



The GNU Radio Companion Changelog

Communications Engineering Lab
Prof. i.R. Dr.rer.nat. Friedrich K. Jondral



Overview

- GNU Radio Companion Intro
 - Graphical Flow Graph Design
 - How to add your own blocks

- Recently added features
 - Bypassed Blocks
 - Embedded Python Blocks / Modules
 - Custom Run Commands
 - Bootstrap depending hier_blocks

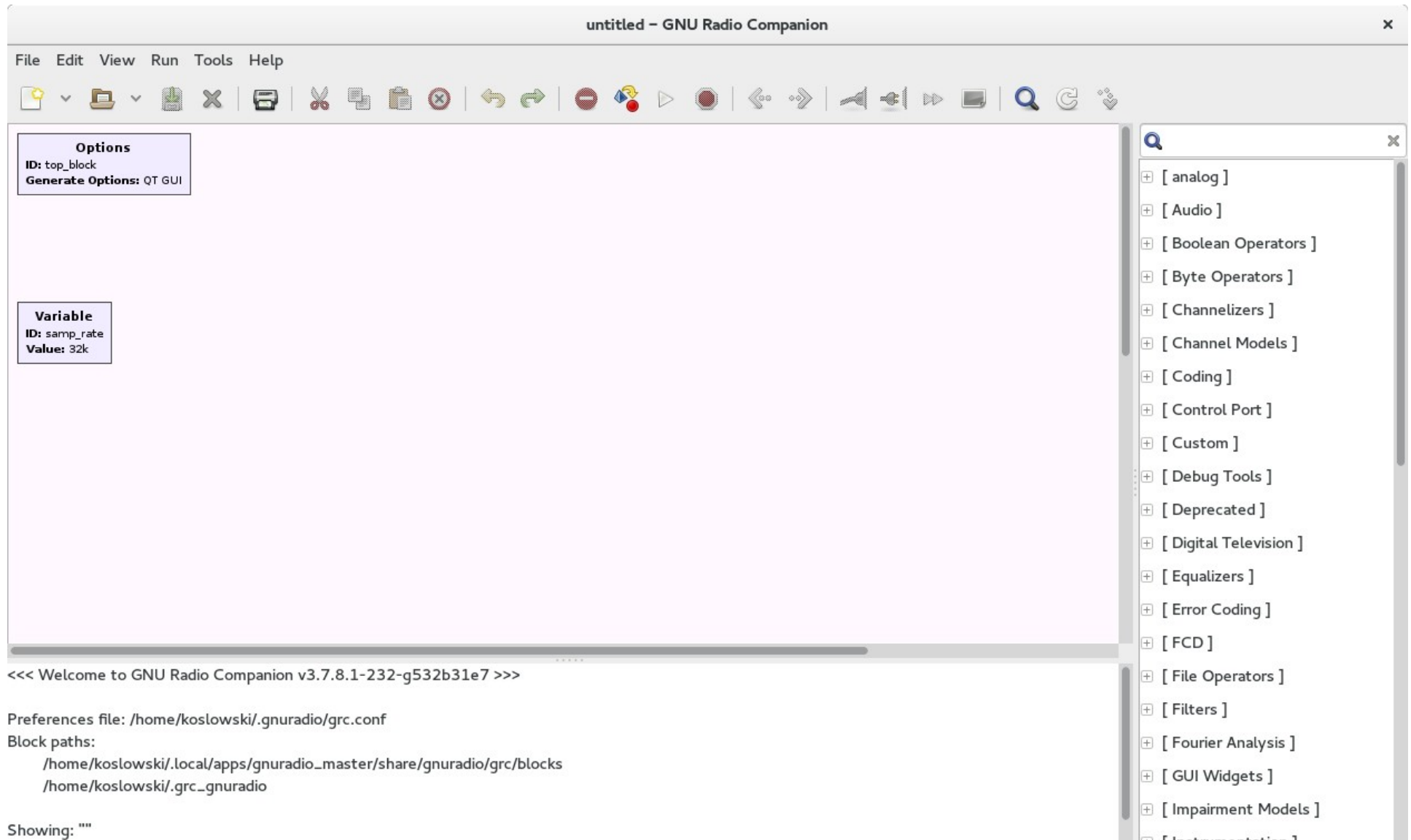
- Current development and plans for future versions



The GNU Radio Companion

- Written by Josh Blum around 2007
 - Extended, patched and tinkered with by many since
 - 45 contributors (~90% of commits by 4 people)
- Maintained by me since 2013
- Written in (legacy) Python
 - PyGTK as GUI
 - Cheetah for templating Python code
 - Code base: ~12k lines, almost no tests =(

The GUI



The screenshot shows the GNU Radio Companion (GRC) interface. The window title is "untitled - GNU Radio Companion". The menu bar includes "File", "Edit", "View", "Run", "Tools", and "Help". The toolbar contains various icons for file operations, navigation, and execution. The main workspace is currently empty, displaying two panels: "Options" with ID: top_block and "Generate Options: QT GUI", and "Variable" with ID: samp_rate and "Value: 32k". A search bar on the right side of the workspace is active, showing a list of block categories such as [analog], [Audio], [Boolean Operators], [Byte Operators], [Channelizers], [Channel Models], [Coding], [Control Port], [Custom], [Debug Tools], [Deprecated], [Digital Television], [Equalizers], [Error Coding], [FCD], [File Operators], [Filters], [Fourier Analysis], [GUI Widgets], and [Impairment Models].

untitled - GNU Radio Companion

File Edit View Run Tools Help

Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

<<< Welcome to GNU Radio Companion v3.7.8.1-232-g532b31e7 >>>

Preferences file: /home/koslowski/.gnuradio/grc.conf
Block paths:
/home/koslowski/.local/apps/gnuradio_master/share/gnuradio/grc/blocks
/home/koslowski/.grc_gnuradio

Showing: ""

- [analog]
- [Audio]
- [Boolean Operators]
- [Byte Operators]
- [Channelizers]
- [Channel Models]
- [Coding]
- [Control Port]
- [Custom]
- [Debug Tools]
- [Deprecated]
- [Digital Television]
- [Equalizers]
- [Error Coding]
- [FCD]
- [File Operators]
- [Filters]
- [Fourier Analysis]
- [GUI Widgets]
- [Impairment Models]

