AN SDR-BASED FRS TRANSCEIVER

PROTOTYPING A MULTI-CHANNEL FRS TRANSCEIVER WITH CTCSS SUPPORT VIA GNU RADIO AND THE NUAND BLADERF

JUNE 21, 2016

License

Authors

Robert Ghilduta
<robert.ghilduta@nuand.com>
Nuand, LLC

Brian Padalino
<brian.padalino@nuand.com>
Nuand, LLC

Jon Szymaniak
<jon.szymaniak@nuand.com>
Nuand, LLC

Contributing Authors

Dr. Juan R. Pimentel
<jpimente@kettering.edu>
Professor of Computer Engineering
Kettering University

Revisions

Comments, feedback, improvements, and fixes may be sent to <bladeRF@nuand.com>.

Revision	Date	Summary
1	2015-06-12	Initial public draft
2	2015-10-25	Incorporated feedback and revisions from Dr. Juan R. Pimentel
3	2016-01-07	Fixed incorrect frequency values in Figure 1
4	2016-06-21	Typo fix in Section 5.4

Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

CTCSS	Continuous Tone-Coded Squelch System
DAC	Digital to Analog Converter
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FRS	Family Radio Service
IIR	Infinite Impulse Response
GRC	GNU Radio Companion
GUI	Graphical User Interface
LNA	Low Noise Amplifier
LPF	Low Pass Filter
RX	Receive
SINAD	Signal-to-Noise and Distoration ratio
SDR	Software Defined Radio
TX	Transmit
VGA	Variable Gain Amplifier
VSA	Vector Signal Analyzer

Intended Audience

The intended audience for this document is someone with a basic understanding of frequency domain analysis, sampling theory, digital filtering, and analog modulation techniques. Prior experience with GNU Radio, GNU Radio Companion (GRC), Software Defined Radio (SDR), and MATLAB is also beneficial. This paper is likely to be useful for SDR users looking for examples of:

- Narrow-band FM transmitters and receivers
- Filter design using software tools
- Usage of GNU Radio's polyphase channelizer and synthesizer blocks
- Full-duplex operation of the bladeRF

Abstract

The Nuand bladeRF is a USB 3.0 SDR featuring full-duplex operation and configurable filter bandwidths that can cover up to 28 MHz. These properties are leveraged to design and implement a real-time Family Radio Service (FRS) transceiver capable of simultaneously transmitting and/or receiving on any subset of its 14 channels, with optional Continuous Tone-Coded Squelch System (CTCSS) support. The open source GNU Radio framework and the GRC tool are used to realize the transceiver design as a graphical flowgraph and to create an interactive GUI interface for the transceiver. This paper presents both a high-level description of the design, as well as a detailed discussion of the GNU Radio blocks utilized and the rationale for their associated parameter values. In doing so, considerations for targeting the bladeRF and accounting for real-world nonidealties are included. Lastly, the approaches for testing and evaluation the design using RF test equipment and FRS handhelds is presented, along with results.

Overview of the Family Radio Service

FRS is a private, 14-channel, two-way data communications service intended for short-distance voice and data communications [1]. Rules governing this service are defined in Title 47, Part 95B [2].

The frequency allocations for each of the 14 channels are listed below. Each channel is allocated 12.5 kHz of bandwidth, and the max deviation of the narrow band FM signals used on these channels is 2.5kHz [2].

Table 1: FRS frequency allocations

Channel	**Frequency (MHz)**	**Channel**	**Frequency (MHz)**
1	462.5625	8	467.5625
2	462.5875	9	467.5875
3	462.6125	10	467.6125
4	462.6375	11	467.6375
5	462.6625	12	467.6624
6	462.6875	13	467.6875
7	462.7125	14	467.7125

Many handheld FRS radios include CTCSS support[1], in which receivers require the presence of a sub-audible tone in order to break squelch. Table 2 presents commonly used frequencies for these sub-audible tones. Note that these CTCSS tones are not part of the FRS standard, but an industry-created and adopted method for co-channel use between multiple parties.

Table 2: Common CTCSS tone frequencies, in Hz

67.0	71.9	74.4	77.0	79.7	82.5	85.4	88.5
91.5	94.8	97.4	100.0	103.5	107.2	110.9	114.8
118.8	123.0	127.3	131.8	136.5	141.3	146.2	151.4
156.7	162.2	167.9	173.8	179.9	186.2	192.8	203.5
210.7	218.1	225.7	233.6	241.8	250.3		

[1] CTCSS support is often marketed as "sub-codes" or "privacy codes."

Transceiver Design

The transceiver presented in this document is designed to simultaneously receive and transmit on any subset of the 14 FRS channels using GNU Radio [3] and the Nuand bladeRF SDR [4]. This section first presents some of the driving factors of the design, followed by a detailed description of both the Transmit (TX) and Receive (RX) portions of the flowgraph. It is recommend that one have the flowgraph (or the screenshots in Section 8) readily accessible while reading this section.

Although the individual FRS channels have a very small bandwidth, there is a 4.85 MHz separation between channels 7 and 8. Instead of re-tuning the transceiver to the desired frequency, the bladeRF is tuned to an intermediate frequency and its bandwidth and sample rate are configured to cover the entire FRS spectrum. As the bladeRF is capable of operating on up to 28 MHz of bandwidth, this does not pose a significant challenge.

This approach has the following advantages:

- No time is spent re-tuning or sweeping across channels.
- Calibration only needs to be performed for the intermediate frequencies used.

The tuning of the RX and TX channels in the context of the FRS spectrum is presented in Figure 1. Note that both the TX and RX center frequencies are not in the "center" of the FRS spectrum, as also shown in Figure 2(a). If the transceiver were tuned directly to the "center" of the FRS spectrum, any IQ imbalance that remains after calibration could cause mirror images of channels 1-7 to interfere with channels 8-14, and vice versa. This situation is depicted in Figure 2(b). To avoid this, the RX and TX channels are tuned at 200 kHz offsets from the "center" of the FRS spectrum, which is illustrated in Figure 2(c).

It is important to be aware of some trade-offs with this design, which must be be kept in mind during the implementation and evaluation of the transceiver:

1. Using over 5 MHz of bandwidth concurrently on TX and RX will exceed USB 2.0 capabilities. A USB 3.0 connection is required.
2. By not tuning hardware to one of the FRS channels, any LO leakage from the transmitter may interfere with other signals.
3. The offset tuning approach places any IQ images out-of-band, which may interfere with neighboring signals.

We argue that #1 is reasonable, given the ever increasing availability of USB 3.x controllers on modern systems.

Trade-offs #2 and #3 include interference and out-of-band emissions and are therefore serious concerns. Fortunately, both DC offset and IQ imbalance can be compensated for using

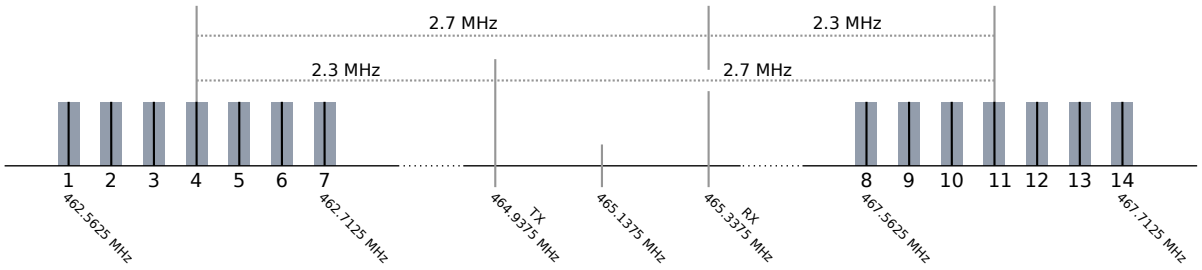
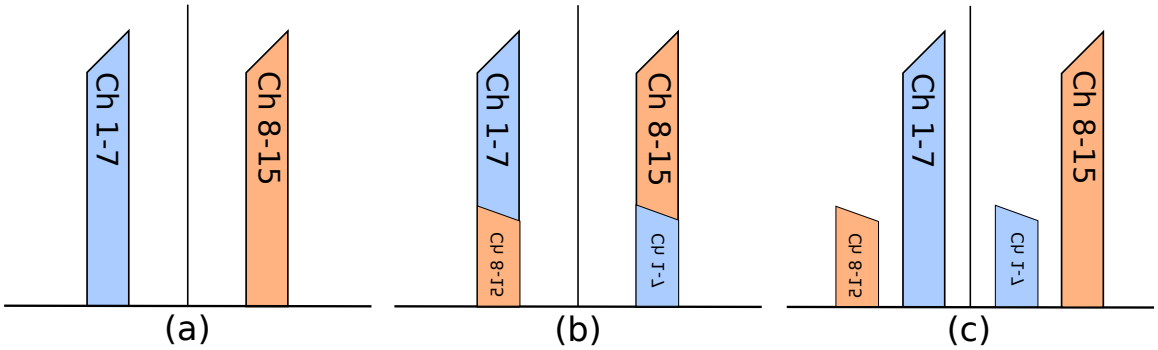


Figure 1: Transceiver tuning with respect to FRS channel allocations



(a) Nominal case: Tuning to "center" and no IQ imbalance.
(b) Realistic case: IQ imbalance yields mirror images interfering with signals.
(c) Offset tuning to ensure IQ imbalance images do not overlap with signals.

Figure 2: Avoiding effect of IQ imbalance via offset tuning

hardware features of the bladeRF. In evaluating this design, these signals will be important to measure and take note of.

In designing with this approach, we must first determine how the transmitter and receiver will utilize the spectrum. Given the 25 kHz separation between channels and each channel's 12.5 kHz of bandwidth, this implies that the bladeRF needs to be configured for a minimum bandwidth defined by the following:

$$\begin{aligned} \mathrm{BW}_{\mathrm{TX,\,min}} &= 2 \times (\mathrm{Max}\,(|F_{TX} - F_{CH1}|, |F_{TX} - F_{CH14}|) + 6.25\,\mathrm{kHZ}) \\ &= 2 \times (\mathrm{Max}\,(|464.9375 - 462.5625|, |464.9375 - 467.7125|)\ \mathrm{MHz} + 6.25\,\mathrm{kHZ}) \\ &= 5.5625\,\mathrm{MHz} \end{aligned}$$

The equation for $\mathrm{BW}_{\mathrm{RX,\,min}}$ is similar, but instead uses an ($F_{RX} = 465.1175\,\mathrm{MHz}$) term.

The Lime Microsystems LMS6002D transceiver [5] on the bladeRF has discrete TX Low Pass Filter (LPF) bandwidth settings, with the next largest value being 6.0 MHz in quadrature,

which passes $[-3\,\text{MHz}, 3\,\text{MHz}]$ at baseband. The rejection of the LMS6002D 3 MHz filter exceeds 50 dB at approximately $\pm$ 4 MHz [6]. On RX, this filter is used as an anti-aliasing filter to ensure incoming signals outside of the sampling bandwidth are sufficiently attenuated before being converted by the ADC. On TX, this filter is used to remove the images created due to the zero-order hold nature of the DAC. Therefore, to avoid these artifacts, an 8 MHz sample rate is used by the TX and RX modules.

These sample rate and bandwidth calculations provide constraints which guide the design of the TX and RX paths detailed in the following subsections.

