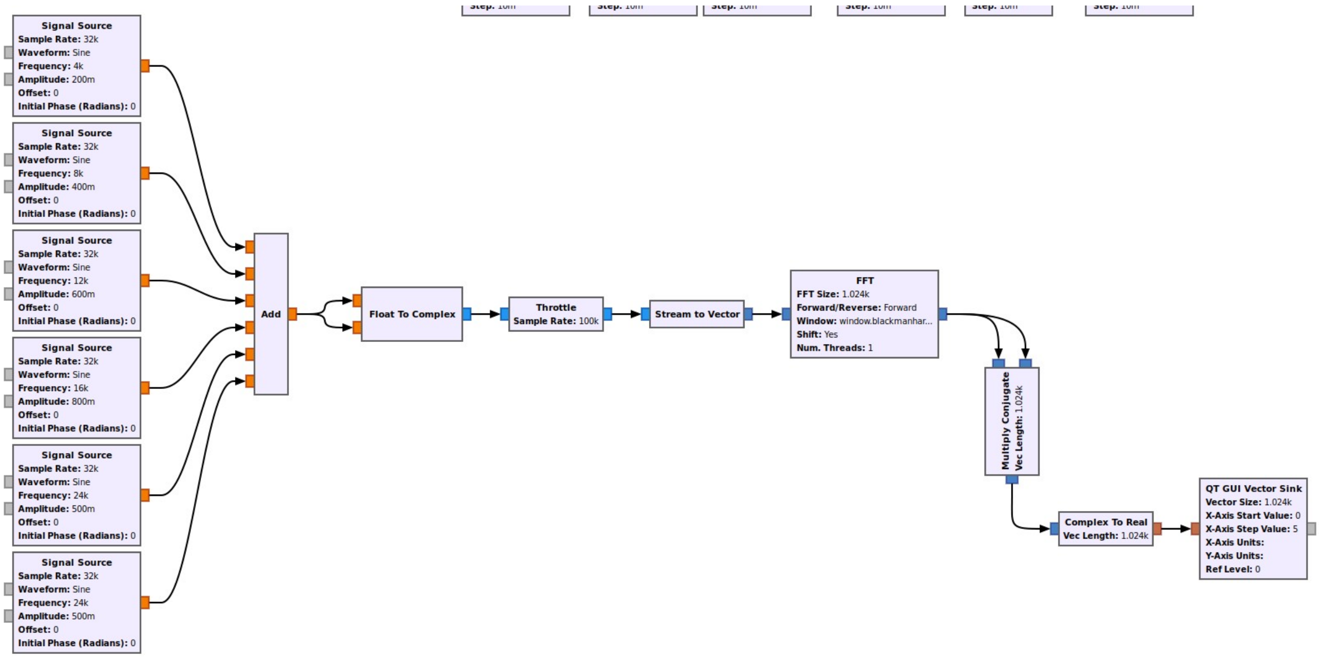
DSPIRA GnuRadio Lessons: Introduction and Simple Spectrometer
Lesson 4: The FFT Block

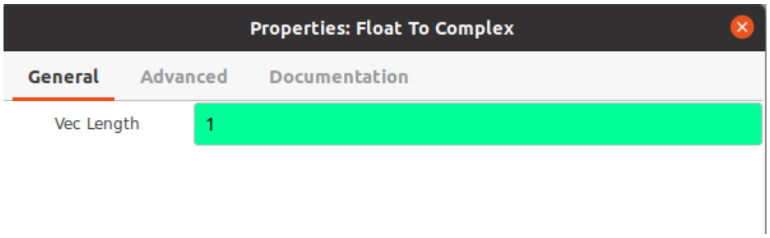
In this activity you will demonstrate how the FFT block in GNURadio works. The FFT (Fast Fourier Transform) determines the frequencies and amplitudes of the sine and cosine functions that make up the Fourier series of a signal input to the FFT block. This block utilizes some mathematics that are beyond the scope of this presentation; it uses vectors and complex numbers to complete its computations. Several of the blocks will have inputs and outputs that are complex.

