# ICARC Fox Transmitter

William Robison
KC0JFQ

December 2, 2024

A proposed article for publishing in QST.

LaTeX
This is pdf-TeX, Version 3.14159265-2.6-1.40.18

**kc0jfq@n952.ooguy.com**.

Documents webpage:
http://n952.ooguy.com/HamRDF

## Contents

# List of Figures

Supporting graphics and image files:

Schematics 102_73181_10

Figure 1, Schematic, Sheet 2
Figure 2, Schematic, Sheet 3
Figure 3, Schematic, Sheet 4
Figure 4, Schematic, Sheet 5

Boards

Figure 7, artwork Top
Figure 8, artwork Bottom
Figure 9, Completed FOX Transmitter

Schematics 102_73181_28

Figure 5, MMIC Amplifier Schematic, Sheet 1

Boards

Figure 11, MMIC Amplifier artwork Top

Schematics 102_73181_36

Figure 6, DRA818 Module Schematic, Sheet 1

Boards

Image 10, Completed DRA818 Module

William Robison KC0JFQ

# 1 Genesis of a New Fox Transmitter

The Iowa City Amateur Radio Club resumed foxhunting activities recently following a long pause, a pause long enough for useful skills to have been lost to the mists of time. We resumed our hunts in a county park near town. The park provided a lodge that was used as a comfortable base of operations. We started our hunts with a set of three transmitters that all operated at 146.565MHz. These are typical fox transmitters that are time multiplexed so that their transmissions do not occur concurrently.

Although these early hunts were comfortable for our event organizers, the novice fox hunters were having a rather difficult time locating the transmitters. With each transmitter being active for only one minute out of five, obtaining fixes was proving rather difficult for these novices. The success rate, that being the ability to find the transmitters, was disappointing.

To address this, I started a design effort (clear back in 2018) to come up with a richly featured fox transmitter that would be able address the shortcomings seen with our hunts. The fox transmitter architecture must be able to reproduce the operating modes used by other fox hunting products as well as satisfy the needs of novice hunters.

A casual set of requirements emerged somewhat as follows:

- Easy to use command and configuration language
  (using a simple verb/noun structure)
- Easily configured and synchronized schedules
  (using a simple serial connection to a host computer)
- Software selectable frequency (dynamic)
  (we don't want to deal with jumpers for this)
- Convenient mechanical packaging
  (easy to transport, setup and physically find; not too small)
- Ability to configure a training mode
  (for my novice hunters to practice with)

Some wag even asked "Why doesn't it talk?". Although this was said in jest at the time, it turned out to be difficult to resist (and easy to implement).

Two designs emerged from the initial effort, one using a single chip processor (SOC or a System-On-Chip), that draws minimal power. A second design replaced the SOC with a Raspberry PI-Zero to add a PWM audio (i.e. voice) capability. The board outlines and the external mechanical interface of the designs were kept identical in order to make use a common enclosure.

## 1.1 A Tale of Two Transmitters

The Raspberry PI-Zero design, perhaps somewhat more sophisticated, was eventually dropped due to a relatively lengthy startup time and the high current requirements (the batteries required replacement after every hunt).

The SOC design proved to be easy to load and control. Access to the system is through a simple serial port (no device-specific USB drivers!). A simple verb/noun command structure allows the device to be tested, configured, and setup for a hunt through the serial port. Also included in the design is a TOY clock to allow synchronous operation of multiple transmitters. This design has progressed through several revisions culminating in the design described here.

# 2 Hardware

The SOC variants all used a ZiLOG zNEO SOC (System on Chip). The zNEO has a comfortable sized program memory (128KB) and a spartan RAM (4KB). These days the only readily available package is a 64-pin flat pack which, of course, is what the latest revision makes use of.

All of these revisions share a common enclosure. The board dimensions for all variants is 124mm by 72mm.

## 2.1 Packaging

The first board produced for the project was built to fit in a Hammond 1599E enclosure along with a 6-cell AAA pack. The 124mm by 72mm board dimension is derived from the 1599E enclosure mechanical drawing . Subsequent revisions retained the same form factor and external connector placement. This enclosure provides room for the circuit board and a battery and may be obtained in several mechanically compatible variations (i.e. varying in color and cost). The final artwork for all of the designs (figure 9) has kept the external connectors locations fixed so holes in the housing align, allowing the use of a common drilling jig.