MATLAB's `fdatool`, shown in Figure 3 is used to design the digital filters presented in this document. However, GNU Radio's `gr_filter_design` program and/or Octave could also be used.

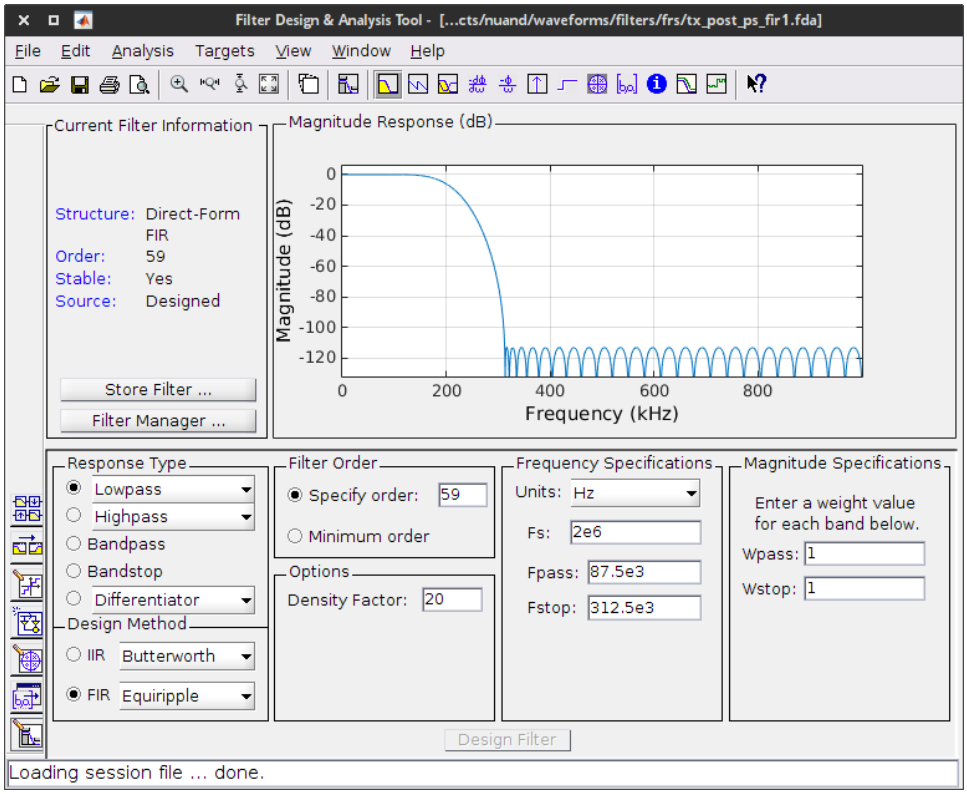


Figure 3: MATLAB Filter Design and Analysis Tool

Transmit Path

Audio Input

At the beginning of the TX path, audio enters through an `Audio Source` block. Through the use of the "pulse" argument provided to this block, the audio from the host machine's PulseAudio [7] subsystem is used. A 25 kHz sample rate is selected since it shares all common factors with the final DAC output sample rate of 8 MHz. This alleviates the need for digital resampling in the flowgraph prior to entering a downstream `Polyphase Synthesizer` block.

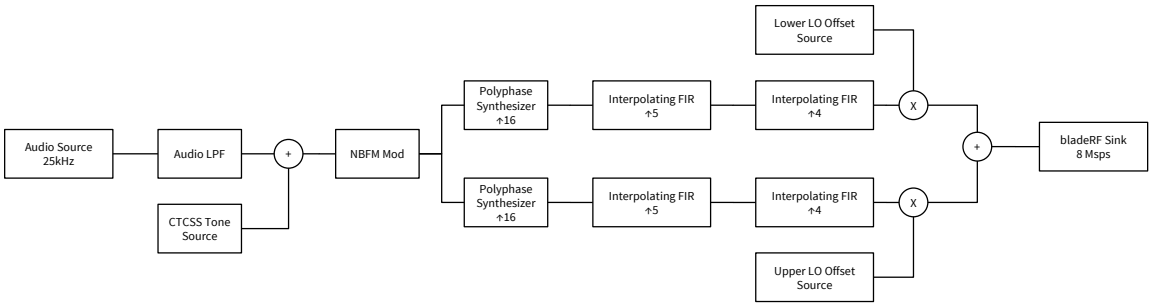


Figure 4: FRS TX block diagram

A `Valve` block is used to start and stop the flow of samples to the downstream portions of the flowgraph. This value is "closed" when the user presses the "Push to Talk" Graphical User Interface (GUI) button or checks the "Continuous TX" checkbox. When the valve is opened, downstream blocks are not processing samples which saves CPU cycles when the transmit functionality is not enabled. When no channels are selected to transmit, the bladeRF's TX front end is turned off. This helps guarantee no residual carrier is being transmitted when there is nothing to transmit.

A `Finite Impulse Response (FIR) Filter` block is used to realize an LPF to limit the audio frequency response to 3.125 kHz [2]. Using `fdatool` and the parameters shown in Table 3, a filter whose magnitude response is shown in Figure 5 is created. From this filter's magnitude response, is is clear that frequencies above the 3.125 kHz limit are attenuated by 60 dB, while the vast majority of the human vocal range is passed.

Optional CTCSS Tone

Next, a `Selector` block is used to pass either the audio, or the audio added with a sub-audible CTCSS tone supplied by a `Signal Source` block, to the next stage. The frequency

Table 3: Filter parameters of TX audio LPF

Design Method	FIR: Equiripple
Density Factor	20
Order	74
Sampling Frequency (kHz)	25
Passband Frequency (kHz)	2
Stopband Frequency (kHz)	3.125
Passband Weight	1
Stopband Weight	1

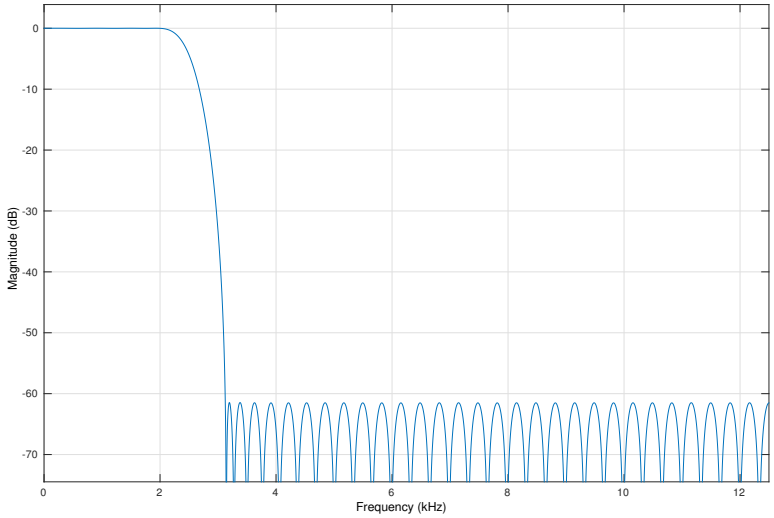


Figure 5: Magnitude response of TX audio LPF

of the CTCSS tone is controlled by a drop-down widget in the GUI. When "None" is selected, the selector passes the audio from the Audio Source block. Otherwise, the `Signal Source` block is configured for the desired CTCSS frequency, and the `Selector` passes the combination of the audio and this tone.

FM Modulation and Mute Blocks

The output of the `Selector` is fed into the `NBFM Transmit` block. There is no need to interpolate here, so the *Quadrature Rate* is left the same as the sample rate of the incoming audio.

No pre-emphasis is desired. As of GNU Radio commit `ee369b92` (2015-03-04), the `NBFM Transmitter` block does not currently implement pre-emphasis[2], so the *tau* parameter is

[2]The IIR filter is constructed with hard-coded feed-forward and feed-back taps set to 1.0.

unused. However, this *tau* value is used simply to match the value used on the RX side of the flowgraph, should this change in the future. The rationale for this value is discussed in Section 4.2.5.

The FM modulated signal is then fed into a series of `Mute` blocks. Each of these blocks is associated with one of the FRS channels, and is controlled by a GUI checkbox. When the channel is not muted, the FM modulated signal is passed to the next stage. When muted, values of $(0+0j)$ are supplied to the next stage. Note the important difference here between the `Mute` and `Valve` blocks – the former zeroes samples and keeps the stream running, whereas the latter stops the flow of samples entirely. Using a `Valve` here would cause the following block to stop running if one or more channels were muted, which is undesirable.

Polyphase Synthesizer

Because the FRS channels within their groupings are evenly spaced, GNU Radio's `Polyphase Synthesizer` [8] provides a very simple solution for combining multiple channels into a single interpolated signal.

Two `Polyphase Synthesizer` blocks are used in conjunction with each other to construct two 400 kHz wide signals – one containing FRS channels 1-7, and a second 400 kHz wide signal containing FRS channels 8-14. Channels 4 and 11 are located at the center frequencies of their respective signals. Each channel in the output is separated by the desired 25 kHz, as this is the input sample rate.

As noted in the previous section, input channels contain either the FM modulated audio signal, or $(0+0j)$, depending on the state of the associated `mute` block.

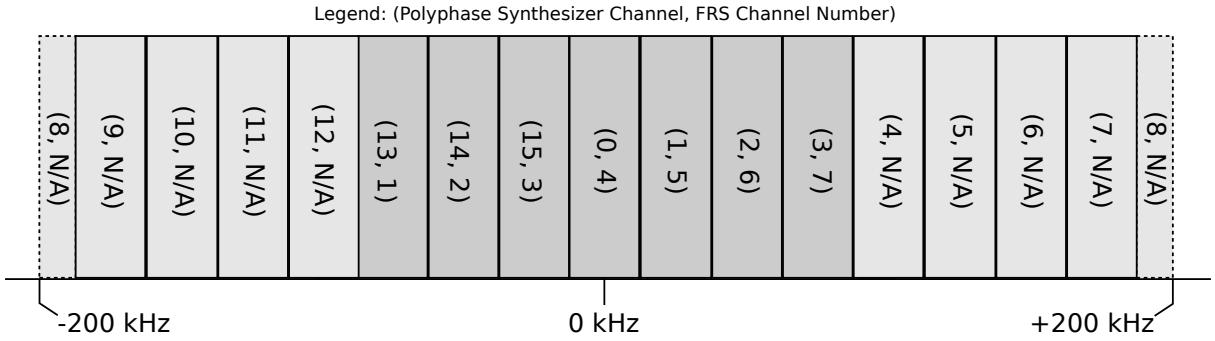


Figure 6: Polyphase Synthesizer configuration for FRS channels 1-7

Although only 7 of these 25 kHz channels are required, these blocks are configured for a total of 16 channels, with 9 of the channels left unused. This is done to alleviate the need for more interpolation in later blocks; a value of "16" is selected to avoid necessitating an additional factor of 2 in later interpolating blocks. The extra unused channels allow for a

wider transition band to be used in the interpolating FIR filter that follows this stage, which reduces the required number of filter taps[3].

The *Channel Map* property is set to `[13, 14, 15, 0, 1, 2, 3]`, in order to position FRS channels 1-7 (on input indices 0 through 6) as shown in Figure 6. With respect to channel mapping, the same approach is taken for FRS channels 8 through 14 in the second `Polyphase Synthesizer` block.

