VHF/UHF testing notebook

Errata

Switch direction of diodes D1 and D2, connect R72 to 3V3 instead of ground. **DONE!**

Design problem of R34 being on wrong side of R35, but that's not a board mod. **DONE!**

Design problem of Q1 being hard to solder, but that's not a board mod. **DONE!**

Replace R55 and R54 with 3k9, same as input level shifter. **DONE!**

~~Interface between AD5387 and readout is wrong? Figure out what and fix.~~ Test setup issue, not the interface. Was not driving it at double LO frequency.

Alternative differential driver circuit? See the LTSpice model new_tx_signal_path.asc

RF switches are wrong. V_EN was +5V – should be +1.8V. V_CTRL was +3.3V – should be +1.8V. Replace with AS179-92LF.

To match the T41, the first IF frequency should be 48 kHz. So the LPF frequency needs to change.

Alternative switch options

Frequency: 30 MHz to 4 GHz SMD

Cheapest option (SKY13453-385LF) needs 1.8V logic. AS179-92LF looks like the winner.

Part	Internal DC block?	Single control line?	3.3V tolerant?	Footprint	Cost
SKYA21012	No	No	Yes	QFNish	$0.98
HMC544AETR	No	No	Yes	SOT26	$3.05
AS179-92LF	No	No	Yes	SOT	$0.84

LO testing

I2C/SPI Pin map

Function	Logic channel
SDA	4
SCL	2
MOSI	0
MISO	1
CLK	3
LO_SEL	5

Function	Logic channel
MOD_SEL	7
LD	N/A
CE	6

Step 1

Look at I2C traffic in Pulseview. I2C bus is at 100 kHz.

This all works.

LO RF testing

Generate signals at 400 & 800 MHz – make sure it's clean.

Result is beautifully clean! Hooray!

Transmit testing

DC levels are wrong. We want the mean DC level to be 1.4V. It is actually 2.21V. This exceeds the maximum common mode voltage level of 2V.

Why is it so high?

Need to adjust IN2+ to be 1.45V, same as the input level. Replace R55 and R54 with 3k9, same as input level shifter.

DC coupled AC coupled

Perfect!

LTC5599 configuration

In the function where I try to read the chipID, this code is executed:

```
mcp.digitalWrite(SRC_PIN, HIGH); // VHF/UHF mode
digitalWrite(TXRX_PIN, HIGH); // TX State
delay(250);
...
if(!this->i2c_write(functionID, txData, txLen))
    return false;
//read in the data that came from MISO
return i2c_read(readBuf, rxLen);
```

What happens on the busses as I do this? First this burst of traffic is seen on the I2C bus:

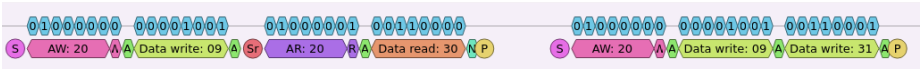


Figure 1: Initial I2C traffic

About 260 ms later we see this traffic on the busses:

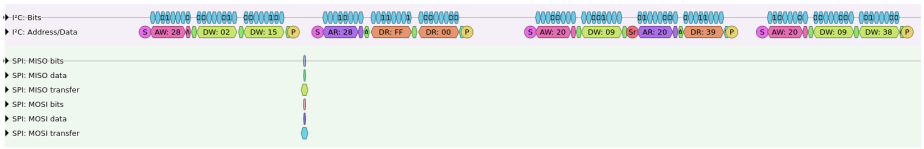


Figure 2: 260ms later traffic

Zooming in on the SPI bus traffic we see that it looks like this:

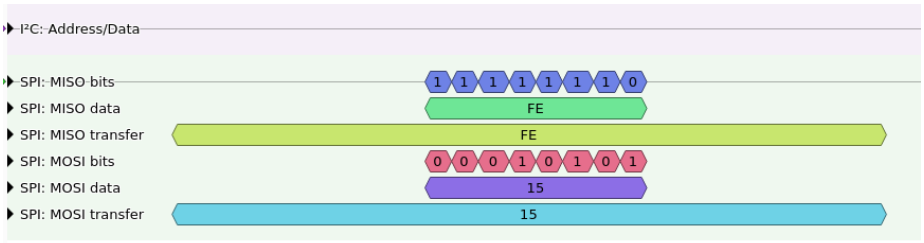


Figure 3: SPI traffic

Let's break it down. The I2C address of the mcp chip is `0x20`.The I2C address of the bridge chip is `0x28`. The slave number of the LTC5599 is `1`.

So the first two bursts of traffic are the MCP configuration. Let's ignore them.

The code to write to the SPI bus and the variables were:

```
this->i2c_write(functionID, txData, txLen)
```

Variable	Code	Value (binary)	Value (hex)
functionID	(1 « slaveNum)	0b00000010	0x02
txData	(0x0A « 1)	0b00010101	0x15
txLen	1	0b00000001	0x01

`0x0A` here is the address of the register we want to read.

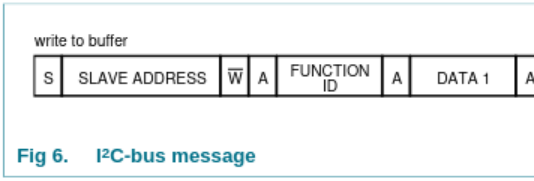
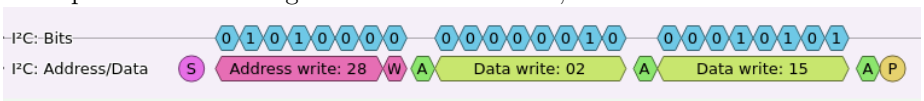


