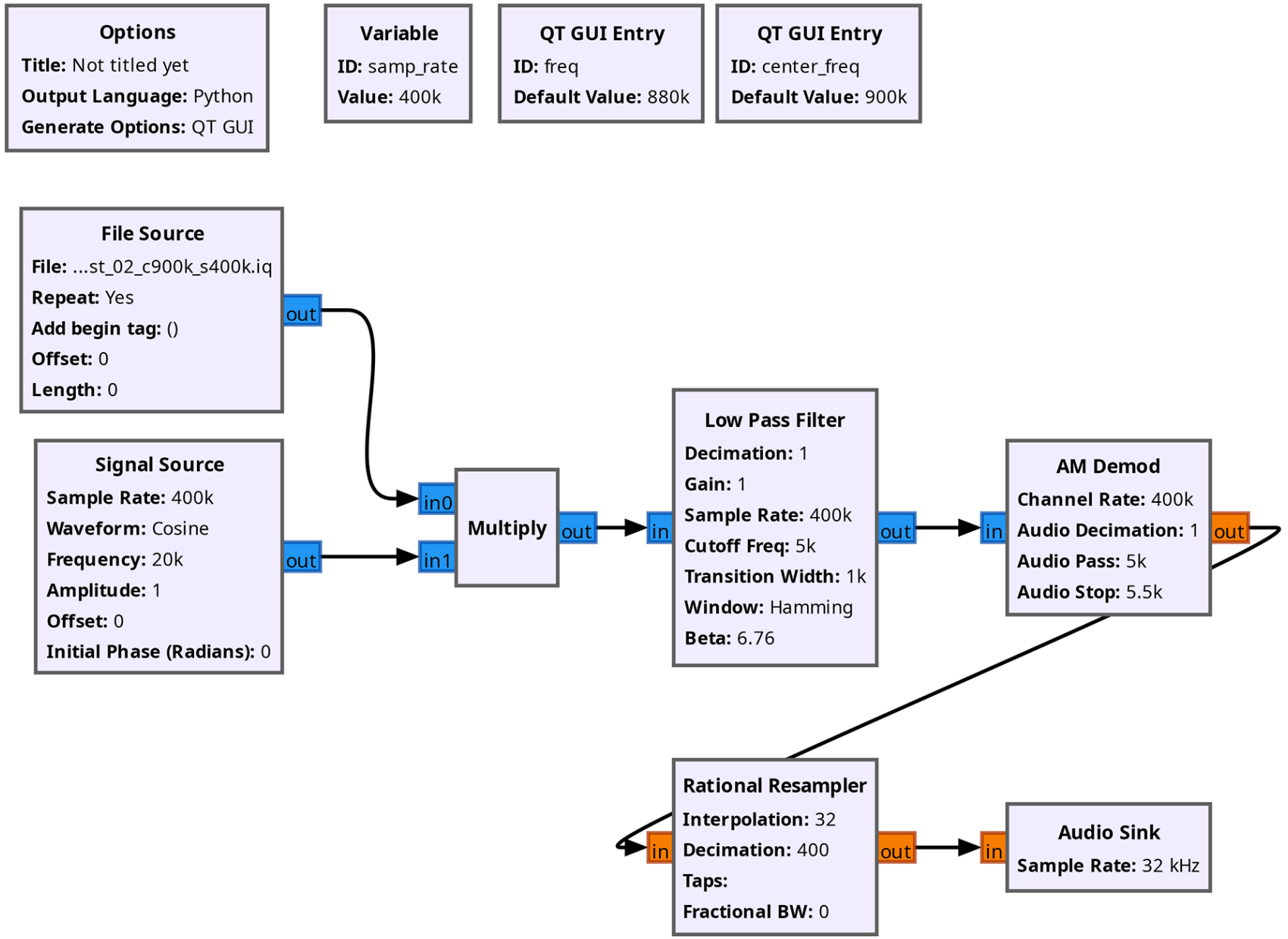
BAA GNU Radio Meeting

by Andrew Thornett

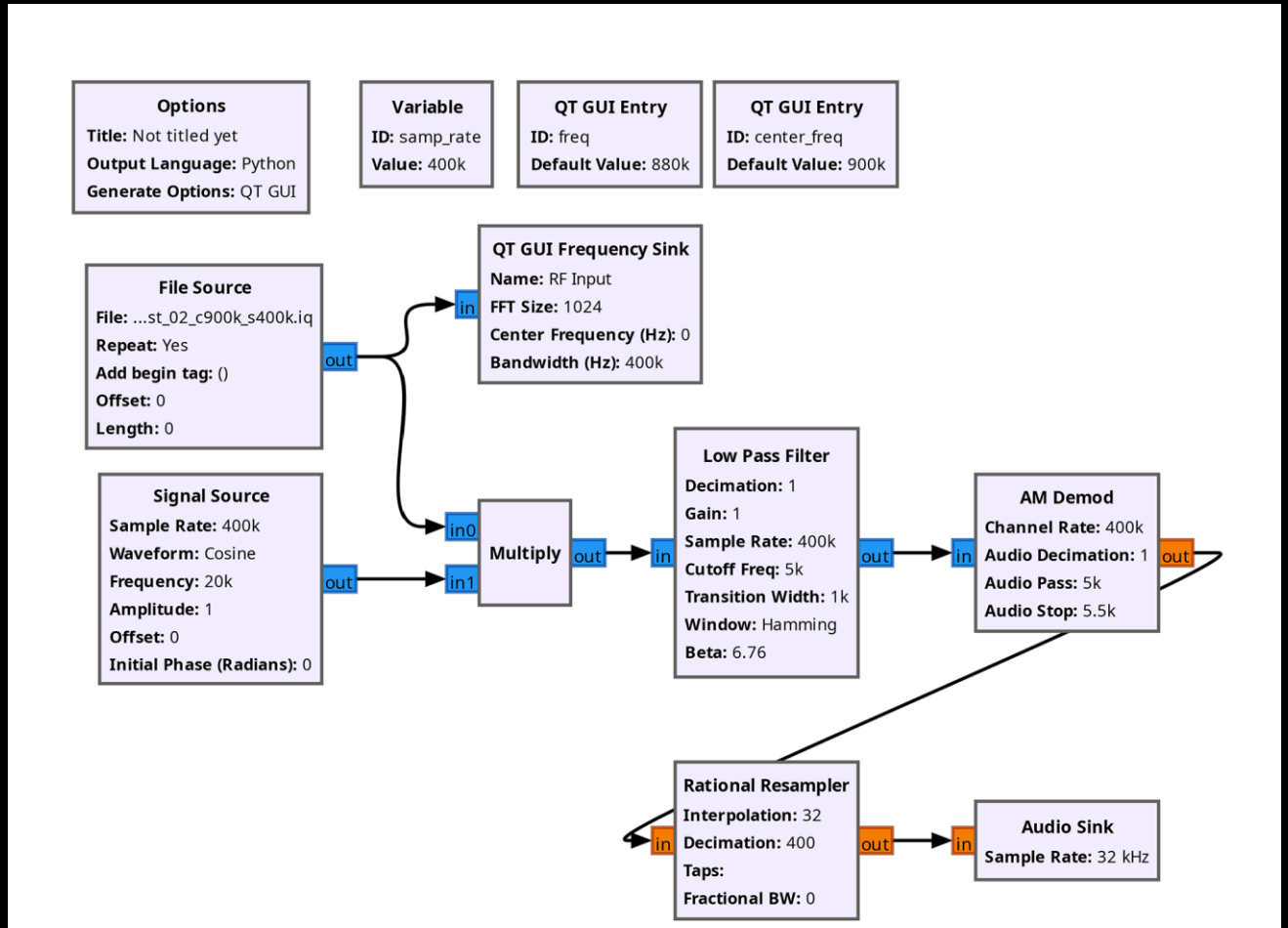
Start by loading up into GNU Radio Companion the AM radio we created in the last practical session I gave



The AM radio flowgraph revisited

Examining Input Radio Frequency Data

- Add QT GUI Frequency Sink to your flowgraph and connect it to the output of the File Source.
- Radio data is entering our flowgraph from this block via a file filled with captured RF data.
- This new QT GUI sink will show frequencies contained in radio data.
- Set new block's Name to "RF Input".

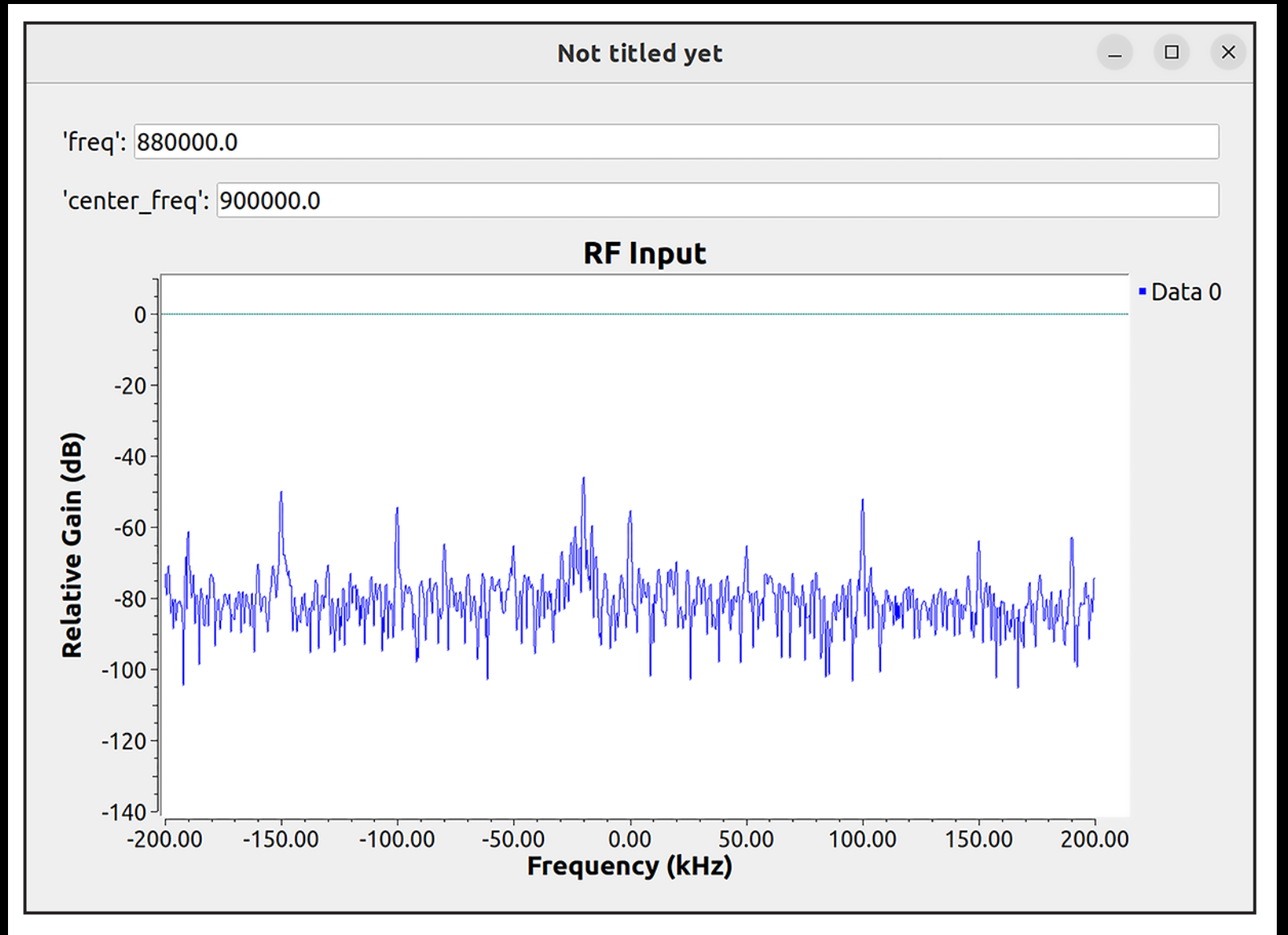


AM flowgraph with a frequency sink

Output Tab Colours

- Output tab on File Source block is blue, indicating that the data coming from input file is complex type.
- Hence, we don’t need to change the Type of the Frequency Sink to Float.
- In general, all radio data is complex.
- Radio data starts out complex and typically transition to a floating-point format as move closer to sinks.

