Stream Tags, Message Passing, and PDUs

Tom Rondeau

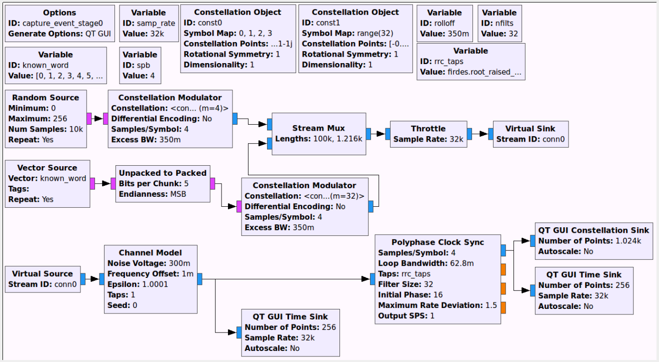
(tom@trondeau.com)

2015-08-24

Introduction

Data Streaming Model

The tried-and-true data streaming system of GNU Radio



Options
ID: capture_event_stage0
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

Constellation Object
ID: const0
Symbol Map: 0, 1, 2, 3
Constellation Points: ...1-1j
Rotational Symmetry: 1
Dimensionality: 1

Constellation Object
ID: const1
Symbol Map: range(32)
Constellation Points: [-0....
Rotational Symmetry: 1
Dimensionality: 1

Variable
ID: rolloff
Value: 350m

Variable
ID: nfilts
Value: 32

Variable
ID: known_word
Value: [0, 1, 2, 3, 4, 5, ...

Variable
ID: spb
Value: 4

Variable
ID: rrc_taps
Value: firdes.root_raised_...

Random Source
Minimum: 0
Maximum: 256
Num Samples: 10k
Repeat: Yes

Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: No
Samples/Symbol: 4
Excess BW: 350m

Stream Mux
Lengths: 100k, 1.216k

Throttle
Sample Rate: 32k

Virtual Sink
Stream ID: conn0

Vector Source
Vector: known_word
Tags:
Repeat: Yes

Unpacked to Packed
Bits per Chunk: 5
Endianness: MSB

Constellation Modulator
Constellation: <con...(m=32)>
Differential Encoding: No
Samples/Symbol: 4
Excess BW: 350m

Virtual Source
Stream ID: conn0

Channel Model
Noise Voltage: 300m
Frequency Offset: 1m
Epsilon: 1.0001
Taps: 1
Seed: 0

QT GUI Time Sink
Number of Points: 256
Sample Rate: 32k
Autoscale: No

Polyphase Clock Sync
Samples/Symbol: 4
Loop Bandwidth: 62.8m
Taps: rrc_taps
Filter Size: 32
Initial Phase: 16
Maximum Rate Deviation: 1.5
Output SPS: 1

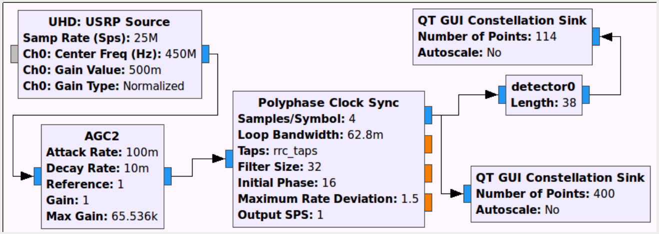
QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

QT GUI Time Sink
Number of Points: 256
Sample Rate: 32k
Autoscale: No

- Data/samples flow downstream from source to sink

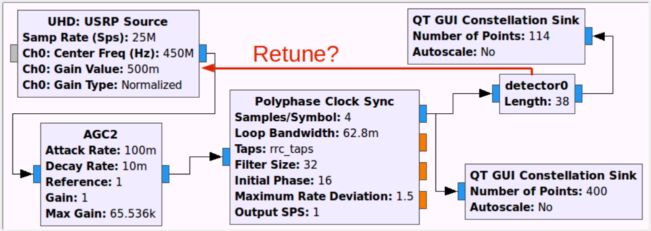
Weaknesses of the data streaming model

- Everything flows downstream.
  - No loops!
- No event signaling between blocks.
  - Annotate samples with info/meta-data
  - Used by other blocks to change behavior
  - Or use to recall/replay events