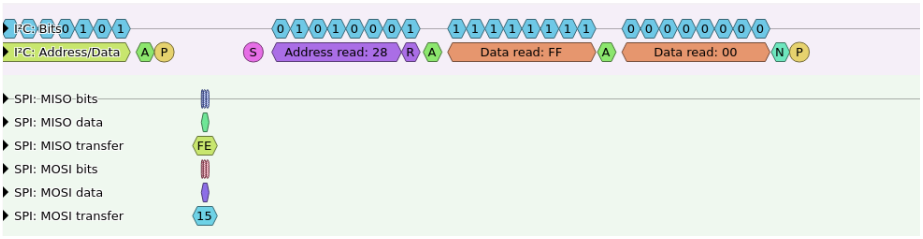
Fig 6. I²C-bus message

We expect the I2C message to have this structure, and it does:

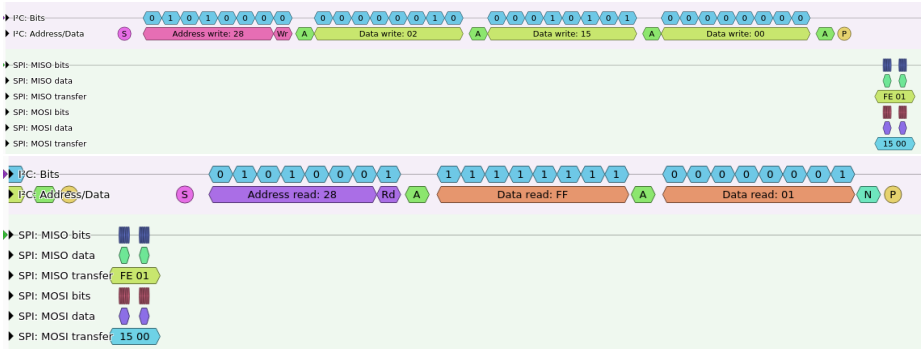


Then the SPI traffic follows 15.5 us later and indeed contains the register address `0x0A` shifted by 1 and or'ed with `1`, as we expect.

But my attempt to read data from the I2C buffer failed – there was nothing to read, because the bridge chip didn't read anything from the SPI line.



How do I get the bridge chip to read more bytes from the SPI line? Simple: write an empty byte after the `0x15` byte to get it to read the MISO line. This works!



Measurements Round 1

With fLO = 440 MHz I set the registers to:

QO = 175 IO = 150 DG = 4

When transmitting USB:

Item	Power [dB]
USB tone	-14.64
Carrier	-47.14
Carrier suppression	30.5

When transmitting LSB:

Item	Power [dB]
LSB tone	-19
Image tone (USB)	-42.7
Carrier	-47
Carrier suppression	28
Sideband suppression	24

With fLO = 146 MHz I set the registers to:

Register	Addr	Value
DG	1	4
IO	2	10010110
QO	3	10110100
IQG	4	1011010
IQP	5	111000

When transmitting LSB:

Item	Power [dB]
LSB tone	-17.5
Image tone (USB)	-53.6
Carrier	-49.1
Carrier suppression	31.6
Sideband suppression	36.1

Can we get to 43 dB? The chip is supposedly capable of 50 dB. See charts on page 6.

Testing round 1 (INVALID?)

Automated testing was performed using the Jupyter notebook `LTC5599_parameter_sweep.ipynb`.

The optimum parameters in each amateur radio band were found to be:

Band	I DC Offset	Q DC Offset	Gain Ratio	Phase Balance
52 MHz	148	174	128	40
146 MHz	152	179	120	-16
222 MHz	151	177	144	-80
430 MHz	151	176	120	8
915 MHz	155	179	120	- 8
1270 MHz	154	181	128	0

Summary is that the gain ratio and phase balance don't make a difference, so keep them at their default values. But change the I and Q DC offsets to these values:

I DC Offset	Q DC Offset
151	177

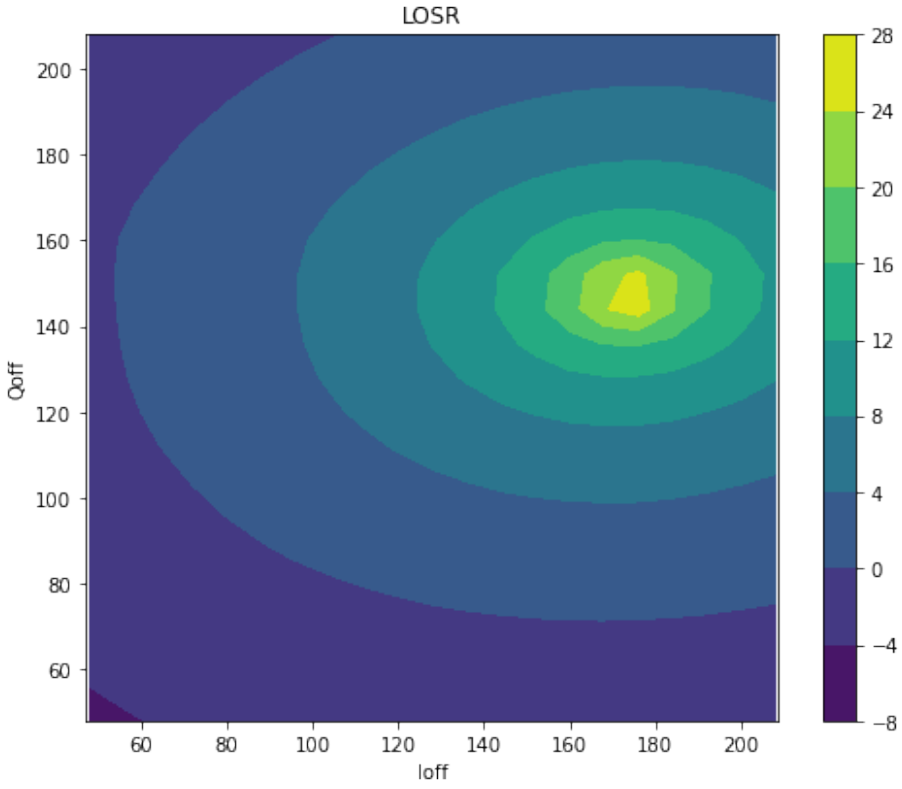
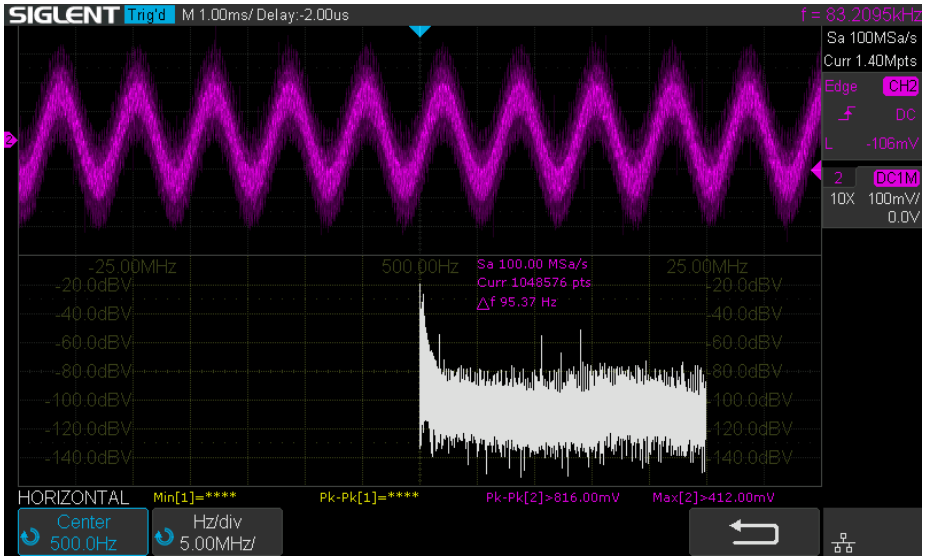