Using An FFT Block

- Open your 6 function signal generator from Lesson 2.
- **Save as . . .** using a new file name.
- Delete all of the blocks in the sequence after the **Add** block.
- Import the blocks blocks shown in the diagram below:

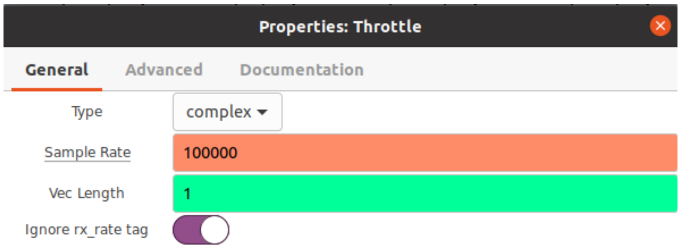


- Open each block and change their settings as illustrated below:
    - **Float to Complex** - This converts the signal data to a complex number.


Properties: Float To Complex

Vec Length: 1

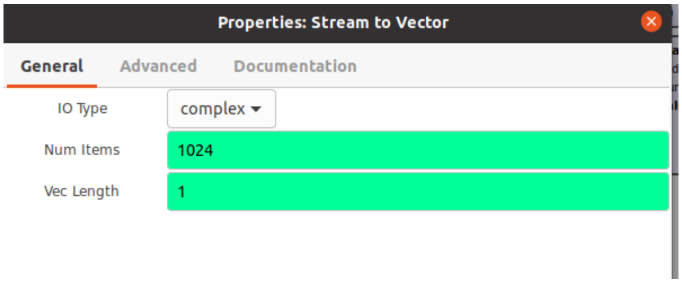
- **Throttle** - This regulates the flow rate of the signal through the sequence; it has no effect on the output. In the previous lessons the speaker regulated the flow of the data through the blocks. As no speaker is used in this program, the **Throttle** block keeps the flow regulated.



Properties: Throttle

Type	complex
Sample Rate	100000
Vec Length	1
Ignore rx_rate tag	

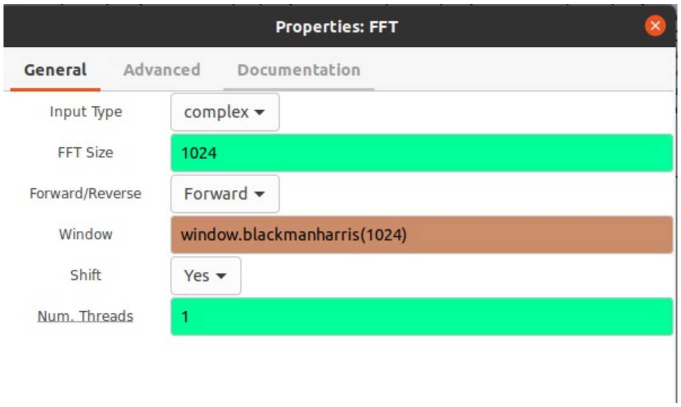
- **Stream to Vector** - This changes the signal source to a vector to be analyzed by the FFT.



Properties: Stream to Vector

IO Type	complex
Num Items	1024
Vec Length	1

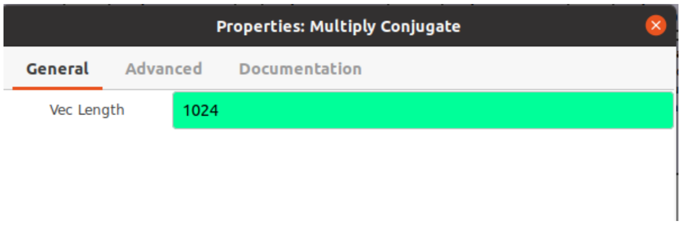
- **FFT** - This is where the magic is done.



Properties: FFT

Input Type	complex
FFT Size	1024
Forward/Reverse	Forward
Window	window.blackmanharris(1024)
Shift	Yes
Num. Threads	1

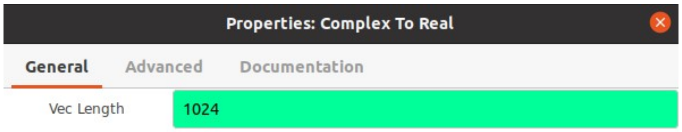
- **Multiply Conjugate** - The output of the FFT block is a complex number. This block calculates a number that represents the magnitude of output at each frequency.