Command and control

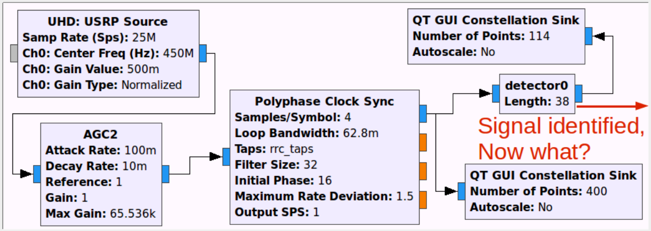
- Signalling upstream events



- What if the detector wanted to retune RFE after finding a signal?

Command and control

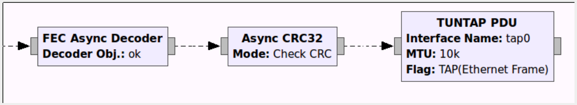
- Signal downstream an event has occurred.



- Preamble detected; tell downstream exactly where.

Working with a full unit of data

- We need to operate on a packet/frame/protocol unit
  - PDU: protocol data unit
- Passed as messages

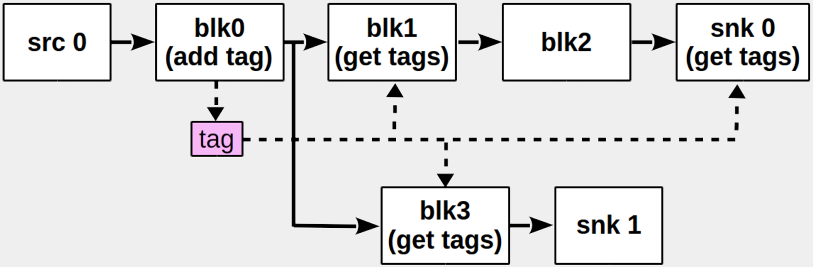


- Example message connections in GRC

Tag Stream Layer

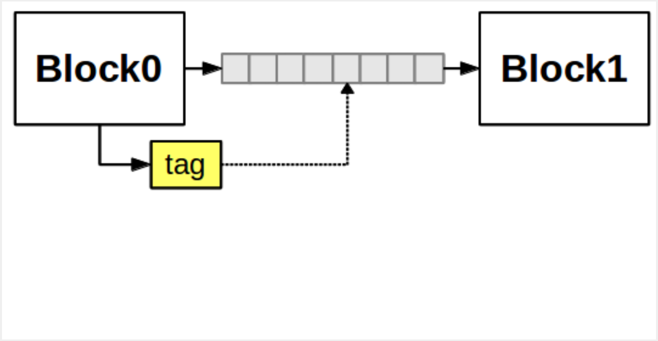
Stream tag layer

Adds a control/logic/synchronous message interface to the data flow layer.



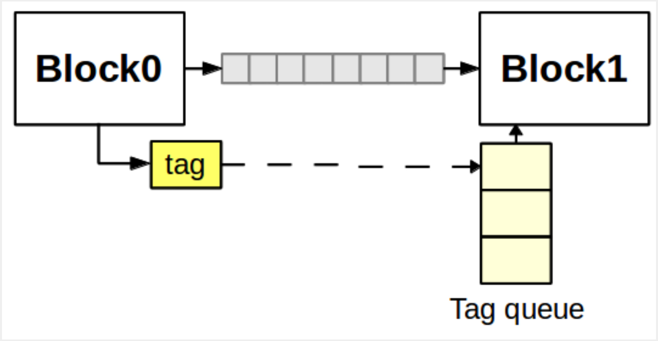
Add Tags to Stream (see: add_item_tag)

Adds a new tag at a specific item.