Figure 4: The LO suppression ratio (axes are switched)

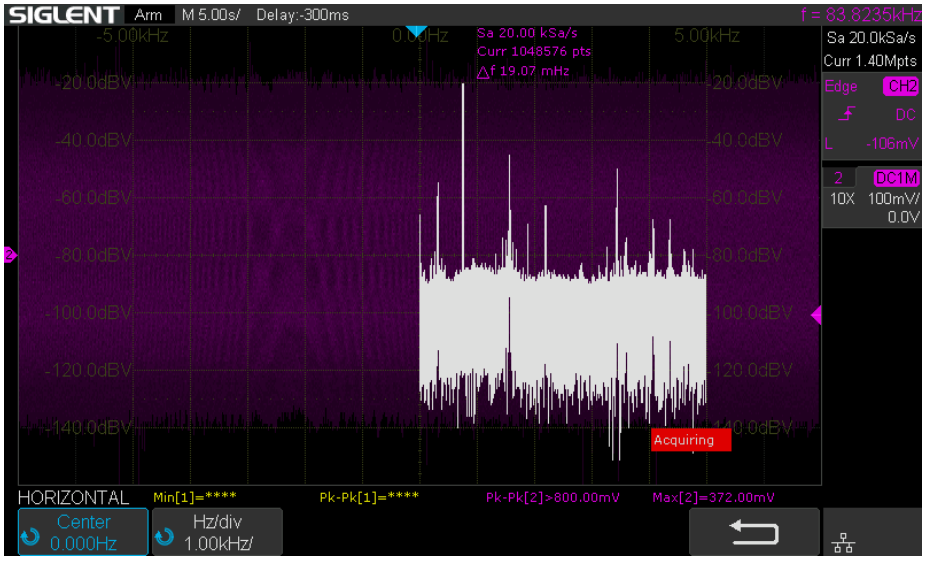
Audio transmit spectrum

What is the spectral output of the teensy audio feed? Screenshots from spectrum analyzer grabbed following these instructions.

```
lxi screenshot --address 192.168.86.61
```

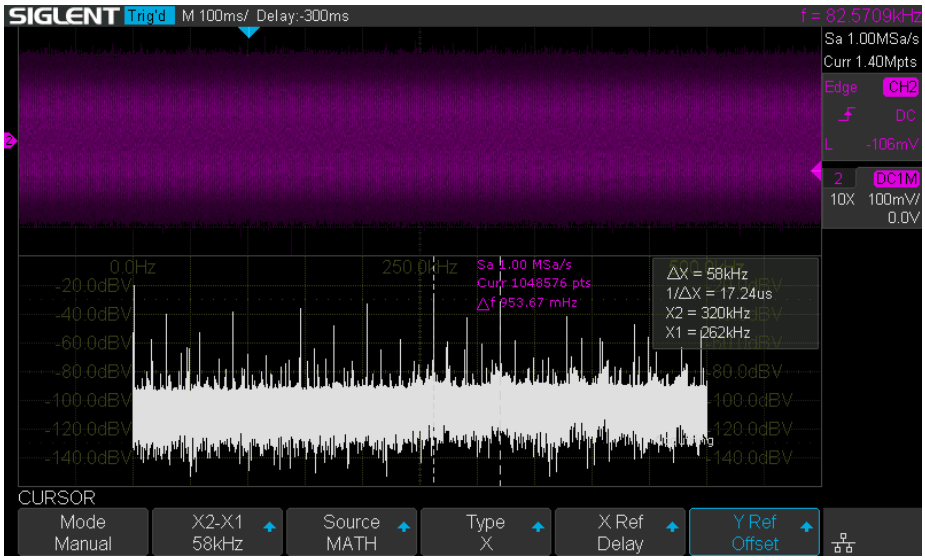


Period is ~1.2 ms, which is 833 Hz. This is right. Zoom in on the spectrum.



Nice pure sine tone. Are there images at the out frequencies? Digital products?