## 2.2 DC Power

The design effort quickly switched to a switch-mode regulator to improve battery life. This also allows us to deal with a higher voltage battery pack without having to manage regulator power dissipation. A secondary linear regulator provides 3.3V to the digital logic.

Nominally power would be supplied using a 6 cell AAA alkaline pack. Although a simple 9V radio battery would work, in practice six AAA cells seems to be a bit less expensive with the added benefit of longer run time. Primary cells are used to simplify battery management, simply replacing cells when necessary. A 3 or 4 cell Li-Po pack that will mechanically fit would also work well as the first stage regulator will tolerate input voltages up to around 24V.

## 2.3 Frequency Control

The current revision makes use of clock synthesizer produced by Skyworks (earlier revisions used synthesizers that are no longer in production). The synthesizer, a SI5351, covers the 2M ham band. The SI5351 isn't terribly difficult to program; with it appearing in many designs. This device gives us access to the entire 2M band in frequency steps encountered in handheld transceivers.

The output from the SI5351 is modulated by varying the load capacitance on the SI5351 crystal using a pair of varactor diodes. The audio from the zNEO is used to change the bias on these varactor diodes, in turn varying the capacitive load across the crystal.

The RF amplifier was, at some point, moved to a separate daughter-board to allow experimenting with the RF amplifier design. In low power applications we use an MMIC to implement a rather simple amplifier of 50mW to 100mW output power.

We can also make use of a commonly available transceiver module that is available on eBay. This module (SA818 or DRA818) has an output of up to one watt and is programmed using a simple serial connection. All we supply to the module (on the RF daughter-board) is power, audio, and setup instructions (through the serial connection). The SA818 and DRA818 are also available in UHF variants.

The output filter and antenna connector (BNC) are located back on the motherboard.

### 2.4 Audio

There are two audio sources coming from the zNEO. The first is a simple programmable timer channel that produces a square wave. A second channel comes from a PWM controller in the zNEO. Audio data is stored in external memory and then transferred into the PWM control register as needed.

These two audio sources are combined (wire-or) before being filtered and passed along to the varactors or the DRA818/SA818 module.

### 2.5 HT Control

Controlling the RF section of the fox transmitter is, for all practical purposes, identical to controlling a handheld transceiver. The audio, a separate Push-To-Talk control, and a serial channel are routed to a header that provides the capability to control an external handheld transceiver. This connector also provides connections to the power rail to allow external power to be delivered to the board through the HT connector (motherboard J6).

The system software can control the HT as is, keying the PTT line and providing audio. HT frequency control is not present in the software at the current revision level. Updates would be required to directly control the HT operating frequency with the FREQ command.

## 2.6 TOY Clock

A Time-Of-Year clock (a Analog/Maxim/Dallas DS1672) is provisioned on the board. This clock is a simple 32 bit seconds counter with a backup battery. The SOC software keeps its system clock in much the same way that the Linux system maintains its system clock, as a 32 bit count of seconds from some epoch.

The battery for the TOY clock is a small lithium battery. The DS1672 has a charge circuit, but the DS1672 must be powered to perform this function so the charge feature is not used. The main battery runs through a low power regulator and then on to the DS1672 battery to keep it charged when the main battery is present. A small current is supplied to the backup battery, on the order of 0.5 micro-amp, which should be compatible with most lithium chemistry cells.

## 2.7 FRAM and FLASH

All of the audio, configuration and sequence data is held in memory external to the zNEO. Two devices are provisioned to make the data management task a bit simpler and also to reduce cost.

The first device, an FRAM, holds configuration and setup commands. A 64Kb device will hold 256 setup/control commands which is usually sufficient to configure the system for a fox hunt. The FRAM allows for replacing individual records in the device.

The second device, a FLASH, holds audio waveform data. The audio files are stored in the much larger device. The FLASH device performs erase operations at the sector or device level.

An 8Mb FLASH, holding about 3 minutes of audio, can be had for less than a dollar whereas this size FRAM would cost over $30.00.

### 2.8 Configuration Port

Loading audio data and setup/control commands is accomplished using the serial port. Physically it appears as a 3.5mm jack that is mechanically and electrically compatible with an FTDI USB-to-serial cable (TTL-232R-3V3-AJ).