Execute your flowgraph to look at Fast Fourier transform (FFT) of radio data.

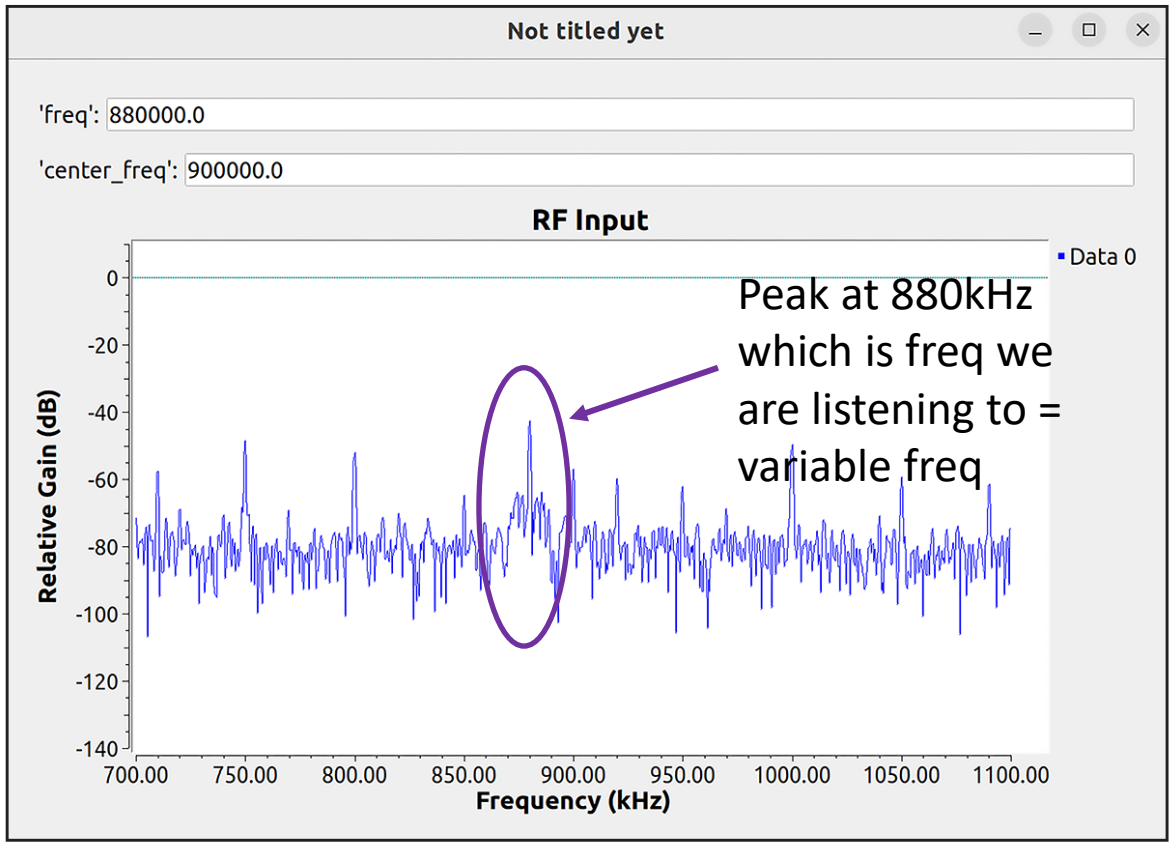


Peaks in AM data

- FFT shows range frequencies contained in RF data.
- Some peaks stand out - commonly seen in raw AM data = several stations transmitting at the same time on different frequencies and creating sharp peaks.
- Plot's horizontal axis = frequencies from –200 kHz to 200 kHz. Why aren't we seeing higher frequencies like 880 kHz in the plot?
- Frequency range of radio data from File Source block doesn't extend down to 0 Hz. File contains data only range 700 kHz-1,100 kHz = 900 kHz ± 200 kHz (flowgraph variable center_freq = 900e3).

Changing horizontal axis to correct frequencies

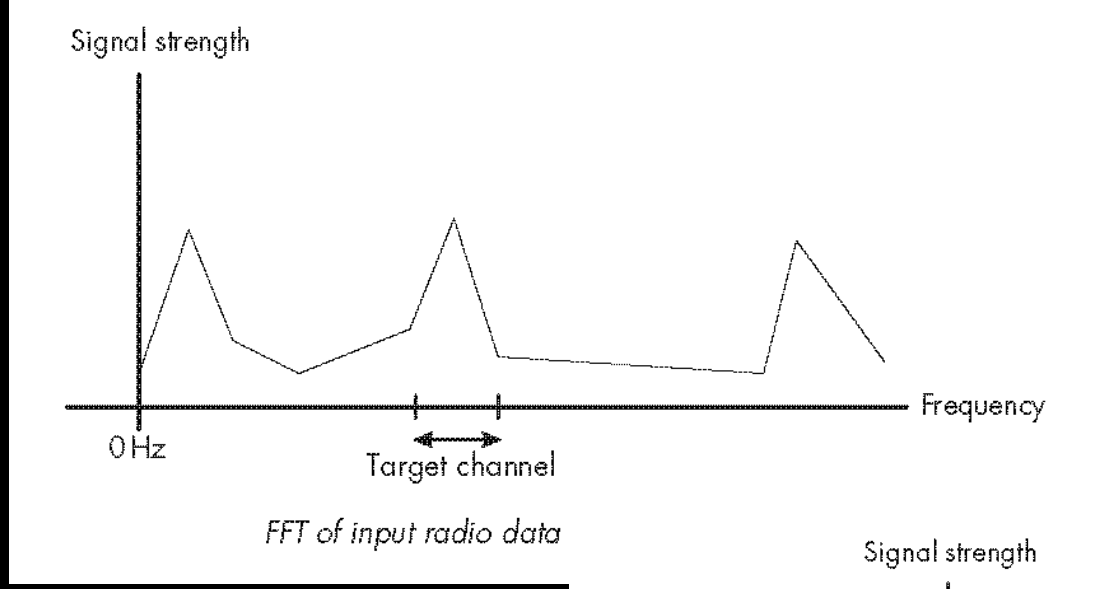
- Frequency range over which radio receiver can grab data called **input bandwidth**, here 400 kHz. Input bandwidth centred 900 kHz, resulting in captured data 700 kHz-1,100 kHz.
- Centre frequency received radio signals is not embedded in raw data, so centre freq on plot = 0 Hz, and on FFT range frequencies = –200 kHz-200 kHz.
- We can correct numbers on plot horizontal axis by providing centre frequency reference to QT GUI Frequency Sink, either by changing block's Centre Frequency (Hz) property to 900e3 (900 kHz) or using a variable like center_freq.
- Make change & re-run your flowgraph.



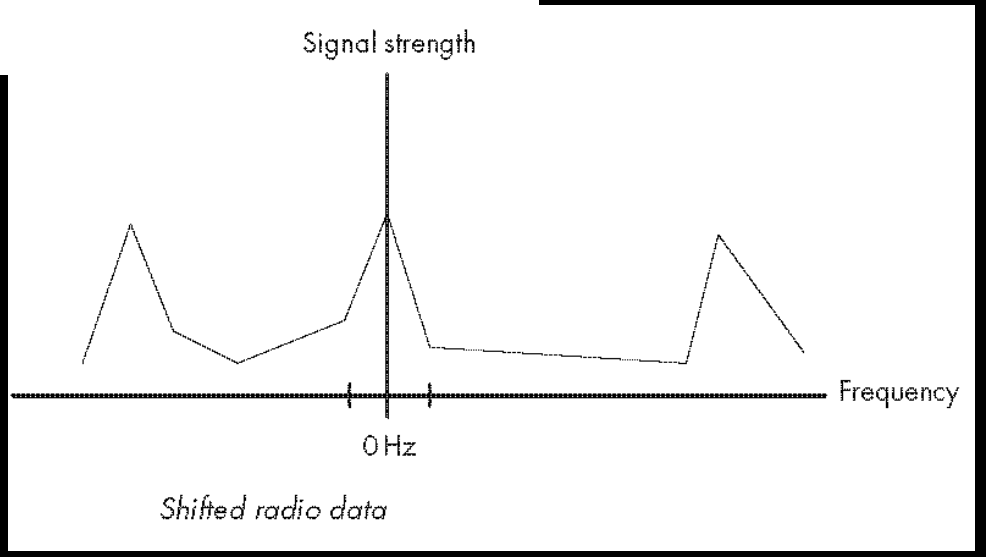
An FFT of the RF input data with a corrected center frequency

Tuning

- Tuning = focusing on one specific signal while excluding any others.
- AM receiver flowgraph implements tuning as 2-step process:
- It shifts input radio signals so that one you want is centred 0 Hz in frequency domain.
- Does this by multiplying input data by sinusoid of specific frequency & then filtering out anything that's *not the* zero-centred signal.



