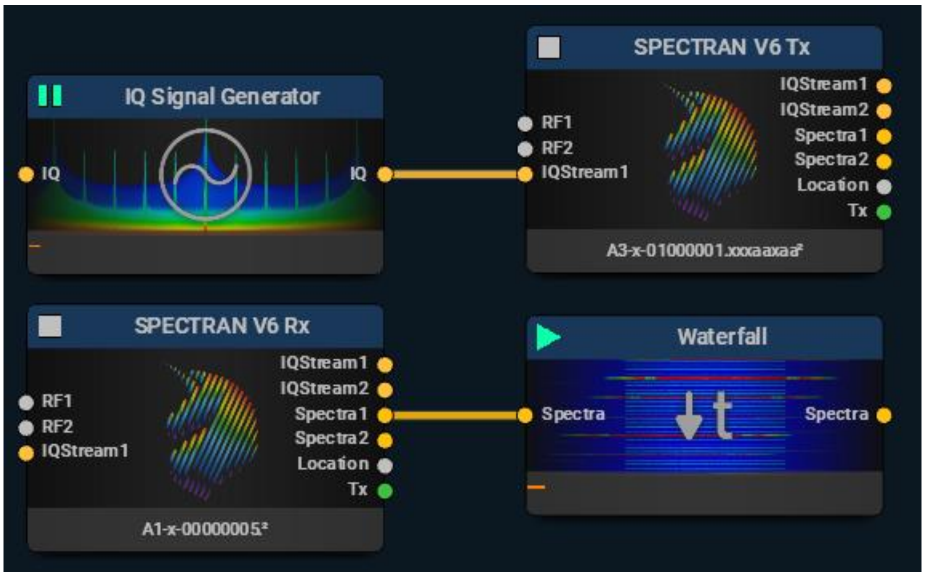
Tx Streaming using Signal Generator

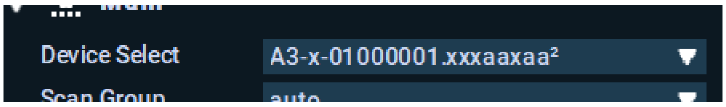
Full Span Stream

This uses “DirectSignalGenerator.rmix”



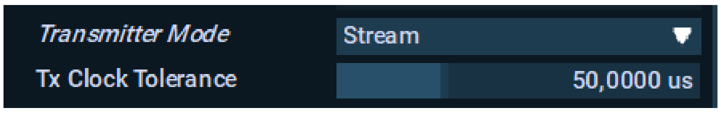
An IQ Signal Generator block is used to directly feed the IQStream input of a SPECTRAN V6. A second SPECTRAN V6 is used to receive the generated stream and a waterfall to display the result.

The correct devices have to be selected in the V6 Tx and V6 Rx blocks for the samples to work.



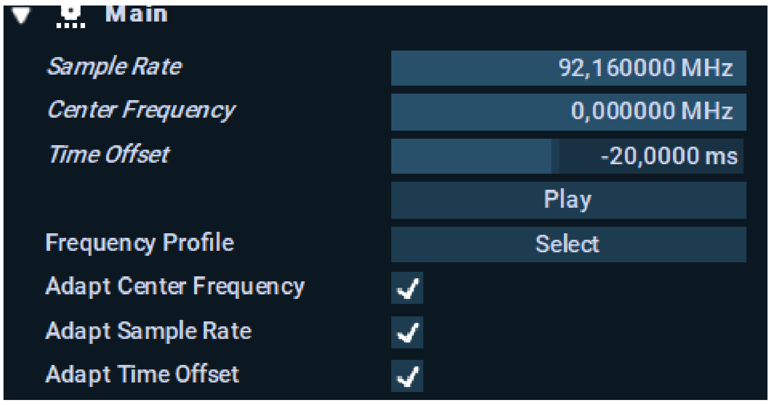
Device Select	A3-x-01000001.xxxaaxaa²
Scan Group	auto

The transmitting V6 is set to send a “Stream” with the “Transmitter Mode” configuration:



Transmitter Mode	Stream
Tx Clock Tolerance	50,0000 us

The Tx Clock Tolerance determines, when to skip data or insert blank to synchronize the output with the clock in the input stream. An unstable clock source, that is not synchronized to the V6 will result in samples, sample rate and time stamps not matching up, thus needing adaptation.