To minimize the impact of aliases, the `Polyphase Synthesizer` applies a per-channel filter at the output sample rate of 400 kHz. The filter used in this block passes signals within the 12.5 kHz FRS bandwidth limit. Therefore, the pass band is located at $\frac{12.5}{2}$ kHz. To reject the image occurring at the next channel, the stop band is place before the "edge" of the image, expected to occur at $18.75\,\text{kHz} = 25\,\text{kHz} - \frac{12.5}{2}\,\text{kHz}$.

The parameters used with `fdatool` to generate these filter taps are shown in Table 4. As shown by the magnitude response in Figure 7, spectral content outside of the desired region should be significantly rejected. However, one must take note of this undesired spectral content when evaluating the design and measuring out-of-band emissions.

Table 4: Filter parameters for polyphase synthesizer LPF

Design Method	FIR: Equiripple
Density Factor	20
Order	191
Sampling Frequency (MHz)	400
Passband Frequency (kHz)	6.25
Stopband Frequency (kHz)	18.75
Passband Weight	1
Stopband Weight	1

Post-PS gain

A `Multiply Const` block follows each `Polyphase Synthesizer`. These apply gain to compensate for the reduction of average power resulting from zero-stuffing during interpolation. However, the gain applied here is not a constant value. Instead, it is a function of the number of enabled channels that are being supplied to the input of the `Polyphase Synthesizer`. The goal here is to apply gain to the signal such that the output IQ values remain within [-1.0, 1.0].

With only one input enabled, a gain that is just slightly less than the interpolation factor is used to avoid clipping. Thus, a value of 16×0.95 is used. When a second input is enabled, the system is combining two signal sources whose amplitudes may be within [0.0, 1.0], yielding a signal with an amplitude within [0.0, 2.0]. Therefore, to keep our signal amplitude within

[3]This reduces the number of convolution operations required.

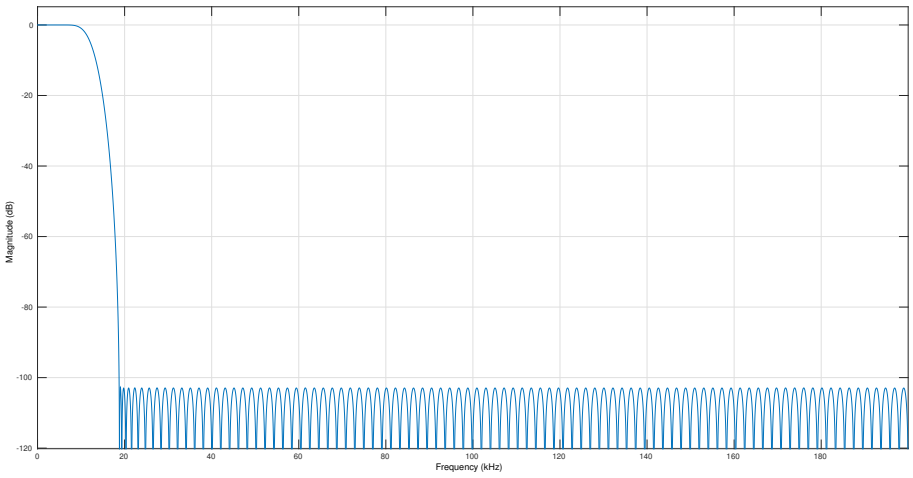


Figure 7: Magnitude response of polyphase synthesizer FIR filter

the desired range, a division by two is applied, yielding a gain factor of $\frac{16 \times 0.95}{2}$. By carrying this logic through for our total of 7 channels, one arrives at the following gain expression used for channels 8-15, with a similar expression being used for channels 1-7.

```
1 if tx_num_enabled_ch8_14 == 0 else 16 * 0.95 / tx_num_enabled_ch8_14
```

`tx_num_enabled_ch8_14` is a variable that represents how many channels are currently enabled, based upon the state of GUI checkboxes. If no channels are enabled, a value of one is used as a default. Note that this expression is written such that the check for this zero case is first. This is done to ensure the logic short circuit does not cause a division by zero to be evaluated.

Technically, the gain could be applied by scaling the `tx_pfs_taps` filter taps by this desired gain value. This would have the advantage of removing extraneous multiplications from the system. However, since these blocks are operating at a lower sample rate (400 kHz), they have been left here to serve as an explicit reminder of this scaling operation.

The outputs of these blocks are connected to `Qt GUI Waterfall Sink` blocks to provide feedback to the user that the channels selected via checkboxes are indeed being transmitted on, as well as to provide visibility of the signal power and bandwidth.

Interpolating Filters

Ultimately, the two groups of FRS signals must be recombined into a single signal that has been interpolated by a factor of 20 to match the hardware's 8 Msps sample rate. This interpolation is achieved utilizing two `Interpolating FIR Filter` blocks. As indicated

by the block name, these first interpolate by zero-stuffing samples and then filter to reject image artifacts that result from zero-stuffing.

The first FIR filter interpolates by a factor of 5, yielding an output sample rate of 2 MHz. The second stage interpolates by a factor of 4, yielding the final output sample rate of 8 MHz.

The parameters for these filters are shown in Tables 5 and 6.

To account for energy lost due to the interpolation, the filter taps obtained via `fdatool` are scaled by their associated interpolation factor. Figure 8 presents the magnitude responses of the individual stages, as well the combined response of both stages.

The passband frequencies are selected to ensure all 7 of the 25 kHz channels are passed without attenuation. Of the 400 kHz of bandwidth provided to the input of the first filter, 175 kHz[4] may contain information. Therefore, we set our passband to $85.7\,\text{kHZ} = \frac{175}{2}\,\text{kHz}$.

The stop band for the first filter is placed at the location where the first image artifact caused by interpolation is expected to be. This is calculated as follows:

$$\begin{aligned} F_{\text{stop}} &= \frac{F_s}{2} + \left(\frac{F_s}{2} - \text{BW}\right) \\ &= 200\,\text{kHz} + (200\,\text{kHz} - 87.5\,\text{kHz}) \\ &= 312.5\,\text{kHz} \end{aligned}$$

The position of the second filter's stop band is selected such that the first null of this filter aligns with null occurring before the image of the previous filter's response. This can be quickly achieved through an iterative process of measuring the location of the first null, changing the F_{stop} value in `fdatool`, and repeating until the nulls align sufficiently.

Table 5: Filter parameters for first TX interpolating (5x) FIR filter

Design Method	FIR: Equiripple
Density Factor	20
Order	59
Sampling Frequency (MHz)	2
Passband Frequency (kHz)	87.5
Stopband Frequency (kHz)	312.5
Passband Weight	1
Stopband Weight	1

The motivation behind using two stages rather than one is to minimize the number of convolution operations performed by the filter per unit of time. If a single interpolation by

[4]Recall that seven 25 kHz inputs are provided to the `Polyphase Synthesizer`.

Table 6: Filter parameters for second TX interpolating (4x) FIR filter

Design Method	FIR: Equiripple
Density Factor	20
Order	27
Sampling Frequency (MHz)	8
Passband Frequency (kHz)	87.5
Stopband Frequency (MHz)	1.67
Passband Weight	1
Stopband Weight	1

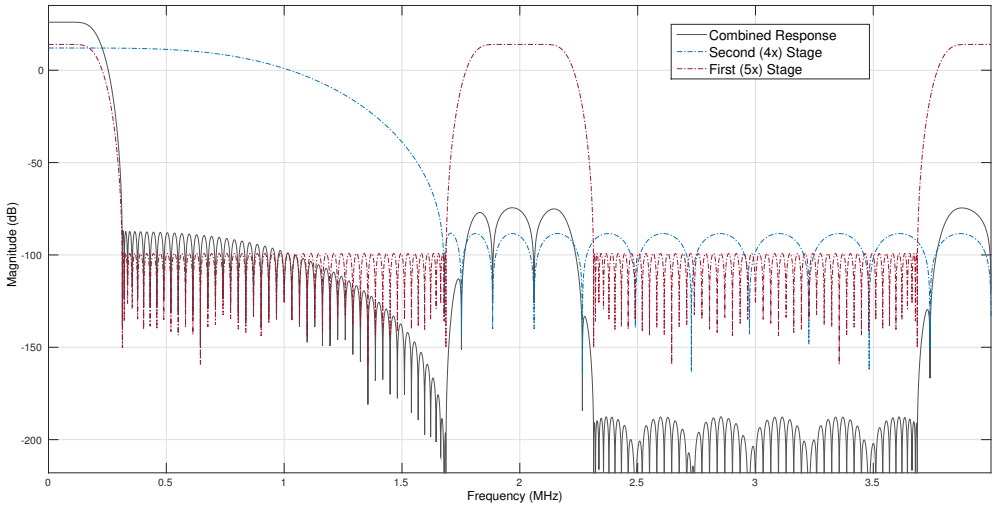


Figure 8: Magnitude response TX interpolating FIR filters

20 were performed, all of the filtering would be done at the final 8 MHz samplerate. Because the FRS spectral content is in a relatively small percentage of the interpolated signal, this requires a large number of taps to achieve the desired transition band and rejection. This combination of higher sample rate and more taps leads to higher computational load. In general, by performing the interpolation and filtering in stages, it is possible to achieve fewer operations per stage, with each stage running at a fraction of the final sample rate.

Mixing and Hardware Sink

After the previous stages of interpolation, there is ample bandwidth to combine the two groups of 7 FRS channels. Each group is mixed from baseband to a target frequency offset, per Figure 1, and then added to form the final output signal. This mixing is performed though multiplication with the output of a `Signal Source` block.

The `Signal source` is configured to provide a cosine at the target offset frequency (-2.3 MHz for Ch 1-7 and 2.7 MHz for Ch 8-14). The amplitude of this cosine is set to 0.5 in order to ensure that the final output is within (-1.0, 1.0).

The final output signal is sent to the bladeRF via an `osmocom Sink` block, which presents a hardware-agnostic sink interface.

The sink's frequency, sample rate, and bandwidth are set to the values described at the beginning of this section – 464.9375 MHz, 8Msps, and 6 MHz, respectively.

The bladeRF has two Variable Gain Amplifier (VGA) stages in its TX path, as shown in [6]. Generally, TXVGA1 should be increased before TXVGA2. Sliders for these parameters have been provided in the GUI, via `QT GUI Range` block. The `osmocom Sink` block maps its *BB Gain* to TXVGA1, and the *RF Gain* to TXVGA2.

The device-specific arguments are provided via the *Device Parameters* property string, which includes the following:

```
bladerf=0,buffers=128,buflen=8192,transfers=32
```

The first argument specifies that the first instance of a bladeRF device found on the system should be used. The remaining items configure properties of the data stream, including number of sample buffers to use, the size of each sample buffer, and the number of USB transfers to pre-allocate. Additional buffering ensures that samples are not dropped during periods of heavy CPU load, with trade-offs of added latency and memory usage.

Receive Path

The design of the RX path is very similar to the TX path, but in reverse. The same sample rate and bandwidth settings from the TX path are applied. Also, the filter designs from the TX path are used as a starting point for the associated filter in the RX path, implying that the decimation rates follow the same interpolation rates.