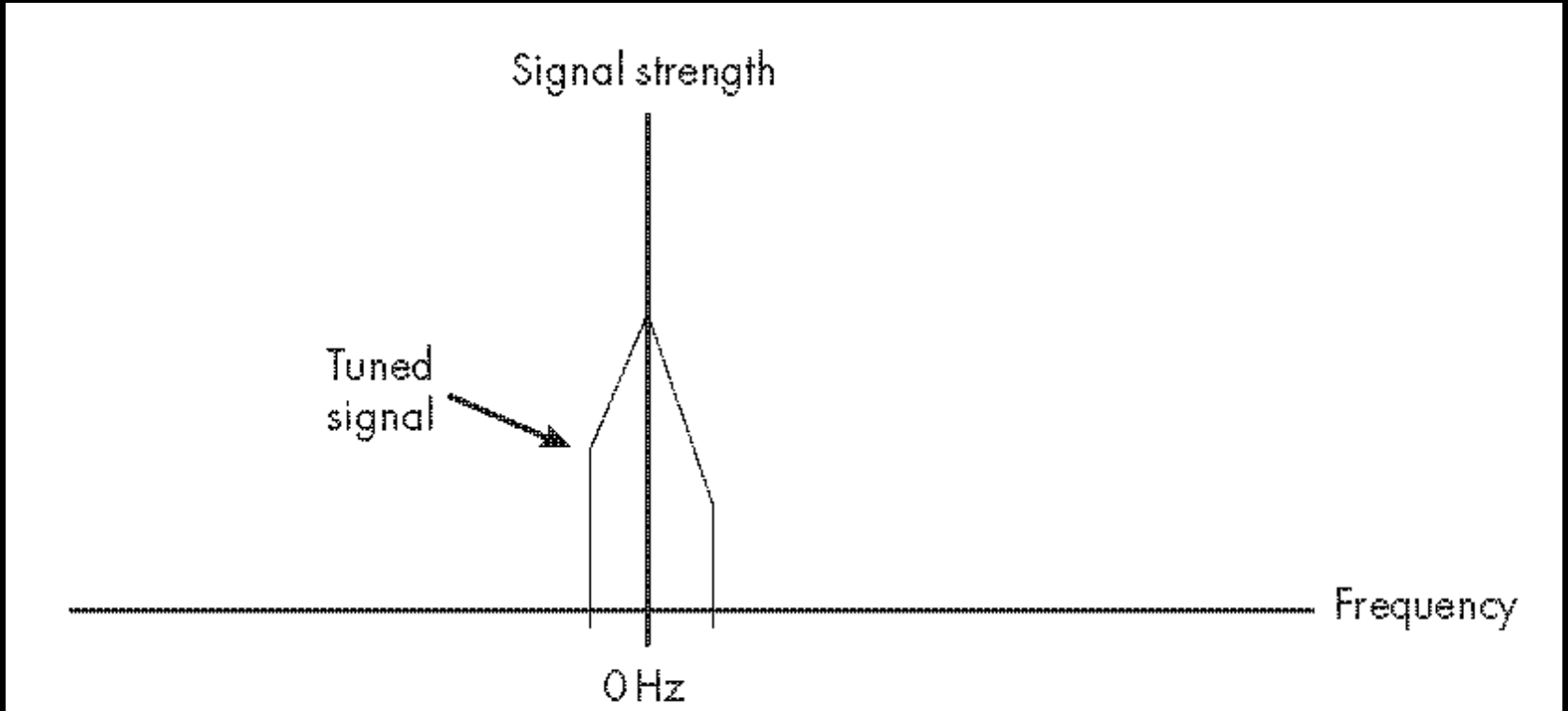
No change in shape of FFT signal



- Input data contains three different signals (each represented by FFT spike).
- First shift frequency of input radio data so that target signal centred at 0 Hz.

Filtering Radio Data

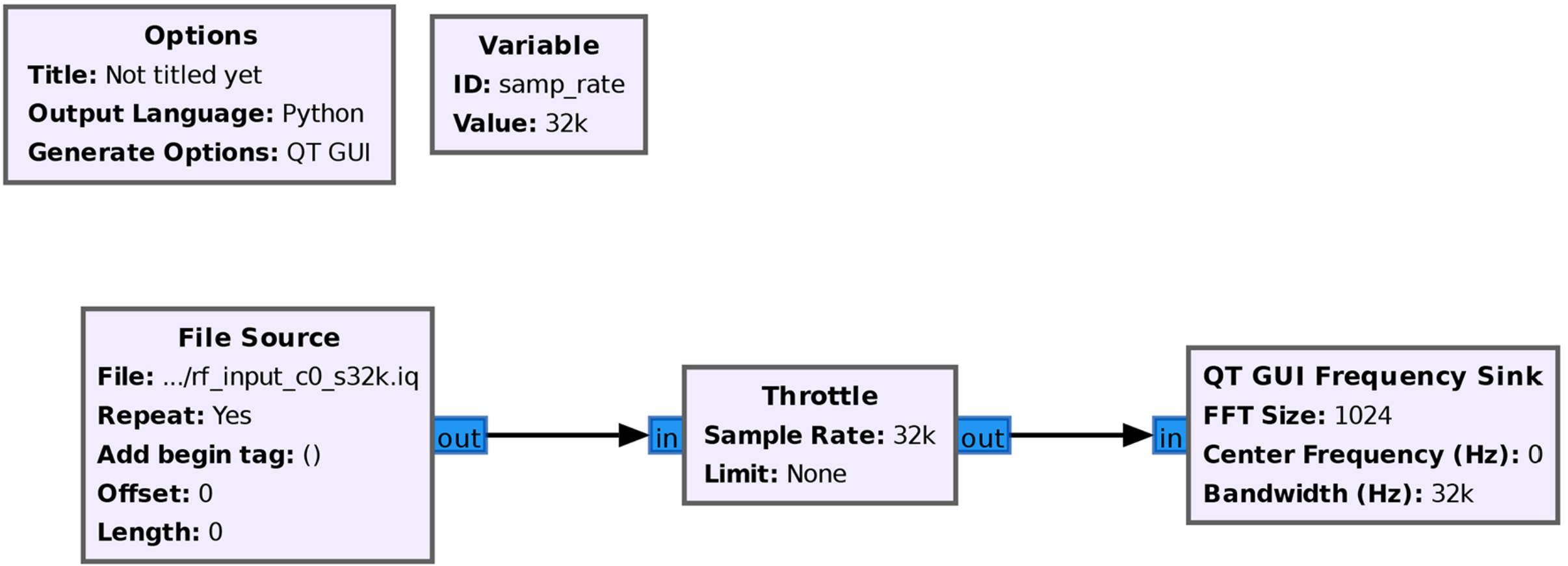
- This changes the FFT signal.
- Removes extraneous signal from frequency range of interest.



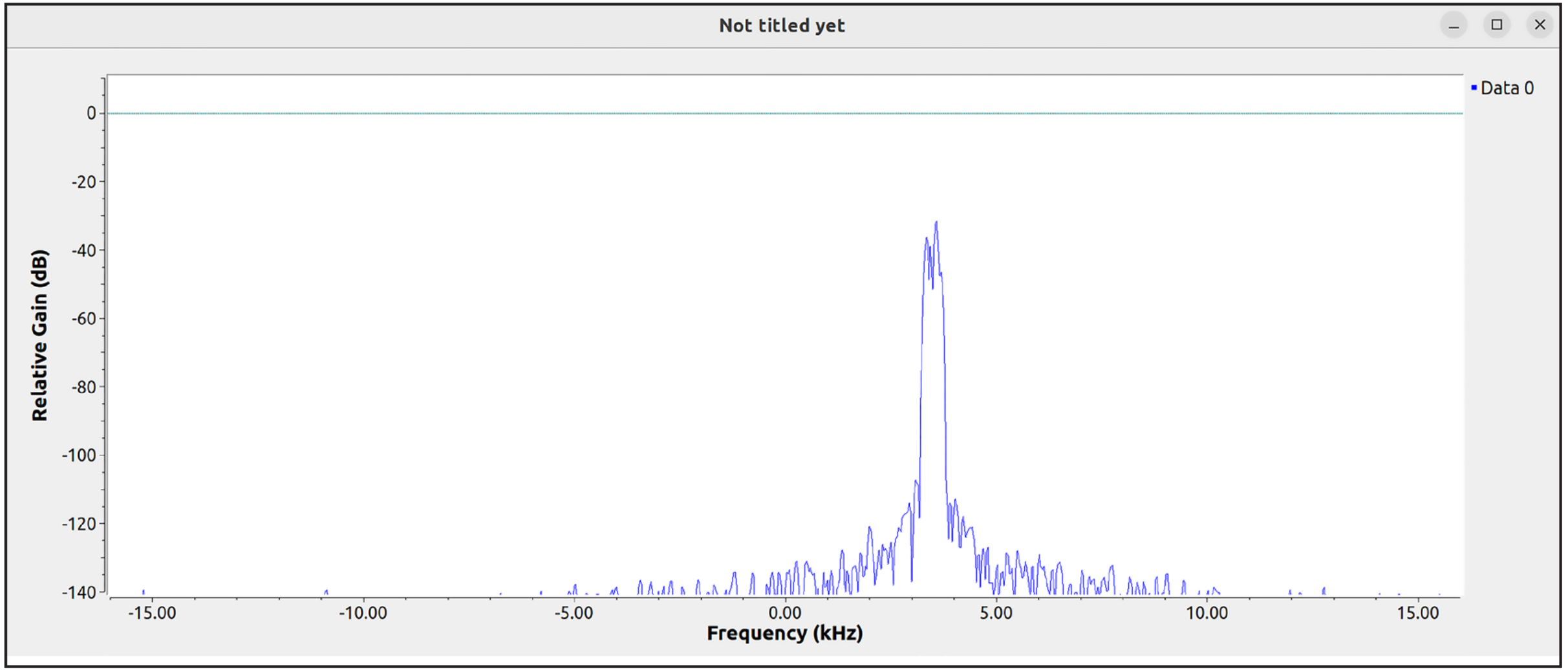
Shifted and filtered radio data

Frequency Shifting

- Way to shift frequency of some radio data is to multiply it by a complex-typed sinusoid.
- In GNU Radio Companion, set aside your AM radio for now, and create a new project called *freq_shift.grc*
- In new project, add File Source & set its File property to *ch_06/rf_input_c0_s32k.iq*.
- Add Throttle block & connect its input to output of File Source. It keeps your computer from working too hard.
- Add QT GUI Frequency Sink & connect its input to Throttle block output.
- Add appropriate options to Options block.



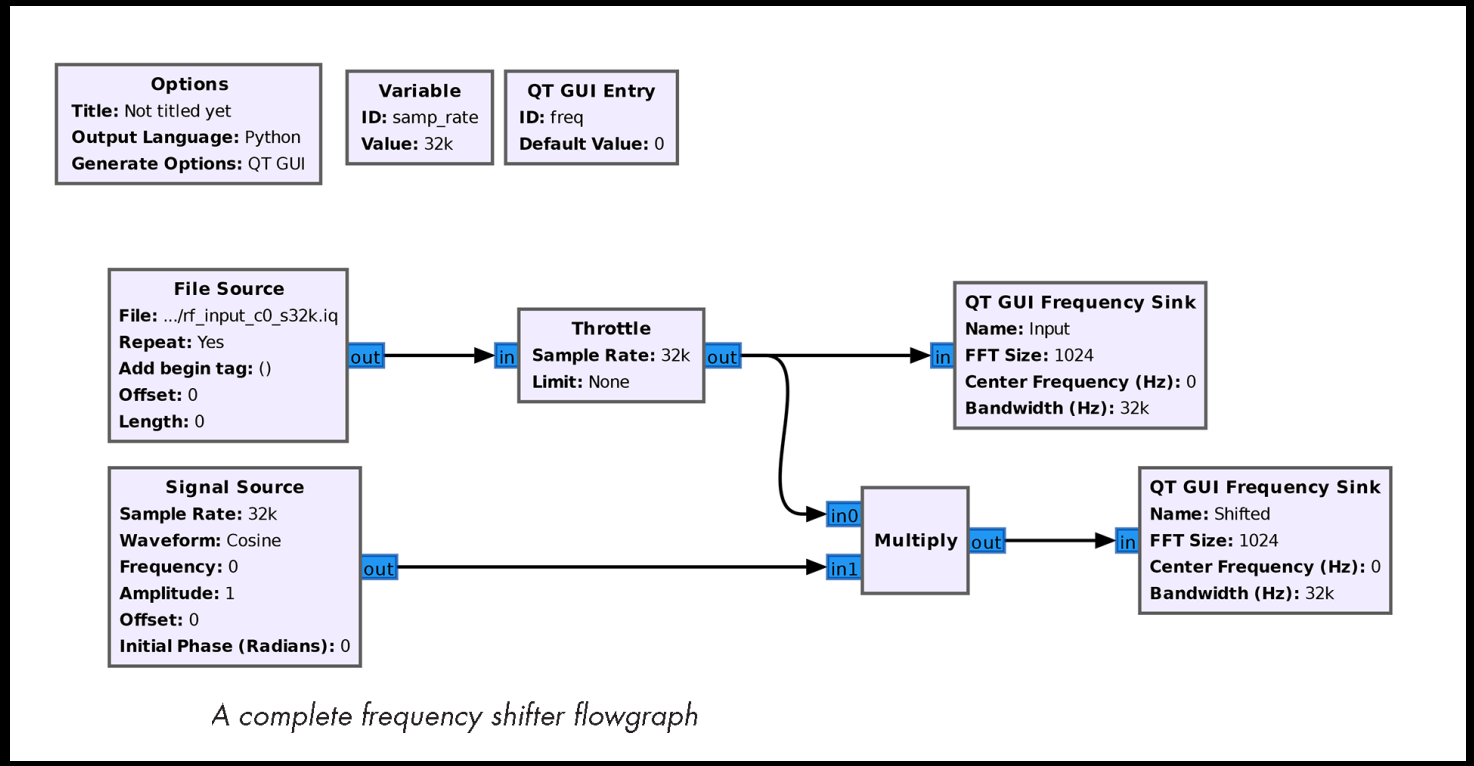
A partial flowgraph for the frequency shifter