SIGLENT
Trig'd
M 100ms/ Delay:-300ms
f = 77.8753kHz
Sa 1.00MSa/s
Curr 1.40Mpts
Edge
CH2
DC
L -106mV
2
DC1M
10X 100mV/
0.0V
0.0Hz
250.0kHz
Sa 1.00 MSa/s
Curr 1048576 pts
Δf 953.67 mHz
ΔX = -58kHz
1/ΔX = -17.24us
X2 = 29kHz
X1 = 87kHz
-20.0dBV
-40.0dBV
-60.0dBV
-80.0dBV
-100.0dBV
-120.0dBV
-140.0dBV
CURSOR
Mode
Manual
X1
87kHz
Source
MATH
Type
X
X Ref
Delay
Y Ref
Offset
SIGLENT
Trig'd
M 100ms/ Delay:-300ms
f = 77.6622kHz
Sa 1.00MSa/s
Curr 1.40Mpts
Edge
CH2
DC
L -106mV
2
DC1M
10X 100mV/
0.0V
0.0Hz
250.0kHz
Sa 1.00 MSa/s
Curr 1048576 pts
Δf 953.67 mHz
ΔX = 58kHz
1/ΔX = 17.24us
X2 = 204kHz
X1 = 146kHz
-20.0dBV
-40.0dBV
-60.0dBV
-80.0dBV
-100.0dBV
-120.0dBV
-140.0dBV
CURSOR
Mode
Manual
X2
204kHz
Source
MATH
Type
X
X Ref
Delay
Y Ref
Offset



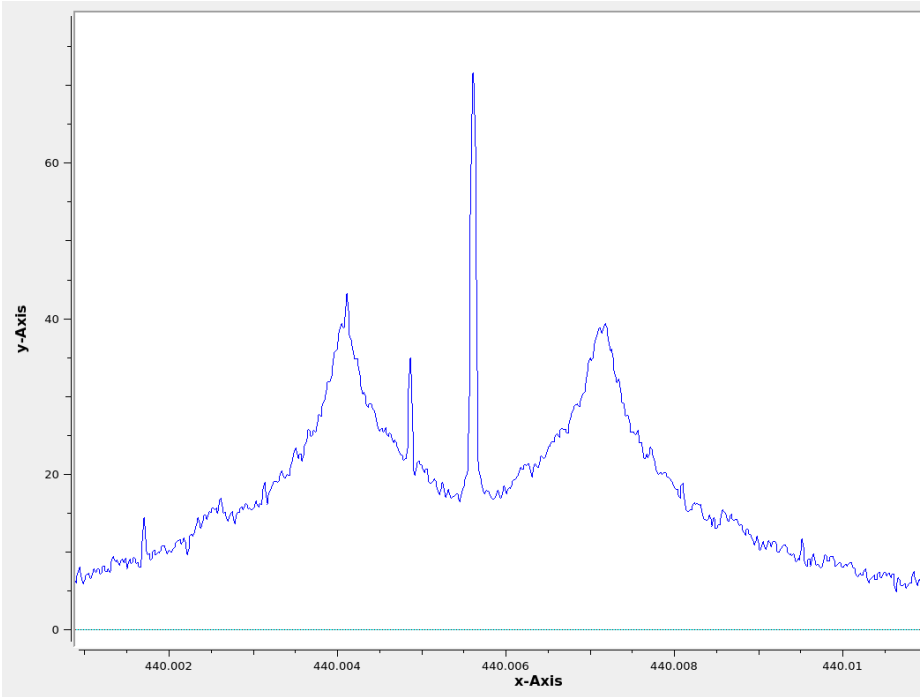
Harmonics are ~58 kHz apart and happen according to this pattern:

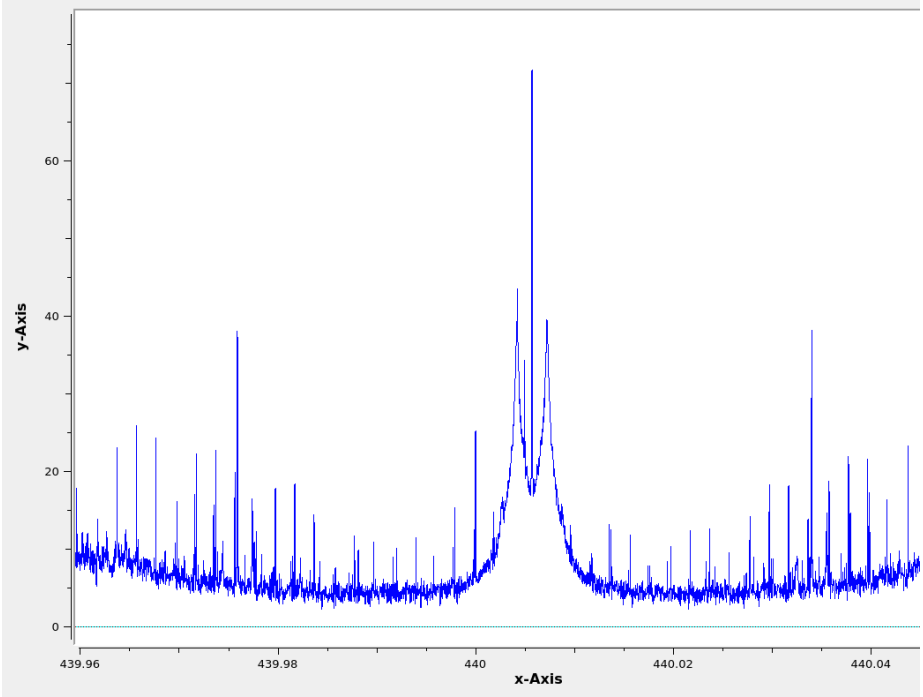
```
N*29 kHz where N = 1, 3, 5, 7, ...
```