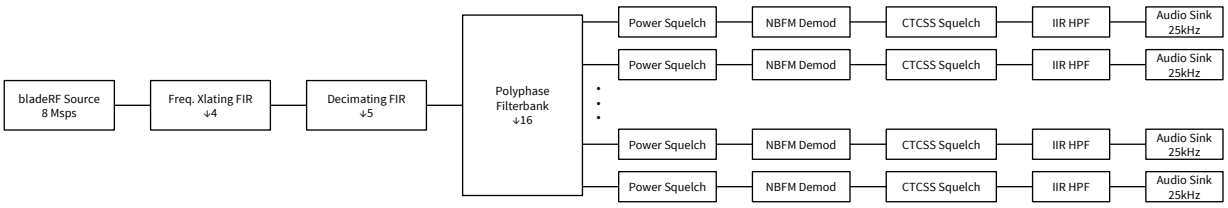


Figure 9: FRS RX block diagram

Hardware Source and Valve

An `osmocom Source` block is used to receive samples from the bladeRF. Per previous calculations, it is tuned to 465.3375 MHz, with a sample rate of 8 Msps, and a bandwidth of 6 MHz.

In the RX path, the LMS6002D provides a Low Noise Amplifier (LNA) gain stage, and two VGA gain stages. The LNA stage should be adjusted first, followed by RXVGA1 and then RXVGA2. The `osmocom Source` block maps the *RF Gain* field to the LNA gain, and the *BB Gain* to the sum of RXVGA1 and RXVGA2, where RXVGA2 is incremented after RXVGA1 reaches its max value.

The output of this hardware source is connected to a `Valve` block allowing all RX processing to be stopped.

Power Meter

The instantaneous power of the received signal passing through the `Valve` is filtered via a `Single Pole IIR Filter`, converted to dBFs, and then displayed via a `QT GUI Number Sink`. This GUI Widget provides a thermometer-style bar that allows the user to quickly gain a sense of the input power over the full 8 MHz captured at the ADC.

If strong in-band signals cause this GUI item to display a value greater than 0 dBFS, the input will saturate and likely result in artifacts and distortion. Upon observing the power meter nearing or exceeding 0dBFS, a user should reduce RX gain values, or introduce external attenuation if gain values are already at a minimum.

Frequency Translation and Decimating FIR Filters

The signals passed by the `Valve` are also routed to sections of the flowgraph that perform frequency translation, filtering, and decimation. Again, like in the TX case, the decimation and filtering is performed in two stages instead of a single stage in order to reduce computational load.

The first block is a `Frequency Xlating FIR Filter`, which efficiently translates a signal to a desired frequency offset, while filtering and decimating.

The `Frequency Xlating FIR Filter` for FRS channels 1-7 centers the input originally at -2.7MHz, and the block for channels 8-14 centers the input originally at 2.3 MHz, per Figure 1.

These blocks are configured to decimate by a factor of 4, yielding an output sample rate of 2 MHz. To avoid aliasing caused by decimation, the FIR filter applied prior to decimation must reject frequencies outside [-1 MHz, 1 MHz]. Frequencies within [-87.5 kHz, 87.5 kHz] must be passed, as this region of the spectrum contains the FRS channels. The parameters used to create a filter that meets these constraints are shown in Table 7.

Table 7: Filter parameters for RX frequency translating and decimating FIR filter

Design Method	FIR: Equiripple
Density Factor	20
Order	39
Sampling Frequency (MHz)	8
Passband Frequency (kHz)	21.875
Stopband Frequency (MHz)	1
Passband Weight	1
Stopband Weight	1

Note that although the pass band frequency specified for use with `fdatool` is 21.875 kHz, the resulting response of this stage (Figure 10) does not impose any significant attenuation up through the 87.5 kHz region of interest. In this case, it is found that relaxing the passband constraint allows `fdatool` to design a filter with better stop band attenuation.

Next, the 2 MHz signal is passed through a `Decimating FIR Filter`, which performs filtering and an additional decimation by a factor of 5, resulting in a 400 kHz signal. This implies that frequencies outside of [-200 kHz, 200 kHz] must be rejected, while ensuring the FRS content in [-87.5 kHz, 87.5 kHz] is passed without attenuation. Table 8 presents the filter design parameters.

As shown in Figure 10, this filter does indeed meet the desired requirements, as does the entire response of the two filtering stages.

Table 8: Filter parameters for RX decimating FIR filter

Design Method	FIR: Equiripple
Density Factor	20
Order	39
Sampling Frequency (MHz)	2
Passband Frequency (kHz)	87.5
Stopband Frequency (kHz)	200
Passband Weight	1
Stopband Weight	1

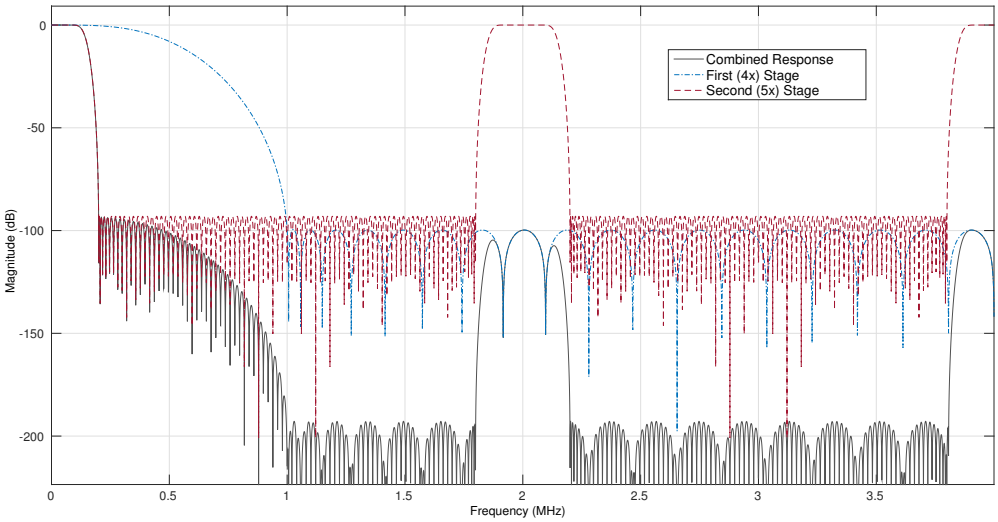


Figure 10: Magnitude response of RX decimating FIR filters

Polyphase Channelizer

In this flowgraph, the usage of the `Polyphase Channelizer` is very similar to that of the `Polyphase Synthesizer` in the TX path.

The input to each channelizer is a 400 kHz wide signal, centered on FRS channels 4 or 11 (see Figure 6). By configuring the channelizer for 16 channels and using the same channel mapping presented in Section 4.1.4, indices 0-6 are associated with FRS channels 1-7 or 8-14.

Each of the outputs containing a single FRS channel are routed to a `Valve` block. Similar to the `Mute` blocks used in the TX path, these are controlled by GUI checkboxes to allow the user to select which FRS channels to listen to. However, the `Valve` is used to disable downstream blocks when opened. This eliminates the unnecessary processing of channels that the user is not interested in.

Bus ports [9] are used to route the remaining 9 unused outputs to a single `Null Sink`, rather than having 9 separate `Null Sinks` displayed on the flowgraph; this is merely a matter of preference to reduce visual clutter in the flowgraph.

Squelch and FM Demodulation

A `Power Squelch` block is placed before the `NBFM Receive` block to prevent the CPU from performing all the demodulations simultaneously when no active signal is being received. The *Gate* option is set to "Yes," meaning that when the incoming signal does not exceed a specified power threshold, no samples will be passed to the output causing downstream blocks to stop. This is intended to further eliminate unnecessary processing.

The squelch level is controllable by the user via a slider provided by a `QT GUI Range` block. The required level for this block is expected to vary with the operating environment. A fairly high squelch threshold default has been selected for to avoid immediately passing noise when the flowgraph is started.

A value of 0.0125 for the `alpha` parameter of the `Power Squelch` is selected to significantly favor past samples in order to avoid breaking squelch due to any transient spikes. This parameter can be tuned in the flowgraph via an `rx_power_squelch_alpha`.

After breaking squelch, a signal then enters the `NBFM Receive` block. No decimation is necessary[5], so the *Audio Rate* (output sample rate) parameter is set equal to the *Quadrature Rate* (input sample rate).

The *Max Deviation* parameter of 2.5 kHz is derived from the FRS specification [2].

The `NBFM Receive` block is designed to perform de-emphasis using an Infinite Impulse Response (IIR) filter, however, de-emphasis is not needed in this design. Therefore a value for the RC time constant *tau* parameter is selected such that the resulting filter response is flat up to at least the 3.125 kHz max audio frequency.

The IIR filter taps (in transfer function from) are calculated by the `NBFM Receive` block as follows[6]:

[5] Decimation by a factor of two here could be performed to further reduce computational load.

[6] As found in fm_emph.py as of GNU Radio commit `ee369b92`

$$\omega = \tan\left(\frac{\frac{1}{\tau}}{F_s \times 2}\right)$$

$$a_0 = 1 \qquad b_0 = \frac{\omega}{1+\omega}$$

$$a_1 = \frac{\omega - 1}{\omega + 1} \qquad b_1 = b_0$$

$$\text{taps}_{\text{feed-back}} = [a_0, a_1] \qquad \text{taps}_{\text{feed-forward}} = [b_0, b_1]$$

Given the $F_s = 25$kHZ sample rate, MATLAB's `fvtool` can be used to view the response for various values of τ. A value of $\tau = 5 \times 10^{-6}$ is found to yield a stable filter with a sufficiently flat response through the frequencies up to 3.125 kHz FRS maximum, as shown in Figure 11

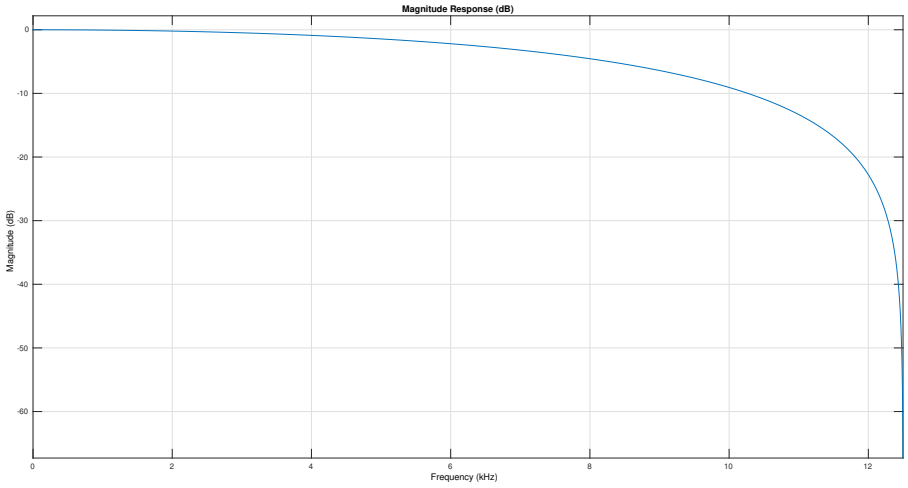


Figure 11: Magnitude response of NBFM de-emphasis filter

As of GNU Radio commit `ee369b92`, the `NBFM Receive` block also performs a LPF operation to limit the audio output. The limit is hard coded at 2.7 kHz with a 0.5 kHz transition band using a Hamming window.