Raw input data for the frequency shifter

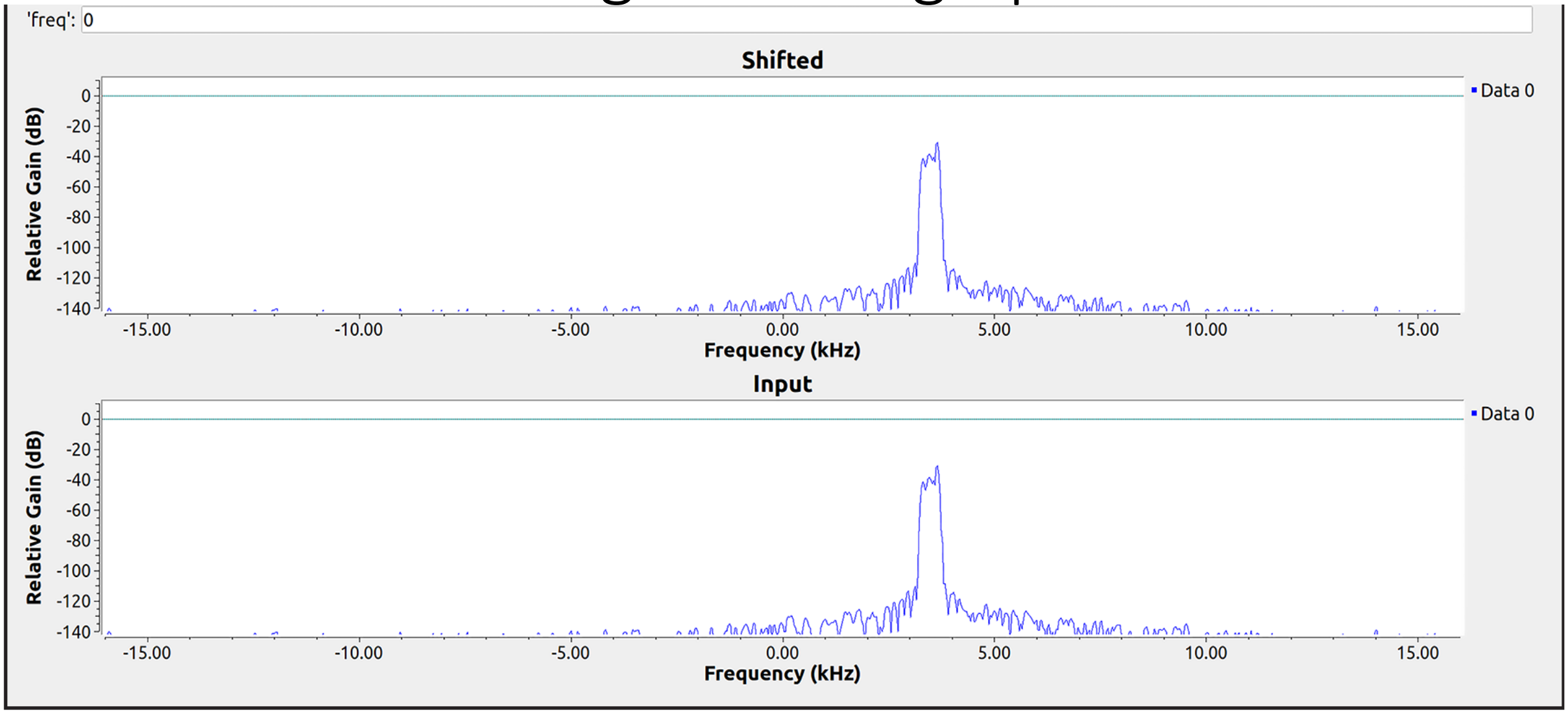
Multiplying by complex-typed sinusoid

- Multiply file data by sinusoid with variable frequency. Add Signal Source with Frequency set to freq. Leave all other properties at default values. Your sinusoid is complex type, rather than floating-point sinusoids.
- Define freq value using a QT GUI Entry, setting ID=freq & leaving Default Value=0.
- Multiply variable-frequency sinusoid by input data, by adding Multiply block, connecting one of the inputs to Throttle output and other to Signal Source output.
- Add 2nd QT GUI Frequency Sink.
- Change name property of 1st QT GUI Frequency Sink to Input.

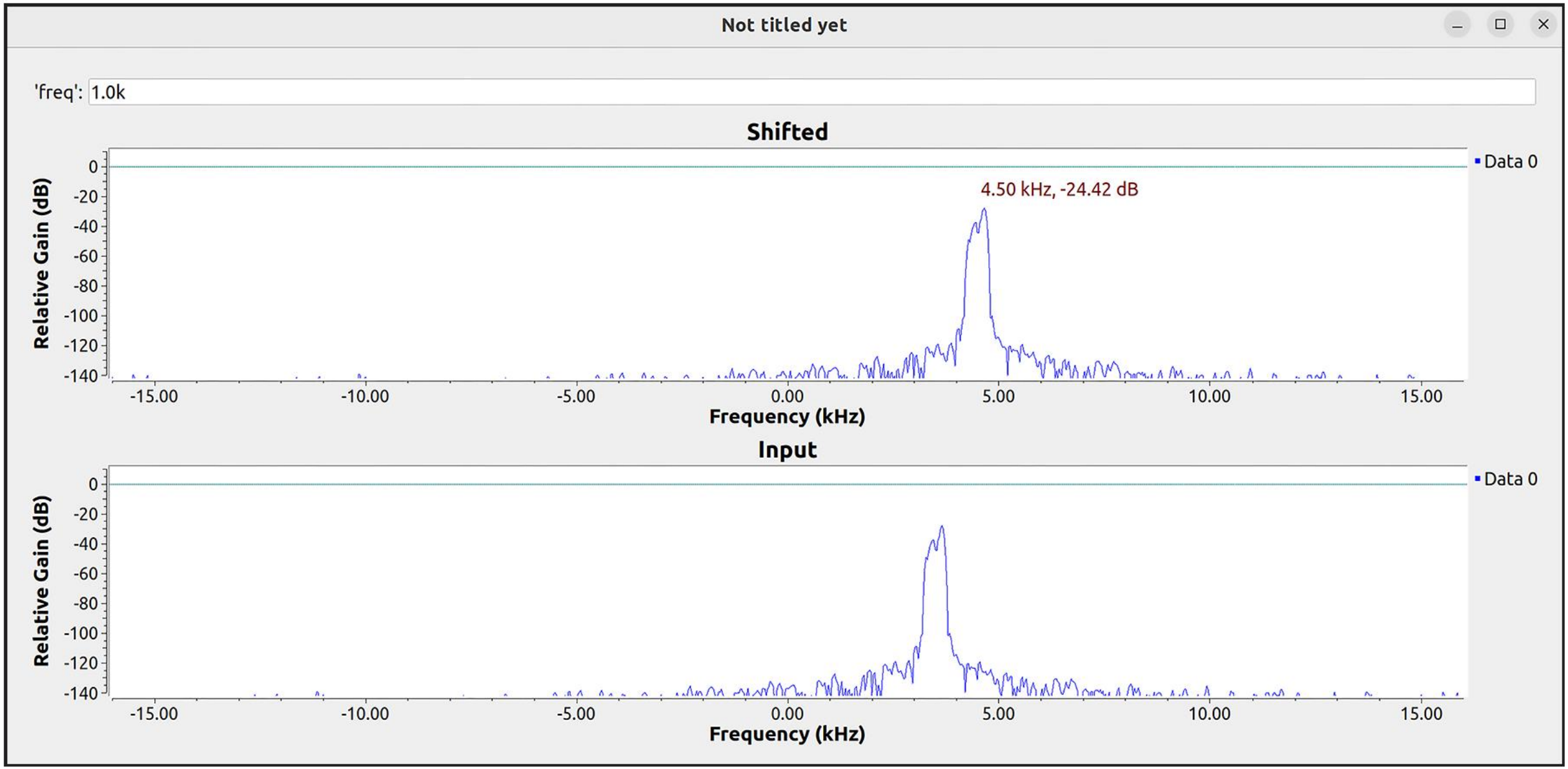


A complete frequency shifter flowgraph

Results of running this flowgraph

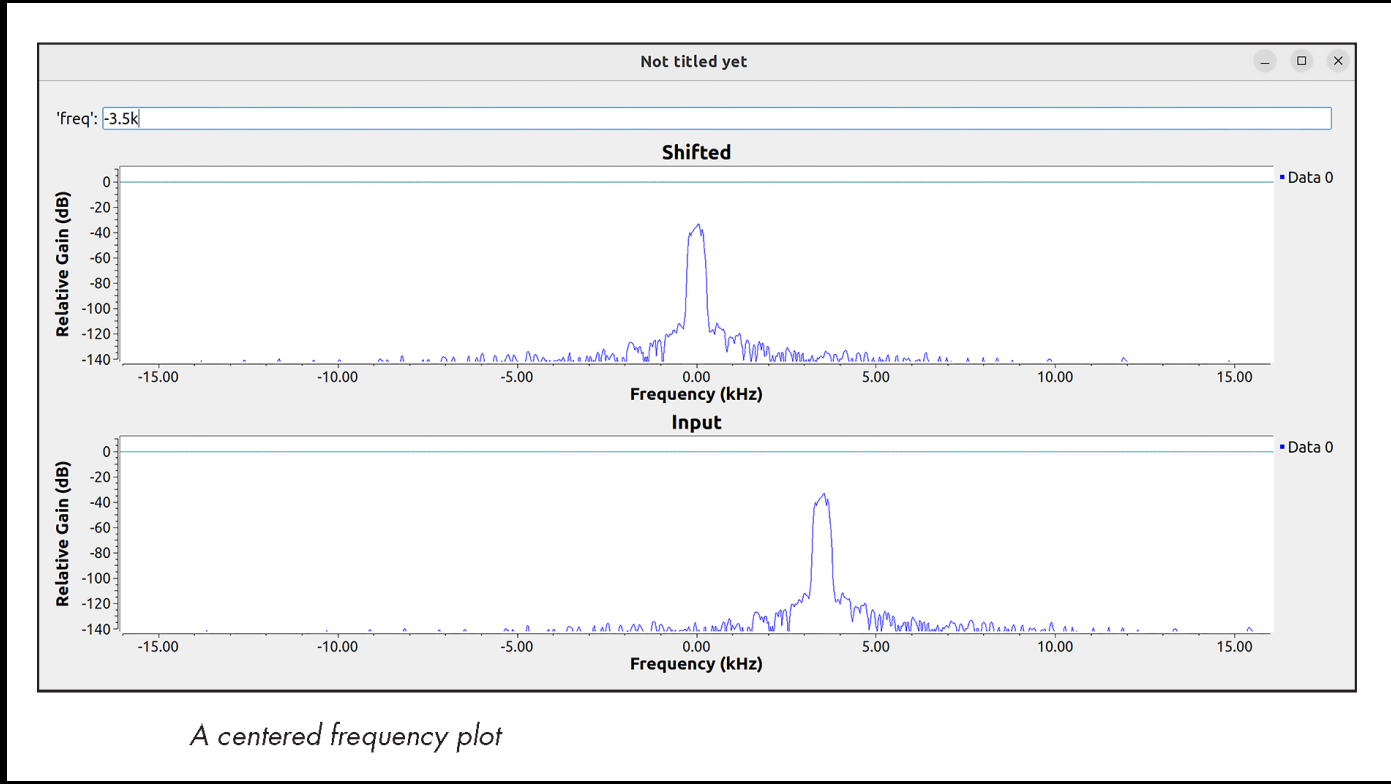


Not a lot of change, is there? Multiplying by 0 Hz leaves the RF data as is. Now try changing the value of `freq` to `1000`. Figure 6-12 shows the result.