Example Flow Graph

example.grc – /home/koslowski/grc – GNU Radio Companion

File Edit View Run Tools Help

Options
 ID: fosdem_intro
 Title: FOSDEM Intro
 Generate Options: QT GUI

Variable
 ID: samp_rate
 Value: 32k

```

  graph LR
    A[Noise Source  
Noise Type: Gaussian  
Amplitude: 1  
Seed: 0] --> B[Throttle  
Sample Rate: 32k]
    B --> C[Low Pass Filter  
Decimation: 1  
Gain: 1  
Sample Rate: 32k  
Cutoff Freq: 8k  
Transition Width: 4k  
Window: Hamming  
Beta: 6.76]
    C --> D[QT GUI Frequency Sink  
Center Frequency (Hz): 0  
Bandwidth (Hz): 32k]
  
```

Generating: '/home/koslowski/grc/fosdem_intro.py'

Executing: '/home/koslowski/.local/apps/gnuradio_master/bin/python -u /home/koslowski/grc/fosdem_intro.py'

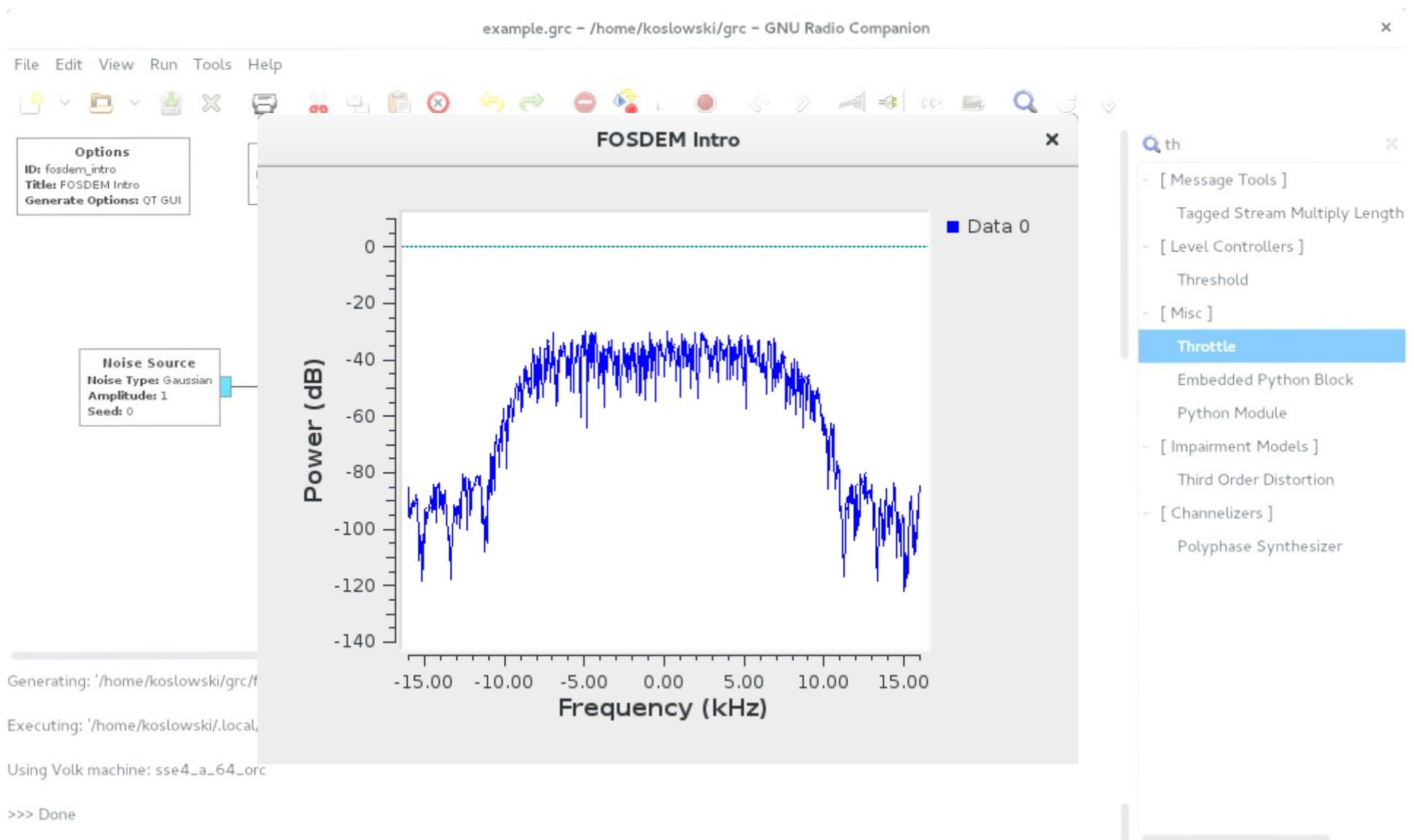
Using Volk machine: sse4_a_64_orc

>>> Done

Search: th

- [Message Tools]
 - Tagged Stream Multiply Length
- [Level Controllers]
 - Threshold
- [Misc]
 - Throttle**
 - Embedded Python Block
 - Python Module
- [Impairment Models]
 - Third Order Distortion
- [Channelizers]
 - Polyphase Synthesizer