Following the FM demodulation, the audio signal is routed to both a `CTCSS Squelch` block and directly to a `Selector` block. When the GUI dropdown for the desired CTCSS tone is set to "None" the `Selector` passes the output of the `NBFM Receive` block to the next stage. Otherwise, the `Selector` passes the output of the `CTCSS Squelch` block, which is configured to pass samples when the CTCSS tone associated with the current GUI dropdown widget selection is detected.

Adequate values for the `CTCSS Squelch` values are determined experimentally by interfacing with handheld FRS radios, such as those described in Section 5. A *Length* value of 6250 samples (250 ms at 25 kHz) and *Level* of 0.010 are found to provide desirable squelch functionality.

CTCSS Tone IIR Filter

An astute listener can hear a number of the higher-frequency CTCSS tones. Therefore, a high-pass filter is needed to reject audio below 250.3 Hz. At a 25 kHz sample rate, an acceptable FIR filter is found to require a large number of taps [7]. Instead, an IIR filter is designed using `fdatool`, as this is expected to yield significantly fewer taps. The design parameters are shown in Table 9, and the filter response is presented in Figure 12.

Table 9: Filter parameters of RX CTCSS-blocking high pass filter

Design Method	IIR: Elliptic
Option: *Match Exactly*	both (pass and stop)
Order	7
Sampling Frequency (kHz)	25
Stopband Frequency (Hz)	250.3
Passband Frequency (Hz)	300
Passband Attenuation (dB)	1
Stopband Attenuation (dB)	50

Audio Sinks

The audio from each FRS channel is provided to the system via the `Audio Sink` block. As done in the TX path, "pulse" is specified for the *Device Name* parameter to utilize the PulseAudio subsystem. It is possible to use one sink per channel because the underlying audio subsystem handles multiple inputs (from each sink). When multiple channels are being received, all of them will be heard and combined by the PulseAudio subsystem.

[7] Approximately 900 taps for a FIR: Equiripple and 50dB of stopband attenuation.

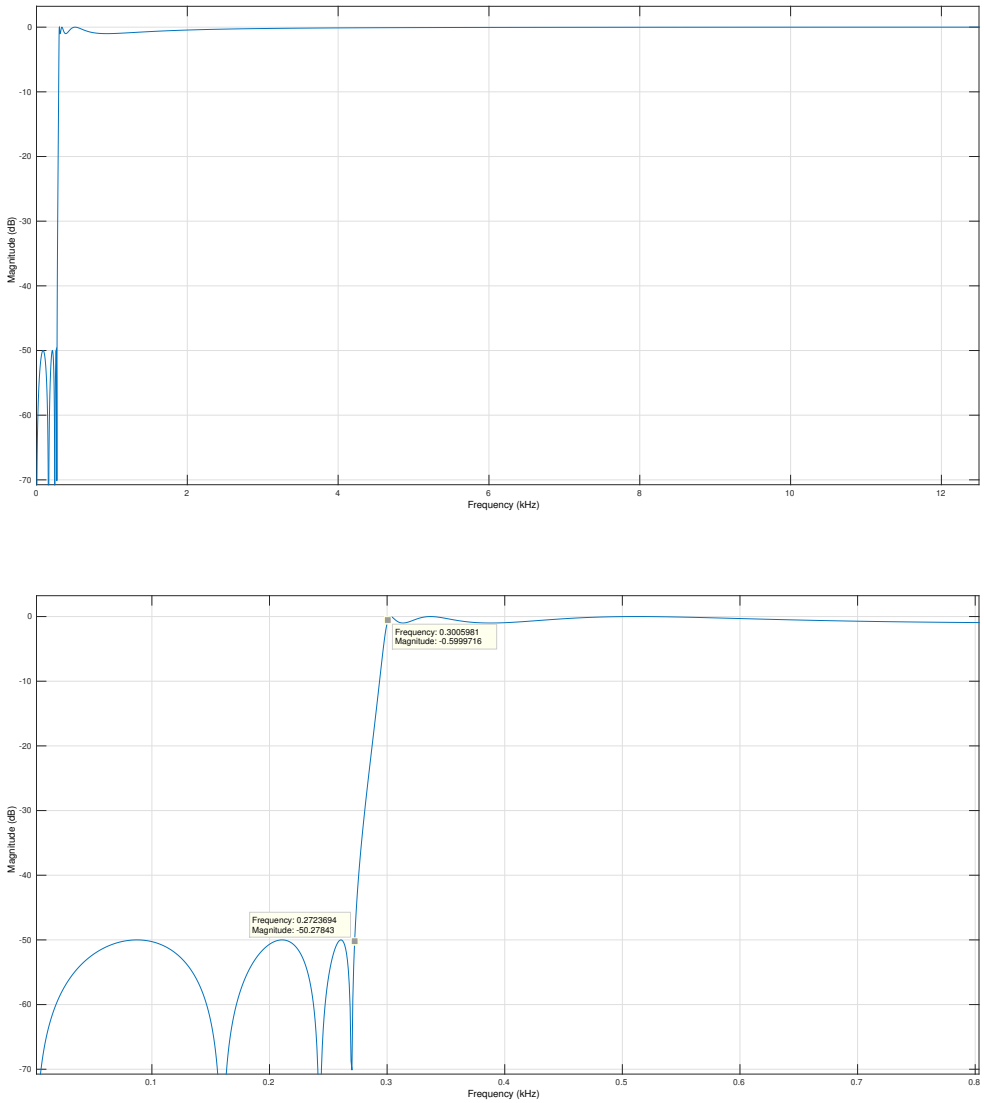


Figure 12: Response of RX CTCSS-blocking high pass filter

Testing and Evaluation

Flowgraph GUI

As noted in previous sections, various blocks provide GUI widgets that may be used to control and interact with the running flowgraph. Screen captures of this GUI are shown in Figures 13 and 14.

The top of the GUI provides quick access to CTCSS tone selection and the ability to enable/disable the RX and TX functionality.

By default, the CTCSS dropdown is set to "None." This can be changed at any time, but it is recommended to make changes while RX and TX are disabled, as portions of the flowgraph will be reconfigured.

The "Enable RX" checkbox is a master enable for all reception. Two master enable options are provided for TX. A "Continuous Enable" checkbox enables the TX front end while the checkbox is marked. The "Push to Talk" button enables the TX front end only while this button is pressed.

The remainder of the GUI is split between an RX tab and a TX tab. Both views provide checkboxes for each FRS channel. Marking the checkbox enables reception/transmission of the associated channel. Gain controls are also provided in each view.

The RX view also provides control over the `Power Squelch` block as the noise floor may vary in different environments.

Waterfall plots are provided for both groups of channels (1-7 and 8-14). These plots display received signals in the frequency domain prior to any squelch blocks. An "Input Power" meter is placed between these plots to provide a quick means to determine if gains are set too high or too low.

The TX side also provides waterfall plots. These are intended to provide feedback in response to the user's checkbox selections.