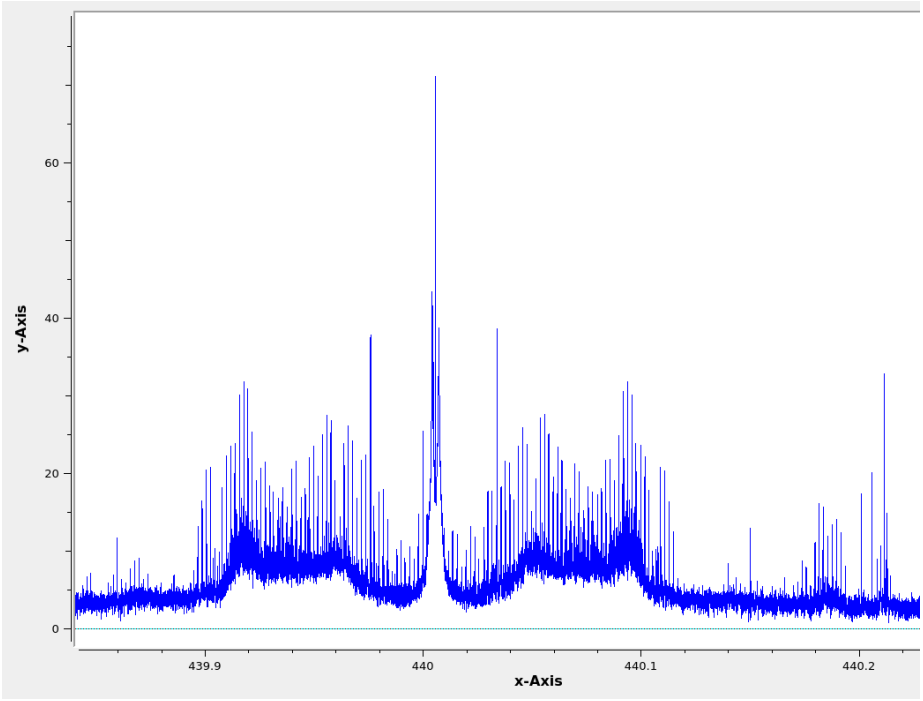
Looks like a square wave with a fundamental at 29 kHz. Not sure this has anything to do with the DAC.

Transmit RF spectrum

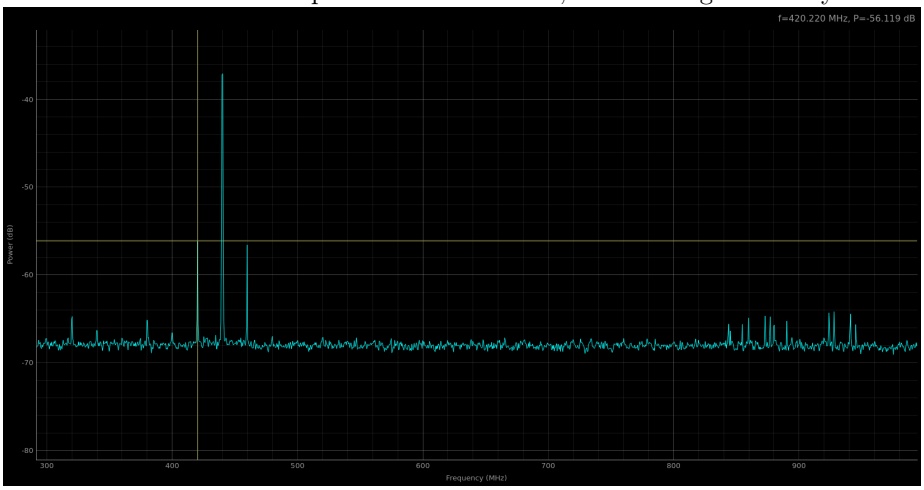
Set it up to transmit with LO frequency of 440 MHz, USB CW tone. I DC Offset = 151. Q DC Offset = 177.


y-Axis
60
40
20
0
440.002
440.004
440.006
440.008
440.01
x-Axis

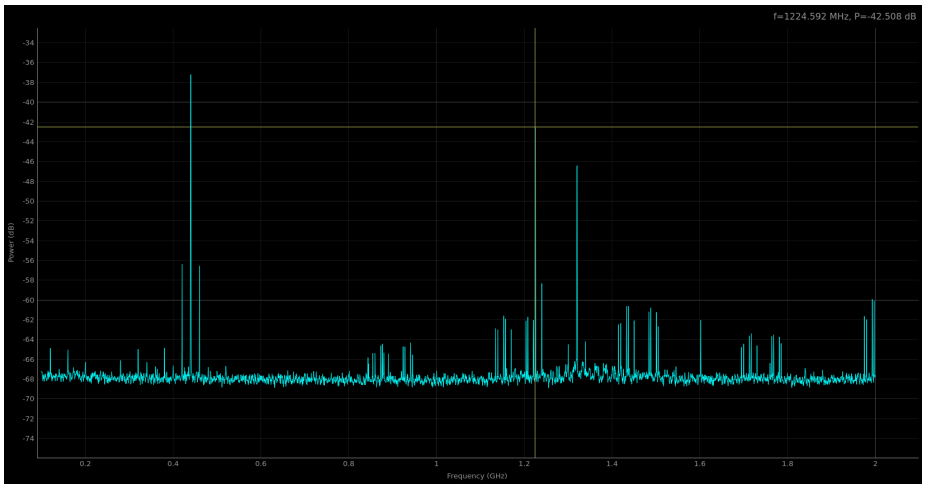

y-Axis
60
40
20
0
439.96
439.98
440
440.02
440.04
x-Axis



Looking at the broader RF spectrum we see tones at +/- 20 MHz and 30dB down from the main tone at 440 MHz. This is an artifact of the HackRF sample rate of 20 MHz, so nothing to worry about.



The wider RF spectrum shows the 3rd harmonic clearly at 1320 MHz. We also see an unexpected spur at 1226 MHz – this is an internal HackRF birdie and is still there when we turn the transmitter off.



Transmit RF power

At 340 MHz using the TinySA. Span 5 MHz, RBW = 35 kHz. Tone power is -54 dBm.

RBW [kHz]	Power [dBm]
3	-53.7
10	-54.2
35	-54.0
100	-53.2

This feels low… investigate.

Voltage in to the mixer is 13 mV RMS. 8dB of attenuation from mixer.

13 mV RMS = -25 dBm. Add in 8dB of attenuation: -33 dBm. I'm measuring a lot less.

Testing round 2

After removing switch from LO, repeat testing since LO power is now much higher. Automated testing was performed using the Jupyter notebook `LTC5599_parameter_sweep.ipynb`.