Audio data is stored in the FLASH using a file with Intel-HEX records. The command decoder recognizes an Intel-HEX record, loading it into the FLASH device.

Setup/control commands are written to the FRAM in a similar fashion. A command may be entered directly to be immediately executed. A "save" command prefix is used to store a setup/control commands into the FRAM memory for later use.

Commands are provided to erase and dump both FLASH and FRAM memory. Audio data and setup/control data, being kept is separate devices, are erased using two different commands. The net result being, of course, one does not need to erase audio data when erasing setup/control data.

### 2.9 Battery monitor

Both voltage and current monitors are present in the design. The zNEO SOC provides the A/D for these measurements. Voltages are obtained using a simple voltage divider.

Current, on the other hand, is measured using a sense resistor (50 m$\Omega$) and a Zetex ZXCT1009. The ZXCT1009 is across the sense resistor in the battery positive. The ZXCT1009 provides an output current proportional to the voltage across the sense resistor. The output of the ZXCT1009 placed across a resistor to develop a voltage that is, in turn, measured by the zNEO.

## 3 RF Amplifiers

We only present a brief description of the RF amplifiers here.

An image of the SA818/DRA818 RF module is shown in figure 10 on page 23. A schematic is shown in figure 6 on page 21.

The MMIC RF daughter-board is shown in figure 11 on page 23. A schematic for this board is found in figure 5 on page 21.

The common mounting points are evident in figures 9, 10 and 11. The motherboard, of course, has sockets for all the interboard signals whereas the daughterboards leave off the connections that aren't required.

You may notice that the daughter-boards don't have all the connection points present on the motherboard. The SA818/DRA818 module does not require the RF signal (RF_DB$2 in figure 3) while the MMIC daughter-board does not require the serial port connection (J18_5, in figure 2).

# 4 Software

The software design aims to eliminate the need for any type of field configuration. We really want to be able to simply turn the device on as it is placed in its hiding spot in the field.

We expect the *Fox Transmitter* to report on its condition when turned on and then settle in to performing its *foxly* duties.

The TOY clock provides the time synchronization required to operate multiple units.

When powered on the system looks for setup commands (in the FRAM) to configure the system.

One of the setup commands will copy the current time from the TOY clock into the system time field. The setup commands must also set the callsign, nickname, and other necessary configuration information.

Thus we configure the transmitters *identity* in preperation for transmitting over the air.

The transmitters status should be reported at turn-on. Typically we would vocalize the callsign, nickname, and battery condition. This status report would typically be sent on a common *startup frequency*. The transmitters will then switch to its unique operating frequency.

Consider, for a moment, a multi-group foxhunt where there are multiple hunt groups. Each group (5 or 6 transmitters) will require a unique frequency to operate on.

## 4.1 Message Traffic

Each transmitter will periodically send out a message. The delivery schedule, having been specified earlier in the initialization commands, determines the *when*, providing for the regular delivery of the *message*. This delivery schedule consists of a cycle time (called "period") and a point in the cycle (called "offset") when the *message* will delivered.

The *message* is simply a set of commands that are executed at the scheduled time. This *message* program would send a sign-on message, some message traffic, and a sign-off message. The sign-on/sign-off messages send the station callsign (rules, rules, rules). The rest of the traffic is defined by the program commands.

# 5 A sample Setup

Operation fox transmitter (hence the "personality" of the station) is entirely controlled by the commands stored in the FRAM. The FRAM is managed as a set of fixed length records, each one holding 32 bytes (thereby limiting the size of any one individual command). Here is a sample setup.

This is a working example, it is complete.

## 5.1 INI=, Initialization commands

The first group of commands are the initialization commands. This group of commands runs when the zNEO is reset (power-on or by mashing the reset button). The operating software scans the FRAM looking for initialization records (those that begin with "INI="). These commands are executed in the order that they are encountered.

```
INI=TIME
INI=WAIT 0.5
INI=TIME
INI=EPOC -5.0
INI=NAME FOX21
INI=CALL KC0JFQ
INI=CONF SI5351
INI=CONF 8MA CLK0
INI=FREQ 144.150
INI=MODS S0 360,60
INI=MODS S1 360,90
INI=STAT
```

### 5.1.1 TIME/EPOC

The TIME command sets the system time from the DS1672 (with no arguments as shown here). The command is sent twice to mitigate an issue with the DS1672.