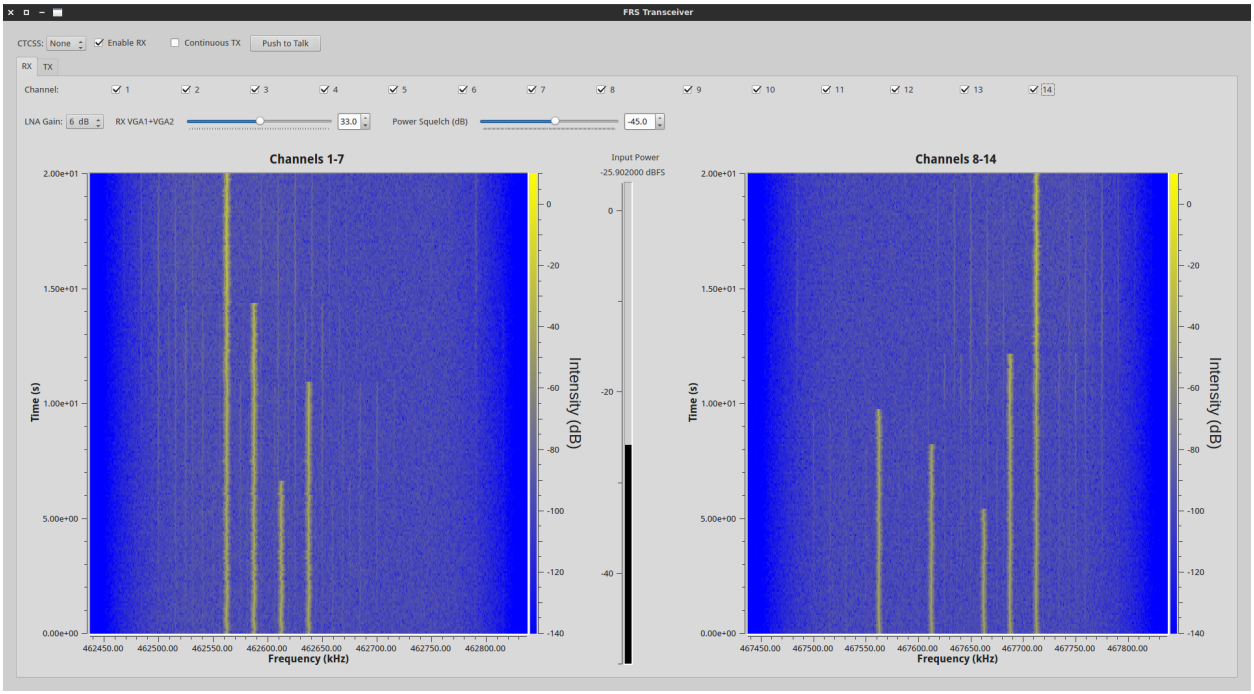


Figure 13: RX Section of the flowgraph GUI

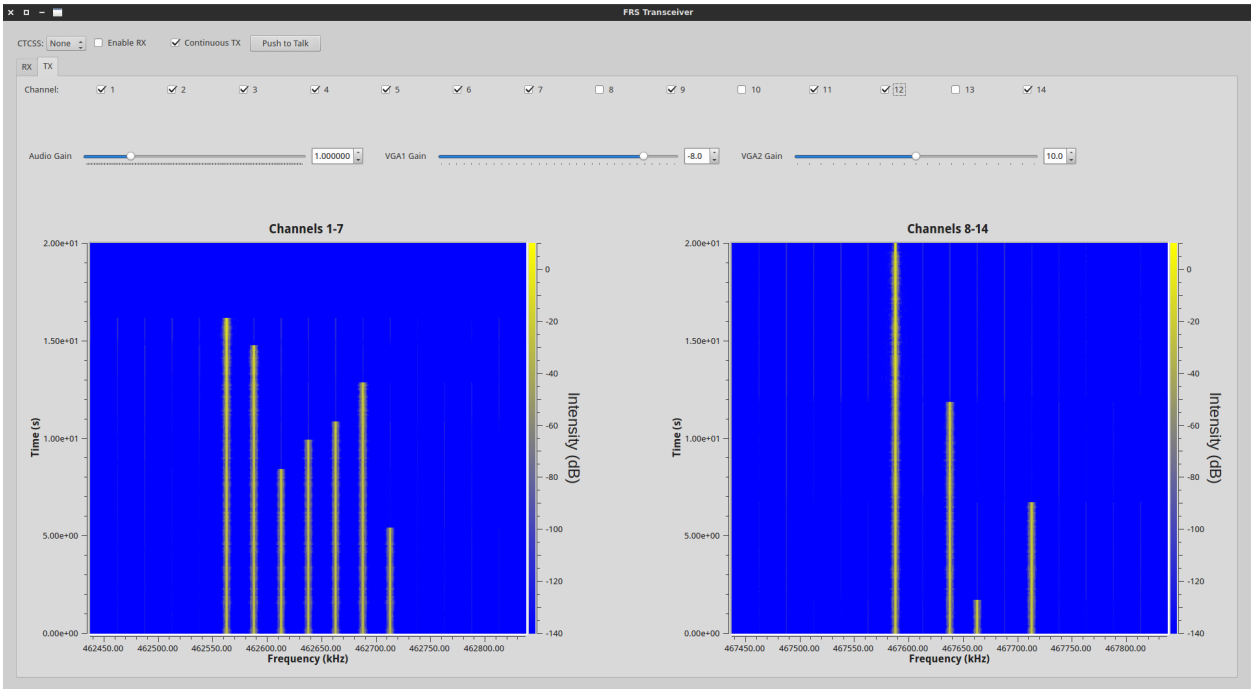


Figure 14: TX Section of the flowgraph GUI

Nuand bladeRF SDR

The Nuand bladeRF is a USB 3.0 SDR with a frequency range of 300 MHz to 3.8 GHz, full-duplex capabilities, a maximum bandwidth of 28 MHz, 12-bit samples, and a maximum sample rate of 40 MHz [10]. The architecture of the platform is illustrated in Figure 15.

The bladeRF utilizes a Cypress FX3 USB3 device controller, which allows data to be transferred between its USB 3.0 SuperSpeed connection and its parallel GPIF II interface (running half-duplex at a speed of 100MHz). This is connected to an Altera Cyclone IV E Field Programmable Gate Array (FPGA). The FPGA provides buffering and additional signal processing, as well as control of the Lime Microsystems LMS6002D Field-Programmable RF Transceiver.

The LMS6002D transceiver IC, whose architecture is depicted in in Figure 16, has multiple input and output RF ports which are tuned to different frequency ranges. The architecture is split into a low band (300 MHz - 1500 MHz) and high band (1500 MHz - 3800 MHz). The LMS6002D contains independently configurable RX and TX signal chains, which allows the bladeRF platform to provide full-duplex capabilities. Also note that the RX and TX mixers each have their own PLLs, which synthesize desired tuning frequencies from a common input clock. These are what allow the device to operate the RX and TX paths at different frequencies.

A single VCTCXO running at 38.4MHz supplies the FX3 and Silicon Labs Si5338 low-jitter clock generator. All sample rate clocks are generated by the Si5338 with high accuracy, allowing for flexible and arbitrary clocking of the ADC/DAC. The Si5338 also supplies clocks to the Altera Cyclone FPGA and to the Lime Microsystems LMS6002D.

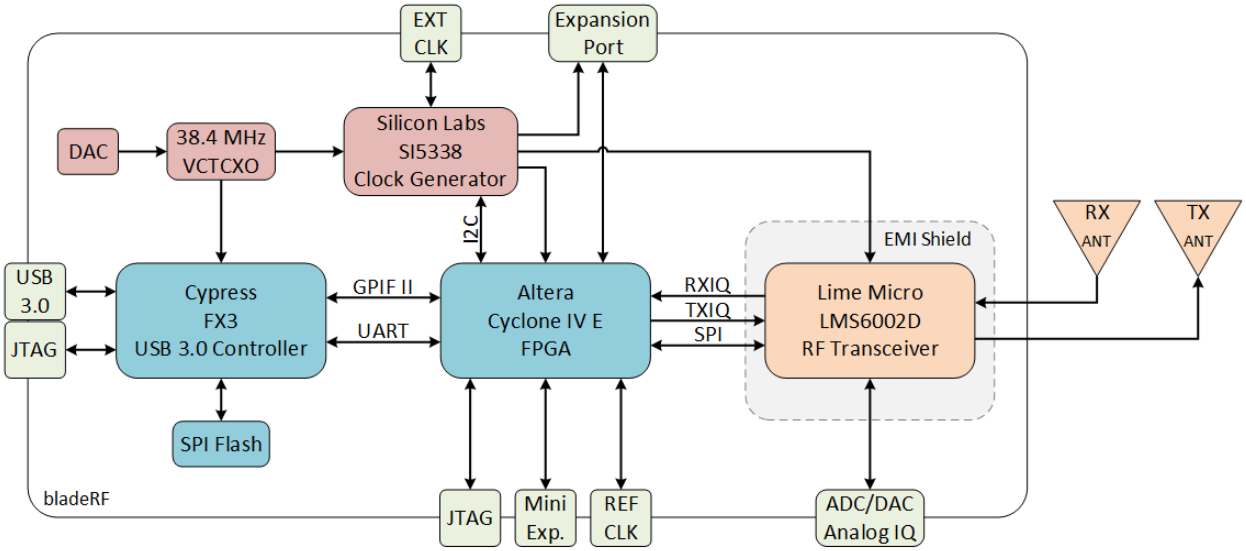


Figure 15: Nuand bladeRF architecture

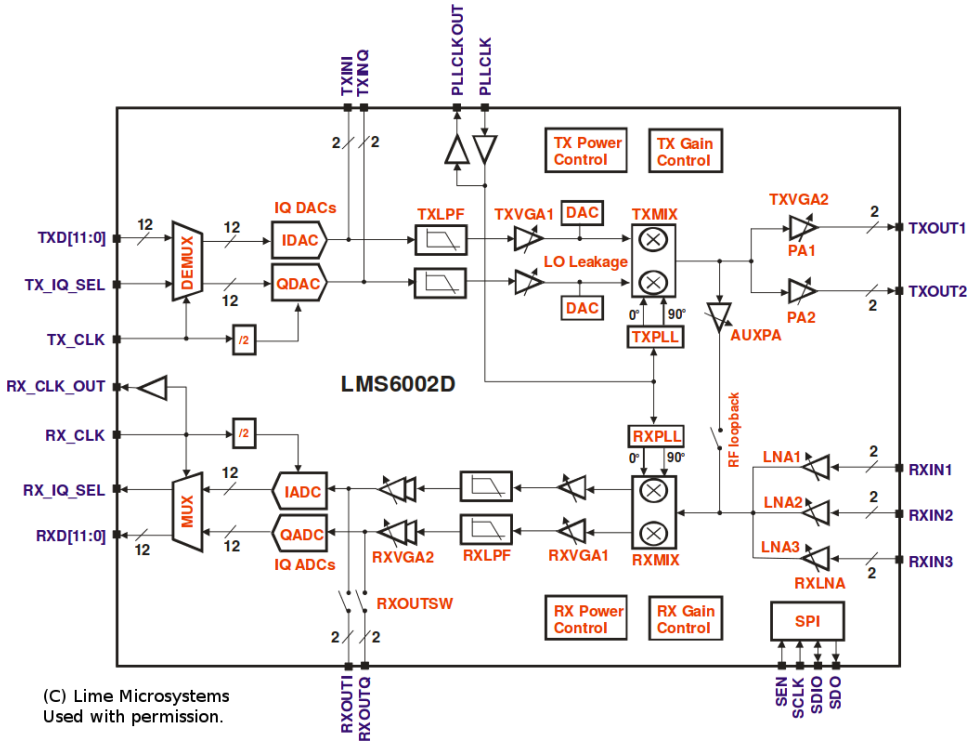


Figure 16: Lime Microsystems LMS6002D architecture

Reception

To initially test reception, an Agilent E4433B signal generator [11] is used to supply FM modulated tones to the bladeRF. This device is configured to supply an FM-modulated 600 Hz tone with a deviation of 2.5 kHz and an amplitude of -60 dBm.

The channel mapping is verified by supplying a signal at a specific FRS channel and checking that the demodulated signal is played only when that associated channel is enabled in the GUI. This is repeated for each of the 14 FRS channels.

The per-channel filters are evaluated in a similar fashion, by verifying that supplying a signal to the bladeRF on one channel does not yield significant interference on neighboring channels.

Next, it is verified that none of the channels are attenuated significantly more than others, as one might expect to occur if the decimating FIR filters had been designed to be too narrow. This is done by sweeping across 200 kHz of bandwidth with the signal generator, and noting the signal level in the waterfall plots provided in the designed FRS Transceiver GUI. As expected, attenuation begins occurring before channels 1 and 8, and after channels 7 and 14. No significant attenuation is observed with the regions containing the groups of 7 FRS channels.

Figure 17: Cobra CXT545 FRS handheld transceiver used to test flowgraph

Next, interoperability with commercially-available handheld FRS transceivers is tested. A pair of Cobra CXT545 [12] walkie talkies, configured for low power operation on FRS channels (Figure 17) are used. These devices support all 38 of the commonly used CTCSS tones listed in Table 2, with 67.0 Hz being CTCSS tone #1, and 250.3 Hz being tone #38.

Within the confines of a lab environment, the output power of these radios is found to saturate the bladeRF input. This is addressed by reducing the gain values used for reception and/or using external attenuators.

The channel mapping and per-channel filter tests are performed again using these devices. The CTCSS squelch functionality is verified by transmitting with each tone on the handheld, and confirming that the audio is played by the flowgraph only when "None" or the associated CTCSS tone is selected in the GUI.

Transmission

The waterfall plots in the TX portion of the flowgraph are used to provide confirmation that each channel's signal is located at the correct offset. To further evaluate the signal being supplied to the bladeRF for transmission, a `Qt GUI Frequency Sink` is connected in parallel to the `osmocom Sink` block. For a higher-resolution view of a group of channels, a frequency sink block is placed in parallel with the first `Interpolating FIR Filter` block.

Figure 18 depicts the digital representation of the signal containing a concurrent transmission on channels 8 through 14 while Figure 19 show a transmission containing only channel 8. A plot containing the simultaneous transmission on all 14 channels is presented in Figure 20.