The optimum parameters in each amateur radio band were found to be:

Band	FFT offset	I DC Offset	Q DC Offset	Gain Ratio	Phase Balance
52 MHz	600	148	173	128	-48
146 MHz	1700	153	179		

Band	FFT offset	I DC Offset	Q DC Offset	Gain Ratio	Phase Balance
222 MHz	260	151	177	112	-80
430 MHz	500	152	176	128	16
915 MHz					
1270 MHz					

Results are the same.

Receive testing

What are the expected power levels? P1dB of amplifier is ~10dBm, so stay below -20dBm input. At first I tried -30 dBm, but it saturated, so switched to 0dB of gain in the HackRF, which equals -50 dBm input. This has -19 dBm input to the ADL5387 (measured).

Stage	Power [dBm]	Voltage [mV rms]	Actual
Input	-50		-50dBm @ 144 MHz
After PSA-8+	-20		-19 dBm
LO @ 288 MHz			-4 dBm
After ADL5387	-15.6	37	Hard to measure
After level shifter		371	640 mV

Note: to match the T41, the first IF frequency should be 48 kHz. So if our RF is at 144,048 kHz, we want the LO to be at 144,000 kHz.

Sweep the bands

Measure the I and Q outputs of the level shifter when varying the RF frequency. This gets the passband. Python notebook that does this is `receive_passband_sweep.ipynb` and data is stored in `freq_sweep_12JULY.pickle`.

Note: you can't trust the 1270 MHz numbers above. The system is not producing any signal above 1060 MHz. The signal generator appears to be working fine, so it's likely the LO? Investigate!

Result: the RF switch was blown, because I select the wrong damn part. Need 3.3V tolerant part.

Wideband sweep

Data for wideband sweep is in `wideband_freq_sweep_13JULY.pickle`. Swept around one frequency, 430 MHz.