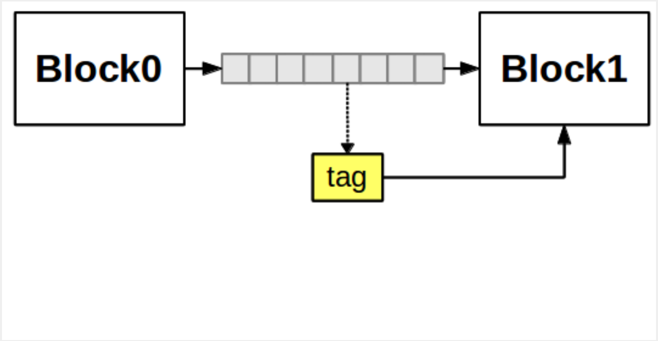
Add Tags to Stream (see: add_item_tag)

Actually, pushes on to next block tag queue.



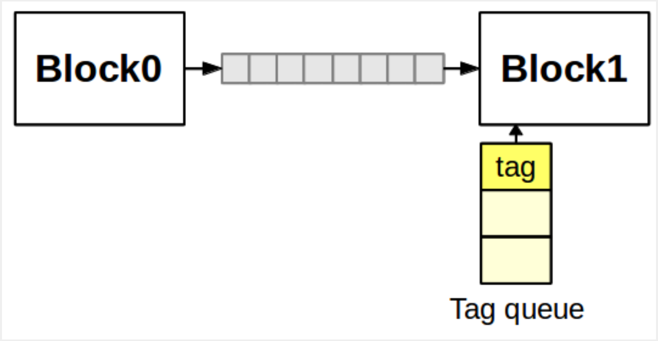
Get Tags (see: get_tags_in_range and get_tags_in_window)

Now, a block needs to get a tag on its input stream.

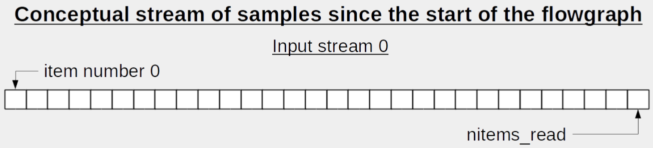


Get Tags (see: get_tags_in_range and get_tags_in_window)

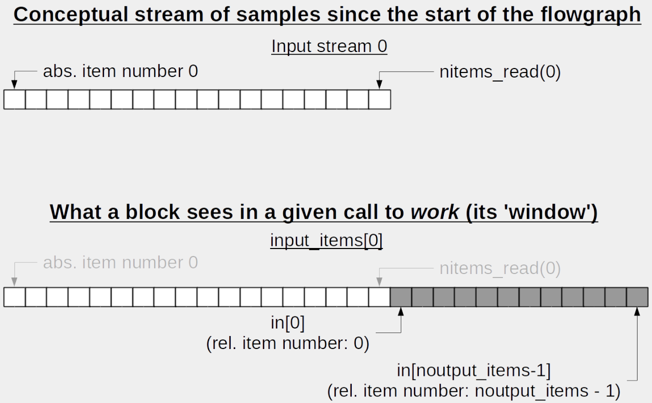
Actually, pulls from its own input queue.



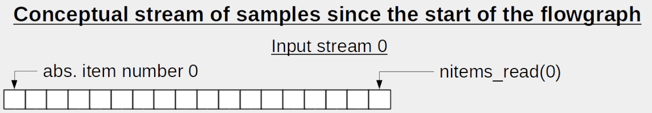
Buffer Aids: using add_item_tag

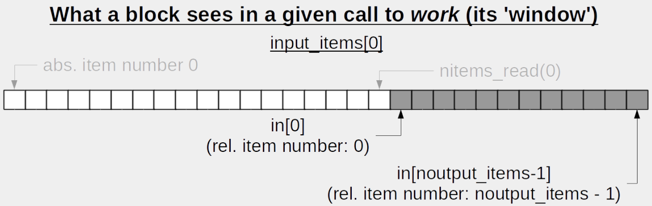

Conceptual stream of samples since the start of the flowgraph
Input stream 0
item number 0
nitems_read