A 1 kHz frequency shift

- You can take any complex RF data, multiply it by a complex sinusoid of frequency *f*, and you'll get the same complex RF data shifted in frequency by *f*. e.g. Multiply by 4 kHz complex sinusoid, and frequency shifts 4 kHz to right. Multiply by 9,341 Hz signal, and it shifts 9,341 Hz to right. Multiply it by –1 kHz signal, and it shifts 1 kHz to left.
- Shift peak frequency of 3,500 Hz down to 0 Hz: Multiply by negative frequency. Plug -3500 into freq box, so peak of shifted FFT now centred around 0 Hz.



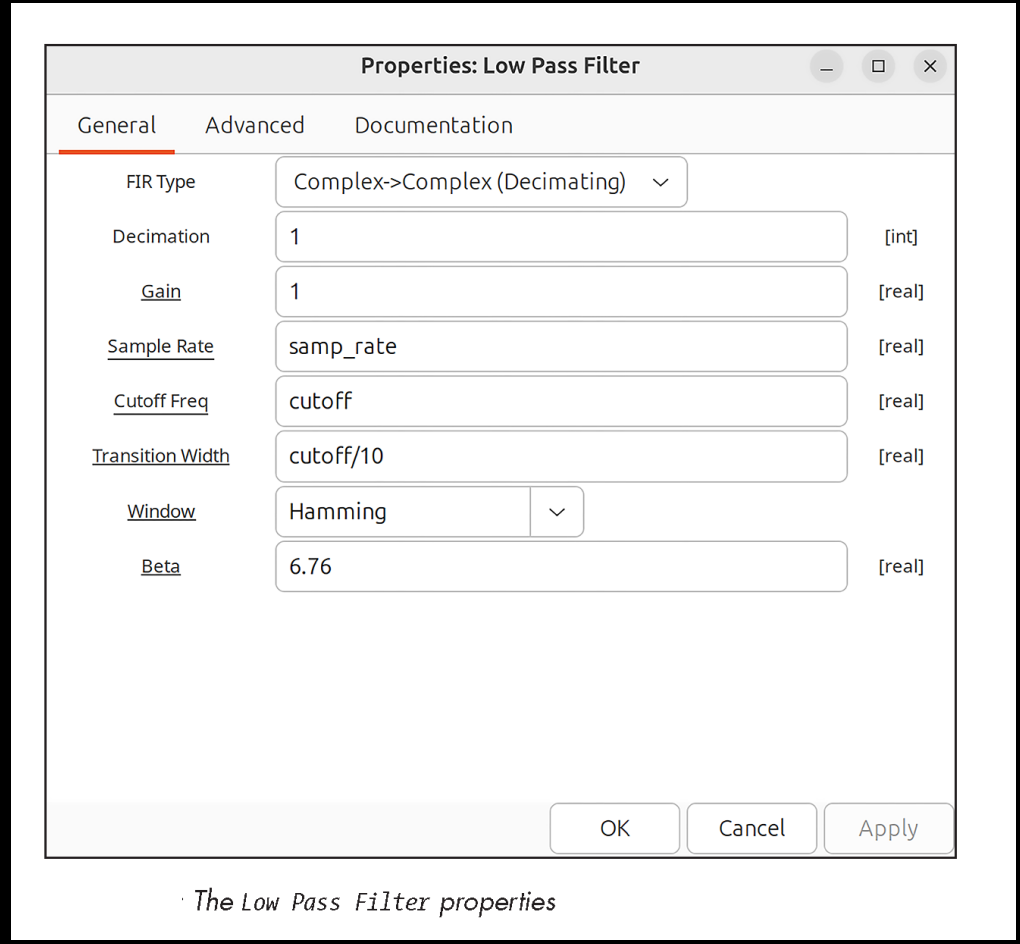
A centered frequency plot

Principals of tuning a frequency

- First step of tuning to a frequency is to multiply the input RF data by a sinusoid whose frequency is –1 times the frequency to which you want to tune.
- E.g. Want to tune to a 5,400 Hz signal? Use a sinusoid with a frequency of –5,400 Hz. Want to tune to a –4 MHz signal? Use a +4 MHz sinusoid.

Filtering

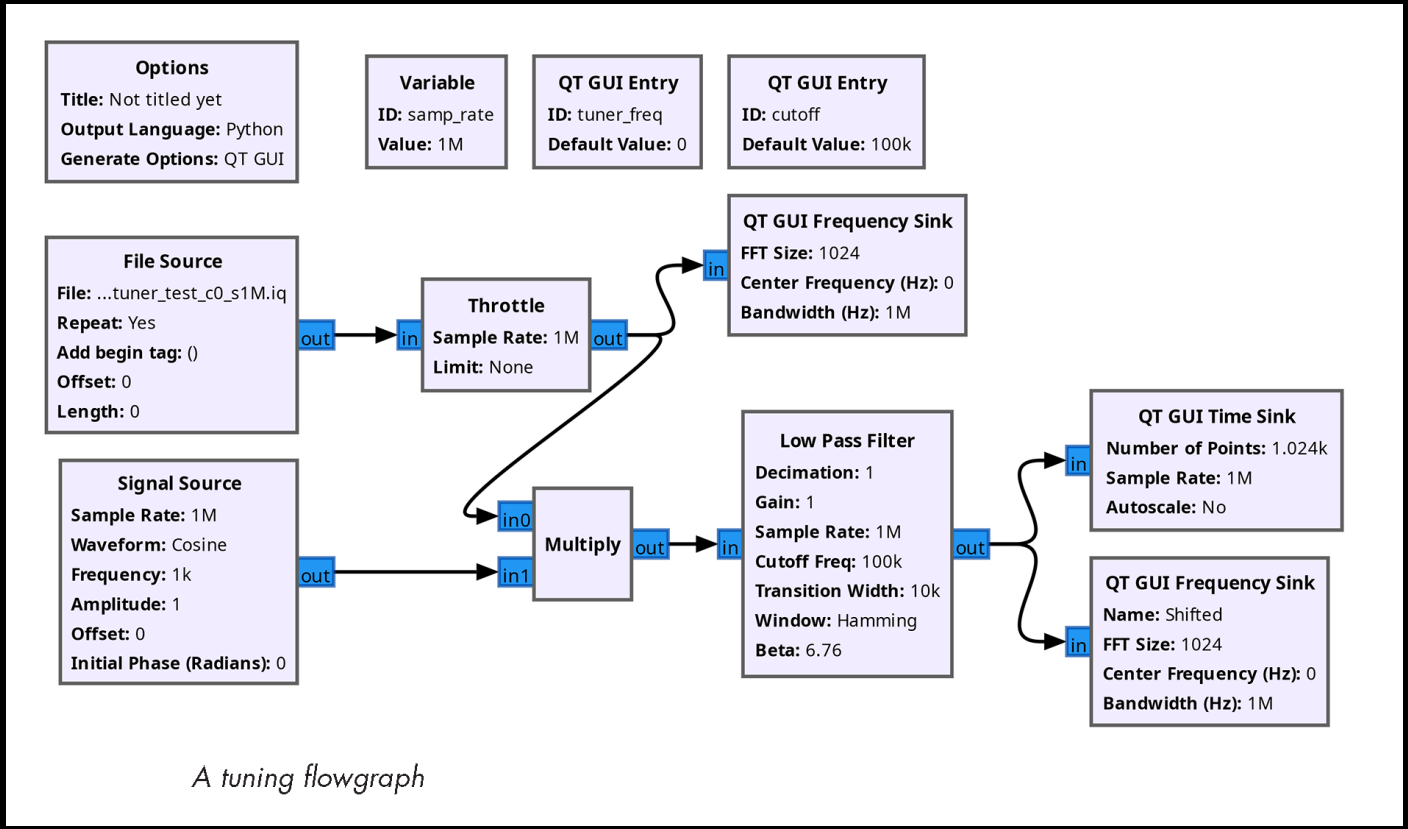
- Now frequencies we want are centred at 0 Hz.
- Now add filter that passes frequencies near zero and eliminates all other ones (filtering out everything except target signal)
- We have negative frequencies, some of which we want to preserve. Filters operating on complex data will affect negative frequencies in same way as positive ones. E.g. low-pass filter will pass frequencies 0 Hz to cutoff frequency & also pass frequencies 0 Hz to –1 times cutoff frequency. It will remove frequencies > cut-off frequency & < –1 times cutoff frequency.
- Rename your frequency-shifting flowgraph as tuner.grc
- Change File property in File Source to ch_06/tuner_test_c0_s1M.iq. This file captured at different sample rate - change Variable with ID samp_rate to value 1e6.
- Add Low Pass Filter with its input connected to output of Multiply block, and QT GUI Time Sink connected to output of Low Pass Filter. There are test patterns in input file data that will be visible in time domain, so new sink will let you see if your tuner is working correctly. The test patterns will be relatively small, however, so to see them clearly, we'll configure the QT GUI Time Sink to automatically set the y-axis of the plot by changing the Autoscale property to Yes.
- Set properties of low-pass filter. Since you don't yet know what kind of value to use for the cutoff frequency, it's a good idea to use a QT GUI Entry so you can adjust the value as the simulation runs. Go ahead and create a QT GUI Entry with ID of cutoff, Type of Float, and Default Value of 100e3. Then double-click Low Pass Filter and set the Cutoff Freq to cutoff.
- You also need to set the filter's transition width. You could create separate QT GUI Entry for this, but when with laboratory SDR work, where the signals pretty clear, you don't have to be careful with transition width. As a rule of thumb, setting transition width to one-tenth of cutoff frequency and change it only if run into problems. The one-tenth will usually give you decent filtering without overworking your CPU. As such, go ahead and set your Transition Width property to cutoff/10.