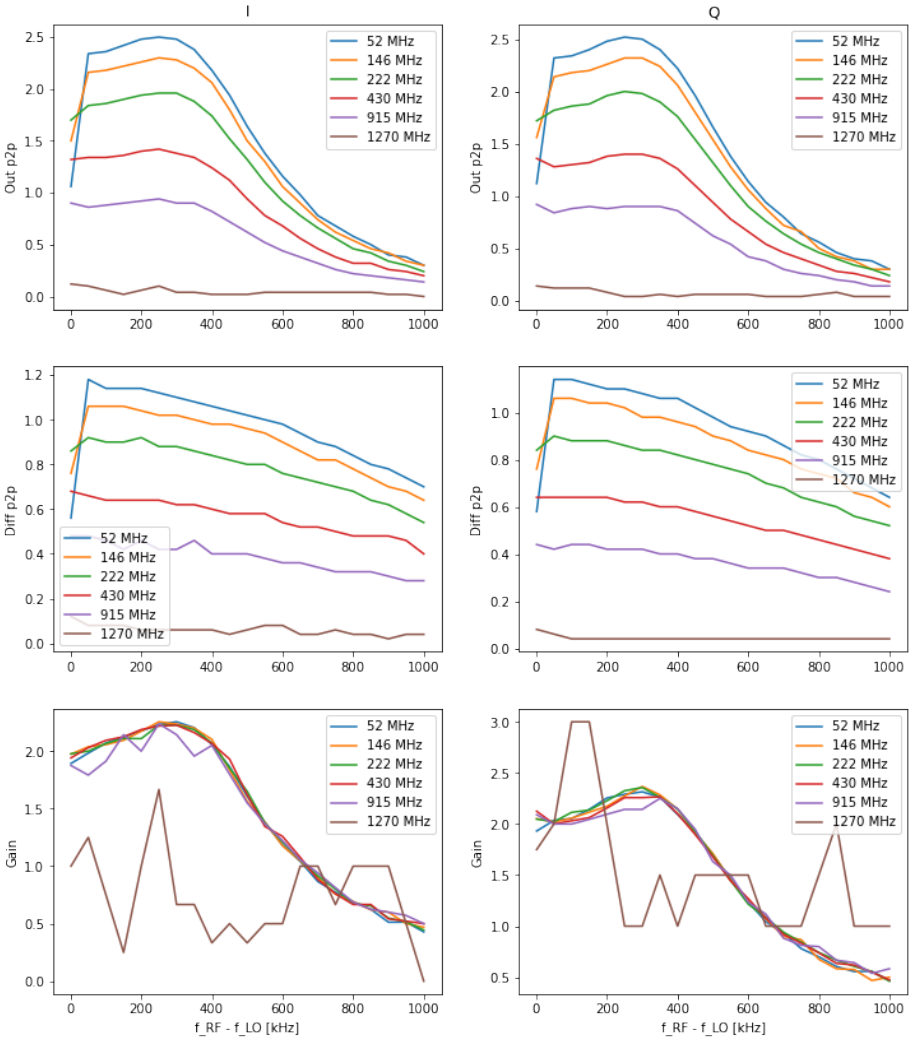


Figure 5: Receiver passbands at each LO frequency

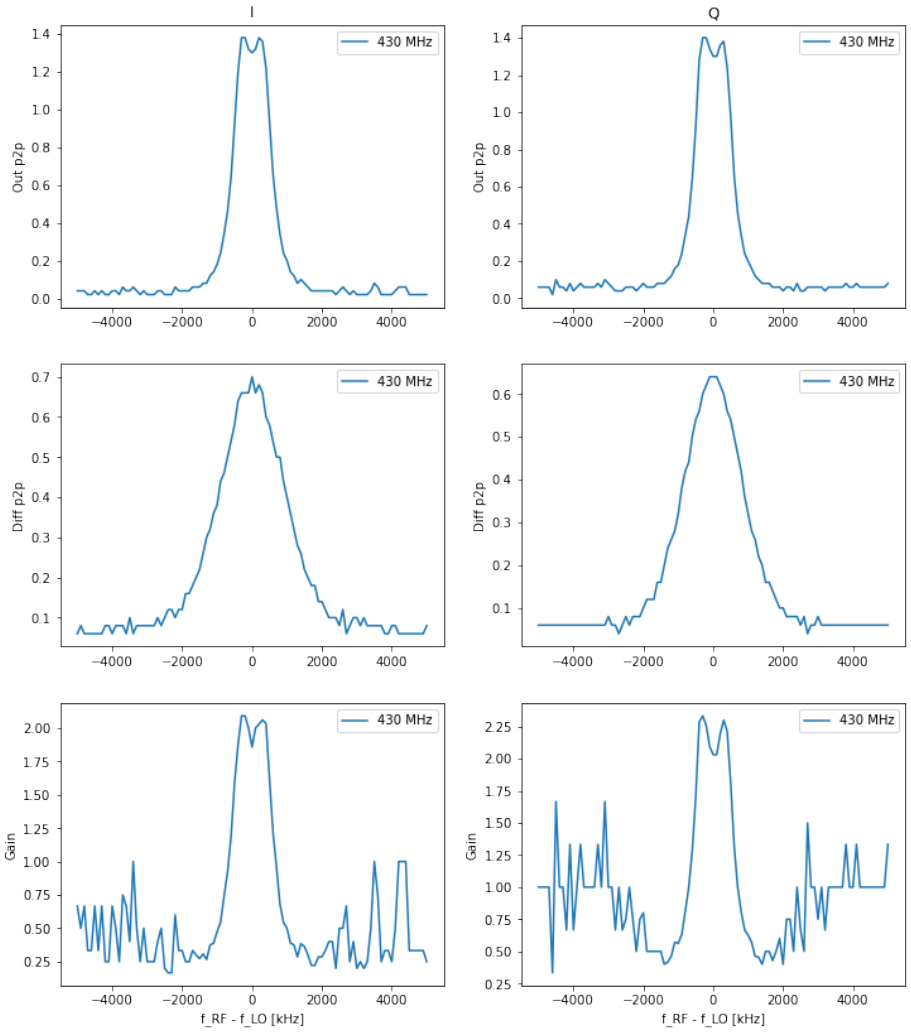
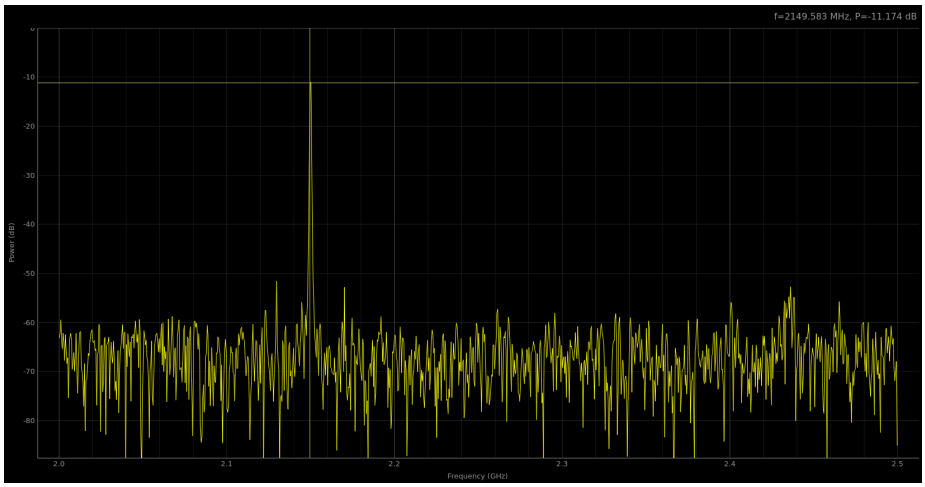


Figure 6: Wideband sweep around 430 MHz

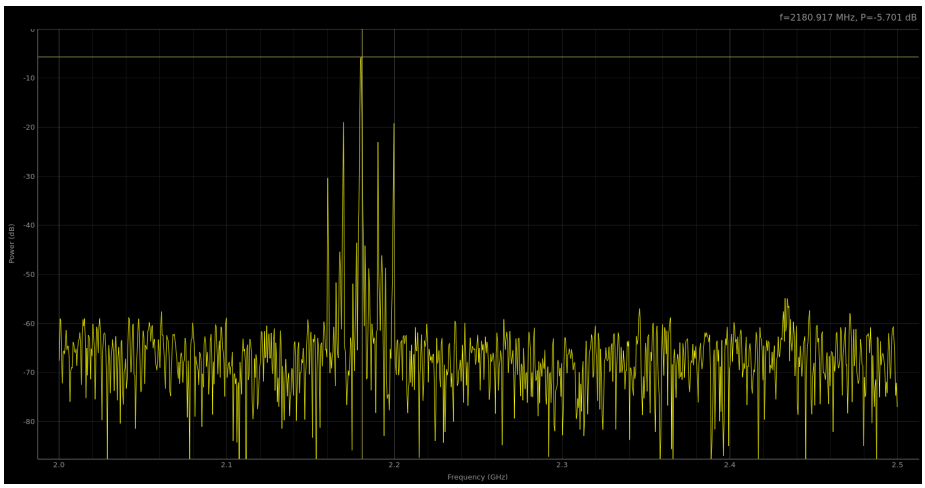
LO research and Maximum frequency

Hooking the LO output up to the HackRF as a spectrum analyzer, using QSpectrumAnalyzer software.