Buffer Aids: using get_tags_in_{range, window}


Conceptual stream of samples since the start of the flowgraph
Input stream 0
abs. item number 0
nitems_read(0)
What a block sees in a given call to *work* (its 'window')
input_items[0]
abs. item number 0
nitems_read(0)
in[0]
(rel. item number: 0)
in[noutput_items-1]
(rel. item number: noutput_items - 1)

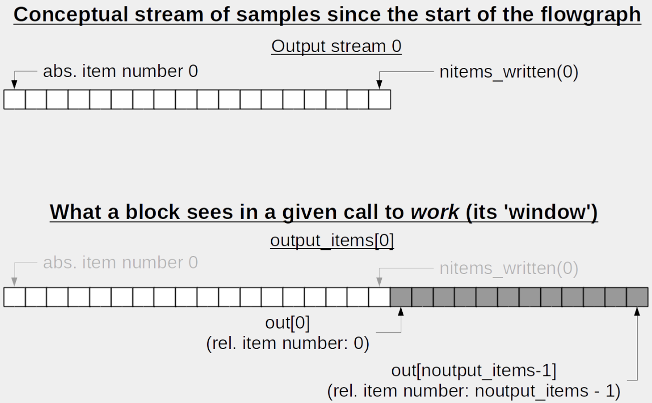
Buffer Aids: using get_tags_in_{range, window}


Conceptual stream of samples since the start of the flowgraph
Input stream 0
abs. item number 0
nitems_read(0)

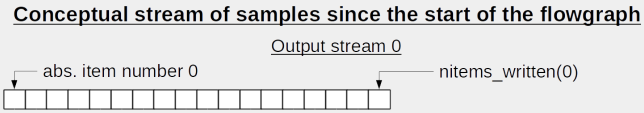

What a block sees in a given call to *work* (its 'window')
input_items[0]
abs. item number 0
nitems_read(0)
in[0]
(rel. item number: 0)
in[noutput_items-1]
(rel. item number: noutput_items - 1)

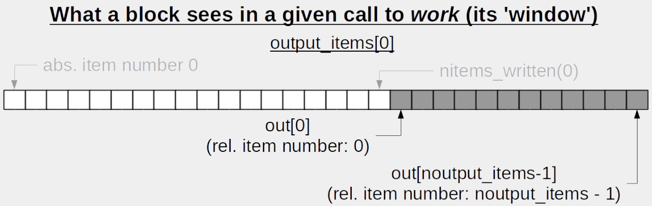
get_tags_in_range using the absolute offset
get_tags_in_window using the relative offset

Buffer Aids: using add_item_tag


Conceptual stream of samples since the start of the flowgraph
Output stream 0
abs. item number 0
nitems_written(0)
What a block sees in a given call to *work* (its 'window')
output_items[0]
abs. item number 0
nitems_written(0)
out[0]
(rel. item number: 0)
out[noutput_items-1]
(rel. item number: noutput_items - 1)

Buffer Aids: using add_item_tag


Conceptual stream of samples since the start of the flowgraph
Output stream 0
abs. item number 0
nitems_written(0)



