AHEAD OF WHAT'S POSSIBLE™

THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

Using GNU Radio to do signal acquisition and analysis with Scopy

ADRIAN SUCIU

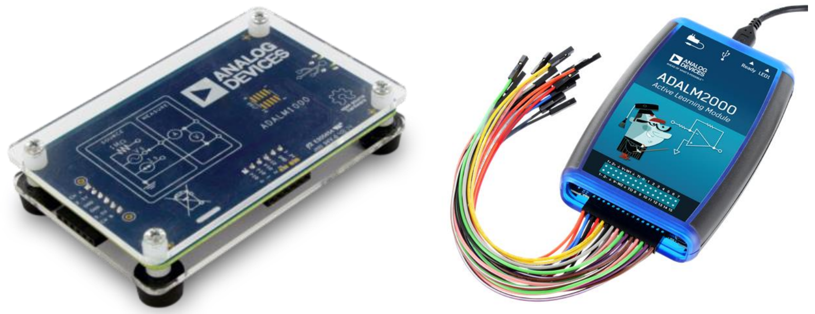
GNU Radio Conference 2018

September 17 - 21, 2018 - Henderson Convention Center, Henderson, Nevada

Analog Devices Active Learning Program

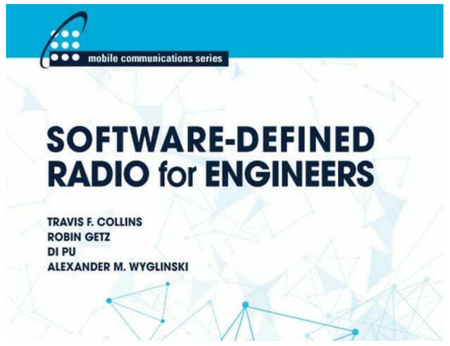
- Dedicated to inspiring students to better understand analog real world signals
- Enables integration of technology into course curricula, design and research projects
- Consist of:
  - Active learning modules
    - ADALM-1000
    - ADALM-2000
    - ADALM-PLUTO (SDR)
  - Over 60 free courseware materials & labs https://wiki.analog.com/university/courses/electronics/labs
  - Free books, webcasts, tutorials, etc.

Entry-Level Hardware Platforms



Analog Parts Kit

Software-Defined Radio for Engineers Book



mobile communications series

SOFTWARE-DEFINED RADIO for ENGINEERS

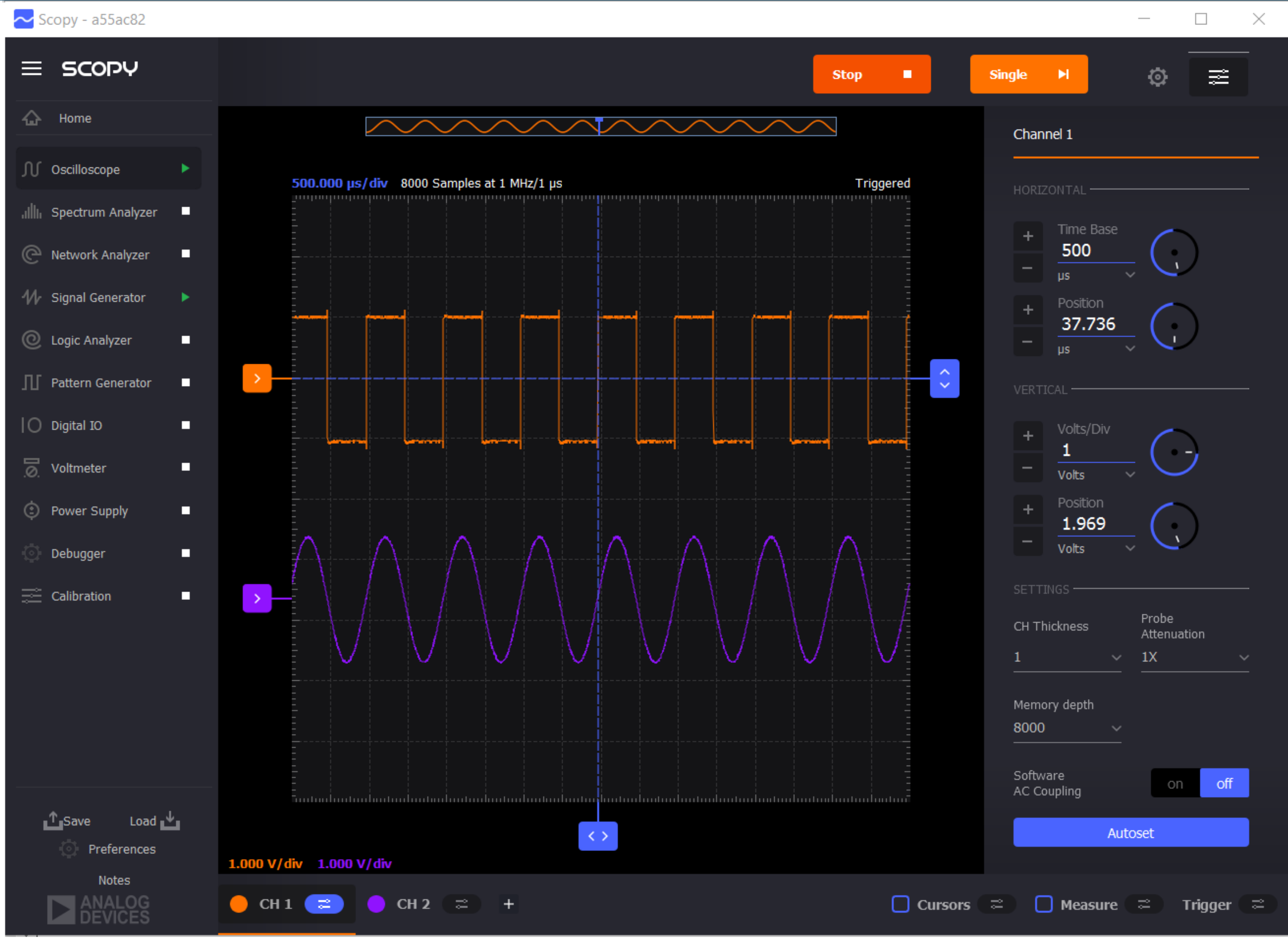
TRAVIS F. COLLINS
ROBIN GETZ
DI PU
ALEXANDER M. WYGLINSKI

The hardware – ADALM2000

- USB powered multifunction instrument
- Software defined measurement platform
  - System components
  - Connectivity options
- Specs:
  - 2 channel 100 MSPS ADC
  - 2 channel 150 MSPS DAC
  - 2 x +/- 5V power supply
  - 16 Digital I/O channels – buffered @ 100MSPS
- Open source hardware and firmware
  - https://github.com/analogdevicesinc/m2k-fw
  - https://wiki.analog.com/university/tools/m2k/hacking/hardware

The software – Scopy

- ► Touch-friendly, modern-looking, Qt based cross-platform GUI (Windows, Linux, MacOS)
- ► Features several virtual instruments:
  - ▪ Oscilloscope
  - ▪ Signal generator
  - ▪ Voltmeter
  - ▪ Power supply
  - ▪ Spectrum analyzer
  - ▪ Network analyzer
  - ▪ Logic analyzer
  - ▪ Pattern generator
- ► GNU Radio is used for signal processing and acquisition
- ► Open source - https://github.com/analogdevicesinc/scopy



