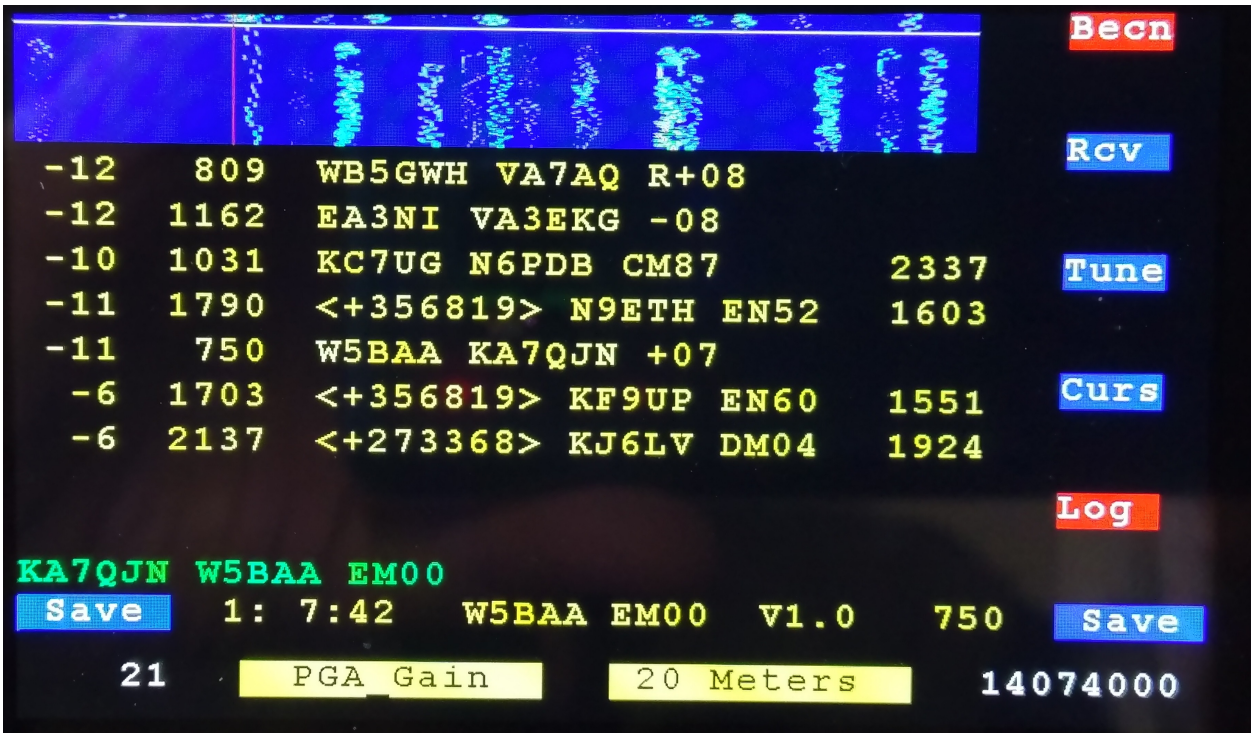
The interface between the F7_FT8 and other Baseband SDR Boards for frequency control and PTT control is via an I2C interface and a 5 Volt PTT signal. For further details please see the interface drawing at the end of this document.

The F7_FT8 project uses an SD Card inserted into the SD Card Socket on the F7_FT8 board for storage of operating parameters and station data.

The Blue User Pushbutton on the lower left corner of the Disco board is used to synchronize the FT8 coding and decoding algorithms.

ALL control of the transceiver is done thru the encoders, the touch screen or the User Pushbutton. The secrets of these controls are revealed below.

Screen Layout



A realtime Waterfall display of received audio is displayed at the top. The Waterfall covers the range of 300 to 2300 Hz. A red cursor is shown which shows the frequency chosen for transmission in the Beacon and Tune modes. The location of the cursor may be set by touching the Waterfall or by rotating the Frequency Encoder when the **Curs** Touch Button is blue.

A column of Touch Buttons are provided along the right hand side which are explained in following sections.

The decoded FT8 messages and data are displayed in yellow immediately below the Waterfall. This display is organized as four colums: **Recieved Signal Level**, **Received Audio Frequency**, **Decoded Message**, and **Distance** between your station Maidenhead Locator and the received station Locator in Km.

A single green data line below the decoded messages displays an automated response to an FT8 station replying to a CQ transmitted by the F7_FT8.

A line of two Touch Buttons and general data is shown below the FT8 message area.

And, finally, at the bottom of the screen are two Menu Bars and associated user input data displays.

Touch Buttons

There are seven touch buttons on the user screen which are shown as **blue** with text labels in white upon boot up and will change to **red** background when touched. The operation of each button is described below:

Becn: When this button is **blue** FT8 decoding is perfromed with no FT8 transmissions. When this button is r**ed** the unit will automatically issue a CQ transmission with your call and locator when the displayed FT8 time is at 00 seconds. So, when in the beacon mode the unit issues a CQ transmission once every 60 seconds. The CQ call is made at the audio frequency indicated by the red cursor line displayed in the waterfall.