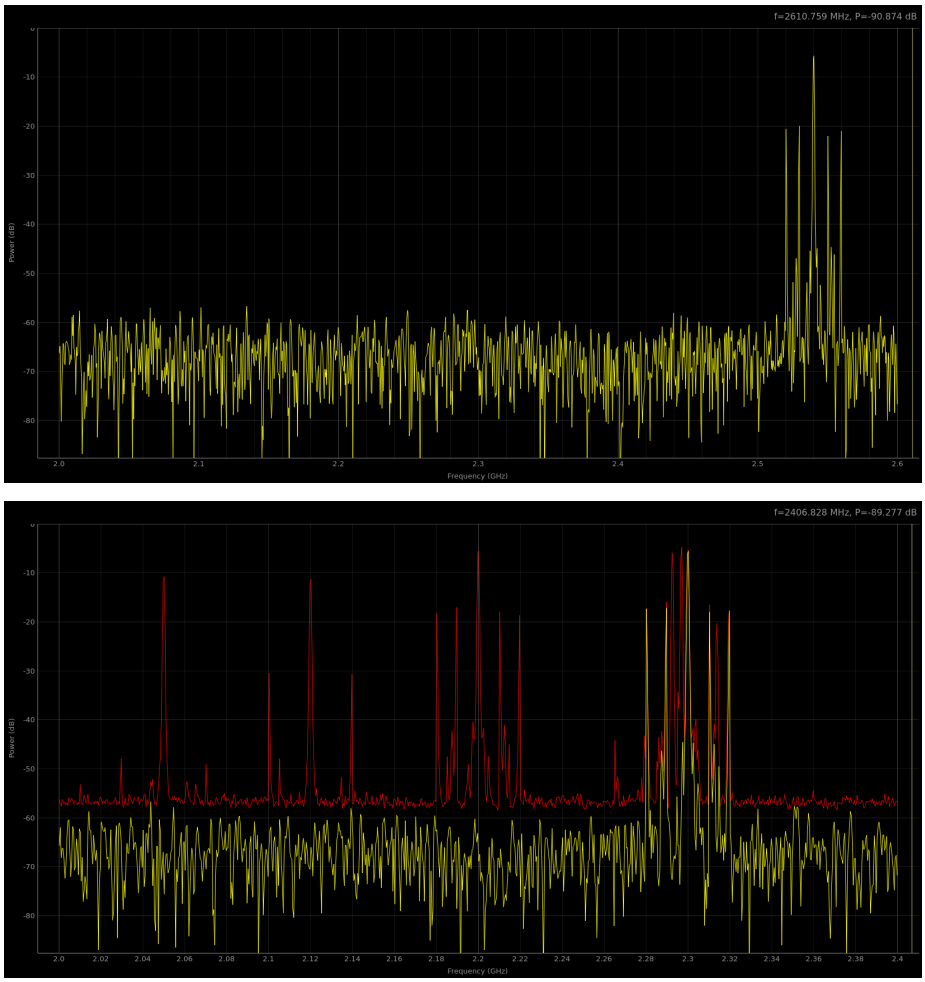
When LO set to 2150 MHz and below, all looks fine, like this:



But move the frequency up just a tiny bit and it looks like this:



And the result is the same at the LO frequency that corresponds to the 1270 MHz band, i.e., 2540 MHz:



Something in the LO command code goes wonky when the RF frequency exceeds 2150 MHz. The sidebands creep up at you approach 2160 MHz, and gets really wonky when you reach 2160 MHz. By 2200 MHz they are really bad and stay about equally bad for a while. And the power gets much lower above ~ 2.8 GHz.

What does the datasheet say? Should the system be changing this much? The datasheet seems to think it's fine. Does the eval board behave the same way? Yes – it does.

What changes in the code around an RF frequency of 2160 MHz to make the sidebands of the LO signal so bad?

```
output frequency [MHz]: 2050.0000000000
pfd_freq = 0.10
N = 41000.00
INT, MOD, FRAC, output_divider, band_select_clock_divider(41000, 2, 0, 2, 1)

output frequency [MHz]: 2060.0000000000
pfd_freq = 0.10
N = 41200.00
```

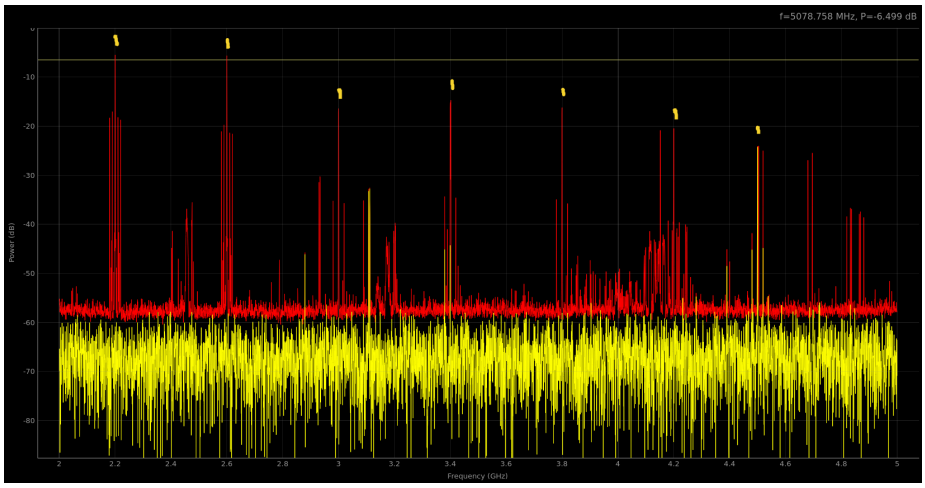


Figure 7: The signals generated by the LO are indicated by yellow ticks. The rest is interference.

```
INT, MOD, FRAC, output_divider, band_select_clock_divider(41200, 2, 0, 2, 1)
```

```
output frequency [MHz]: 2200.0000000000
pfd_freq = 0.10
N = 22000.00
INT, MOD, FRAC, output_divider, band_select_clock_divider(22000, 2, 0, 1, 1)
```

Video of making the sweep is `sweeping_LO_video.mp4`.

Looking at the parameters, they make a change from one division state to another at 2200 MHz:

Freq [MHz]	pfd_freq	N	INT	MOD	FRAC	output_divider	band_select_clock_divider
2192	0.1	43840	43840	2	0	2	1
2194	0.1	43880	43880	2	0	2	1
2196	0.1	43920	43920	2	0	2	1
2198	0.1	43960	43960	2	0	2	1
2200	0.1	22000	22000	2	0	1	1
2202	0.1	22020	22020	2	0	1	1
2204	0.1	22040	22040	2	0	1	1

That doesn't explain the poor quality around 2160 MHz though. How does the signal quality vary with frequency? Are there other places where it is poor?