500.000 µs/div 8000 Samples at 1 MHz/1 µs Triggered

1.000 V/div 1.000 V/div

Channel 1

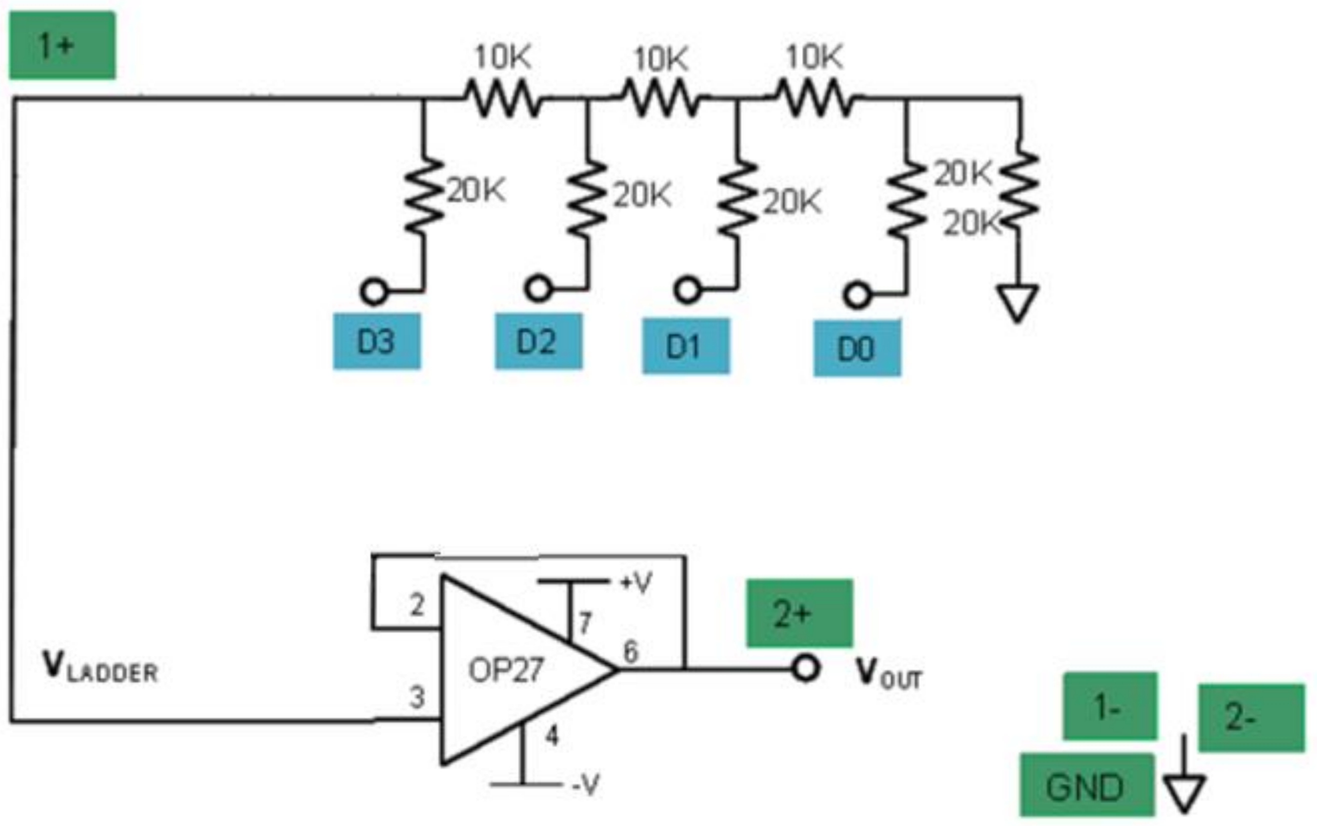
HORIZONTAL
Time Base 500 µs
Position 37.736 µs

VERTICAL
Volts/Div 1 Volts
Position 1.969 Volts

SETTINGS
CH Thickness 1
Probe Attenuation 1X
Memory depth 8000
Software AC Coupling on [✓ off]

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

R-2R Ladder DAC DEMO

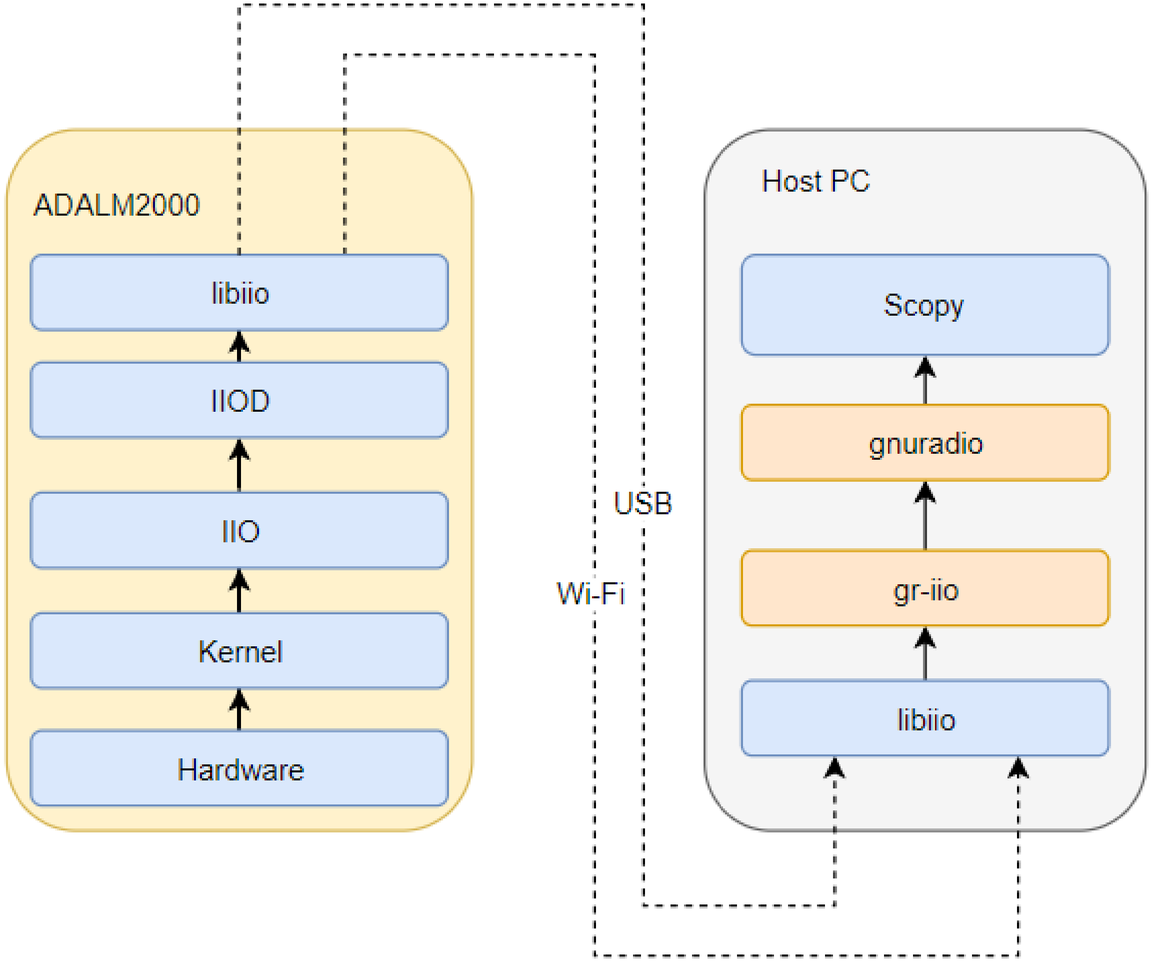


Why we chose GNU Radio ?

- Provides lots of signal processing blocks relevant to our application
- Library is geared towards performance
- Already has Qwt based visualization blocks