Rcv/ / Xmit: This button does not respond to your touch. However, it is linked to the Becn and Tune buttons so that when it is **blue** the unit is in the receive mode. When it is **red** the unit is transmitting in either the beacon or tune mode.

Tune: When this button is **blue**, the unit is in the receive mode. When it is **red** a continuous tone is transmitted at the audio frequency indicated by the red cursor line displayed in the waterfall. You may use this to tune your antenna or amplifier.

Curs / Frequency: When **blue**, the Frequency Encoder adjusts the transmit audio frequency which is shown in the waterfall and as a numerical value displayed immediately to the left of the right hand Save button. When **red**, the Frequency Encoder adjusts the RF frequency of the associated RF SDR unit's clock. The RF frequency is displayed in the extreme lower right hand corner of the display.

Log: When this button is **red** each FT8 QSO conducted by your unit will be saved to a log file on the SD card.

Right Hand Save: When this button is touched it will briefly flash as red to indicate that the RF frequency displayed immediately below is saved to a file on the SD card.

Left Hand Save: When this button is touched it will briefly flash as red to indicate that the Parameter Value displayed immediately below is saved to a file on the SD card.

Data Display Line: Nestled between the Left and Right Save buttons five data items are displayed:

FT8 Session Relative Time in Hours, Minutes & Seconds (FT8_Time)

Your Station Call which you enter via a file you place on the SD Card

Your Station Maidenhead Locator which you enter via the SD Card

The revision level of the firmware.

The current transmit audio frequency in Hz.

Touch Menus

There are two touch menus on the user screen which are shown as yellow with text labels in black upon boot up. Pressing the left side of a menu will cause the selection to go backwards. Pressing the right side of a menu will cause the selection to go forward. The left hand menu controls the unit's operating parameters while the right hand menu controls the RF Frequency Band. The operation of each menu is described below:

Parameter Menu: This menu displays the current parameter that can be modified or saved. The parameter selected can be changed with the Parameter Encoder.

Band Menu: The menu displays the current band. The current frequency associated with the band selected is displayed immediately to the right of the menu. The RF frequency of the associated SDR unit clock is immediately changed when this menu is manipulated.

Parameter Data Items

The data items associated with the Parameter Encoder are described below:

PGA_Gain: This sets the gain of the Codec first gain stage of the analog IQ Audio inputs.

ADC_Gain: This sets the gain of the Codec Analog to Digital Converter (ADC) of the analog IQ Auddio inputs.

TX _Gain: This value controls the audio level of the IQ Audio outputs sent to the RF board when the transceiver is in the transmit mode.

Rx Amp: This value is used to adjust the Q signal amplitude so that suppressed receive signal level is minimized.

Rx Phase: This value is used to adjust the Q signal phase so that suppressed receive signal level is minimized.

Tx Amp: This value is used to adjust the Q signal amplitude so that suppressed transmit signal level is minimized.

Tx Phase: This value is used to adjust the Q signal phase so that suppressed transmit signal level is minimized.

Clk_Src: This value controls the interface between the F7 board and the RF board. A zero value invokes the UART interface for the HobbyPCB RS-HFIQ SDR Transceiver Board. A value of 1 invokes an I2C interface for an SI5351 clock. A value of 2 invokes an I2C interface for an Si570 clock.

ClockMult: This value may be set to 2 or 4 and is used to provide a clock signal at either 2 or 4 times the desired receive frequency as required by your SDR RF board.

SD Card Installation

The application utilizes a micro SD. A micro SD must be installed in the micro SD slot. The system displays an error message if no micro SD disk is present.

For proper operation, three files must exist on the SD:

Frequencies.txt

SaveParams.txt

StationData.txt

Upon initial startup of the application you do not need to have the Frequencies.txt or SaveParams.txt files installed. The application will recognize that these files are not present and will automatically generate them using default data stored in the firmware. After these files are initialized you may modify them using the menus and encoders as described in the previous sections.

However, you must create and install the StationData.txt file manually. Notepad is a good Windows application for creating this simple file The file requires a single line of data as shown below:

W5BAA:EM00

Replace my call with your call and the EM00 with your Maidenhead Locator.

Logging QSOs

When you use the application you may log your FT8 transactions by using the Log Touch Button.

The log file will be automatically generated and it will have the file name Log1,txt,

An example of the contents of a log file are shown below:

W5BAA KA7QJN +07

KA7QJN W5BAA EM00

W5BAA AB6OR DM13

AB6OR W5BAA EM00

W5BAA W4LRN EM90

W4LRN W5BAA EM00

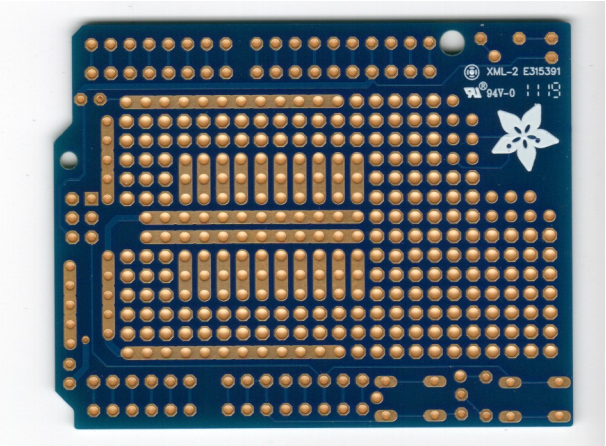
W5BAA W4LRN EM90

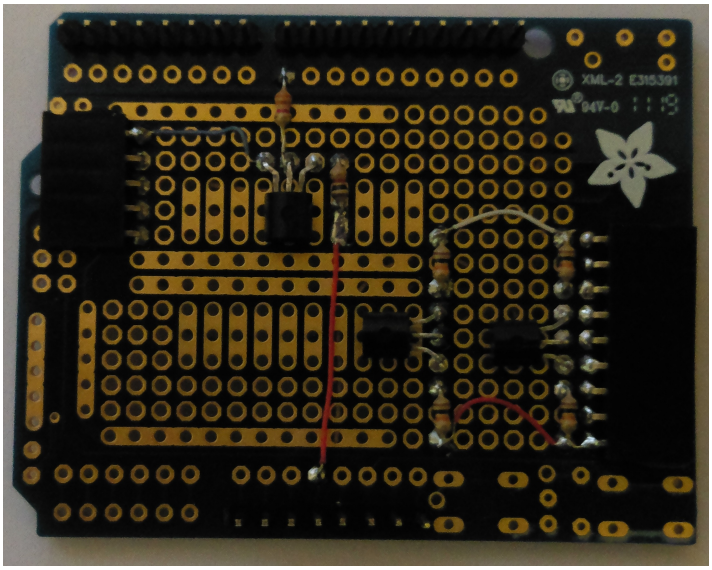
A Word About the HobbyPCB RS-HFIQ SDR Transceiver Board

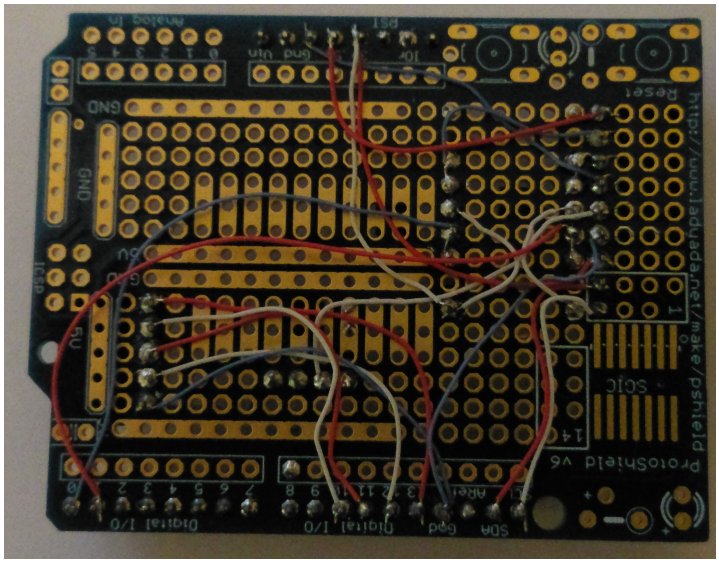
This project has been developed around the HobbyPCB board. Our initial testing shows that this transceiver board is very quiet when used with the F769 Disco board and the code developed by this project. And the band switching is great. So, we want to congratulate and thank the HobbyPCB folks for designing such a great board!

Construction Hints

The STM user manual for the F769 Disco Board is provided as a PDF file. Described in the manual are four header pin connectors which are laid out to match the connectors on the Adafruit Shield Board shown below. All of the interface connections required for this project except for the Audio I/Q signals are brought out on the four header pin connectors. Also, photos of my project board are shown.







Other Baseband Boards

The Project Code and F769 Disco board have been tested with two UHFSDR transceiver boards. Both boards performed well with the F7_FT8.

One UHFSDR board had been modified to use a SI5351 Clock in lieu of the standard SI570 clock. Tuning is smooth and noise free. The noise floor approaches -130 dBm.

The other UHFSDR board uses the original SI570 clock. It performs similarly to the other UHFSDR board.

Installing Firmware on F769 Disco Board

The firmware is supplied as a STM32f binary file labeled Velvet.bin

This file must be installed on the STMF769 board using the STM Utility called ST Link which may be obtained at this link: https://www.st.com/en/development-tools/stsw-ink004.html .

ST Link is free and only requires an USB cable for connection between your PC and the F769 Disco board.

Installing Revised Firmware on your HobbyPCB RS-HFIQ SDR Transceiver Board

A zip file labeled “RS_HFIQ_REV_F7_FT8 is included with this project for modifying the Ardunio Nano to accept UART frequency settings from the F7_FT8. Both source code and hex files are provided. In the case of compiling the source code, the required SI5351 drivers are included with the INO file so that you should not have to chase them down.

F7_FT8 Interconnections