The EPOC command establishes the local time zone. It is expressed as hours from "ZULU" and assumes that time is stored as "ZULU" (not local time). The author is in the Midwest and using CDT.

#### 5.1.2 NAME

This command defines the stations nickname.

This stores the nickname which will be substituted into commands when <NAME> is found in a command. This nickname will be unique for each station.

#### 5.1.3 CALL

This command define the stations callsign.

The callsign, stored here, will always sent as part of the sign-on message and the sign-off message to comply with part 95 identification rules.

The callsign which will be substituted into commands when <CALL> is found in a command.

#### 5.1.4 CONF

Configuration commands. As shown, this command selects the SI5351 synthesizer and output configuration.

When using the DRA818 RF daughter-board, the CONF command will, of course, differ.

#### 5.1.5 FREQ

Selects the operating frequency.

It is possible to change frequency during operation. As an example of this use, during our hunts all transmitters start at 144.150MHz and transmit a startup message to let the hunt organizer know that the station is operational and what the current battery condition is. The system then switches to its operating frequency before sending additional message traffic.

#### 5.1.6 MODS

The MODS (modular schedule) command defines the message schedule. Up to 10 schedules can be stored (S0..S9).

The two arguments are the aforementioned ”period” and ”offset”. The fox hunt message repeats every 360 seconds in our example here. The 360 second cycle is synchronous with the time of day (from the DS1672), so think of it as starting at midnight and repeating every ”period” seconds. Our example here starts our fox hunt message 60 seconds into the 360 second ”period”. So, for example, we would transmit at 10:01:00, 10:07:00, 10:13:00, etc. We then stagger additional transmitters in the group on 60 seconds boundaries and limit the message length to less than 60 seconds, so that only one transmitter in the group is active at any time.

## 5.2 ANN=, Announcement Commands

These ANN= commands also run when the system is powered on or reset. This reason for this separation is not discussed here, these command run after the INI= commands.

```
ANN=REM- fox_ann_V2023.fox
ANN=TONE 1.0
ANN=CWPM 30,-1,-1,-1,-1
ANN=BEGN
ANN=TALK <CALL>
ANN=TALK <NAME>
ANN=WAIT 1.0
ANN=BATV V
ANN=BATV I
ANN=WAIT 0.3
ANN=TALK 144
ANN=TALK 225
ANN=TONE 1.0
ANN=CWPM 30,-1,-1,-1,-1
ANN=DONE
ANN=FREQ 144.225
ANN=STAT
ANN=RUNO S0
```

### 5.2.1 TONE

Set the audio tone frequency to 1.0KHz.

### 5.2.2 CWPM

Set the code generator to 30WPM with standard spacing.

### 5.2.3 BEGN

Turn on the transmitter and send our callsign and nickname in code.

### 5.2.4 TALK

Verbalize our callsign. This example substitutes the callsign set in the INI= commands.

We also verbalize our nickname using a similar substitution.

### 5.2.5 DONE

Send our callsign and nickname in code and then power off the transmitter.

#### 5.2.6 FREQ

Change to our group frequency.

This frequency is unique to the group. A multi-group hunt will be operating on multiple frequencies.

#### 5.2.7 RUN0

Enable (or turn-on, if you wish) the S0 schedule.

### 5.3 TALK directory

A small sample showing a few *TALK Directory* entries from the FRAM.

The single parameter is the data starting address in FLASH.

In this example the data in the FLASH is an 8 bit mono WAV file. As such, the WAV header (stored in FLASH) provides additional information necessary to process the file (i.e. length and sample rate).

```
. . .
  TALK=KC0JFQ 51200
. . .
  TALK=FOX21 66048
  TALK=FOX22 70272
  TALK=FOX23 74624
  TALK=FOX24 79872
. . .
```

### 5.4 Waveform Data

A (very) short clip of the hex filed used to load audio data into FLASH memory.