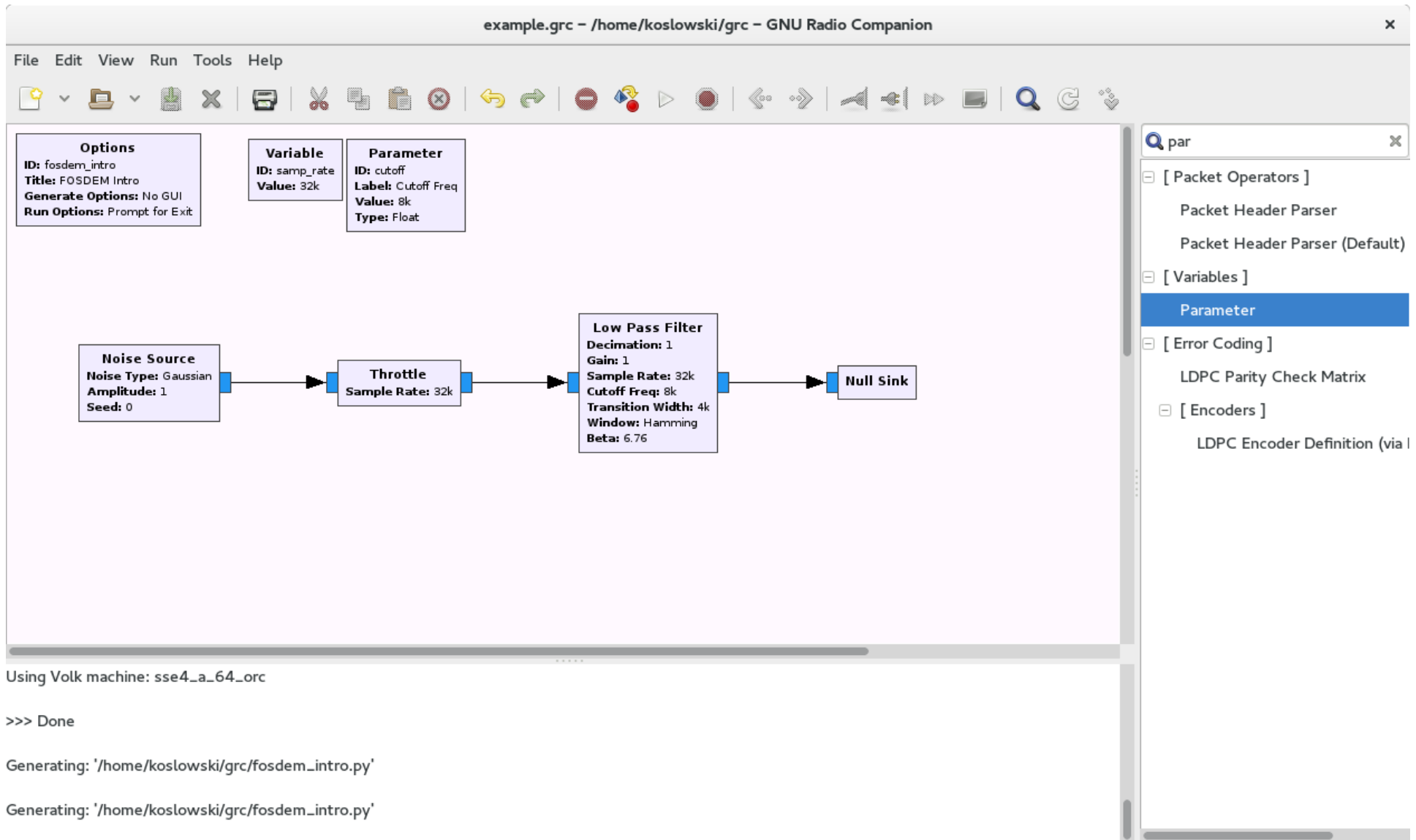
Example Flow Graph in Action



Behind the Scenes Example

example.grc - /home/koslowski/grc - GNU Radio Companion

File Edit View Run Tools Help



Options
ID: fosdem_intro
Title: FOSDEM Intro
Generate Options: No GUI
Run Options: Prompt for Exit

Variable
ID: samp_rate
Value: 32k

Parameter
ID: cutoff
Label: Cutoff Freq
Value: 8k
Type: Float

Noise Source
Noise Type: Gaussian
Amplitude: 1
Seed: 0

Throttle
Sample Rate: 32k

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 32k
Cutoff Freq: 8k
Transition Width: 4k
Window: Hamming
Beta: 6.76

Null Sink

Using Volk machine: sse4_a_64_orc

>>> Done

Generating: '/home/koslowski/grc/fosdem_intro.py'

Generating: '/home/koslowski/grc/fosdem_intro.py'

Search: par

- [Packet Operators]
 - Packet Header Parser
 - Packet Header Parser (Default)
- [Variables]
 - Parameter**
- [Error Coding]
 - LDPC Parity Check Matrix
- [Encoders]
 - LDPC Encoder Definition (via I



Generated Python

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
#####
# GNU Radio Python Flow Graph
# Title: FOSDEM Intro
# Generated: Thu Jan 28 16:35:35 2016
#####
```

```
from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser
```

```
class fosdem_intro(gr.top_block):

    def __init__(self, cutoff=8e3):
        gr.top_block.__init__(self, "FOSDEM Intro")
```

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
#####
# GNU Radio Python Flow Graph
# Title: FOSDEM Intro
# Generated: Thu Jan 28 16:35:35 2016
#####

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser

class fosdem_intro(gr.top_block):

    def __init__(self, cutoff=8e3):
        gr.top_block.__init__(self, "FOSDEM Intro")

    # Parameters
    #####
    self.cutoff = cutoff

    # Variables
    #####
    self.samp_rate = samp_rate = 32000

    # Blocks
    #####
    self.low_pass_filter_0 = filter_fft_filter_ccff1_firdes_low_pass(
        1, samp_rate, cutoff, firdes.WIN_HAMMING, 6.76)
    self.blocks_throttle_0 = blocks_throttle_sizeof_gr_complex*1, samp_rate,True)
    self.blocks_null_sink_0 = blocks_null_sink(sizeof_gr_complex*1)
    self.analog_noise_source_x_0 = analog_noise_source_c(analog.GP_GAUSSIAN, 1, 0)

    # Connections
    #####
    self.connect((self.analog_noise_source_x_0, 0), (self.blocks_throttle_0, 0))
    self.connect((self.blocks_throttle_0, 0), (self.low_pass_filter_0, 0))
    self.connect((self.low_pass_filter_0, 0), (self.blocks_null_sink_0, 0))

    def get_cutoff(self):
        return self.cutoff

    def set_cutoff(self, cutoff):
        self.cutoff = cutoff
        self.low_pass_filter_0.set_fft_size(self.samp_rate, self.cutoff, 443, firdes.WIN_HAMMING, 6.76)

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.blocks_throttle_0.set_sample_rate(self.samp_rate)
        self.low_pass_filter_0.set_fft_size(firdes.low_pass(1, self.samp_rate, self.cutoff, 443, firdes.WIN_HAMMING, 6.76))

    def argument_parser():
        parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
        parser.add_option(
            "--cutoff", dest="cutoff", type="eng_float", default=eng_notation.num_to_str(self.cutoff),
            help="Set Cutoff Freq [default=8000.0]")
        return parser

    def main(top_block_cls=fosdem_intro, options=None):
        # options is None:
        options = argument_parser().parse_args()

        tb = top_block_cls(cutoff=options.cutoff)
        tb.start()

        try:
            time.sleep(1) # Press Enter to quit.
        except KeyboardInterrupt:
            pass
        tb.stop()
        tb.wait()

    # __name__ == "__main__":
    main()
```