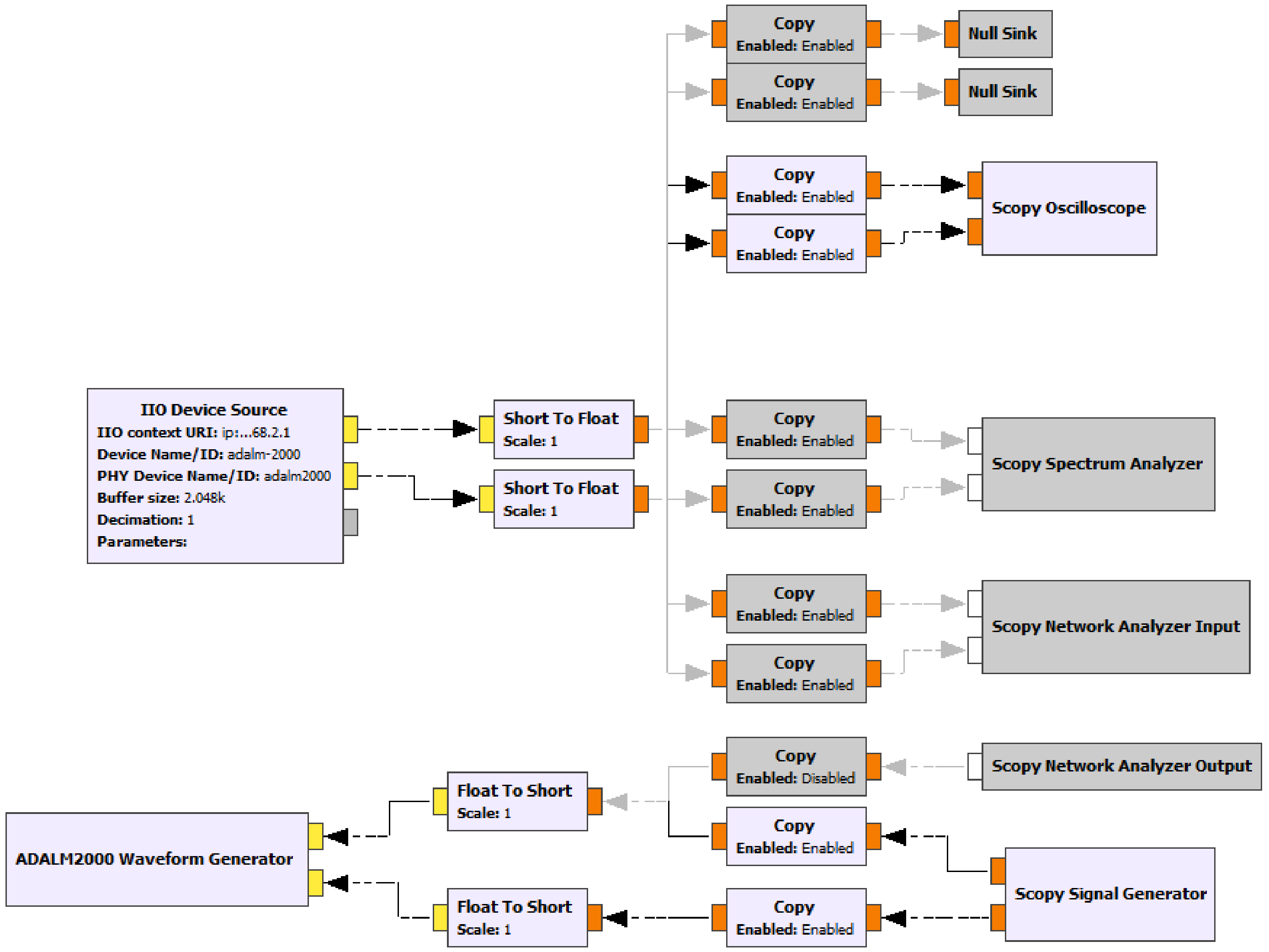
Analog Signal Chain

- ADALM2000 is an embedded Linux host
- Uses IIO subsystem to manage its inputs and outputs (not only used for Industrial I/O)
- libiio is a system library that abstracts the low-level details of the IIO subsystem
- IIOD provides IIO data remotely to clients via USB, IP or even Serial
- gr-iio is used as an interface between GNU Radio and IIO devices
- Scopy uses GNU Radio and gr-iio block to acquire and process data



How do we use GNU Radio ?

- IIO Manager is basically a multiplexer between the I/O and the instruments
- GNU Radio flowgraphs are really not meant to be modified on the fly
- Copy block restricts some parts of the flowgraph, so not all signal paths are enabled at a time
- The IIO Manager remembers all the created paths and destroys them recursively when required

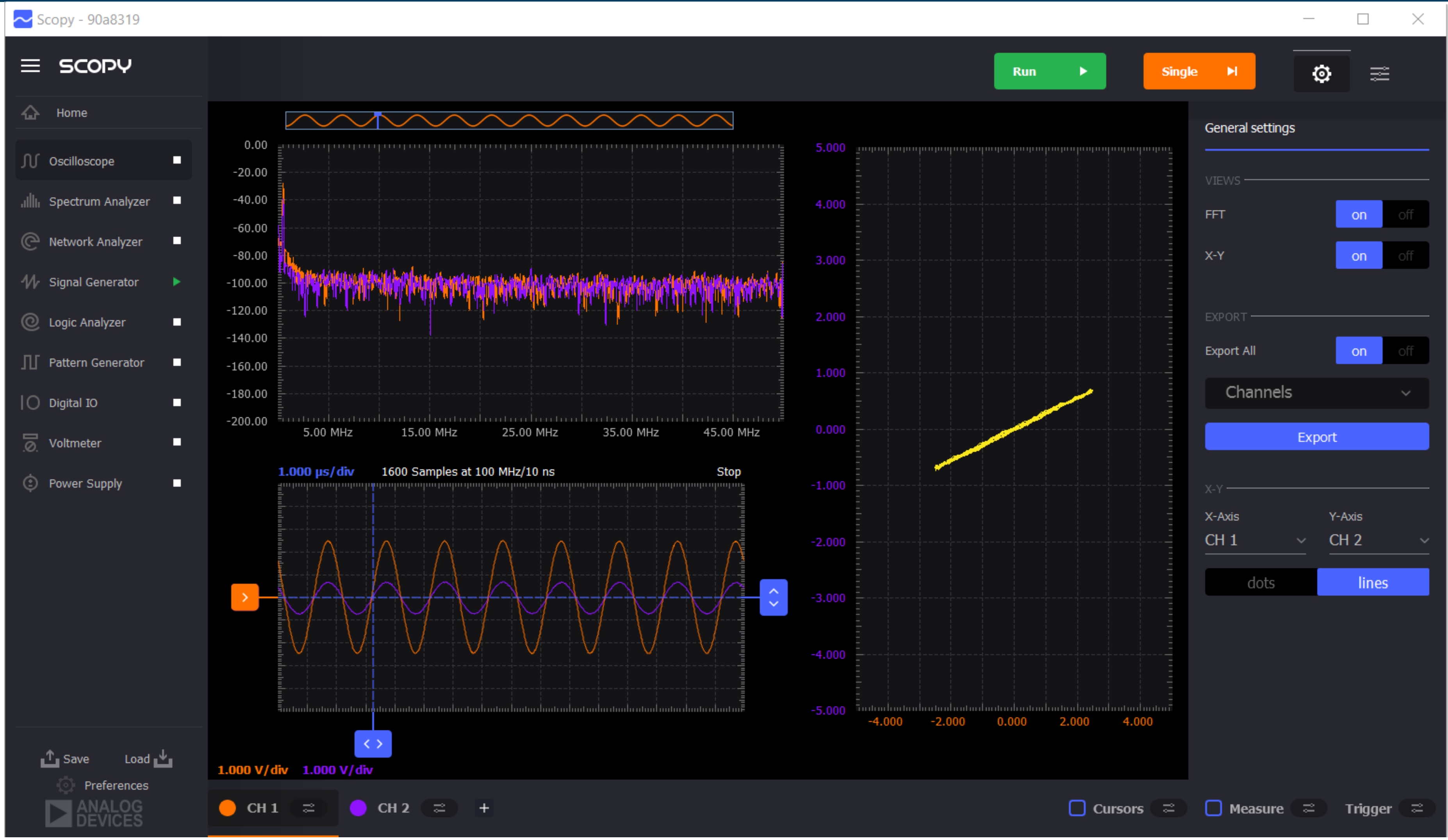


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Oscilloscope

- 2 channel, 100 MSPS
- +/- 20V input range
- Triggered operation

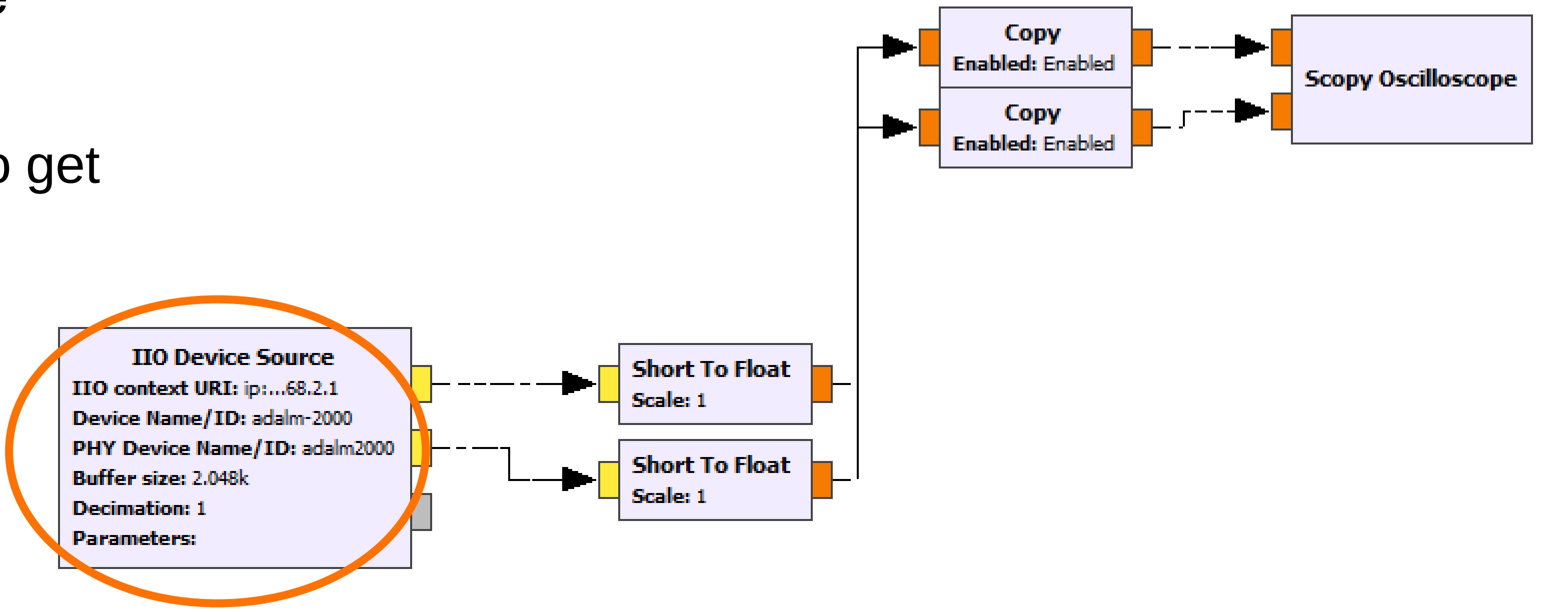
- Features:
  - Autoset
  - AC Coupling
  - Reference channels
  - Math channels