The type-4 records provide extended address information that is required to deal with data above 64K.

```
:02 0000 04 0000 FA
:20 0000 00 524946465010000057415645666D74201000000001000100A00F0000A00F0000 4F
:20 0020 00 01000800646174612B1000008080808080807F7F7F808080808180808080818080 E3
    . . .
:00 0000 01 FF
```

### 5.5 FREQ table

The SI5351 frequency table stored in the zNEO program memory is limited to just a few records. We store corrections and additional frequencies as commands in the FRAM.

```
    INI=FOFF -15.000
    144.FOFF -15.000
    144.100=139C,D2EFF,F4240
    144.105=139D,0DAC0,F4240
    144.110=139D,3C8BF,F4240
. . .
    144.150=139E,BF680,F4240
. . .
    144.225=13A1,A21BF,F4240
. . .
```

This external table allows us to select any frequency in the 2M band and to correct for errors in the SI5351 reference crystal.

This example table corrects for a 15KHz offset that appears on the output pins of the SI5351.

## 5.6 S0=, Schedule 0

We will now move on to look at an example command sequence that implements the fox transmitters on-air "personality".

This example is from a 6 unit group with each transmitter allocated a 60 second window in the 360 second cycle. The **MODS S0 360,60** from above runs a cycle period of 360 seconds with an offset of 60 seconds. Other schedules in the group will all have the same 360 second cycle period but offsets that increase in 60 second steps.

The S0= in this example corresponds with the S0 in the MODS command. These are the commands that are issued when the S0 scheduling point occurs.

```
S0=BATR
S0=TONE 1.0
S0=CWPM 30,-1,-1,-1,-1
S0=BEGN
S0=TALK <CALL>
S0=TALK <NAME>
S0=WAIT 0.5
S0=TONE 1.5
S0=CWPM 25,-1,-1,-1,-1
S0=WAIT 0.15
S0=BATC EV 7.2
S0=WAIT 0.5
S0=CODE IOWA CITY
S0=CODE AMATEUR RADIO
S0=CODE CLUB FOXHUNT
S0=CODE F W KENT PARK
S0=BATR
S0=TONE 1.0
S0=CWPM 30,-1,-1,-1,-1
S0=DONE
```

### 5.6.1 BATR

Generate battery voltage/current report for (external) battery analysis.

This command appears twice, one before the RF section is powered and one when the RF section is powered.

#### 5.6.2 TONE/CWPM

Set the audio tone frequency to 1.0KHz.
Set the code generator to 30WPM with standard spacing.

Later, we change the audio frequency and the word rate for the body of the message traffic (to make it sound "different").

#### 5.6.3 TALK

Verbalize our callsign and nickname.

This example replaces the <CALL> with the callsign set in the INI= commands.

We also replaces the <NAME> with the nickname set in the INI= commands.

The TALK command then expects to find the callsign and nickname in the TALK Directory in order to verbalize them.

#### 5.6.4 BATC

Another battery report, this one in code for the hunters to hear.

The V calls out a voltage reading (rather than current).
The E indicates we want the voltage encoded where volts comes out as a series of dash, and the tenths as a series of dots.
The 7.2 is a trip point. If the battery voltage reading falls below 7.2V the battery message includes an "SOS" to let us know we need to replace the battery.

#### 5.6.5 WAIT

A simple delay, expressed in seconds.

#### 5.6.6 CODE

The message text to be sent in code.

We simply string together as many CODE commands as needed to send our message out. For our typical hunt, we place enough CODE traffic here to keep the transmitter active for a total of 50 to 55 seconds.

# 6 Conclusion

The first hunt where these new units were deployed resulted in a dramatic increase in the number of transmitters that were actually located by the hunters. Managing the transmitters by the event organizers is rather trivial, simply turn the unit on as it is dropped in its hiding place and move on. A brightly colored antenna provides an easy to find visual for the hunter as well as for the event organizer at the end of the hunt. No synchronization procedure is required on the day of the hunt.

Recovering from an accidental power-off only requires that the unit be switched back on. It goes through start-up reporting and then resumes operations on its assigned schedule.

For the hunters, we have moved away from a paper punch that was used to validate a find on a log card. Rather, we simply have them record an ID number (unique to each hunt and unit) that appears on each unit. This method requires that a set of labels be generated for each hunt. It eliminates the need to provide additional hardware (i.e. a paper punch) at the fox transmitter.

## 6.1 Acknowledgments

The author would like to offer thanks to all those that have helped in bringing the project into an operational state.

In particular, George Carsner, W0PPF, for the simple question "Why doesn't it talk?". This simple question triggered the voice capability!