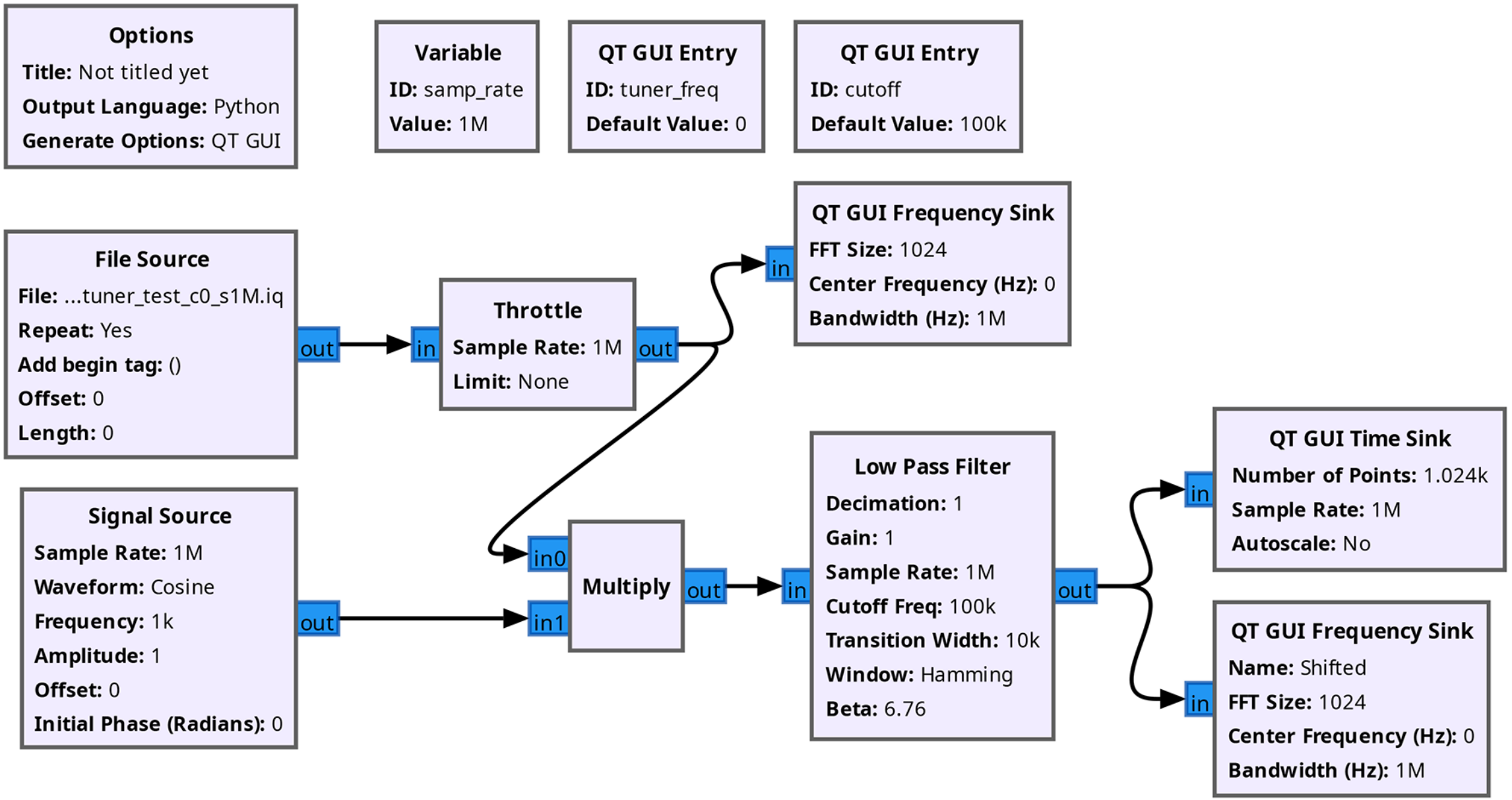
The Low Pass Filter properties

Specifying target frequency in flowgraph

- Rather than specify amount you want to shift RF data, more intuitive to specify tuned frequency – reverse of what done so far.
- Change ID of QT GUI Entry from freq to tune _freq, then change its Type to Float.
- Change Frequency property of Signal Source to -1 * tune_freq. This way, entering in value for tune_freq will cause the data to re-centre on that frequency.

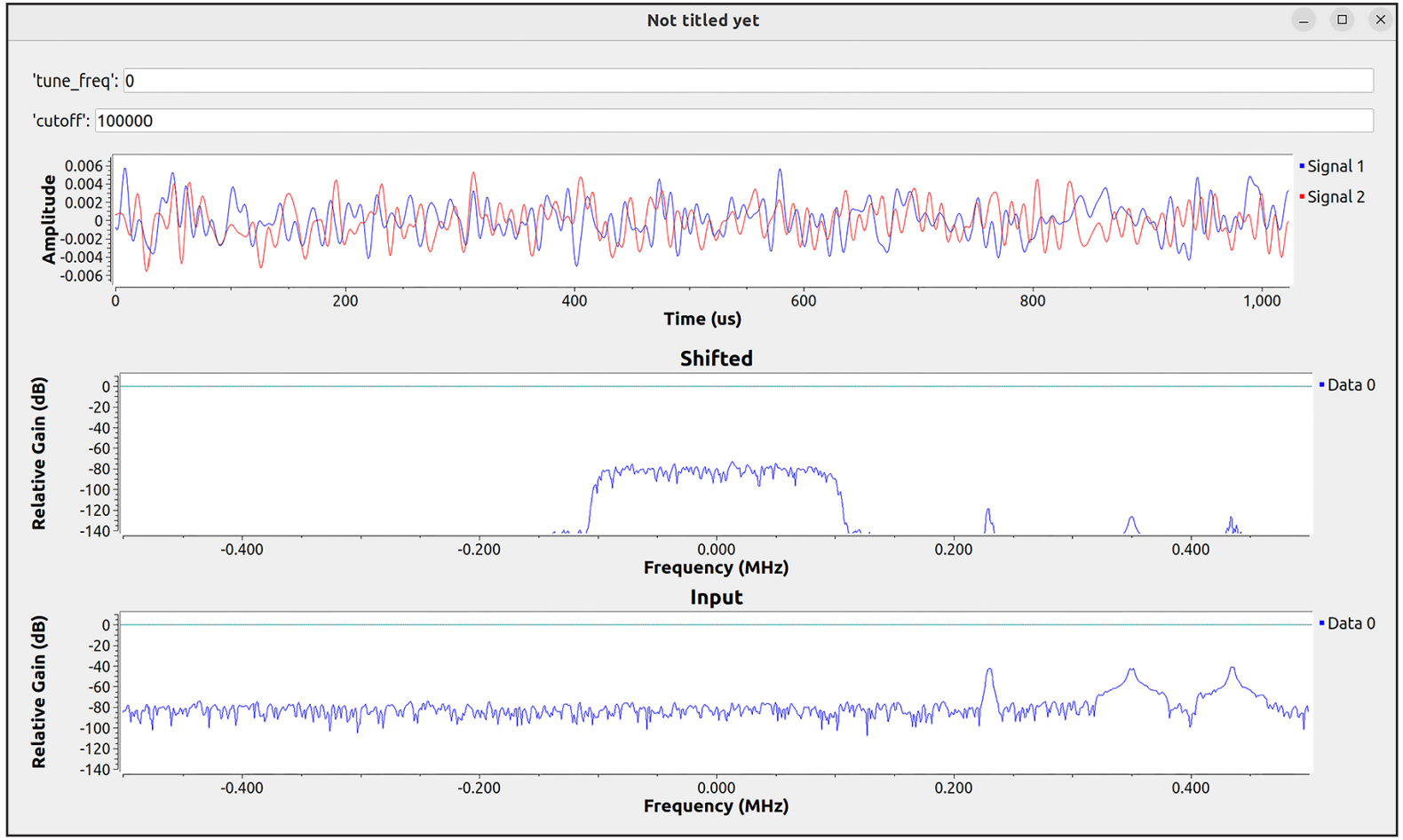


A tuning flowgraph



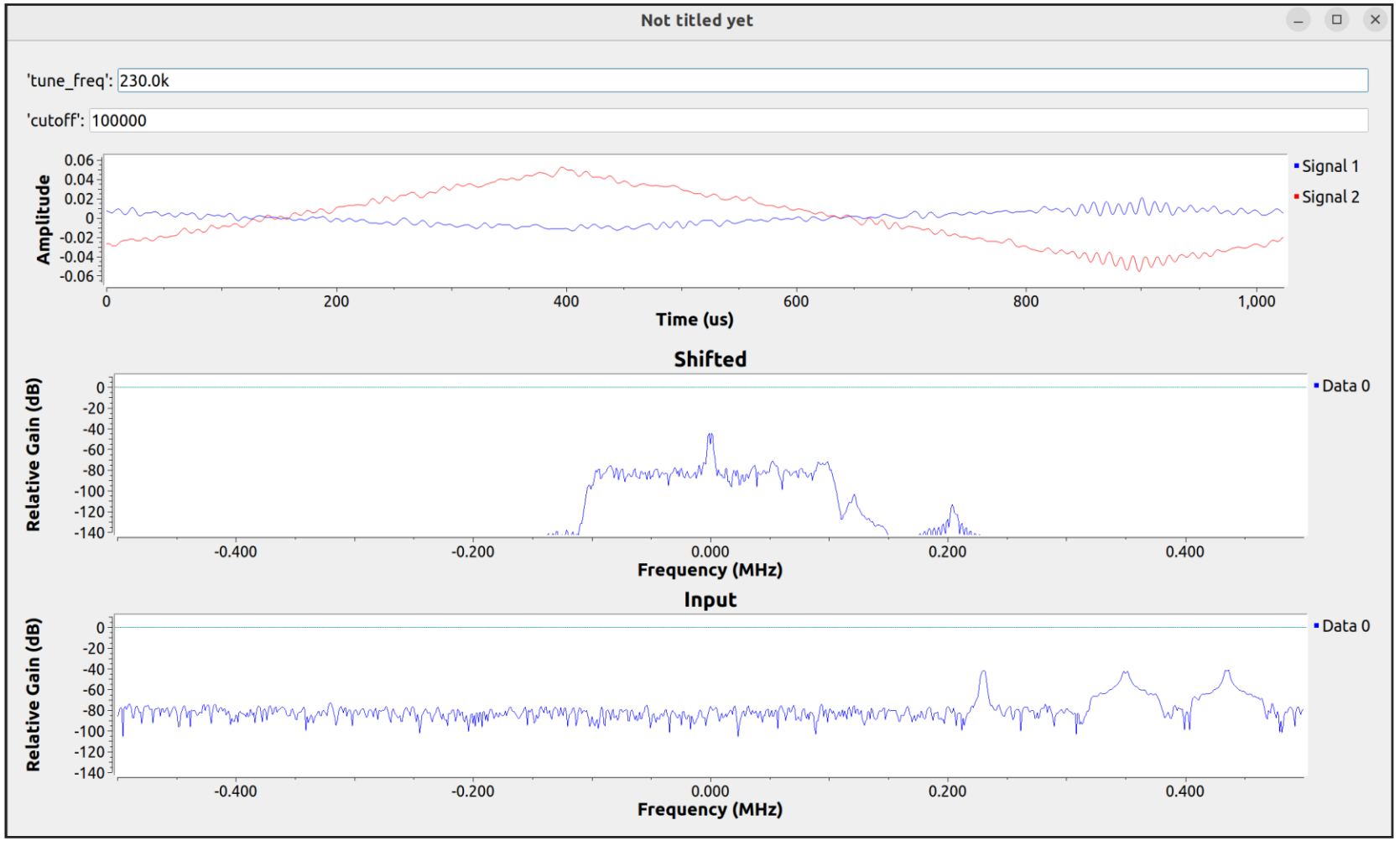
A tuning flowgraph

- Run flowgraph.
- New input file contains 3 peaks, each with different width.
- Time domain plot just shows waveforms moving around randomly, without consistent shape, as not tuned.



The initial output for the tuner flowgraph

- Adjust tune_freq & cutoff values to tune to 3 different signals.
- Hover mouse over each signal peak, you can see that the three peaks in the frequency plot have frequencies of 230 kHz, 350 kHz, and 435 kHz.
- Use your mouse to estimate of width of each peak: first peak about 20 kHz wide, while the other two about 60 kHz wide.
- Tune lowest frequency first. In flowgraph execution window, enter 230k into tune_freq box. After hitting enter, observe that Shifted waveform now has first peak centred around zero.