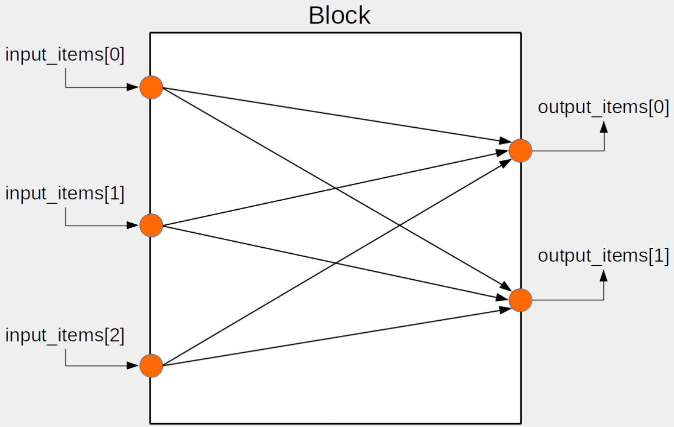
add_item_tag using the absolute offset

often: nitems_read(0) + i

where i indexes a for-loop

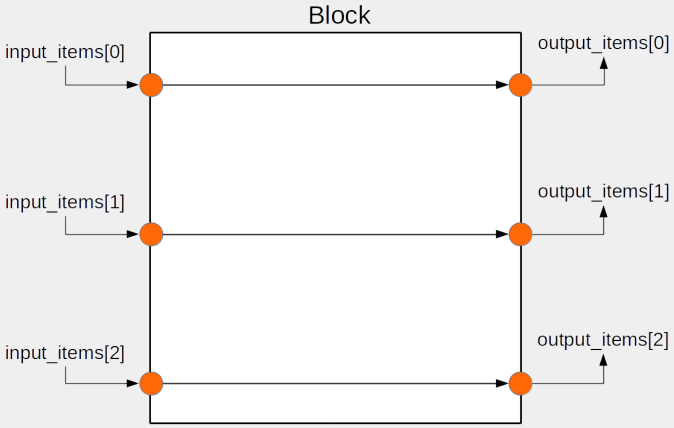
Tag Propagation Policies

Tag Propagation Policy: All – to – All
(TPP_ALL_TO_ALL)

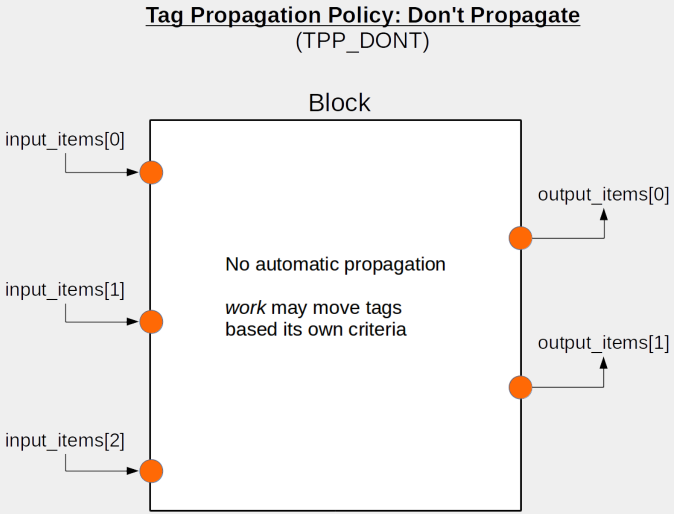


Tag Propagation Policies

Tag Propagation Policy: One – to – One
(TPP_ONE_TO_ONE)



Tag Propagation Policies



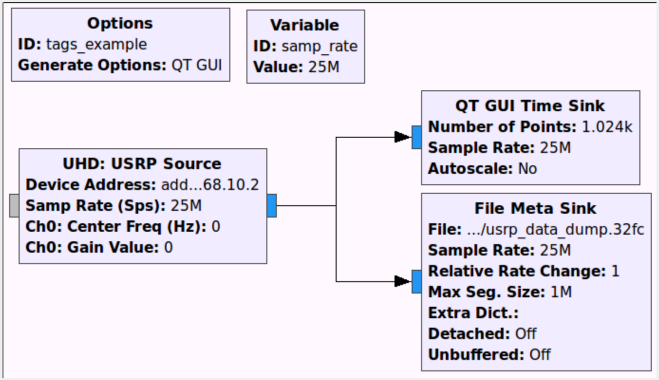
Tags Through Rate Changes

All blocks have a relative_rate()

- **gr::sync_block**: 1.0
- **gr::sync_decimator**: 1.0/decim
- **gr::sync_interpolator**: (float)interp
- **gr::block**: must call set_relative_rate (defaults to 1.0)

Tags Example

USRPs emit tags when rate and frequency change



Options
ID: tags_example
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 25M