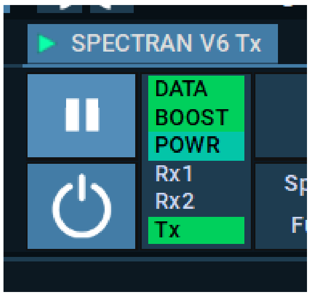
Main

Sample Rate	92,160000 MHz
Center Frequency	0,000000 MHz
Time Offset	-20,0000 ms
	Play
Frequency Profile	Select
Adapt Center Frequency	✓
Adapt Sample Rate	✓
Adapt Time Offset	✓

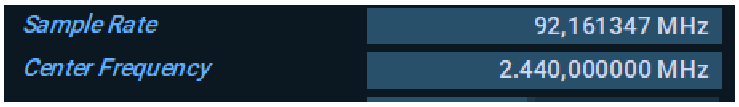
The IQ Signal Generator, when connected to the V6, can adjust to a system clock that does not agree with the V6 sample clock by continuously adjusting the sample rate. This is enabled with the “Adapt Sample Rate” checkbox. The IQ data provided to the V6 must have the same center frequency, this will also be adjusted automatically in the signal generator, when the “Adapt Center Frequency” is checked.

The final config to consider is the “Time Offset”. This will ensure that the generated data is slightly in the future, thus allowing some buffering, to avoid dropouts caused by a non real-time OS such as Windows or Linux. If the default of 20ms is not sufficient, the amount can be increased by checking off the “Adapt Time Offset” checkbox.

When the sample is running, there are some elements, that can be monitored:

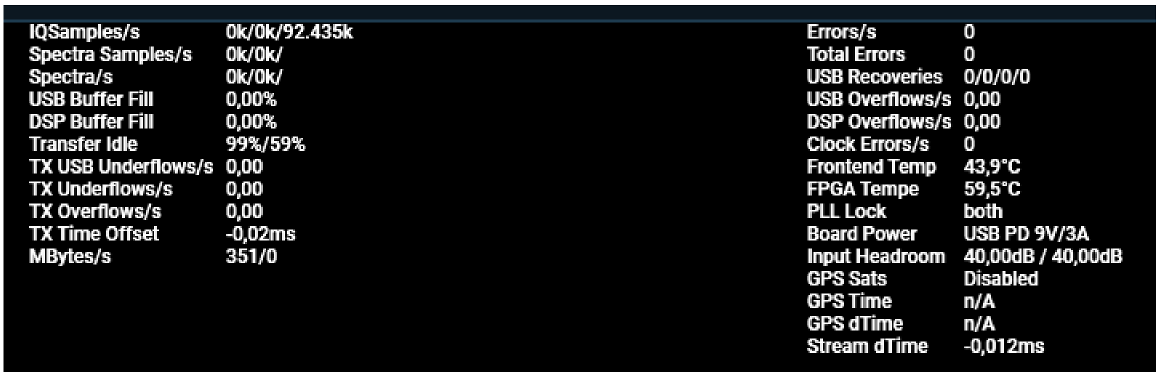


The Tx field should stay green.