Properties: Multiply Conjugate

Vec Length	1024

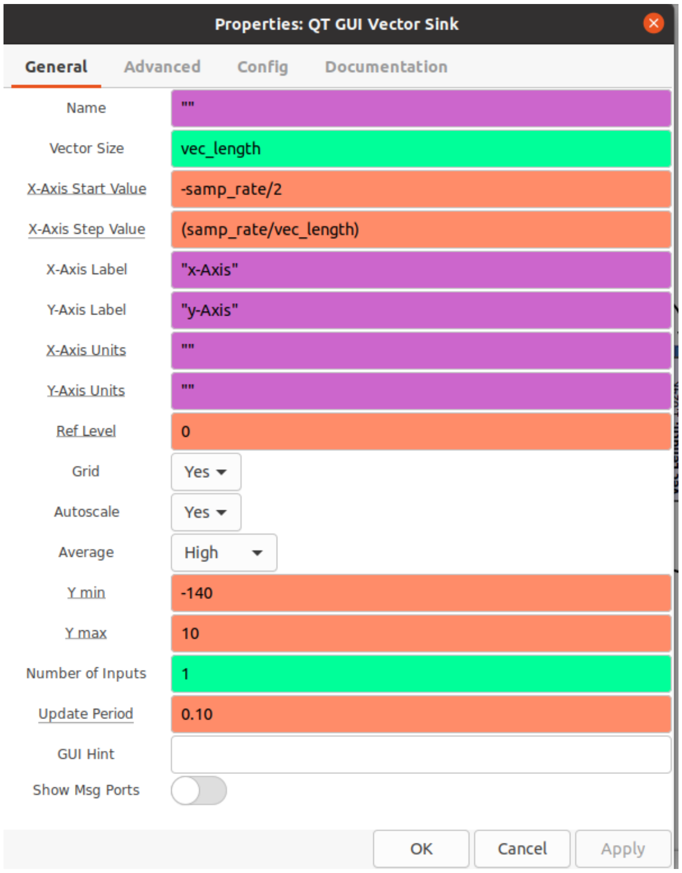
- **Complex to Real** - This converts the data stream back to a real number.



Properties: Complex To Real

Vec Length	1024

- **QT GUI Vector Sink** - This displays the output of the FFT graphically, showing the amplitudes of each frequency that makes up the signal that was the input to the FFT.



Properties: QT GUI Vector Sink

Name	""
Vector Size	vec_length
X-Axis Start Value	-samp_rate/2
X-Axis Step Value	(samp_rate/vec_length)
X-Axis Label	"x-Axis"
Y-Axis Label	"y-Axis"
X-Axis Units	""
Y-Axis Units	""
Ref Level	0
Grid	Yes
Autoscale	Yes
Average	High
Y min	-140
Y max	10
Number of Inputs	1
Update Period	0.10
GUI Hint	
Show Msg Ports	○

- The "X-Axis Start Value" and "X-Axis Step Value" are needed for the Vector Sink to interpret the frequencies correctly.

Test the FFT Block

- Run the program. Choose different frequencies and amplitudes of sources 1-6. These can be adjusted using the sliders or counters for each frequency and amplitude.
- Notice how the output in the graph changes as the frequency and amplitude values are changed.
- Also, notice that the zero frequency on the x-axis is in the middle of the axis, and both positive and negative frequencies are possible. This is an interesting possibility with SDR that we will not go into.

- If all 6 frequencies set in the frequency **QT GUI Range** blocks are at different values, then there should be 12 different frequency values displayed - 6 corresponding to the positive values of these frequencies and 6 that are mirror images of these frequencies in the negative range.
- By placing the cursor over a peak, its frequency can be read as the x-coordinate displayed.
- The peak heights are related to the Fourier coefficients of each frequency. The y-axis is not scaled on this graph.
- Set all of the amplitudes to 1. The peak heights should all be the same, or very similar. (These may not be exactly the same.)
- Now change one of the signal amplitudes to 0.5. Notice that the corresponding peak height at this frequency drops to approximately ¼ of its original height. This is because the **Multiply Conjugate** block actually calculates the power of the signal, which is proportional to the square of the Fourier coefficient. So dropping the amplitude of one of the input frequencies to ½ results in a power at that frequency that is $(½)^2$ = ¼ of its original power.

This activity should give you an idea of what an FFT block can accomplish. Deciphering the frequencies contained in a signal is a very important tool in DSP in general, as it is in radio astronomy. So the FFT block is at the heart of the GNURADIO spectrometers we use.