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Oscilloscope – Acquisition and triggering

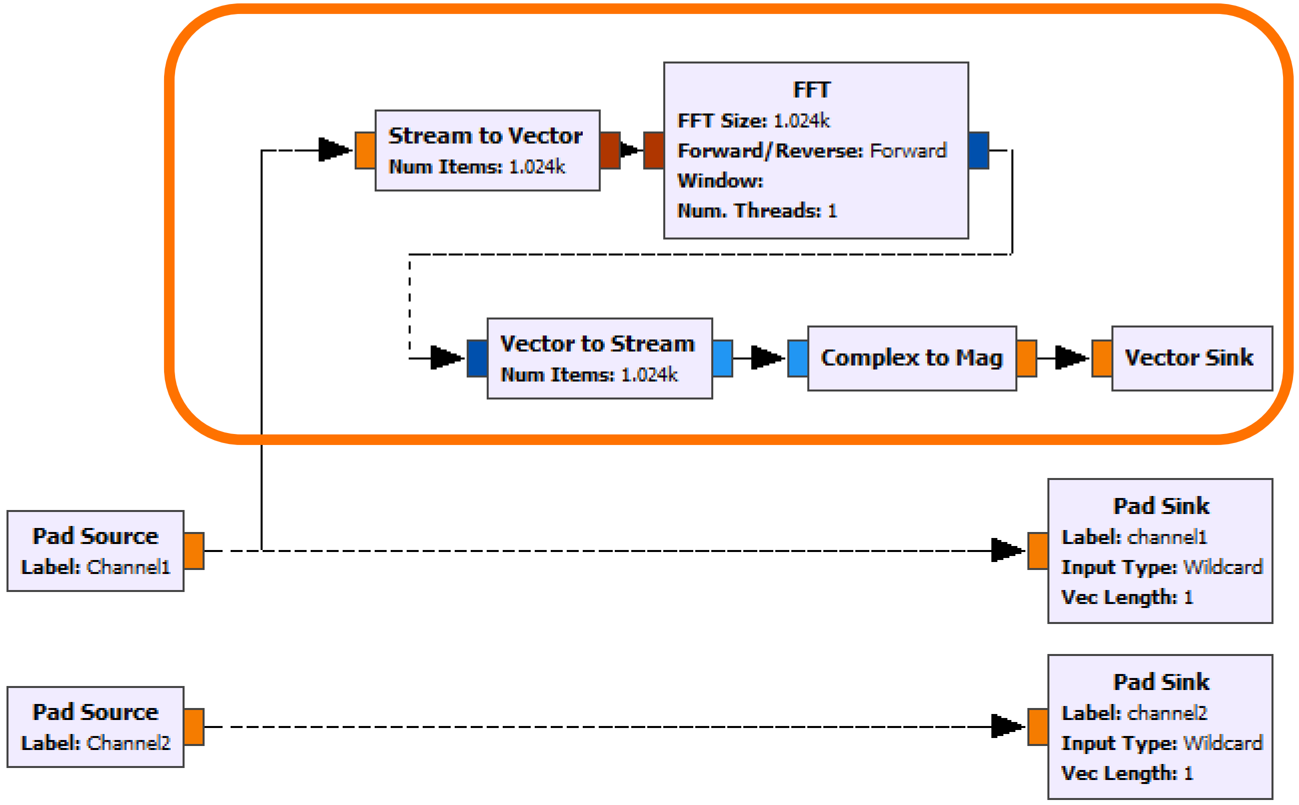
- M2K precisely controls the trigger position using it’s trigger logic core
- gr-iio calls libiio iio_buffer_refill() to get data from the device
- When data is pushed into the flowgraph, gr-iio marks the start of buffer with a tag
- Having buffer_start tag and knowing where the trigger position is, we can precisely draw the triggered waveform on the screen.



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Oscilloscope autoset

- Helps the user set time base, volts/div and trigger settings
- Uses GNU Radio FFT block to find the highest amplitude tone
- It sweeps through all the available sample rates to improve the range of the autoset.
- When a suitable tone is found, it sets the time base.
- Volts/Div and trigger settings are set by analyzing the peaks of the signals in the buffer.



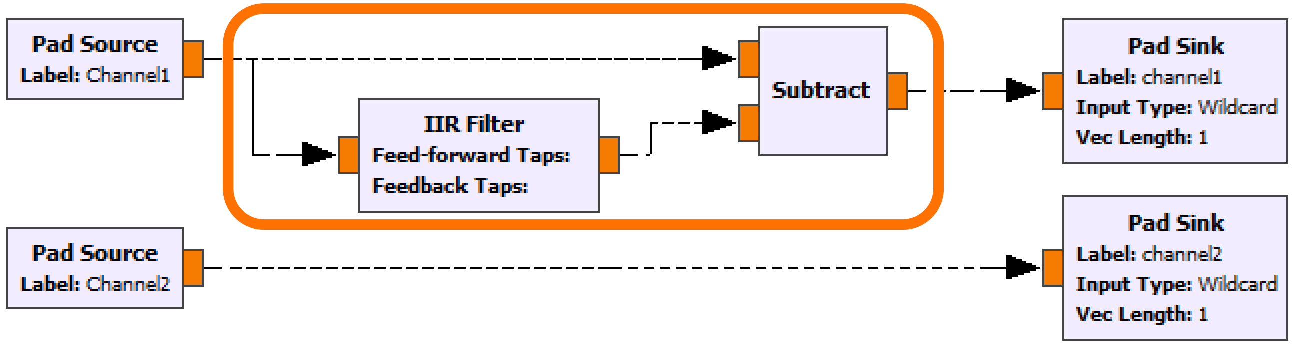
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Oscilloscope AC coupling

- AC coupling basically removes the DC component from the acquired signal

- We implemented a digital LPF using the IIR filter block and subtracted the original signal from the filtered result.

- Since this is a digital filter there are certain limitations
  - Original signal needs to be within hardware input range
  - Setting the trigger is kind of complicated