Sample Rate	92,161347 MHz
Center Frequency	2.440,000000 MHz

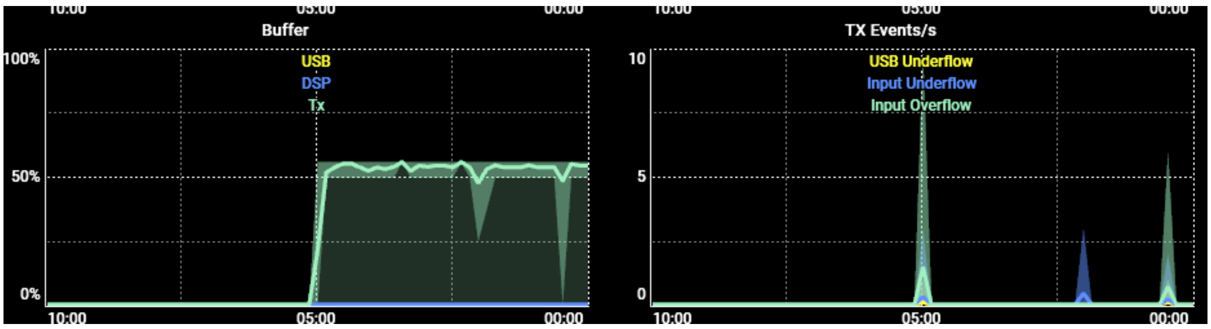
The center frequency of the signal generator should match the center frequency of the V6 and the sample rate will slightly vary due to clock source discrepancies of the system block and the V6 sample clock.



IQSamples/s	0k/0k/92.435k
Spectra Samples/s	0k/0k/
Spectra/s	0k/0k/
USB Buffer Fill	0,00%
DSP Buffer Fill	0,00%
Transfer Idle	99%/59%
TX USB Underflows/s	0,00
TX Underflows/s	0,00
TX Overflows/s	0,00
TX Time Offset	-0,02ms
MBytes/s	351/0

Errors/s	0
Total Errors	0
USB Recoveries	0/0/0/0
USB Overflows/s	0,00
DSP Overflows/s	0,00
Clock Errors/s	0
Frontend Temp	43,9°C
FPGA Tempe	59,5°C
PLL Lock	both
Board Power	USB PD 9V/3A
Input Headroom	40,00dB / 40,00dB
GPS Sats	Disabled
GPS Time	n/A
GPS dTime	n/A
Stream dTime	-0,012ms

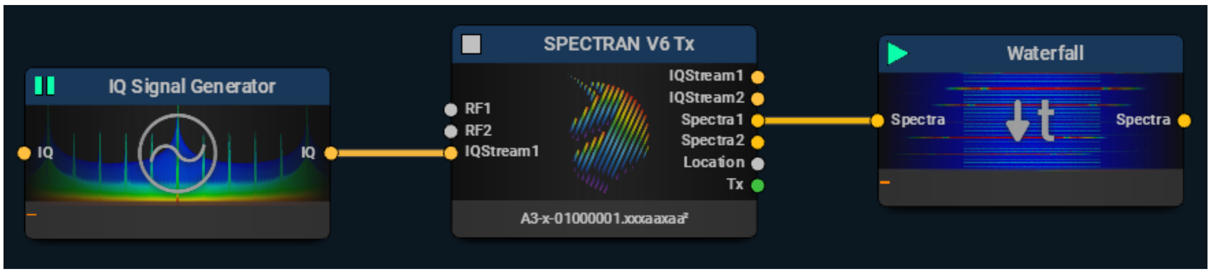
There should be no Tx over or underflows. The Tx time offset will stay in the range of +/- the value allowed by the tx clock tolerance setting.



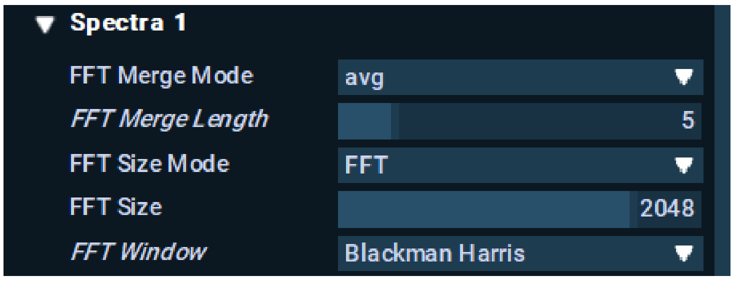
The Tx buffer should be at approx. 50% and the number of Tx events should be rare to none. Frequent input under or overflows might be reduced by increasing the buffer time in the IQ signal generator (if caused by OS problems) or with the clock tolerance setting (if caused by an unstable system clock).