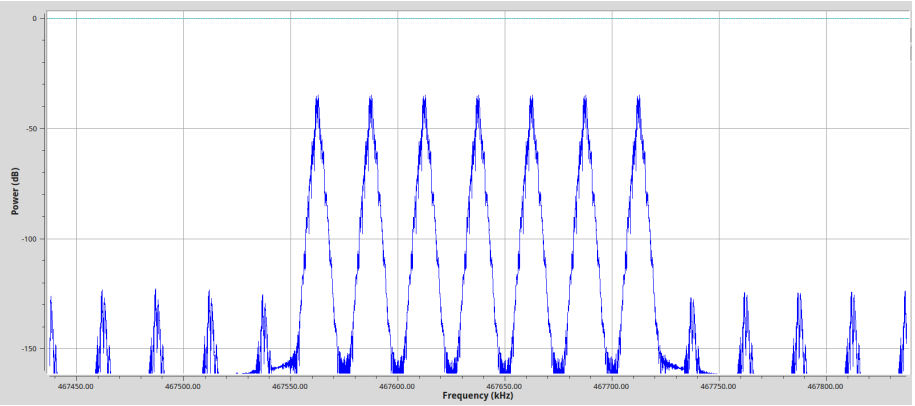


Figure 18: FFT plot of digital signal containing FRS channels 8-14

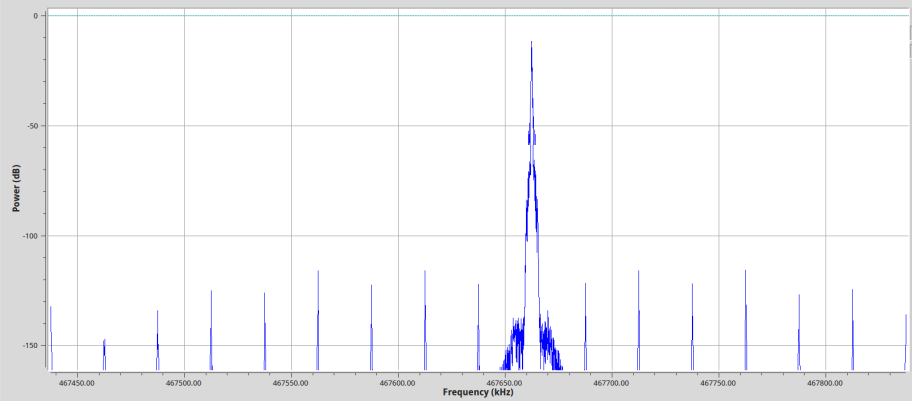


Figure 19: FFT plot of digital signal containing FRS channel 8

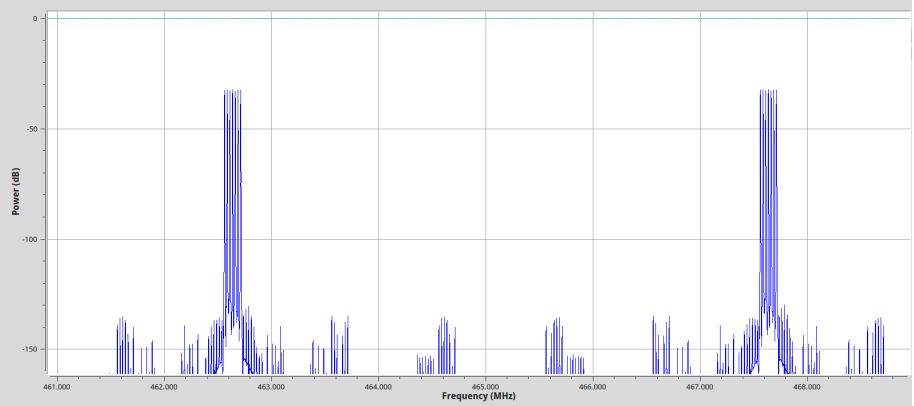


Figure 20: FFT plot of digital signal containing all FRS channels

Per these plots, the channels are located at the expected frequencies, with bandwidths within the 12.kHz maximum. The expected aliases that occur as a side effect of polyphase synthesis [8] are significantly rejected by the designed filters and are observed to occur 100 dB below the signals of interest.

Next, the output of the bladeRF's TX port is observed, using an Agilent E4406A Vector Signal Analyzer (VSA) [13]. Again, the locations and widths channels are first verified to be correct.

The signal levels of the FRS signals and the expected aliases are found to be within acceptable limits. It is noted that the magnitudes do differ from the digital representation to some degree. This may be attributed to quantization error that is inherent in converting a digital floating point signal to an analog signal through the hardware's 12-bit Digital to Analog Converter (DAC). This can be simulated in GRC by adding a `Quantizer` block (found under "Impairment Models") prior to the `osmocom Sink` block.

As discussed in Section 4, the bladeRF's TX frequency is tuned 200 kHz under 465.1375 MHz – the frequency that is equidistant from both groups of channels – to avoid distortion caused by any IQ imbalance.

The effectiveness of this strategy is made clear by Figure 22. Note that to the left of each group of FRS signals, a significantly attenuated image from the other group is present. By ensuring these images do not overlap with the intended signal, a higher Signal-to-Noise and Distoration ratio (SINAD) is achieved. For a laboratory experiment, 40 dB of rejection for these images and the carrier caused by residual DC offset is certainly acceptable. Possible approaches to further reduce the undesired spectral content is to utilize more aggressive digital filters (likely requiring a large number of taps), external filters, or a combination of both.

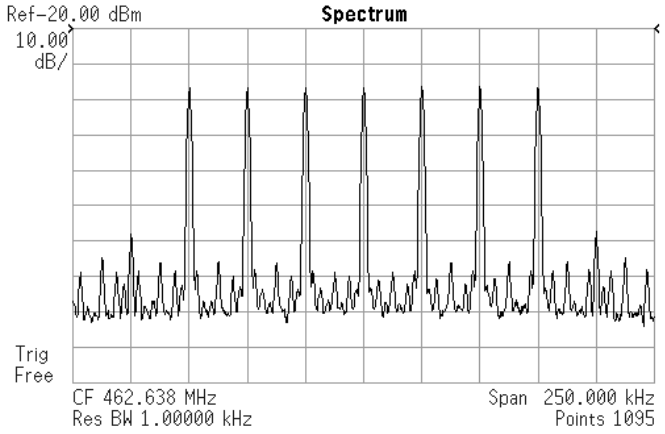


Figure 21: FFT plot of analog signal containing FRS channels 1-7

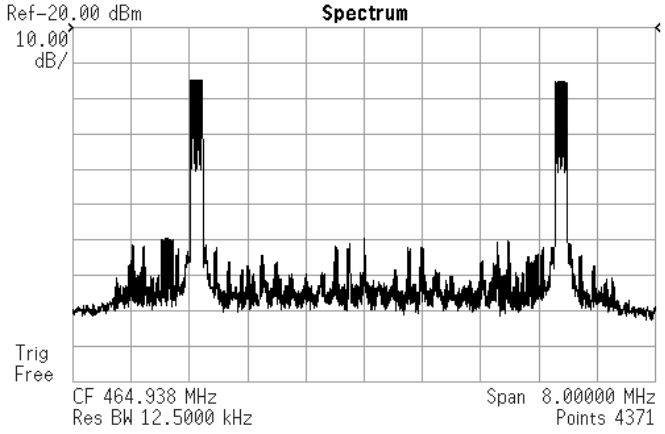


Figure 22: FFT plot of analog signal containing all 14 FRS channels

Software and System Configuration

The design and evaluation of the flowgraph described in this document is performed with XUbuntu 14.10 [14], on machines with Intel i7 processors, more than 4 GB of RAM, and either Intel XHCI 7 Series or Renesas uPD720202 USB 3.0 host controllers. Similar or better systems are recommended.

Due to the sample rates used in this flowgraph, USB 3.0 is required for full-duplex operation. Be aware that even for half-duplex operation, USB 2.0 controllers may not be able to support the required 32 MB/s[8]

Table 10 lists the minimum required software versions needed to use this transceiver. Newer versions of the listed items should also be compatible.

Table 10: Minimum required component versions

Component	Version
bladeRF FPGA	0.1.2
bladeRF Firmware	1.8.0
libbladeRF	1.2.1
GNU Radio	3.7.7
gr-osmosdr	0.1.4-g48045b59

The "Getting Started" guide on the bladeRF wiki [15] provides instructions for installing the latest versions of the above items. Those new to the bladeRF and GNU Radio may wish to use the GNU Radio Live SDR Environment [16] , an Ubuntu-based live image with GNU Radio and SDR support pre-installed.

Conclusions

This paper has demonstrated a successful design and implementation of a prototype FRS radio system in a manner that takes advantage of the features provided by the Nuand bladeRF SDR platform. In doing so, it has exemplified topics including performance-conscious filter design and operation on multiple narrow-band channels via GNU Radio's Polyphase Channelizer/Synthesizer blocks. Furthermore, constraints and challenges presented by real-world nonidealities have been addressed throughout the design, implementation, and evaluation processes. Upon developing a level of comfort with these topics, readers should be able to experiment with, modify, and further extend the GNU Radio flowgraph presented in this work, as well as apply these concepts to new designs.

[8] Although the theoretical max throughput of USB 2.0 is 60 MB/s, the practical achievable throughput is significantly less due to protocol overheads.