Rich Haendel, W3ACO, for advice on implementing and testing the RF section.

And finally, Don Kirchner (KD0L), who I've had the pleasure to work with professionally for many years. Don and I, along with a highly skilled team at the Department of Physics and Astronomy at the Univerity of Iowa have sent many instruments to the outer planets.

## 6.2 Authors email

kc0jfq@n952.ooguy.com

## 6.3 Additional Documents

See the authors web site at:

http://n952.ooguy.com/HamRDF

The boards are predominantly surface mount. Assembly requires some skill with an iron and a static safe work area. An inexpensive bench microscope (from eBay) is usually necessary to find pin 1!

A complete set of build documents are available on the web site as a compressed tar archive. The latest software for the zNEO in included in the tar archive.

## 6.4 Board Availability

Transmitter boards may be obtained from the author for $10 each plus actual shipping costs.

Boards can also be ordered from JLCPCB using the zip files included in the tar archive.

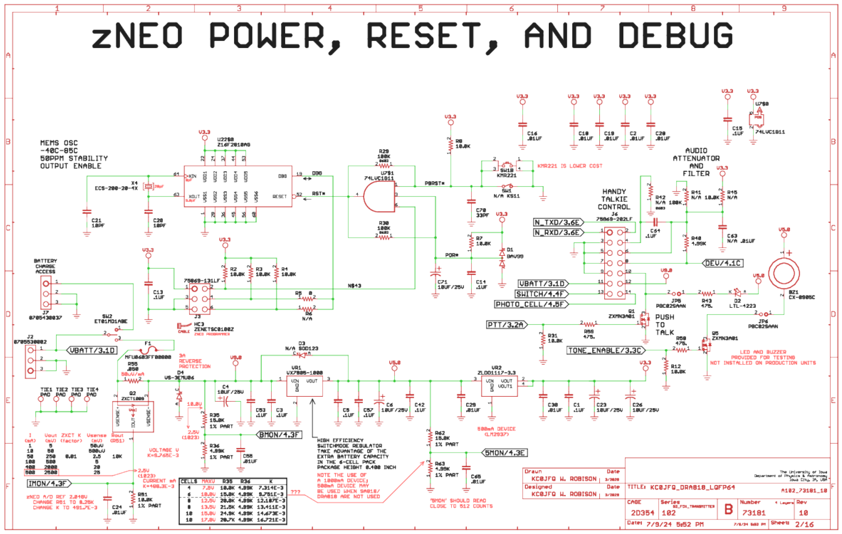


Figure 1: 102-73181-10 Regulators, Reset, HT connector

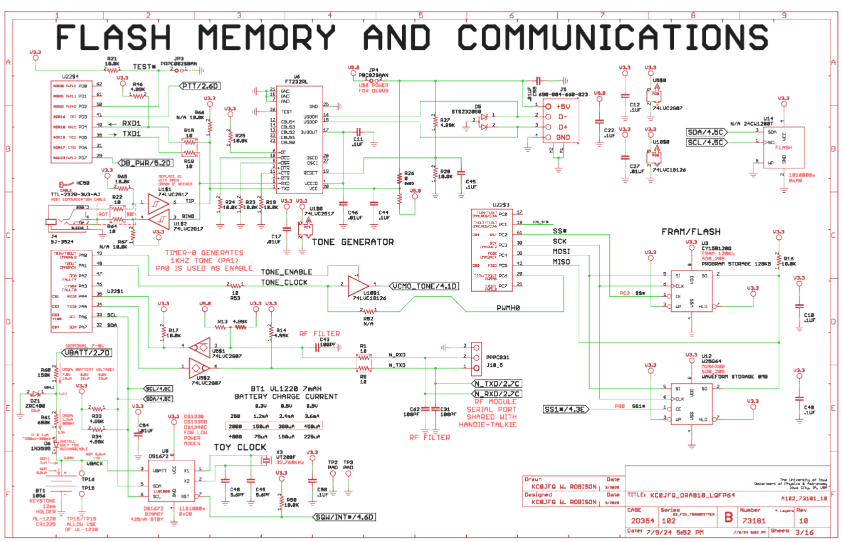


Figure 2: 102-73181-10 Peripherals

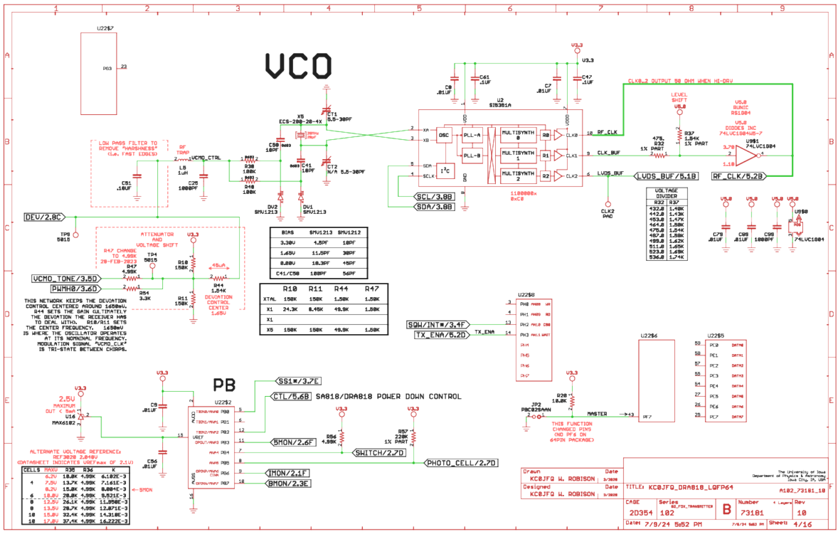

Figure 3: 102-73181-10 Modulation and VCO

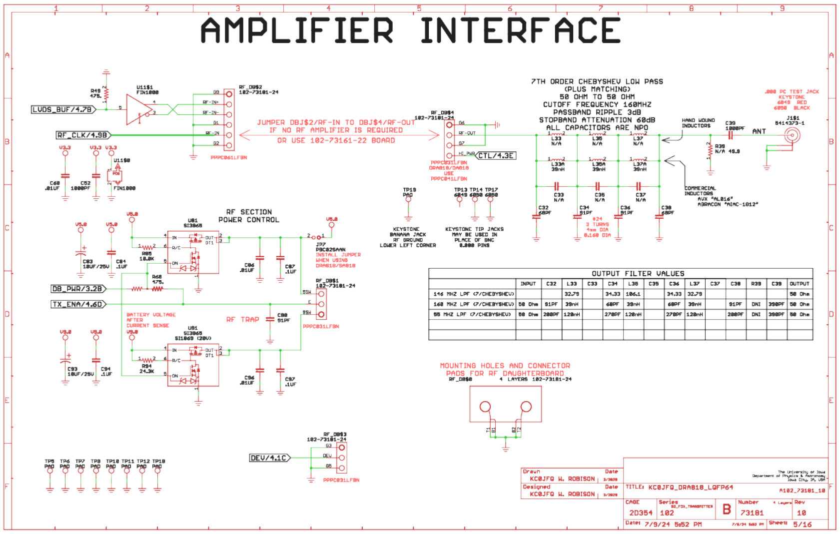

Figure 4: 102-73181-10 Amplifier Interface and Output Filter

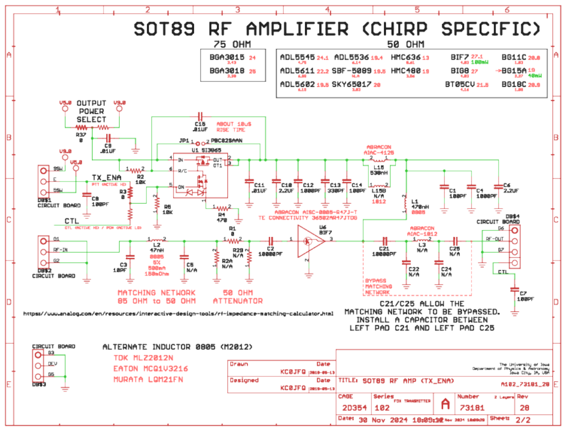


Figure 5: 102-73181-28 MMIC Module

The additional power switch on the board mimics the behavior of the SA818/DRA818 module when emulating a wildlife tracker.

This is the standard MMIC RF daughter-board.