Single SPECTRAN V6

This uses “DirectSignalGeneratorOneV6.rmix”



An alternative to two separate SPECTRAN V6 is to use a single device for transmission and receiving. This requires two USB data connections (DATA and BOOST) and may result in lower fidelity due to some crosstalk.



Spectra 1

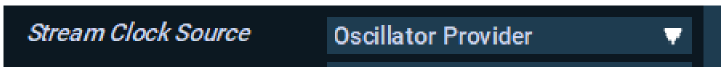
FFT Merge Mode	avg
FFT Merge Length	5
FFT Size Mode	FFT
FFT Size	2048
FFT Window	Blackman Harris

It is recommended to reduce the data rate by using FFT merge on the incoming stream.

Using the SPECTRAN V6 sample clock

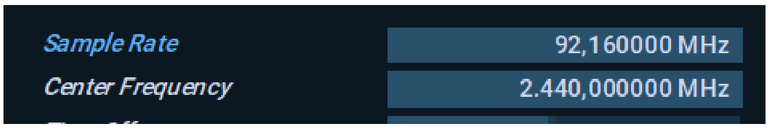
This uses “DirectSignalGeneratorSClock.rmix”

An alternative to using the system clock as the master of the RTSA Suite PRO is to use the sample clock of one of the SPECTRAN V6 as the major clock source. This results in a more stable sample rate for the signal generator, but effectively decouples the streaming from wall clock time.



Stream Clock Source	Oscillator Provider

There is still some clock fluctuation in the beginning due to thermal effects, but the clock will stabilize after some seconds.

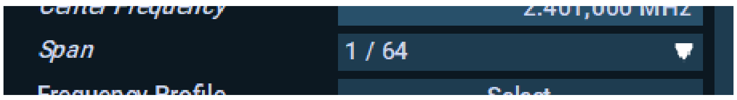


Sample Rate	92,160000 MHz
Center Frequency	2.440,000000 MHz

Signal Generator with up-sampled IQ Data

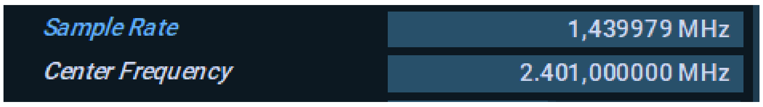
This uses “UpsampledSignalGenerator.rmix”

The incoming IQ stream into the V6 can be upconverted, similar to the sample rate decimation on the input IQ side. The up conversion is performed with a half band cascade, thus covers powers of two.



Center Frequency	2.401,000 MHz
Span	1 / 64
Frequency Profile	Select

The signal generator now generates an IQ signal with a lower sample rate.

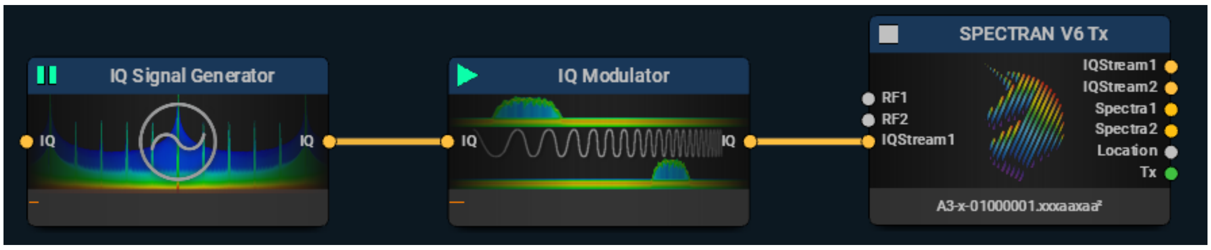


Sample Rate	1,439979 MHz
Center Frequency	2.401,000000 MHz

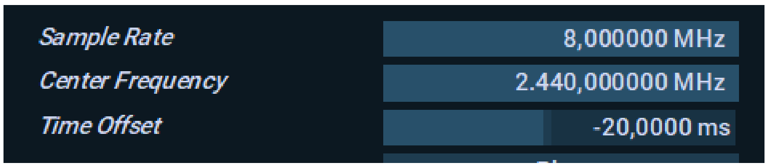
Signal Generator Modulated

This uses “ModulatedSignalGenerator.rmix”

This sample uses the modulator block to generate the full rate IQ stream.