Generated Python

```
class fosdem_intro(gr.top_block):
```

```
def __init__(self, cutoff=8e3):
    gr.top_block.__init__(self, "FOSDEM Intro")
```

```
#####
```

```
# Parameters
```

```
#####
```

```
self.cutoff = cutoff
```

```
#####
```

```
# Variables
```

```
#####
```

```
self.samp_rate = samp_rate = 32000
```

```
#####
```

```
# Blocks
```

```
#####
```

```
self.low_pass_filter_0 = filter.fir_filter_ccf(1, firdes.low_pass(
    1, samp_rate, cutoff, 4e3, firdes.WIN_HAMMING, 6.76))
```

```
self.blocks_throttle_0 = blocks.throttle(gr.sizeof_gr_complex*1, samp_rate, T)
```

```
self.blocks_null_sink_0 = blocks.null_sink(gr.sizeof_gr_complex*1)
```

```
self.analog_noise_source_x_0 = analog.noise_source_c(analog.GR_GAUSS
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#####
# GNU Radio Python Flow Graph
# Title: FOSDEM Intro
# Generated: Thu Jan 28 16:35:35 2016
#####

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_rotator
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser

class fosdem_intro(gr.top_block):
    def __init__(self, cutoff=8e3):
        gr.top_block.__init__(self, "FOSDEM Intro")

        # Parameters
        self.cutoff = cutoff

        # Variables
        self.samp_rate = samp_rate = 32000

        # Blocks
        self.low_pass_filter_0 = filter.fir_filter_ccf(1, firdes.low_pass(
            1, samp_rate, cutoff, 4e3, firdes.WIN_HAMMING, 6.76))
        self.blocks_throttle_0 = blocks.throttle(gr.sizeof_gr_complex*1, samp_rate, T)
        self.blocks_null_sink_0 = blocks.null_sink(gr.sizeof_gr_complex*1)
        self.analog_noise_source_x_0 = analog.noise_source_c(analog.GR_GAUSSIAN, 1, 0)

        # Connections
        self.connect((self.analog_noise_source_x_0, 0), (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.low_pass_filter_0, 0))
        self.connect((self.low_pass_filter_0, 0), (self.blocks_null_sink_0, 0))

    def get_cutoff(self):
        return self.cutoff

    def set_cutoff(self, cutoff):
        self.cutoff = cutoff
        self.low_pass_filter_0.set_topo(firdes.low_pass(1, self.samp_rate, self.cutoff, 4e3, firdes.WIN_HAMMING, 6.76))

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.blocks_throttle_0.set_sample_rate(self.samp_rate)
        self.low_pass_filter_0.set_topo(firdes.low_pass(1, self.samp_rate, self.cutoff, 4e3, firdes.WIN_HAMMING, 6.76))

def argument_parser():
    parser = OptionParser(option_class=eng_option, usage="%prog [options]")
    parser.add_option(
        "--cutoff", dest="cutoff", type="eng_float", default=eng_rotator.num_to_arg(blocks)
        help="Set Cutoff Freq [default: firdes.WIN_HAMMING]")
    return parser

def main(top_block_cls=fosdem_intro, options=None):
    # options is None
    options = argument_parser().parse_args()

    tb = top_block_cls(cutoff=options.cutoff)
    tb.start()

    try:
        raise KeyboardInterrupt
    except EOFError:
        pass
    tb.stop()
    tb.wait()

if __name__ == "__main__":
    main()
```



Generated Python

```
def __init__(self, cutoff=8e3):
```

```
...
```

```
#####
```

```
# Connections
```

```
#####
```

```
self.connect((self.analog_noise_source_x_0, 0), (self.blocks_throttle_0, 0))
self.connect((self.blocks_throttle_0, 0), (self.low_pass_filter_0, 0))
self.connect((self.low_pass_filter_0, 0), (self.blocks_null_sink_0, 0))
```

```
def get_cutoff(self):
    return self.cutoff
```

```
def set_cutoff(self, cutoff):
    self.cutoff = cutoff
    self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, self.cutoff))
```

```
def get_samp_rate(self):
    return self.samp_rate
```

```
def set_samp_rate(self, samp_rate):
    self.samp_rate = samp_rate
    self.blocks_throttle_0.set_sample_rate(self.samp_rate)
    self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, self.cutoff))
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#####
# GNU Radio Python Flow Graph
# Title: FOSDEM intro
# Generated: Thu Jan 28 16:35:35 2016
#####

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_rotator
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser

class firdes_intro(gr_top_block):
    def __init__(self, cutoff=8e3):
        gr_top_block.__init__(self, "FOSDEM intro")

        # Parameters
        self.cutoff = cutoff

        # Variables
        self.samp_rate = samp_rate = 32000

        # Blocks
        self.low_pass_filter_0 = filter_fft_filter_cofl_firdes_low_pass(
            1, samp_rate, cutoff, firdes.WIN_HAMMING, 6, 70)
        self.blocks_throttle_0 = blocks_throttle_sizeof_gr_complex*1, samp_rate, True)
        self.blocks_null_sink_0 = blocks_null_sink(sizeof_gr_complex*1)
        self.analog_noise_source_x_0 = analog_noise_source_c(analog.GR_GAUSSIAN, 1, 0)

        # Connections
        self.connect((self.analog_noise_source_x_0, 0), (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.low_pass_filter_0, 0))
        self.connect((self.low_pass_filter_0, 0), (self.blocks_null_sink_0, 0))

    def get_cutoff(self):
        return self.cutoff

    def set_cutoff(self, cutoff):
        self.cutoff = cutoff
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, self.cutoff, 400, firdes.WIN_HAMMING, 6, 70))

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.blocks_throttle_0.set_sample_rate(self.samp_rate)
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate, self.cutoff, 400, firdes.WIN_HAMMING, 6, 70))

def argument_parser():
    parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
    parser.add_option(
        "--cutoff", dest="cutoff", type="eng_float", default=eng_rotator.num_to_arg(blocks),
        help="Set Cutoff Freq [default=8000.0]")
    return parser

def main(top_block_cls=firdes_intro, options=None):
    # options is None:
    options = argument_parser().parse_args()

    tb = top_block_cls(cutoff=options.cutoff)
    tb.start()

    try:
        raw_input("Press Enter to quit.")
    except EOFError:
        pass
    tb.stop()
    tb.wait()

if __name__ == "__main__":
    main()
```