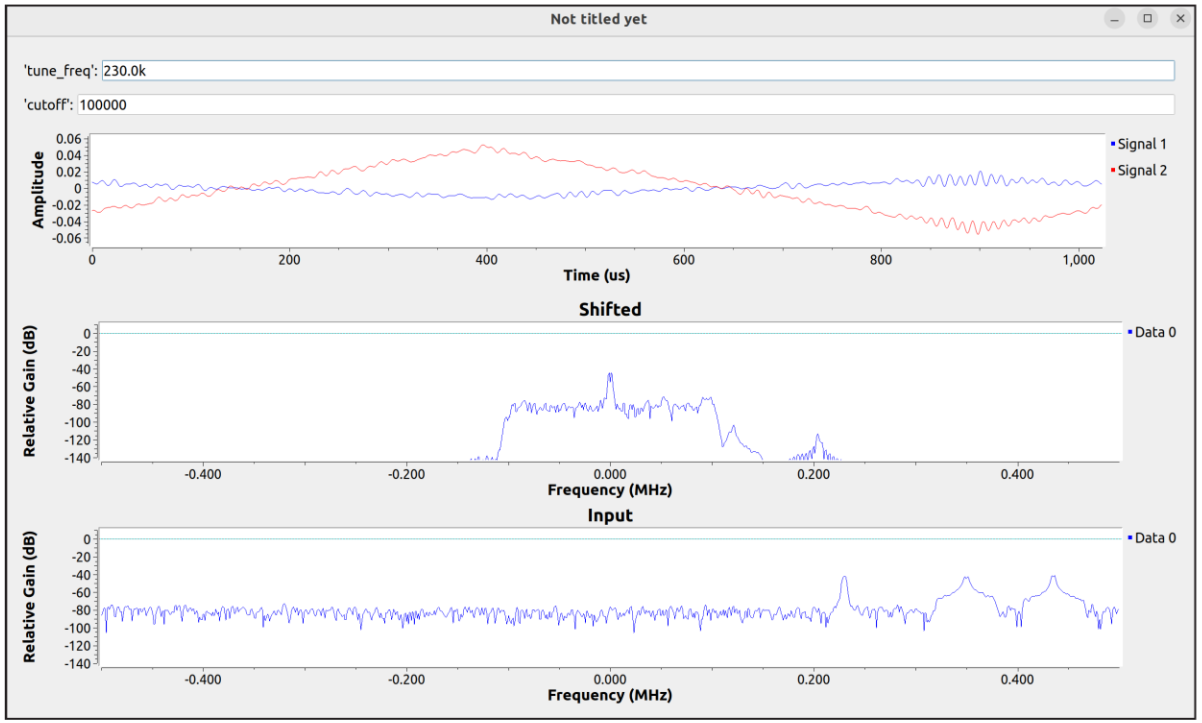
Centering the first peak

If error in console area GNU Radio Companion when press enter, ensure Type of QT GUI Entry for tune_freq = Float.

Notation in blocks & widgets in GNU Companion

- Block properties use exponential notation
- Values entered into QT GUI widgets during execution use metric units like k, M, G, etc. (Hence entered 230k into the tune_freq box rather than 230e3).

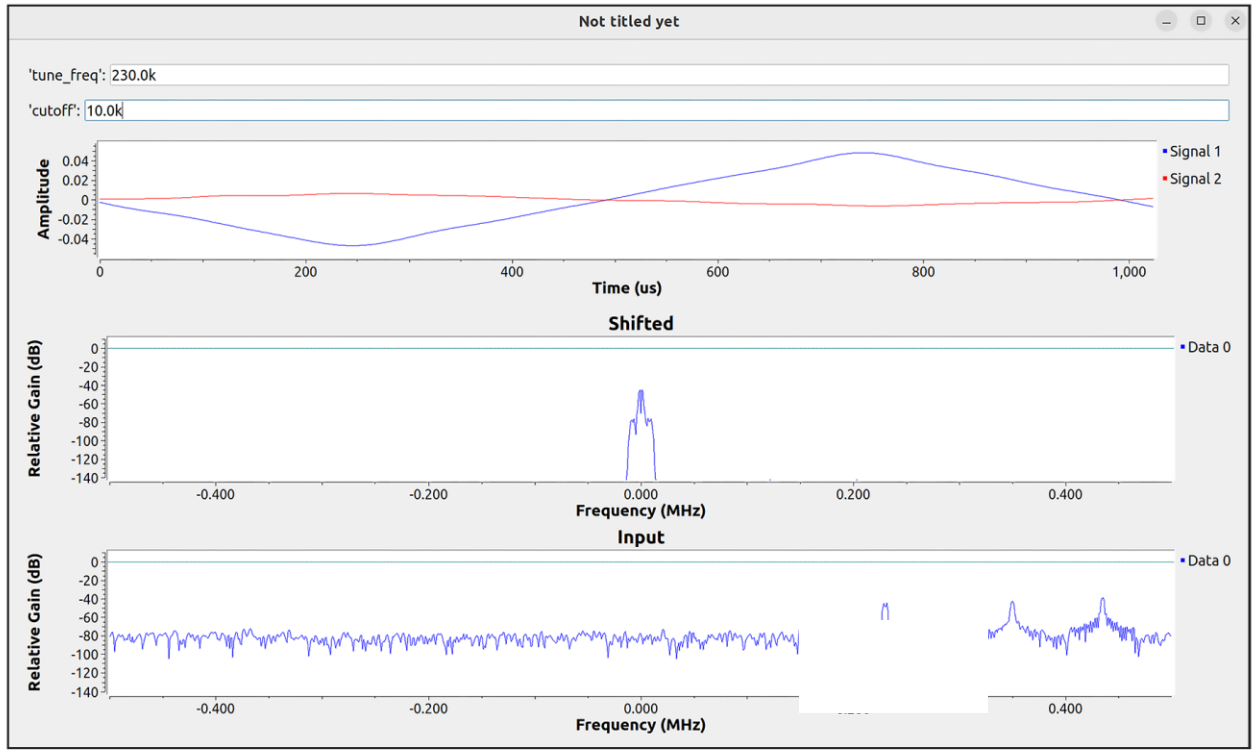
- Triangular-shaped signal appear in time domain plot, doesn’t look clean, so adjust cutoff frequency.
- First peak ca. 20 kHz wide, so might think 20 kHz decent cutoff value.
- BUT Signal peak centred zero, & complex low-pass filter passes frequencies from –cutoff to +cutoff frequencies. Better use 10 kHz as cutoff, in which case filter will pass everything from –10 kHz to +10 kHz, for total 20 kHz. With clear signals, not catastrophic to have wider filter than necessary, as you’d end up with same result if you used 20 kHz for cutoff: Important to understand where the numbers coming from.
- Once you set the cutoff to 10k, triangular wave should clean up significantly.



Centering the first peak

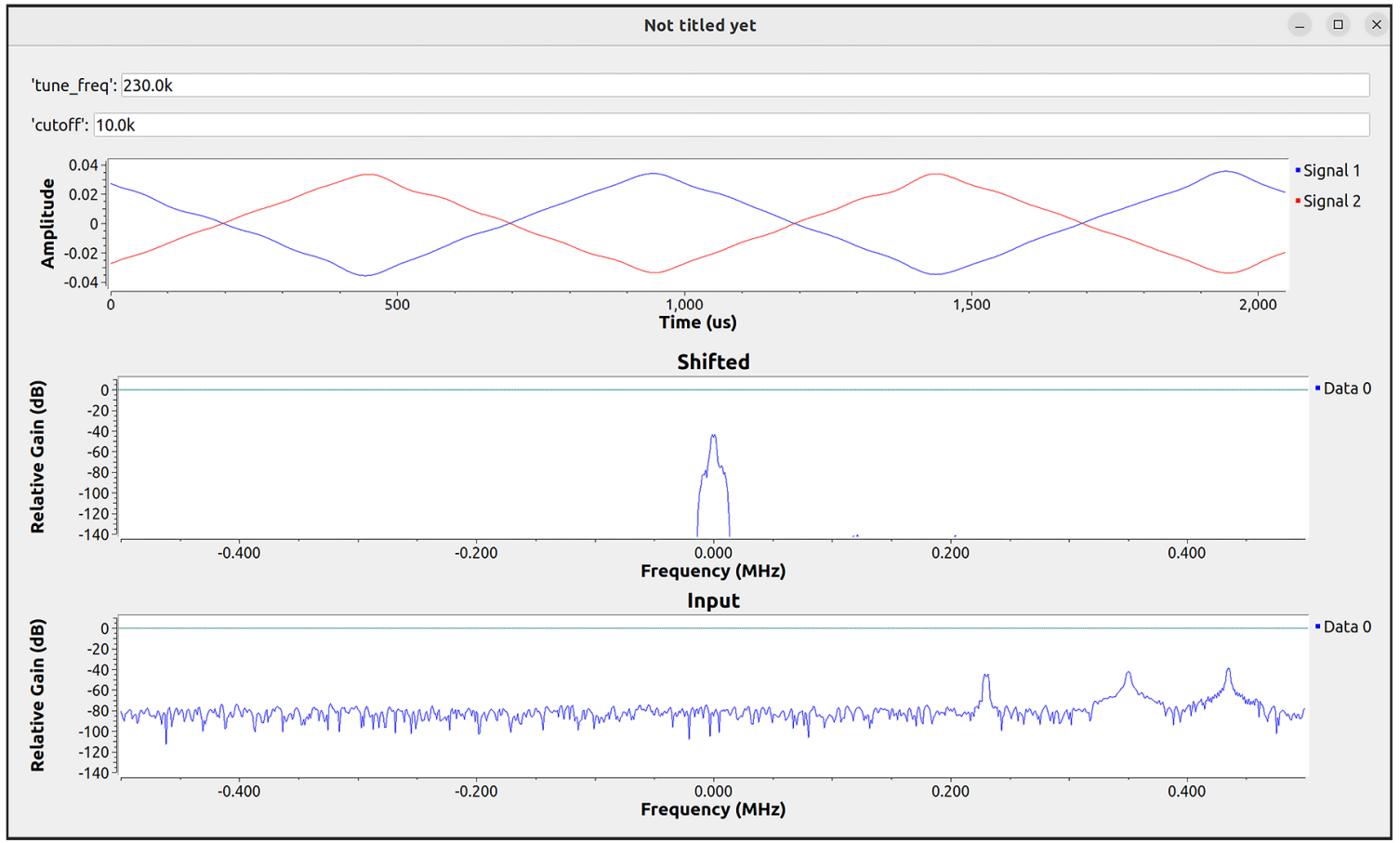
Before (above)

After (below)