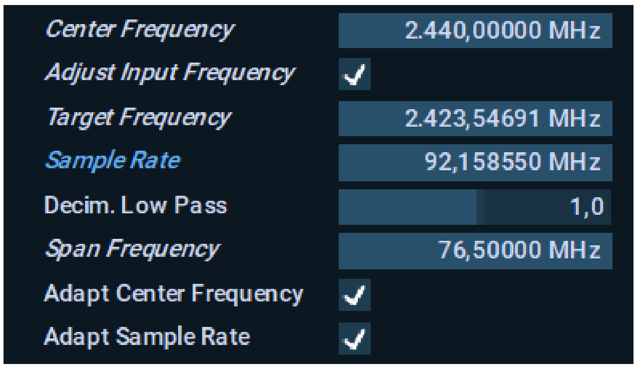


The signal generator provides a narrow band signal to the modulator.



Sample Rate	8,000000 MHz
Center Frequency	2.440,000000 MHz
Time Offset	-20,0000 ms

The time offset is still generated by the signal generator, but the sample rate adaption is now performed in the modulator block.



Center Frequency	2.440,00000 MHz
Adjust Input Frequency	✓
Target Frequency	2.423,54691 MHz
Sample Rate	92,158550 MHz
Decim. Low Pass	1,0
Span Frequency	76,50000 MHz
Adapt Center Frequency	✓
Adapt Sample Rate	✓

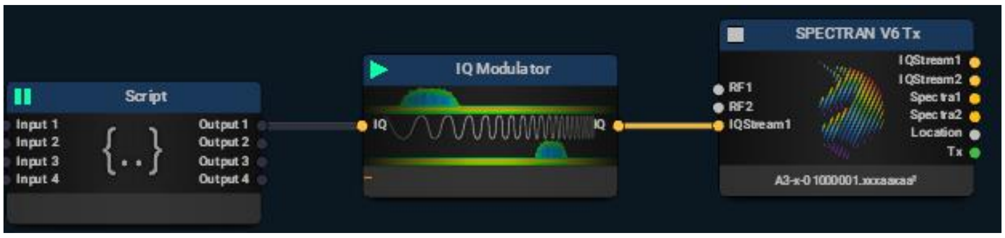
The adjust input frequency checkbox and the target frequency enable an additional frequency shift of the incoming signal, allowing e.g. baseband IQ.

Tx Streaming using Script Block

Fixed Center Frequency

This uses “ModulatedScriptStaticData.rmix”

The signal generator provides a mostly static signal. Using the script block provides the ability to generate the IQ data on the fly.



The modulator block is used for up-conversion, data rate adaption and frequency shift.

The IQ data is prepared in a typed array:

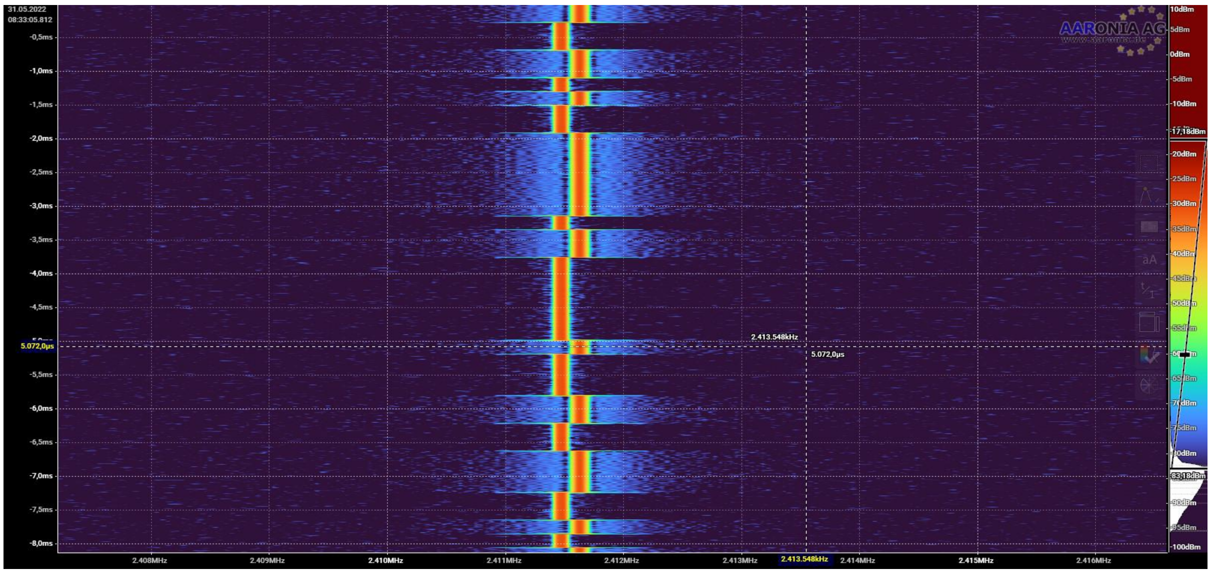
```
// Allocate to samples of complex float data
sample = [new Cplx32Array(256),  new Cplx32Array(256)];

// Fill in sample data
for(let i=0; i<256; i++) {
    let phi = i * Math.PI / 8;
    sample[0][i] =  0.1 * (Math.cos(phi) + 1.0i * Math.sin(phi));
    sample[1][i] =  0.1 * (Math.cos(phi) - 1.0i * Math.sin(phi));
}
```

In this case two variants, one with a negative and one with a positive frequency are generated. A continuous loop is then used to pump this data into the stream, split into packets.

```
    // Build packet meta data
    let packet = {
        payload: "iq",
        startTime: streamTime,
        endTime:   ntime,
        stepFrequency: SampleRate,
        startFrequency: CenterFrequency - 0.5 * SampleRate,
        endFrequency: CenterFrequency + 0.5 * SampleRate,
        samples: sample[Math.floor(Math.random() * 2)]
    };
```

An improved version would use four IQ arrays to provide a smooth transition between the two offset frequencies.



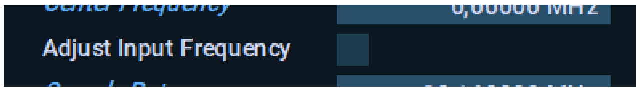
The script block has to ensure that there is always enough data available for the upstream components. It therefore synchronizes the data pump with the stream clock, and delays streaming for 100ms if there is more than 200ms worth of data in flight.

```
  // Delay to avoid buffer overflow, targeting 100ms to 200ms buffer
  let ctime = timer();
  if (streamTime - ctime > 0.2)
      await delay(100);
```

Hopping center frequency

This uses “ModulatedScriptHoppingData.rmix”

The IQ modulator block can change the center frequency of the incoming stream or keep it, and modulate the small band signal according to its center frequency into the full span IQ stream. This allows frequency hopping without the need to actually change the IQ data in the script block.