Generated Python

```
def argument_parser():
    parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
    parser.add_option(
        "", "--cutoff", dest="cutoff", type="eng_float", default=eng_notation.num_to_
        help="Set Cutoff Freq [default=%default]")
    return parser
```

```
def main(top_block_cls=fosdem_intro, options=None):
    if options is None:
        options, _ = argument_parser().parse_args()
```

```
tb = top_block_cls(cutoff=options.cutoff)
tb.start()
try:
    raw_input('Press Enter to quit: ')
except EOFError:
    pass
tb.stop()
tb.wait()
```

```
if __name__ == '__main__':
    main()
```

```
#!usr/bin/python
# -*- coding: utf-8 -*-
# GNU Radio Python Flow Graph
# Title: FOSDEM intro
# Generated: Thu Jan 28 16:35:35 2016

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser

class fosdem_intro(gr_top_block):
    def __init__(self, cutoff=400):
        gr_top_block.__init__(self, "FOSDEM intro")

        # Parameters
        self.cutoff = cutoff

        # Variables
        self.samp_rate = samp_rate = 32000

        # Blocks
        self.low_pass_filter_0 = filter_fir_filter_ccff_1(firdes.low_pass(
            1, samp_rate, cutoff, 400, firdes.WIN_HAMMING, 6, 70))
        self.blocks_throttle_0 = blocks.throttle(gr_complex*1, samp_rate,True)
        self.blocks_null_sink_0 = blocks.null_sink(gr_complex*1)
        self.analog_noise_source_x_0 = analog.noise_source_c(analog.GR_GAUSSIAN, 1, 0)

        # Connections
        self.connect((self.analog_noise_source_x_0, 0), (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0), (self.low_pass_filter_0, 0))
        self.connect((self.low_pass_filter_0, 0), (self.blocks_null_sink_0, 0))

    def get_cutoff(self):
        return self.cutoff

    def set_cutoff(self, cutoff):
        self.cutoff = cutoff
        self.low_pass_filter_0.set_samp_rate(self.samp_rate, self.cutoff, 400, firdes.WIN_HAMMING, 6, 70)

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.blocks_throttle_0.set_sample_rate(self.samp_rate)
        self.low_pass_filter_0.set_samp_rate(self.samp_rate, self.cutoff, 400, firdes.WIN_HAMMING, 6, 70)

def argument_parser():
    parser = OptionParser(option_class=eng_option, usage="%prog: [options]")
    parser.add_option(
        "", "--cutoff", dest="cutoff", type="eng_float", default=eng_notation.num_to_
        help="Set Cutoff Freq [default=%default]")
    return parser

def main(top_block_cls=fosdem_intro, options=None):
    # options is None:
    options, _ = argument_parser().parse_args()

    tb = top_block_cls(cutoff=options.cutoff)
    tb.start()
    try:
        raw_input('Press Enter to quit: ')
    except EOFError:
        pass
    tb.stop()
    tb.wait()

if __name__ == '__main__':
    main()
```



How to add your own block

- *Block Wrapper/ XML*



- Metadata:

- name, category, docs, flags

- Parameters:

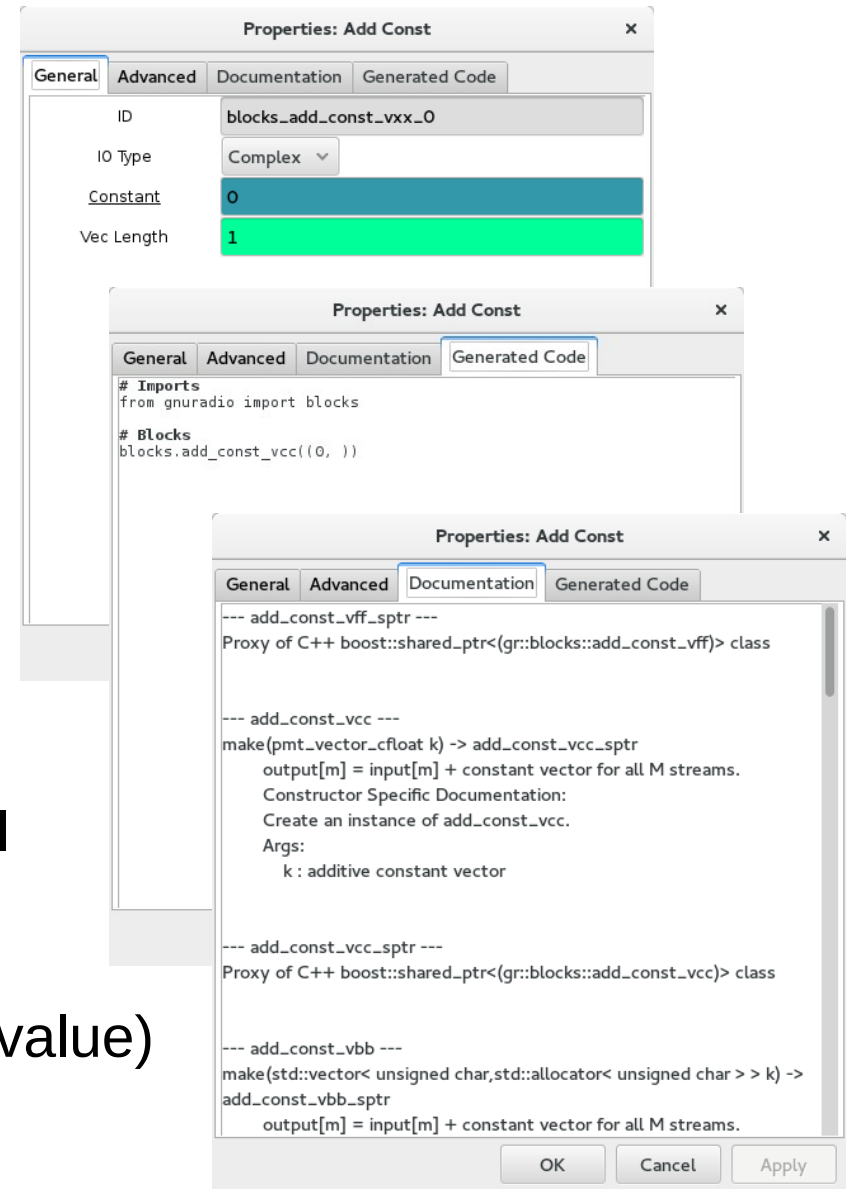
- key, name, type, default value, GUI

- Ports:

- key, name, type (vlen), domain, GUI

- Templates:

- imports, make, callbacks, (variable value)



The image shows three overlapping screenshots of the 'Properties: Add Const' dialog box in a software interface.

- Top Screenshot (General tab):** Shows the 'ID' field with the value 'blocks_add_const_vxx_0'. The 'IO Type' is set to 'Complex'. The 'Constant' field is set to '0' and is highlighted in blue. The 'Vec Length' field is set to '1' and is highlighted in green.
- Middle Screenshot (Generated Code tab):** Shows the generated C++ code. It includes an import statement for 'gnuradio::blocks' and a block definition: `blocks.add_const_vcc((0,))`.
- Bottom Screenshot (Documentation tab):** Shows the documentation for the block. It includes a proxy class definition for `add_const_vff_sptr`, a `make` function for `add_const_vcc_sptr` that takes a `pmt_vector_cfloat k` and returns an `add_const_vcc_sptr`. The documentation describes the output as `output[m] = input[m] + constant vector for all M streams.` and provides constructor-specific documentation: 'Create an instance of add_const_vcc. Args: k : additive constant vector'. It also shows a proxy class for `add_const_vcc_sptr` and a `make` function for `add_const_vbb_sptr` that takes a `std::vector< unsigned char, std::allocator< unsigned char > > k` and returns an `add_const_vbb_sptr`. The documentation for this function is also `output[m] = input[m] + constant vector for all M streams.`

How to add your own block

```

<block>
  <name>Add Const</name>
  <key>blocks_add_const_vxx</key>
  <import>from gnuradio import blocks</import>
  <make>blocks.add_const_v$(type.fcn)($const)</make>
  <callback>set_k($const)</callback>
  <param>
    <name>IO Type</name>
    <key>type</key>
    <type>enum</type>
    <option>
      <name>Complex</name>
      <key>complex</key>
      <opt>const_type:complex_vector</opt>
      <opt>fcn:cc</opt>
    </option>
    <option>
      <name>Float</name>
      <key>float</key>
      <opt>const_type:real_vector</opt>
      <opt>fcn:ff</opt>
    </option>
  </param>
  ....
  <param>
    <name>Constant</name>
    <key>const</key>
    <value>0</value>
    <type>$type.const_type</type>
  </param>
  <param>
    <name>Vec Length</name>
    <key>vlen</key>
    <value>1</value>
    <type>int</type>
  </param>
  <check>len($const) == $vlen</check>
  <check>$vlen > 0</check>
  <sink>
    <name>in</name>
    <type>$type</type>
    <vlen>$vlen</vlen>
  </sink>
  <source>
    <name>out</name>
    <type>$type</type>
    <vlen>$vlen</vlen>
  </source>
</block>

```



Overview

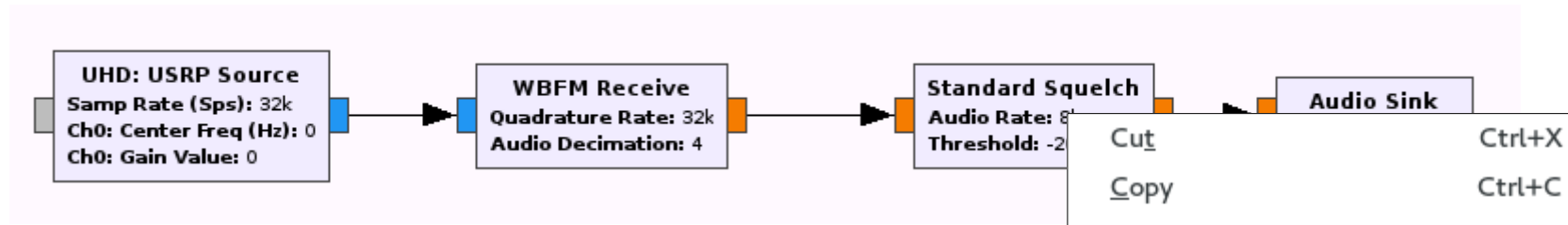
- GNU Radio Companion Intro
 - Graphical Flow Graph Design
 - How to add your own blocks

- Recently added features
 - Bypassed Blocks
 - Embedded Python Blocks / Modules
 - Custom Run Commands
 - Bootstrap depending hier_blocks

- Current development and plans for future versions

Bypassed Blocks

- Say you're streaming data through some blocks...