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

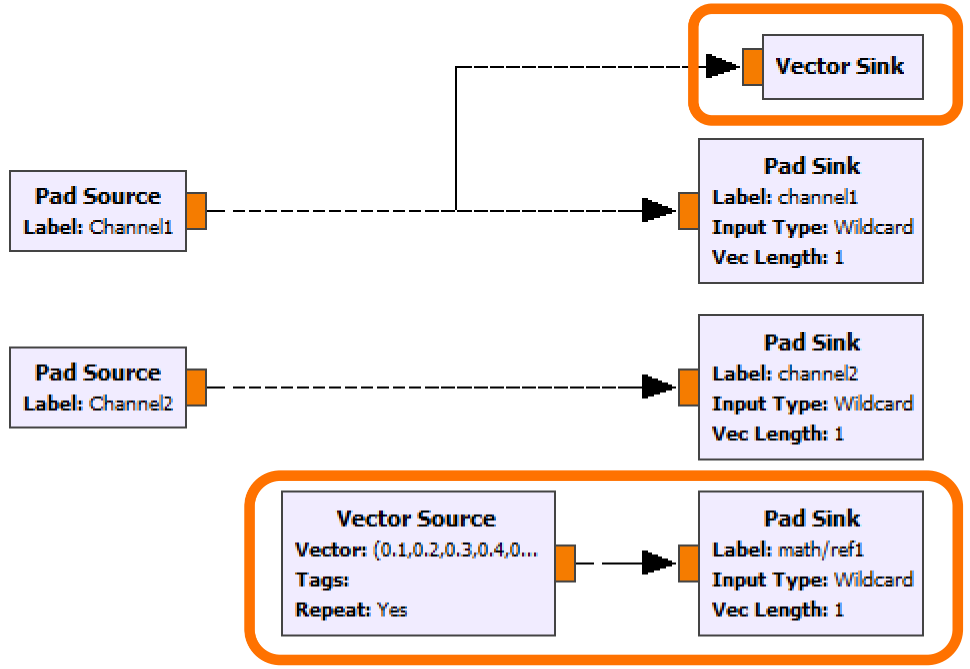
Oscilloscope reference waveform

- It is possible to save waveforms and load them later

- Can be useful to compare current signals to signals acquired in the past

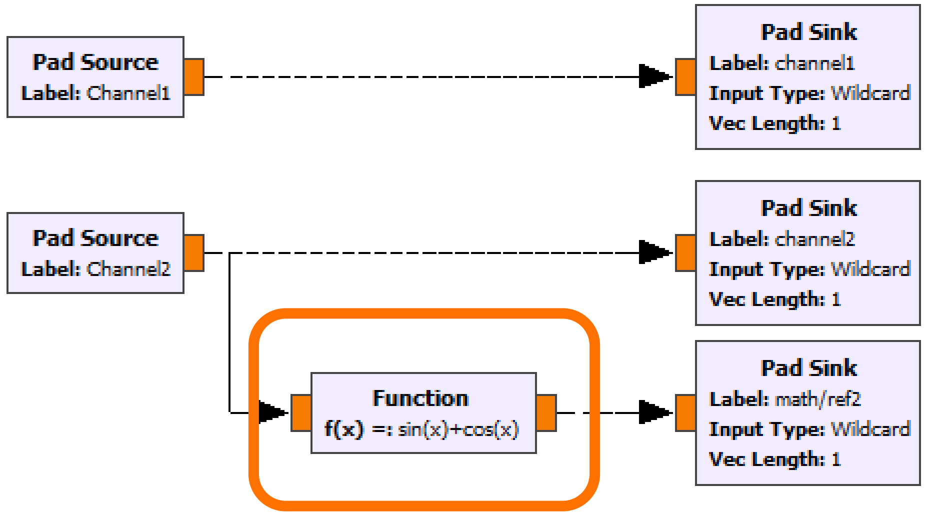
- Using the vector sink we extract data from the flow, encode it in a read-friendly CSV format and save it to a file.

- Eventually, we decode the CSV file, load it into a vector source and display it on the screen.



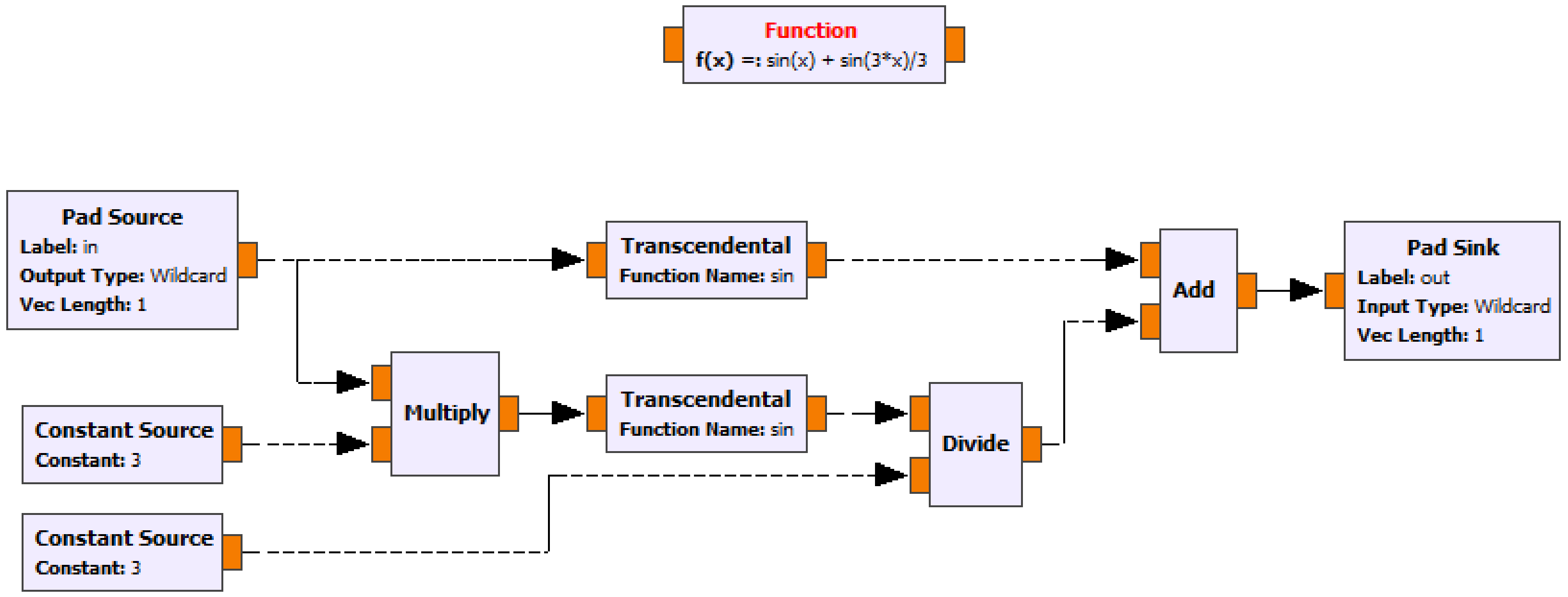
Oscilloscope math channels

- Uses iio_math block from gr-iio
- Math channels basically use GR blocks to compute the result of a mathematical equation from a string
- It was implemented using a flex & bison parser that creates a GR hierarchical block dynamically
- It can take the acquired signal as an input