Flowgraph Diagrams

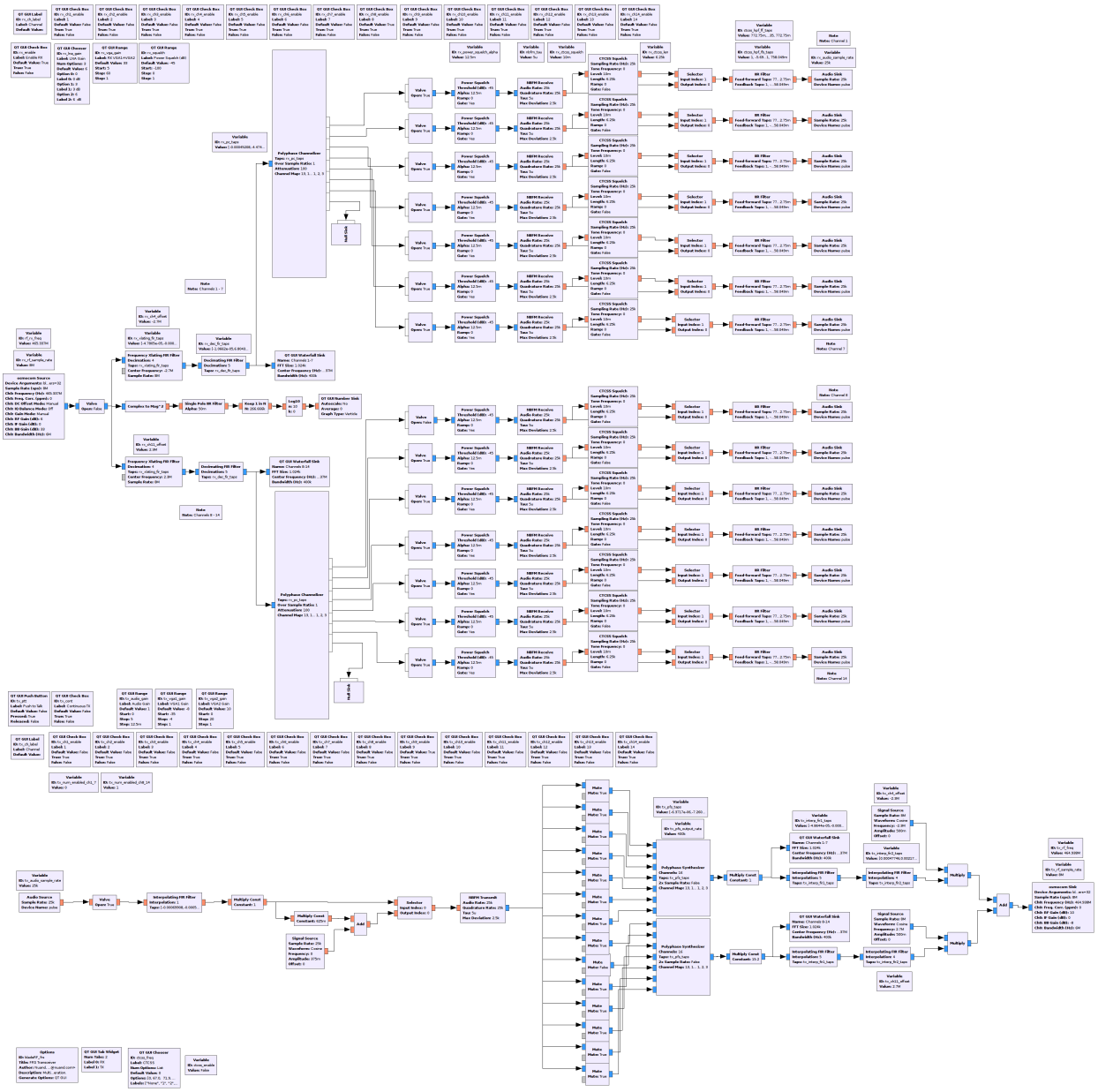


Figure 23: Full FRS transceiver flowgraph

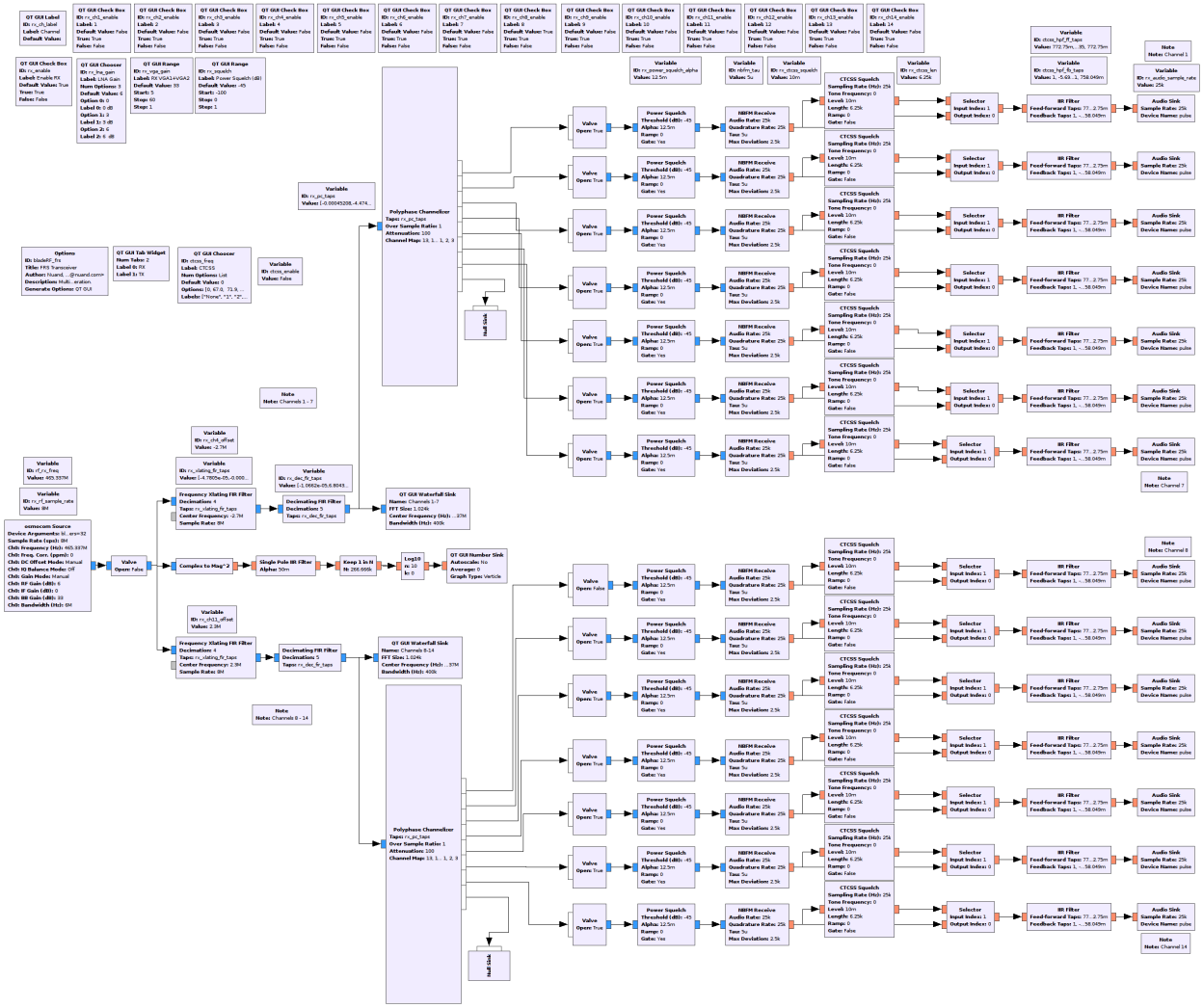


Figure 24: RX path of FRS transceiver

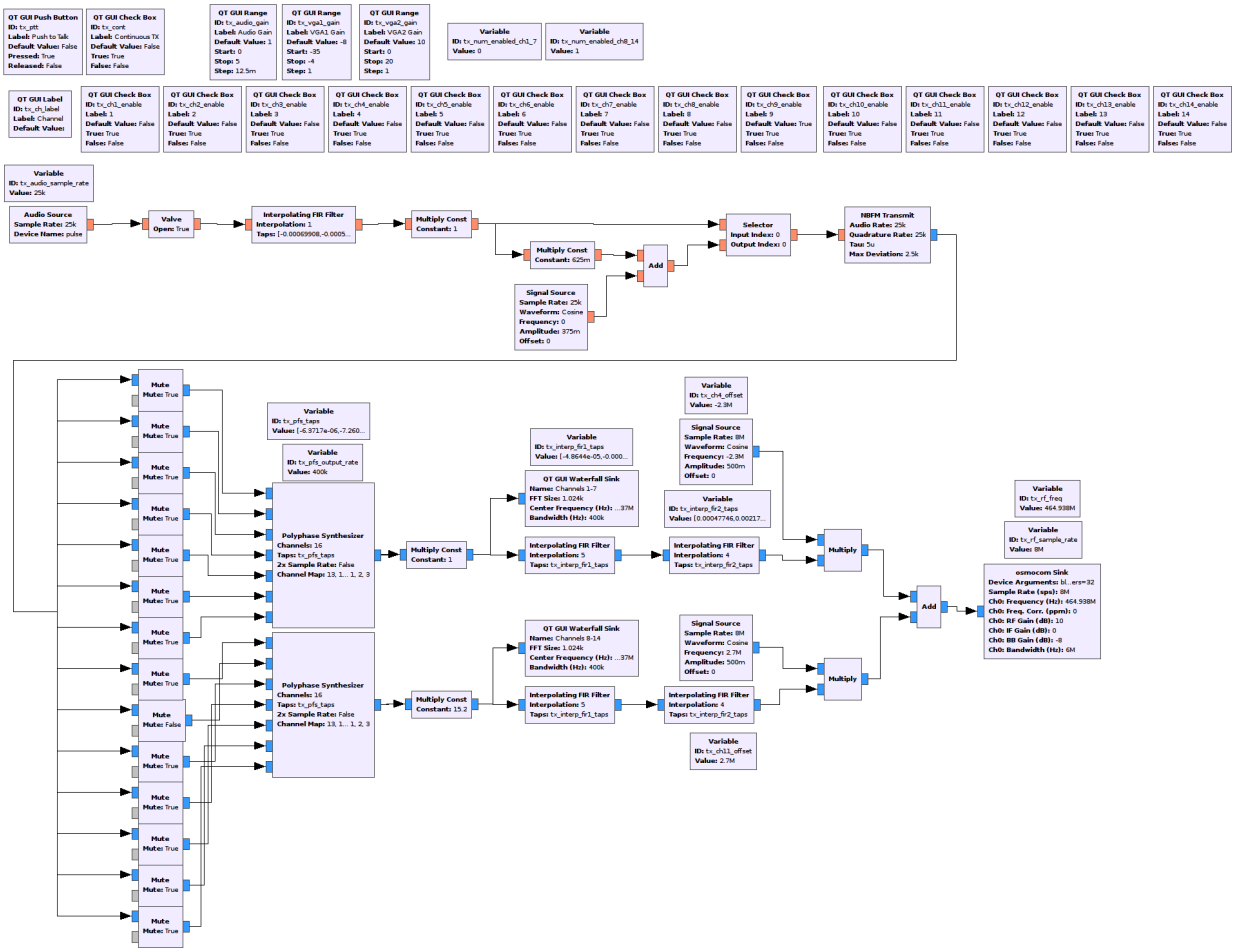


Figure 25: TX path of FRS transceiver, slightly re-arranged to fit page

References

[1] Family Radio Service (FRS). https://www.fcc.gov/encyclopedia/family-radio-service-frs. Accessed: 2015-03-02.

[2] eCRF – Code of Federal Regulations, Part 95 - Personal Radio Services, Subpart B - Family Radio Service (FRS). http://www.ecfr.gov/cgi-bin/text-idx?SID=5ddd3bab07204521939c4b1580b7a692&node=pt47.5.95&rgn=div5#sp47.5.95.b. Accessed: 2015-03-02.

[3] GNU Radio. http://gnuradio.org/. Accessed: 2015-03-02.

[4] Nuand bladeRF. https://www.nuand.com/bladeRF. Accessed: 2015-03-02.

[5] Field Programmable RF ICs: LMS6002D. http://www.limemicro.com/products/field-programmable-rf-ics-lms6002d. Accessed: 2015-03-09.

[6] LMS6002 Datasheet (1.1.0). http://www.limemicro.com/download/LMS6002Dr2-DataSheet-1.2r0.pdf. Accessed: 2015-03-09.

[7] PulseAudio. http://www.freedesktop.org/wiki/Software/PulseAudio. Accessed: 2015-03-03.

[8] PFB Channelizers and Synthesizers. http://www.trondeau.com/examples/2014/1/23/pfb-channelizers-and-synthesizers.html. Accessed: 2015-03-02.

[9] Working with GRC Busports. http://www.trondeau.com/blog/2014/2/27/working-with-grc-busports.html. Accessed: 2015-03-10.

[10] Nuand bladeRF Product Brief. https://www.nuand.com/bladeRF-brief.pdf. Accessed: 2015-10-24.

[11] Keysight E4433B ESG-D Series Digital RF Signal Generator. http://www.keysight.com/en/pd-1000002824%3Aepsg%3Apro-pn-E4433B/esg-d-series-digital-rf-signal-generator-4-ghz?cc=US&lc=eng. Accessed: 2015-03-10.

[12] Cobra CXT545. https://www.cobra.com/products/max-performance/cxt-545. Accessed: 2015-03-10.

[13] Keysight E4406A VSA Transmitter Tester. http://www.keysight.com/en/pd-1000002790%3Aepsg%3Apro-pn-E4406A/vsa-transmitter-tester-7-mhz-to-4-ghz?cc=US&lc=eng. Accessed: 2015-03-10.

[14] XUbuntu 14.10 released! http://xubuntu.org/news/14-10-release/. Accessed: 2015-03-11.

[15] bladeRF "Getting Started" guide for Linux. https://github.com/Nuand/bladeRF/wiki/Getting-Started:-Linux. Accessed: 2015-10-24.

[16] GNU Radio Live SDR Environment. https://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioLiveDVD. Accessed: 2015-10-24.