... and want to bypass one

- Works for
 - Single I/O of the same type
 - Can be explicitly disabled in XML

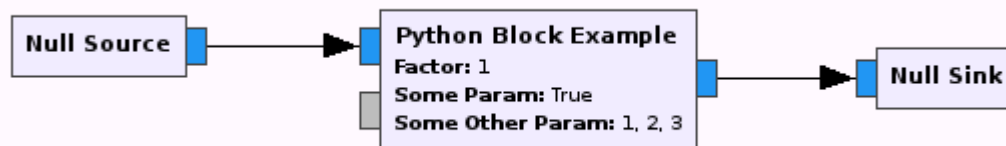
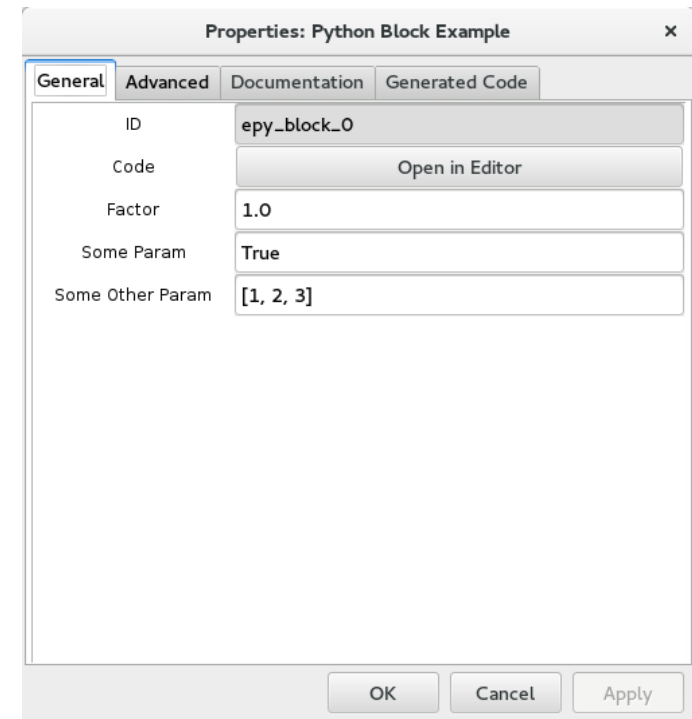
- Thanks to Seth Hitefield



Embedded Python Blocks

- Say you want ...
 - to quickly try something
 - a self-contained (tutorial) Flowgraph with custom DSP

- Python Blocks
 - Stored in the Flowgraph
 - Edited directly from GRC
 - Live, on-the-fly Block Wrappers



Embedded Python Blocks

```

Open ▾ [Icon] epy_block_0_JvUKuA.py Save [Menu] x
/tmp

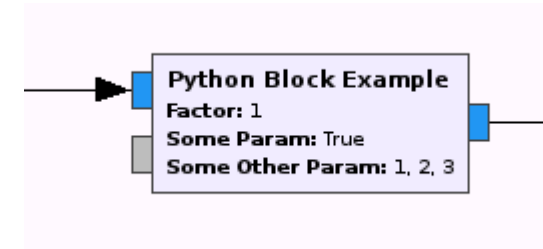
import numpy as np
from gnuradio import gr
import pmt

class blk(gr.sync_block):
    def __init__(self, factor=1.0, some_param=True,
                 some_other_param=[1,2,3]):
        gr.sync_block.__init__(
            self,
            name='Python Block Example',
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        self.factor = factor
        port_key = pmt.intern('control')
        self.message_port_register_in(port_key)
        self.set_msg_handler(port_key, self.handle_msg)

    def handle_msg(self, msg):
        pass

    def work(self, input_items, output_items):
        output_items[0][:] = input_items[0] * self.factor
        return len(output_items[0])

Python ▾ Tab Width: 8 ▾ Ln 16, Col 10 ▾ INS
  
```



Properties: Python Block Example

General | Advanced | Documentation | Generated Code

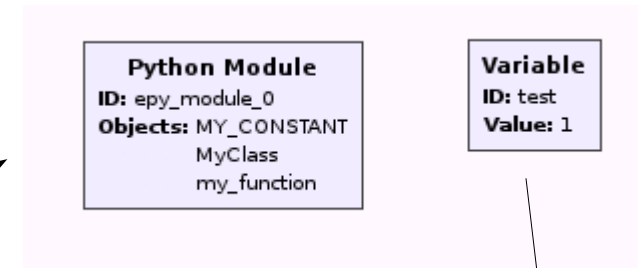
ID	epy_block_0
Code	Open in Editor
Factor	1.0
Some Param	True
Some Other Param	[1, 2, 3]

OK Cancel Apply



Embedded Python Modules

- Say you quickly want ...
 - to include longer python code
 - keep it in the GRC file



```

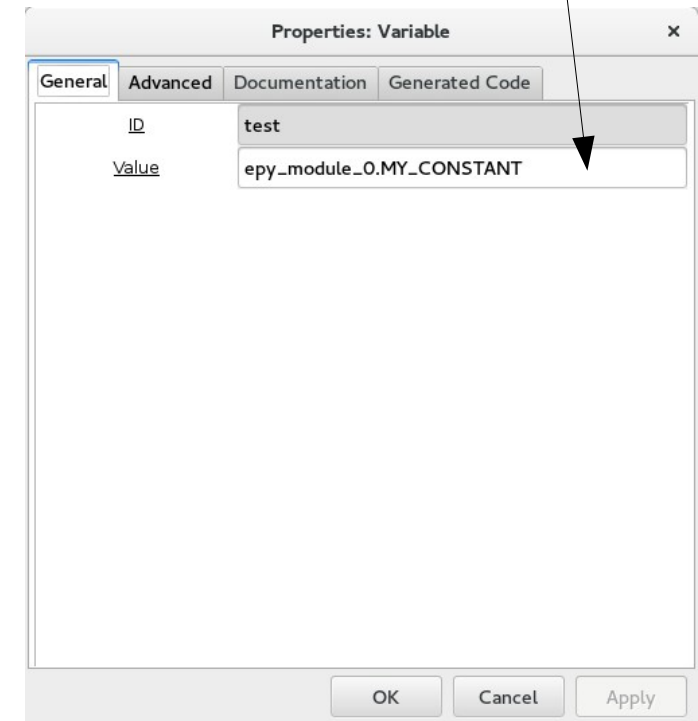
Open  [+  epy_module_0__CfRFA.py  Save  [≡]  x
/tmp
# this module will be imported in the into your flowgraph

MY_CONSTANT = 1

|
def my_function(a):
    pass

class MyClass(object):
    pass

Python  Tab Width: 8  Ln 5, Col 1  INS
  