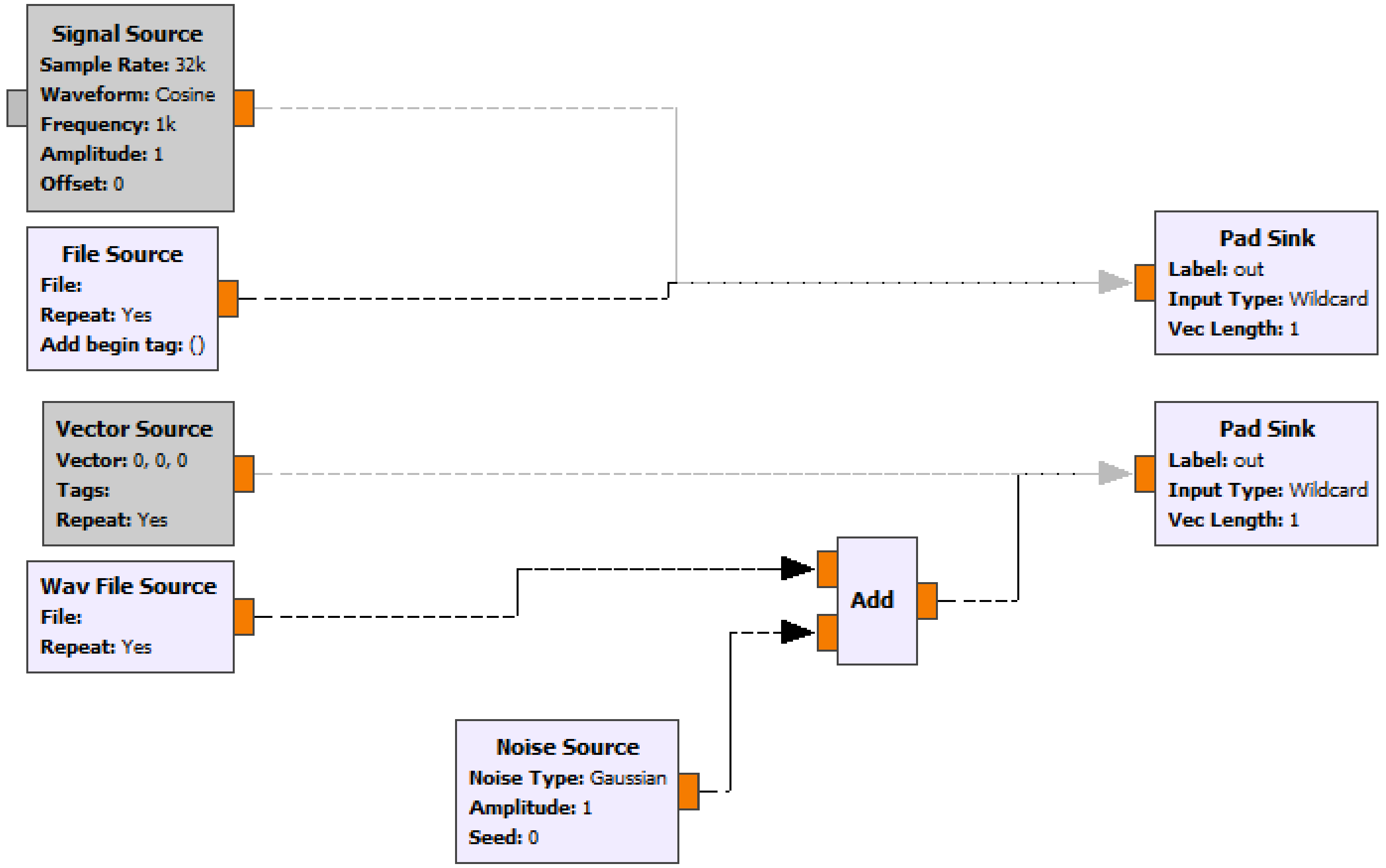
iio_math example

- f(x) = sin(x) + sin(3*x)/3



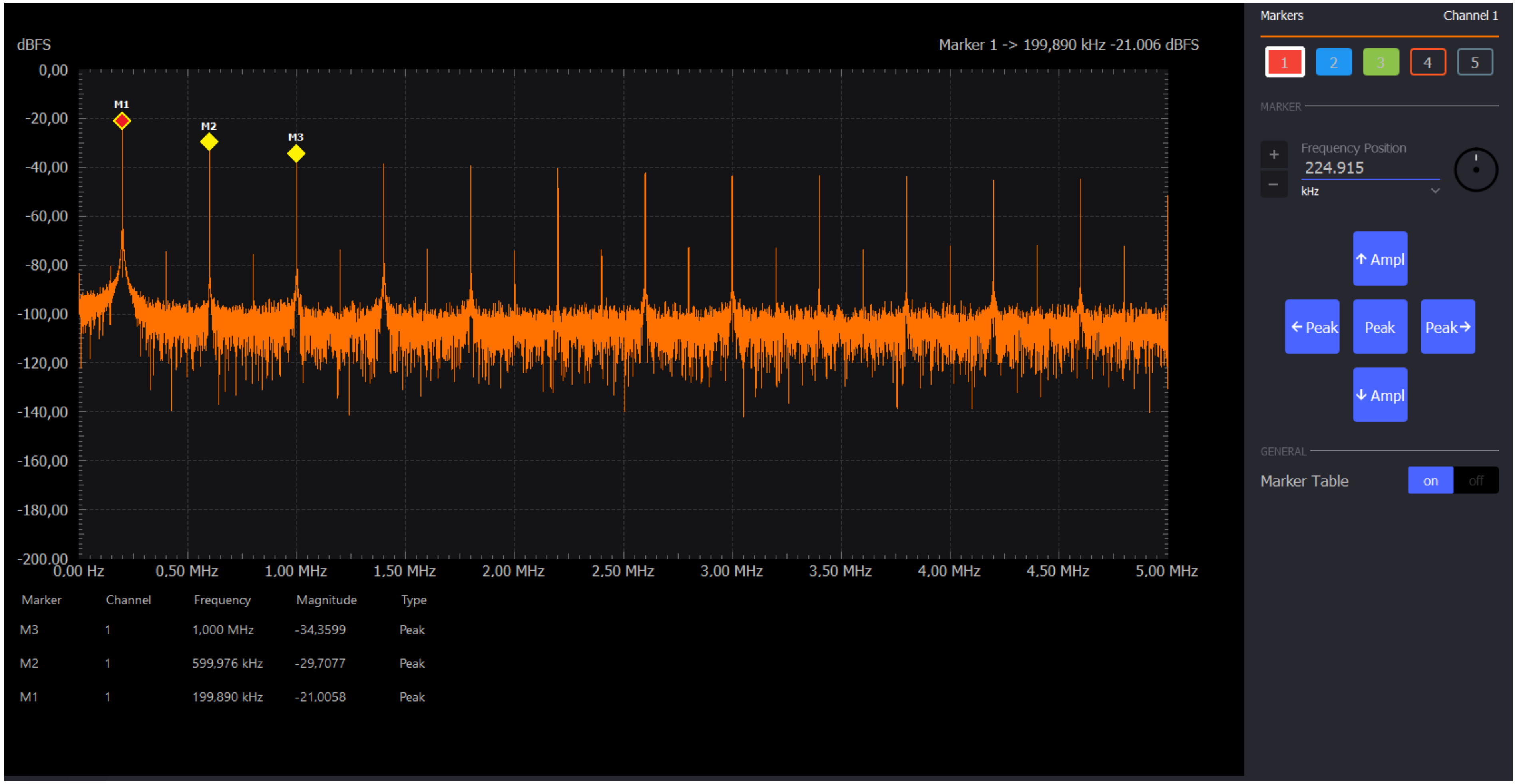
Signal Generator

- Generate basic signal types such as (sine, triangular, square, sawtooth) provided by sig_source_block
- Supports file inputs (binary, WAVE, CSV and MAT formats)
- Can generate signals from math functions using iio_math block
- Noise can be added on top of the signal provided by the noise_source block



Spectrum analyzer

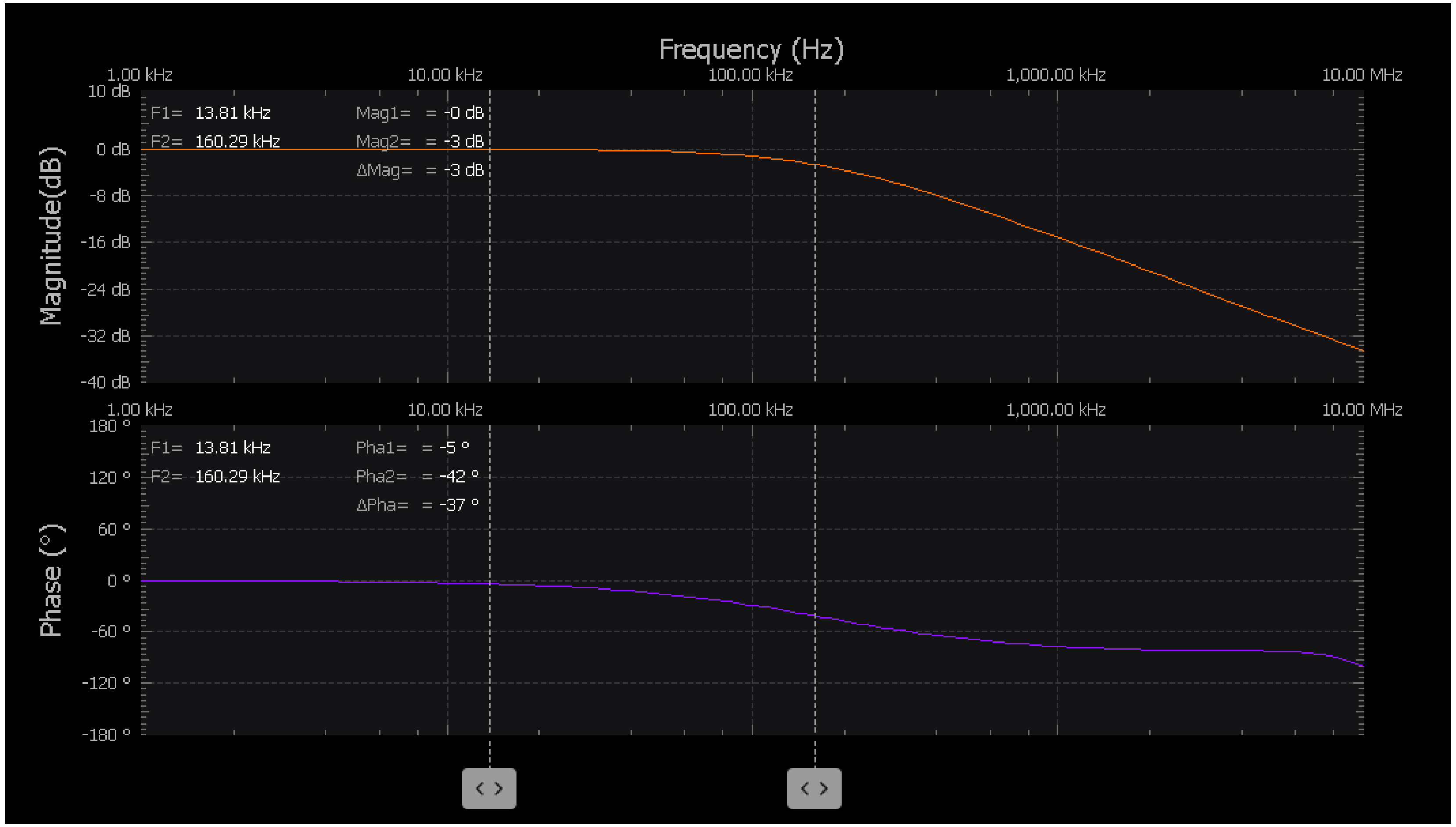
- Up to 50MHz range
- Uses fft block to compute the spectrum
- Uses GNU Radio windowing
- Also has averaging and marker functionality