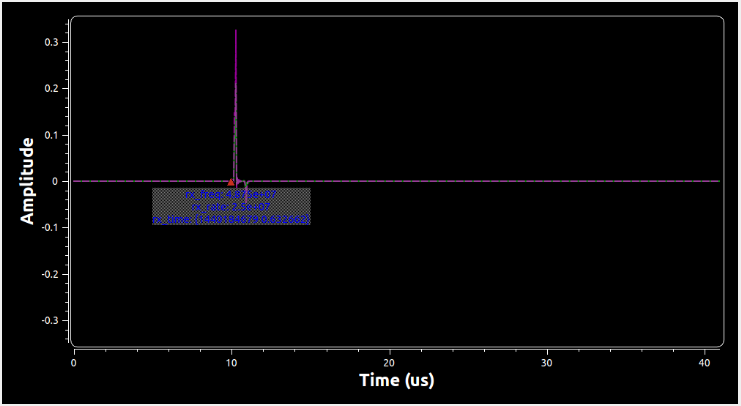
UHD: USRP Source
Device Address: add...68.10.2
Samp Rate (Sps): 25M
Ch0: Center Freq (Hz): 0
Ch0: Gain Value: 0

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 25M
Autoscale: No

File Meta Sink
File: .../usrp_data_dump.32fc
Sample Rate: 25M
Relative Rate Change: 1
Max Seg. Size: 1M
Extra Dict.:
Detached: Off
Unbuffered: Off

Tags Example

USRPs emit tags when rate and frequency change

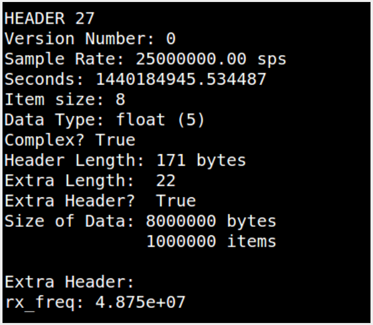


rx_freq: 4.875e+07
rx_rate: 2.5e+07
rx_time: {1440184679 0.632662}

- QTGUI Time Sink can trigger off a tag name

Tags Example

File Meta Sink keeps metadata in headers



```
HEADER 27
Version Number: 0
Sample Rate: 25000000.00 sps
Seconds: 1440184945.534487
Item size: 8
Data Type: float (5)
Complex? True
Header Length: 171 bytes
Extra Length:  22
Extra Header?  True
Size of Data: 8000000 bytes
              1000000 items

Extra Header:
rx_freq: 4.875e+07
```

- gr_read_file_metadata to extract and print
- Python tool: parse_file_metadata
- gnuradio.org/doc/doxygen/page_metadata.html

Message Passing Layer

Messages Intro

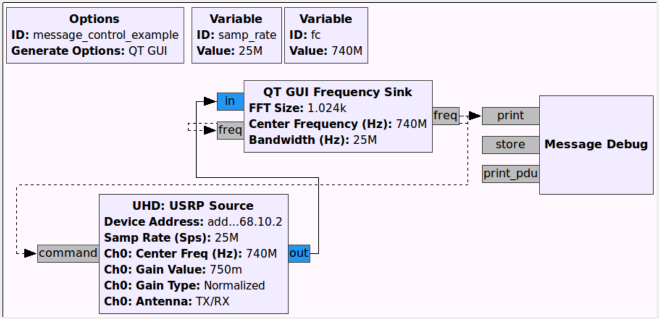
- What are messages and how do they move?
  - pub/sub concept
  - Publish once, many possible subscribers
  - Subscriber can receiver from multiple publishers
  - Thread safety issues

Messages

- A block publishes a message.
  - typically, but not necessarily, in its work function.
- Other blocks subscribe to the publisher.
- Like data streams, we connect them.
  - tb.msg_connect(publisher, “pub port name”, subscriber, “sub port name”)
- Subscriber has messages pushed onto its message queue.
- Block will check the queue and fire an appropriate message handler function to deal with the message.
- Message are Polymorphic Types (PMTs) and can contain anything.
  - commonly vectors of bytes for PDUs
  - Or a dictionary (Key: value pair)