```



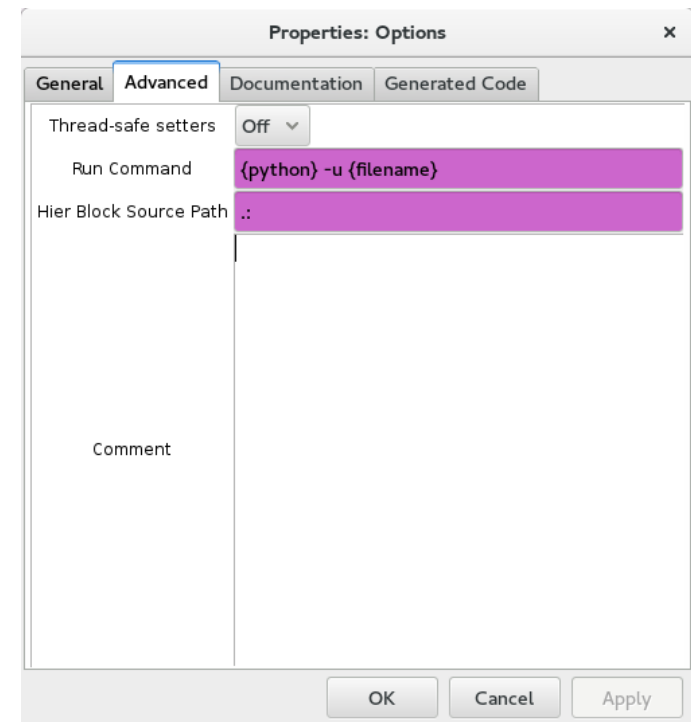
The screenshot shows the 'Properties: Variable' dialog box with the 'General' tab selected. The 'ID' field contains 'test' and the 'Value' field contains 'epy_module_0.MY_CONSTANT'. An arrow from the variable box in the diagram above points to the 'Value' field in this dialog.



Custom Run Commands

- Say you want ...
 - to modify/extend/embedded your Flowgraph and still run in from GRC
 - deploy and execute it remotely

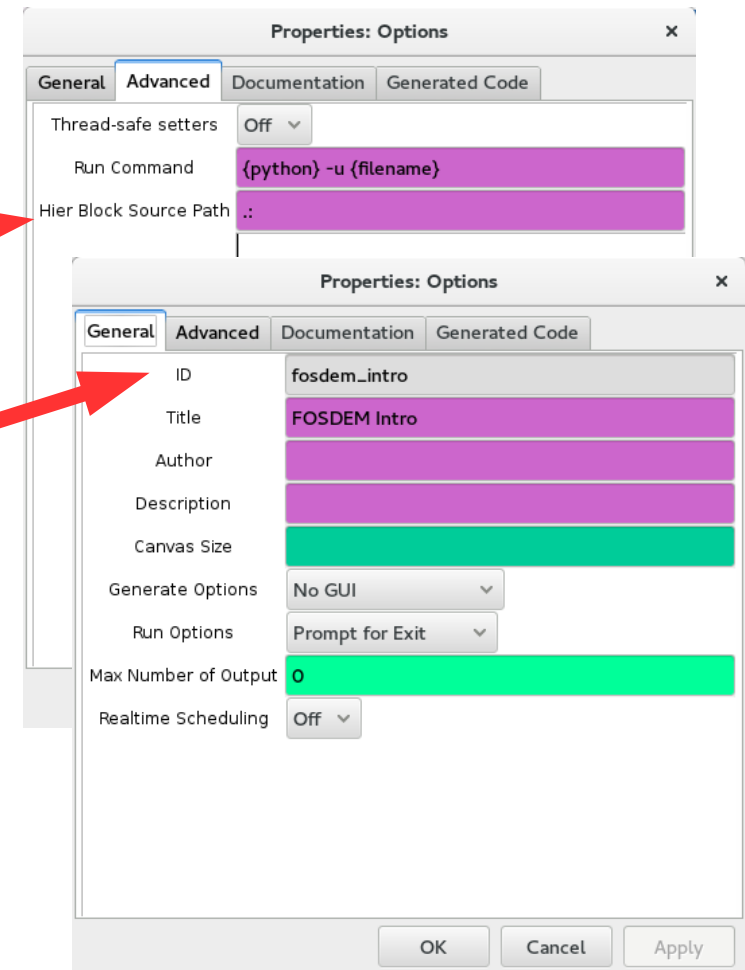
- You can have GRC run any command
 - Oh oh...
 - import the generated code in new model and reuse/extend the
 - top_block, main, arg_parser
 - Have some script run SCP and SSH for remote execution



Bootstrap depending hier_blocks

- Say you distribute a OOT module including a Flowgraph which depends on other GRC generated hier_blocks
 - GRC will auto-generate these now if
 - it knows where to look
 - “.” is default, multiple allowed
 - the Flowgraph ID and filename match

- Of course, embedded hierarchical blocks would be nicer...



Overview

- GNU Radio Companion Intro
 - Graphical Flow Graph Design
 - How to add your own blocks

- Recently added features
 - Bypassed Blocks
 - Embedded Python Blocks / Modules
 - Custom Run Commands
 - Bootstrap depending hier_blocks

- **Current development and plans for future versions**

Current development and future versions

- Long term goals
 - Switch to a QT-based GUI
 - Rewrite/clean/modularize the core
 - get rid of quick fixes, endless special cases and side-effects
 - ease entry for new contributors
 - Switch to Python3
 - Replace XML-based Block Wrappers

- Want to help? Join the GRC Working Group!

The end