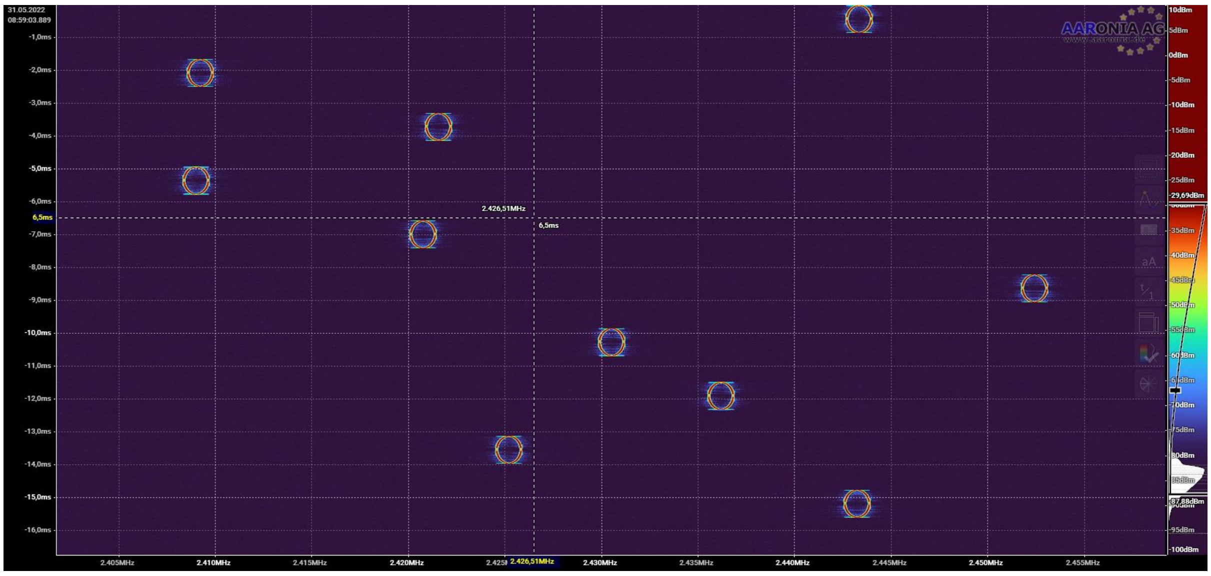

Adjust Input Frequency

The script block selects a random center frequency for each packet.

```
    // Random center frequency for next chunk
    let centerFrequency = Math.random() * 80.0e6 + 2.4e9;

    // Build packet meta data
    let packet = {
        payload: "iq",
        startTime: streamTime,
        endTime:   ntime,
        stepFrequency: SampleRate,
        startFrequency: centerFrequency - 0.5 * SampleRate,
        endFrequency: centerFrequency + 0.5 * SampleRate,
        samples: sample,
        segment: true
    };
```

Setting the segment flag tells the stream, that the frequency has changed from the previous packet.



Playing raw IQ data from a file

This uses “ModulatedScriptFileData.rmix”

The script block can also be used to playback raw IQ data from a file.

```
// Extract path of the mission to get path to the IQ data
let mpath = finfo.mission.split("/");
mpath.pop();
mpath.push("pulse3.iq");

// Read the IQ data into an array buffer
let data = await File.read(mpath.join("/"));

// Use the data as complex 32bit array
sample = new Cplx32Array(data);
```

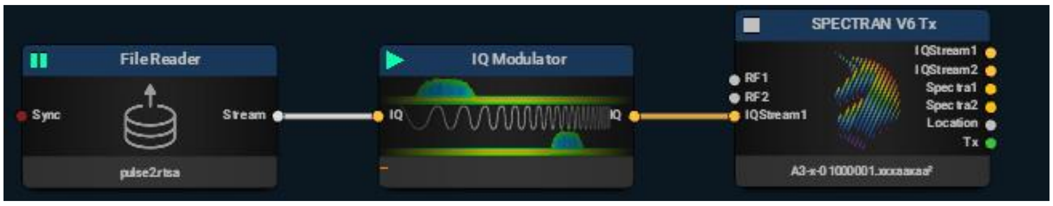
The IQ data is read into an array buffer and then treated as a complex 32bit float array.

Streaming from a File Reader

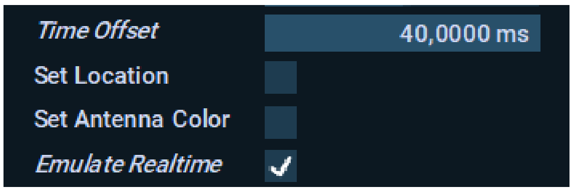
Continuous replay an IQ sample

This uses “ModulatedFileReader.rmix”

An alternative for file playback for .rtsa files is to use a file reader block as source.



The recording time of the stream has to be changed to the current stream time and a buffer added:



Time Offset	40,0000 ms
Set Location	
Set Antenna Color	
Emulate Realtime	✓

The IQ modulator takes care of the sample rate adaption. This sample also uses up-sampling in the SPECTRAN V6 and the modulator block.