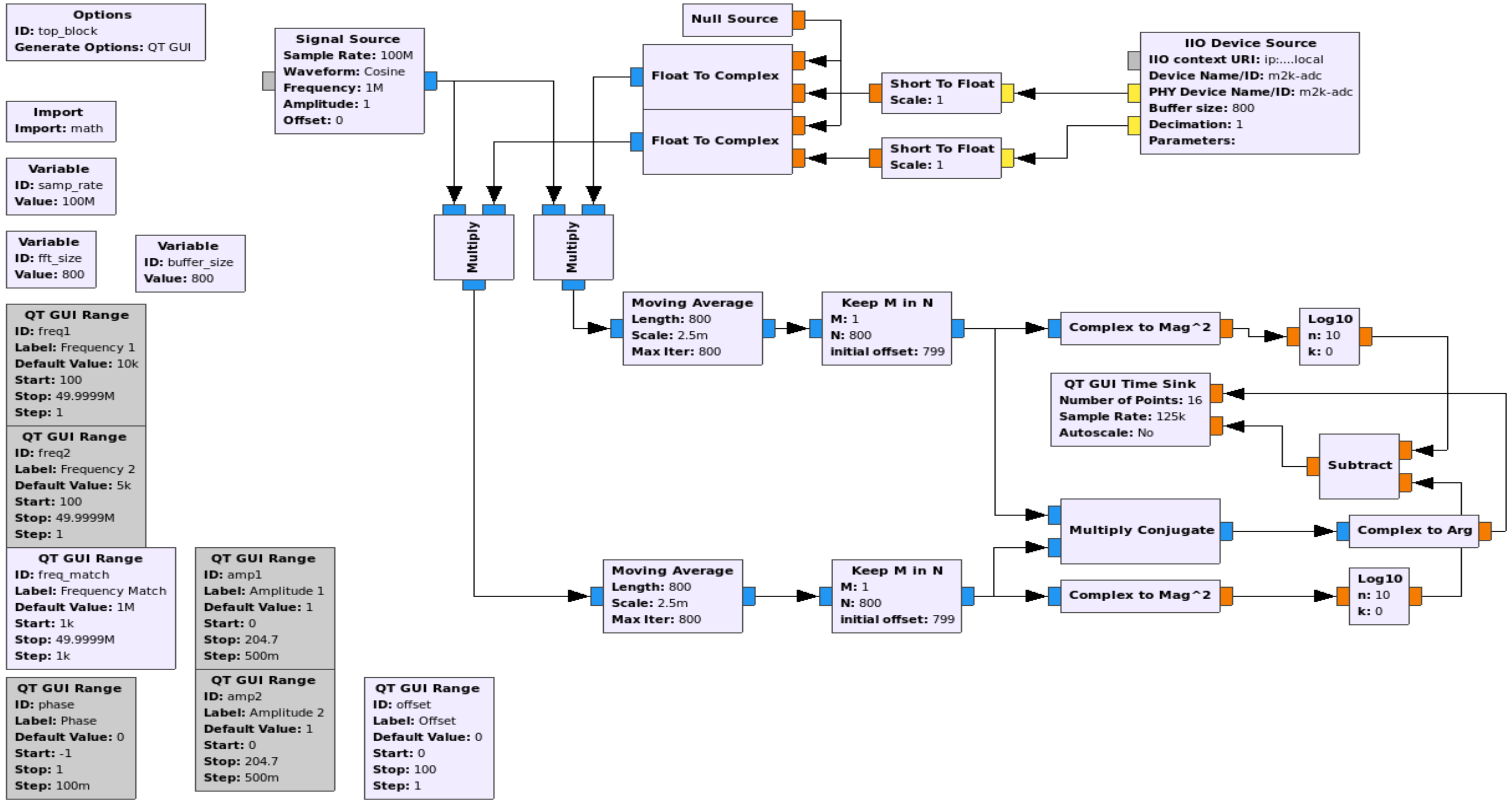
Marker	Channel	Frequency	Magnitude	Type
M3	1	1,000 MHz	-34,3599	Peak
M2	1	599,976 kHz	-29,7077	Peak
M1	1	199,890 kHz	-21,0058	Peak

Network Analyzer

- A network analyzer computes the frequency and phase response of a circuit
- Sweep through the frequencies using the AWG
- For each frequency compute the frequency and phase response and plot it on the screen
- Uses GNU Radio to do that



Network Analyzer



Options
ID: top_block
Generate Options: QT GUI

Import
Import: math

Variable
ID: samp_rate
Value: 100M

Variable
ID: fft_size
Value: 800

Variable
ID: buffer_size
Value: 800

QT GUI Range
ID: freq1
Label: Frequency 1
Default Value: 10k
Start: 100
Stop: 49.9999M
Step: 1

QT GUI Range
ID: freq2
Label: Frequency 2
Default Value: 5k
Start: 100
Stop: 49.9999M
Step: 1

QT GUI Range
ID: freq_match
Label: Frequency Match
Default Value: 1M
Start: 1k
Stop: 49.9999M
Step: 1k

QT GUI Range
ID: amp1
Label: Amplitude 1
Default Value: 1
Start: 0
Stop: 204.7
Step: 500m

QT GUI Range
ID: phase
Label: Phase
Default Value: 0
Start: -1
Stop: 1
Step: 100m

QT GUI Range
ID: amp2
Label: Amplitude 2
Default Value: 1
Start: 0
Stop: 204.7
Step: 500m

QT GUI Range
ID: offset
Label: Offset
Default Value: 0
Start: 0
Stop: 100
Step: 1

Signal Source
Sample Rate: 100M
Waveform: Cosine
Frequency: 1M
Amplitude: 1
Offset: 0

Null Source

Float To Complex

Float To Complex

Short To Float
Scale: 1

Short To Float
Scale: 1

IIO Device Source
IIO context URI: ip:....local
Device Name/ID: m2k-adc
PHY Device Name/ID: m2k-adc
Buffer size: 800
Decimation: 1
Parameters:

Moving Average
Length: 800
Scale: 2.5m
Max Iter: 800

Keep M in N
M: 1
N: 800
initial offset: 799

Complex to Mag^2

Log10
n: 10
k: 0

QT GUI Time Sink
Number of Points: 16
Sample Rate: 125k
Autoscale: No

Subtract

Multiply Conjugate

Complex to Arg

Moving Average
Length: 800
Scale: 2.5m
Max Iter: 800

Keep M in N
M: 1
N: 800
initial offset: 799

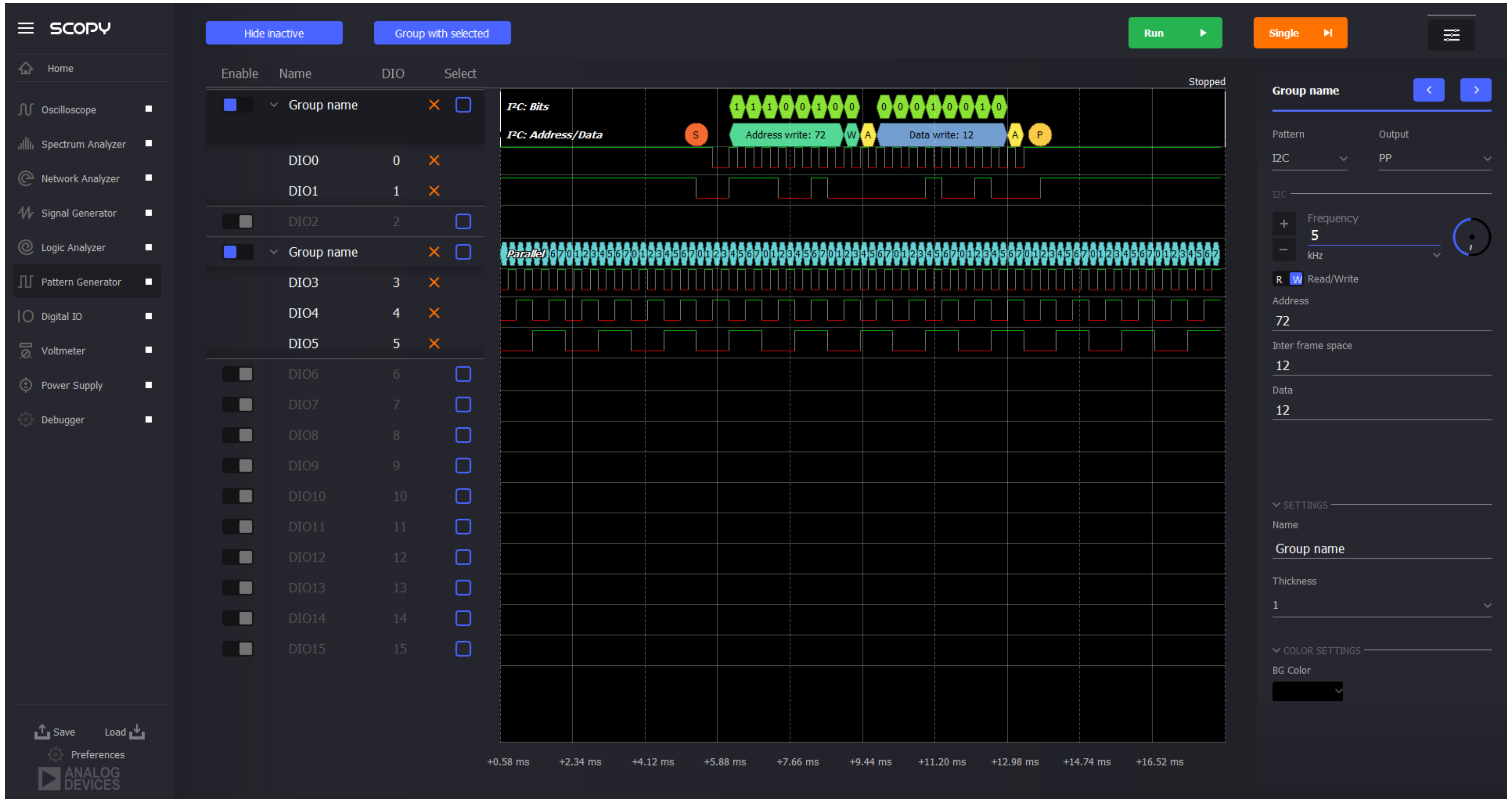
Complex to Mag^2

Log10
n: 10
k: 0

Logic Instruments

- 16 Channel digital logic analyzer and pattern generator
- Up to 100MSPS

- Plotting - PulseView
- Signal acquisition – libiio and libsigrok
- decoding - libsigrok and libsigrokdecode
- Handles countless protocols such as I2C/SPI/UART/JTAG



Name	DIO
Group name	
DIO0	0
DIO1	1
DIO2	2
Group name	
DIO3	3
DIO4	4
DIO5	5
DIO6	6
DIO7	7
DIO8	8
DIO9	9
DIO10	10
DIO11	11
DIO12	12
DIO13	13
DIO14	14
DIO15	15

+0.58 ms +2.34 ms +4.12 ms +5.88 ms +7.66 ms +9.44 ms +11.20 ms +12.98 ms +14.74 ms +16.52 ms

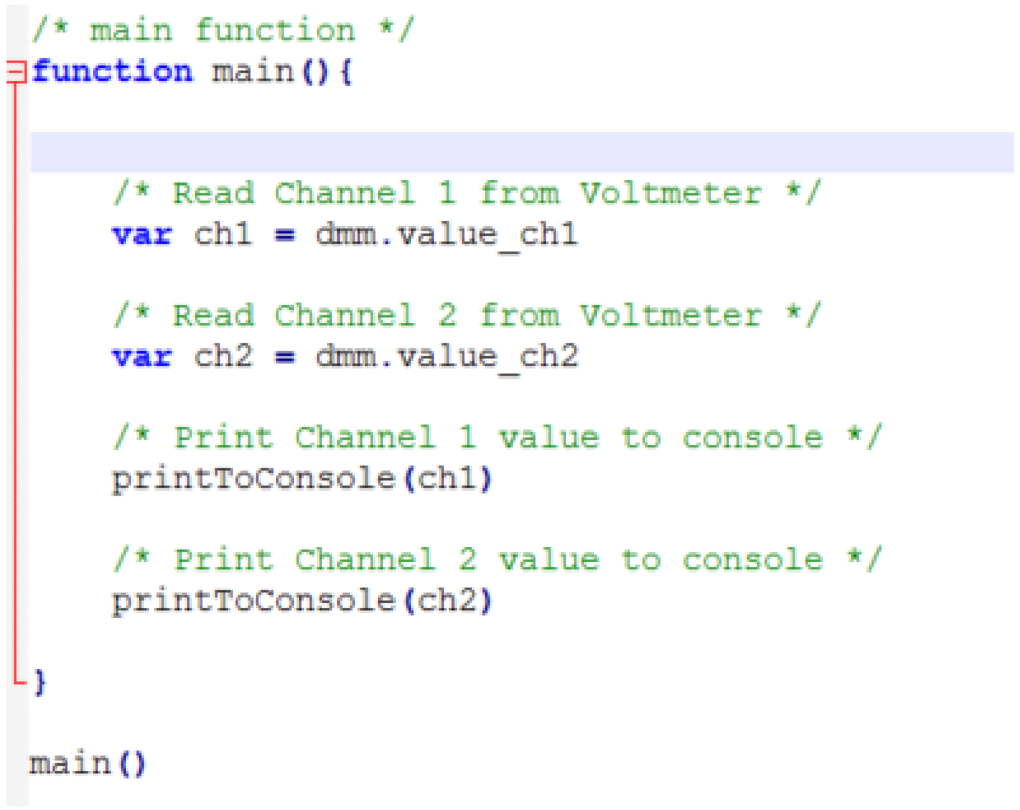
Group name

Pattern: I2C
Output: PP
I2C
Frequency: 5 kHz
R W Read/Write
Address: 72
Inter frame space: 12
Data: 12
SETTINGS
Name: Group name
Thickness: 1
COLOR SETTINGS
BG Color

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Scopy Scripting Environment

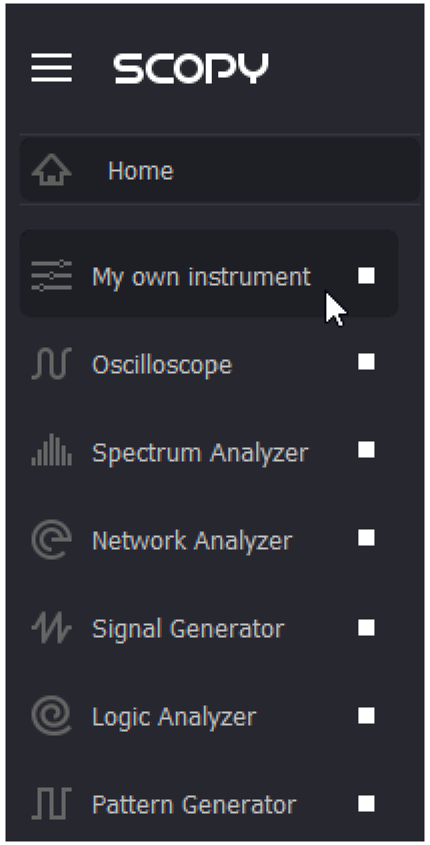
- Provides scripting capabilities using a stripped down version of JavaScript provided by Qt
- Can set/run instruments use measurements and values from buffers
- We actually used scripting to do automated hardware tests for the boards.
- Downside: Scopy still has to run to automate the instruments



```
/* main function */
function main(){


    /* Read Channel 1 from Voltmeter */
    var ch1 = dmm.value_ch1

    /* Read Channel 2 from Voltmeter */
    var ch2 = dmm.value_ch2

    /* Print Channel 1 value to console */
    printToConsole(ch1)

    /* Print Channel 2 value to console */
    printToConsole(ch2)

}

main()
```

Scopy plugins

- Developers can create their own instruments
- ToolLauncher class provides the container for all the instruments
- Tool class provides the base for all instruments
- IIO Context provided by Scopy
- Example on the dummy-instrument branch https://github.com/analogdevicesinc/scopy/tree/dummy-instrument



Conclusion – likes & dislikes

- What we like about GNU Radio
  - ➕ We can easily modify the signal chain, to modify signal acquisition or add specific signal processing features - very few lines of code change
  - ➕ GNU Radio provides lots of blocks that can be used right out of the box. When implementing you only have to think about the big picture, as implementation details are abstracted inside the blocks.
  - ➕ Works great !

- What we don't like about GNU Radio
  - ➖ There is no easy way to do flow reconfiguration and enabling/disabling channels or instruments require flow reconfiguration
  - ➖ When developing, if the flowgraph hangs, it's hard to tell which block chokes, or what you're doing wrong.

Future plans

- We intend to add support to other iio devices (such as Pluto SDR)
- Better plugin support (no need to recompile everything, make use of JavaScript)
- We intend to create libm2k - a library that will make interfacing with the M2K easier.

Questions ?