Message Example

QTGUI Frequency Sink can emit messages when double-clicked


Options
ID: message_control_example
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 25M

Variable
ID: fc
Value: 740M

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 740M
Bandwidth (Hz): 25M

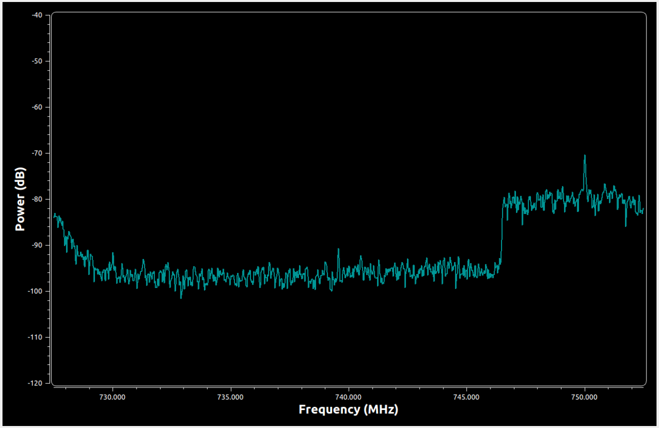
Message Debug

UHD: USRP Source
Device Address: add...68.10.2
Samp Rate (Sps): 25M
Ch0: Center Freq (Hz): 740M
Ch0: Gain Value: 750m
Ch0: Gain Type: Normalized
Ch0: Antenna: TX/RX

- USRPs take message control commands to adjust parameters.
- Frequency sink takes same style control to adjust x-axis.
- Message debug to see output messages.

Message Example

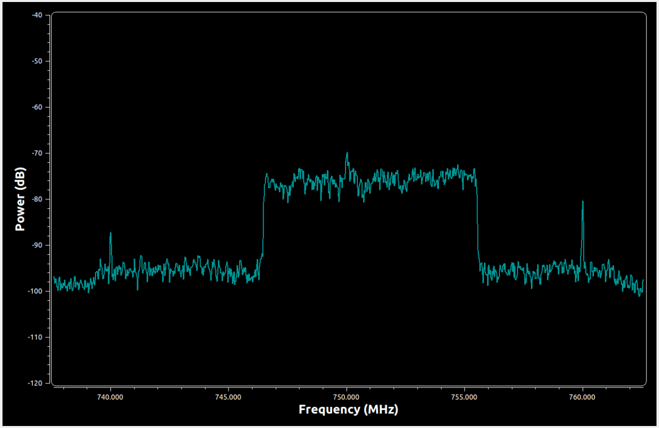
Started with USRP tuned to 740 MHz



- Double-click on right peak.

Message Example

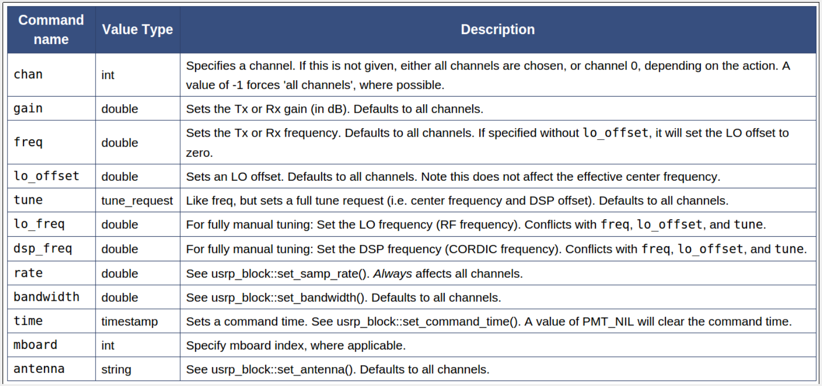
Retuned USRP to 750 MHz



- X-axis now recentered at 750 MHz as well.