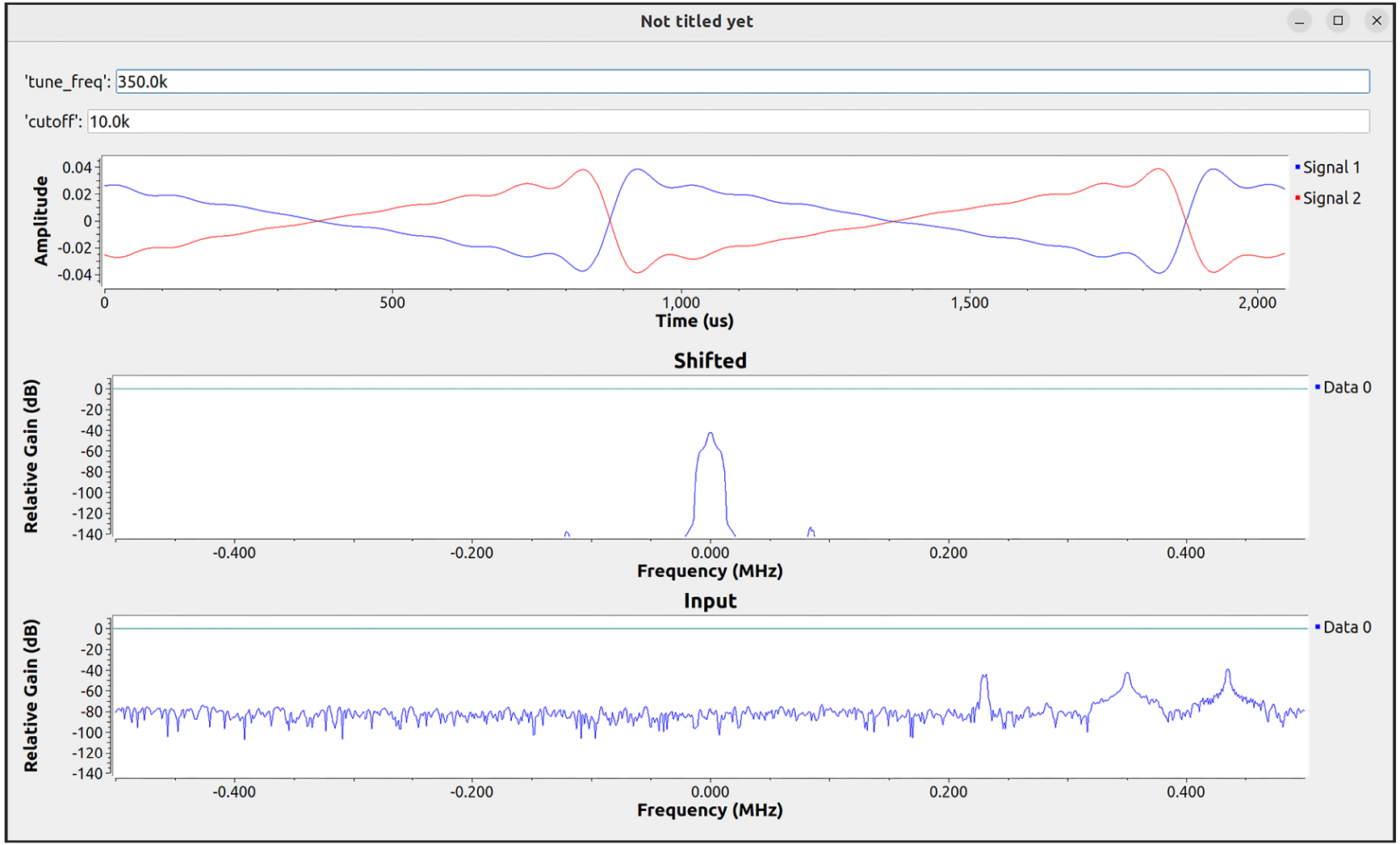
Tuning to the first peak

- Time domain signal = complex triangle wave = simple test pattern to show that tuner functioning correctly.
- To zoom out on the horizontal axis, click your middle mouse button anywhere in time domain plot to bring up context menu. Click Number of Points, and change number 1024 in dialog box to larger number & display more samples on plot.
- Double number by entering 2048, and look at result.

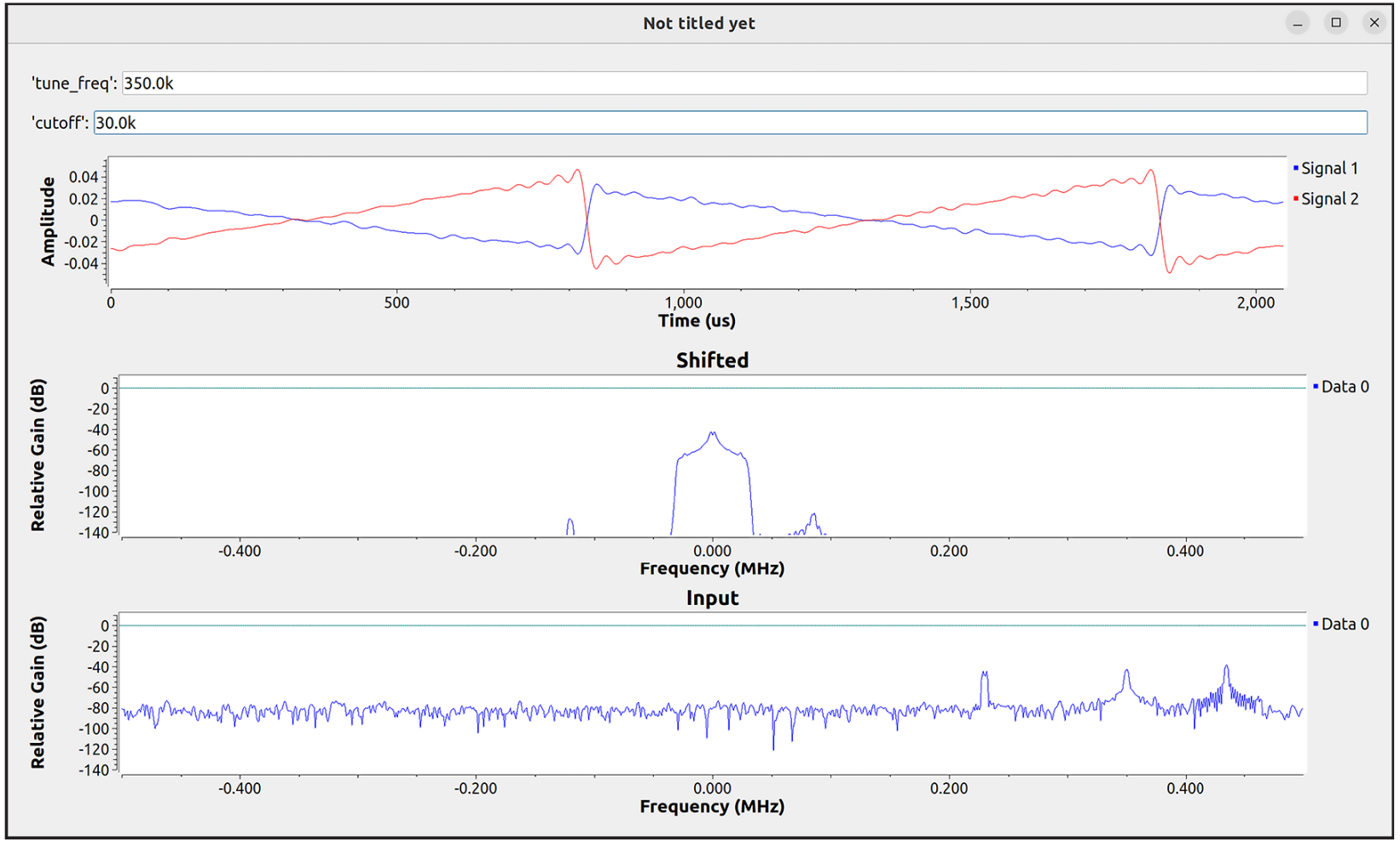


Tuning to the first signal

Next, tune to the second signal by changing `tune_freq` to `350k`. The time domain waveform will change to something with a sharper upward ramp and a slower downward ramp than the triangle wave (Figure 6-20).



Tuning to the second signal, with poor filtering

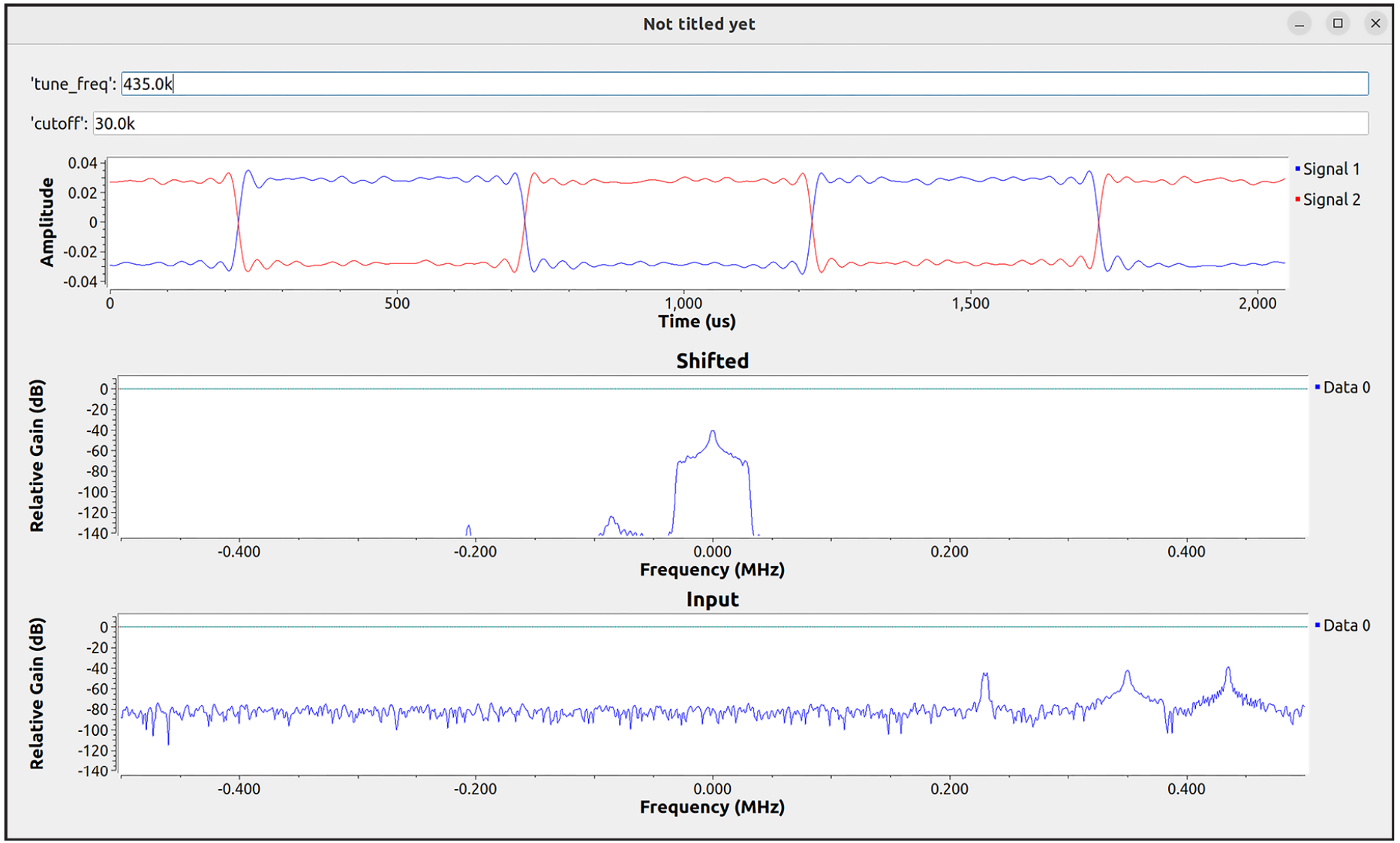


Tuning to the second signal, with better filtering

2nd & 3rd peaks wider than 1st ➔ update cutoff value to 30k (half peaks' 60k width) & press enter ➔ waveform gets sharper.

This waveform is known in electronics as a *sawtooth wave*, with a sharp rise and a slow ramp-down.

change the tune_freq to 435k to see the last signal's test pattern



Tuning to the third signal

waveform is a complex square wave.