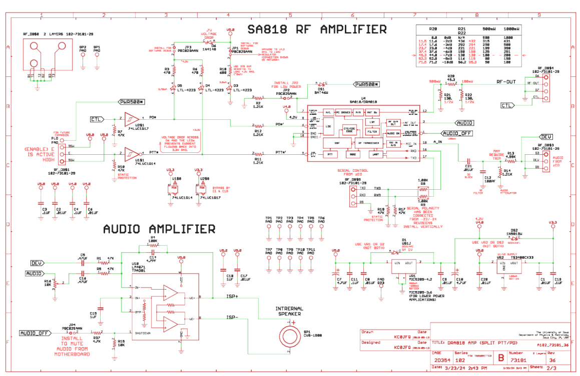


Figure 6: 102-73181-36 DRA818 Module

The LEDs and the Audio Amplifier would be populated only for software development and testing.

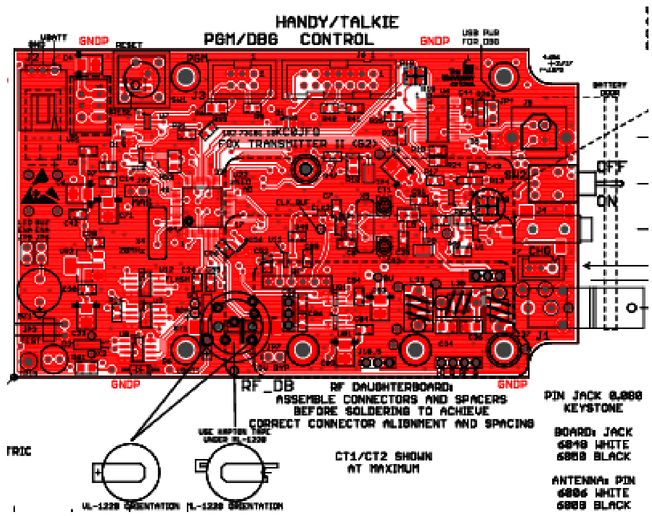


Figure 7: 102-73181-10 Artwork, TOP

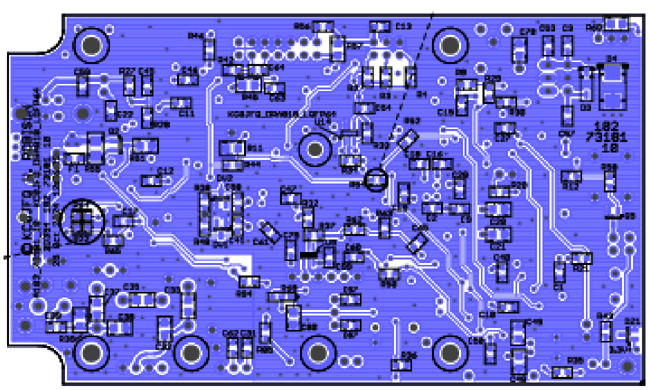

Figure 8: 102-73181-10 Artwork, BOT

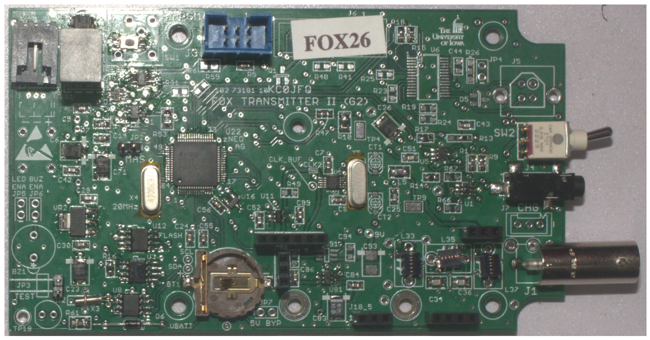


Figure 9: 102-73181-10 Board Image

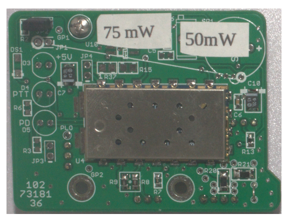


Figure 10: 102-73181-36 Board Image

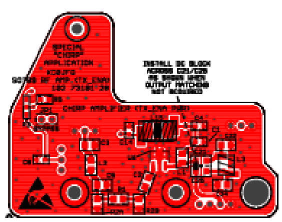

Figure 11: 102-73181-28 Board Artwork, TOP