UHD Command Messages

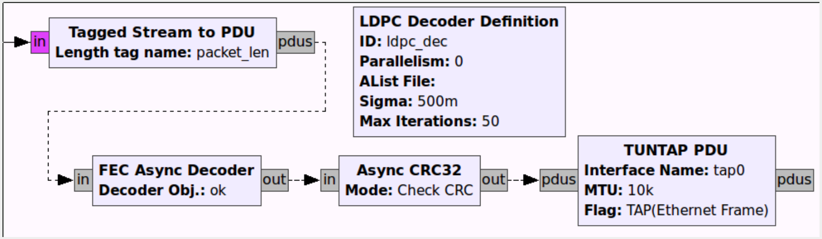
See UHD Interface page in the GNU Radio Manual



Command name	Value Type	Description
`chan`	int	Specifies a channel. If this is not given, either all channels are chosen, or channel 0, depending on the action. A value of -1 forces 'all channels', where possible.
`gain`	double	Sets the Tx or Rx gain (in dB). Defaults to all channels.
`freq`	double	Sets the Tx or Rx frequency. Defaults to all channels. If specified without `lo_offset`, it will set the LO offset to zero.
`lo_offset`	double	Sets an LO offset. Defaults to all channels. Note this does not affect the effective center frequency.
`tune`	tune_request	Like freq, but sets a full tune request (i.e. center frequency and DSP offset). Defaults to all channels.
`lo_freq`	double	For fully manual tuning: Set the LO frequency (RF frequency). Conflicts with `freq`, `lo_offset`, and `tune`.
`dsp_freq`	double	For fully manual tuning: Set the DSP frequency (CORDIC frequency). Conflicts with `freq`, `lo_offset`, and `tune`.
`rate`	double	See usrp_block::set_samp_rate(). *Always* affects all channels.
`bandwidth`	double	See usrp_block::set_bandwidth(). Defaults to all channels.
`time`	timestamp	Sets a command time. See usrp_block::set_command_time(). A value of PMT_NIL will clear the command time.
`mboard`	int	Specify mboard index, where applicable.
`antenna`	string	See usrp_block::set_antenna(). Defaults to all channels.

Moving from Streaming to Message Passing

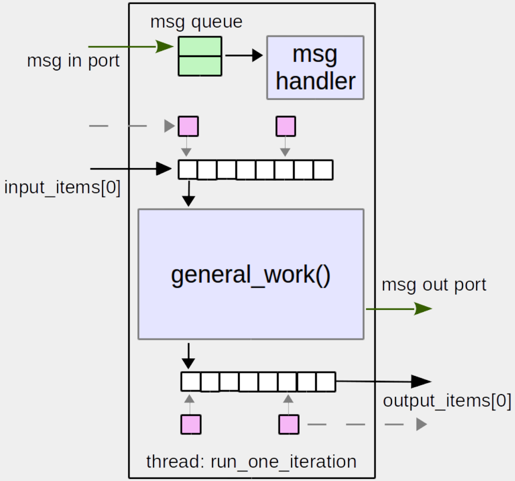
Streaming mode closer to the antenna and into PDUs



- Tagged stream carries PDU length info with it.
- Create a tagged stream by declaring a "length name tag" and issuing the length of the PDU as a tag at the start.

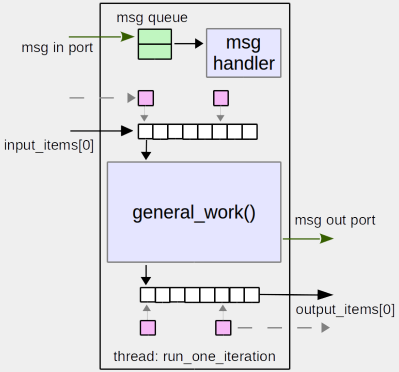
Conclusions

Overview of Data Movement Models in GNU Radio



Review of Data Movement Models in GNU Radio

- Message handlers are called by the scheduler
  - In same thread as work
  - Makes operations in the two inherently thread safe
- Tags operated on in work
  - based on item offsets
  - makes no sense to think of offsets outside work


msg queue
msg in port
msg handler
input_items[0]
general_work()
msg out port
output_items[0]
thread: run_one_iteration