Lab 4

Introduction

This lab explores super–heterodyne single and dual conversion receiver subsystems for analog and digital modulation. Two VHF (30––300 MHz) FM receivers are considered. The first receiver employs a wideband (about 200 kHz) IF subsystem centered at 10.7 MHz, while the second employs a narrowband (about 10 kHz) IF subsystem centered at 455 kHz. The narrowband FM receiver also utilizes dual conversion, with the first IF at 10.7 MHz and the second IF at 455 kHz. Both receivers have been constructed using readily available radio frequency integrated circuits (RFICs) from NXP semiconductor (http://www.nxp.com/). The receivers are presently in prototype form, constructed on an RF breadboard. In the future the receivers will be fabricated using a custom PCB.

The high sensitivity of these receivers allows the wideband receiver to easily tune in FM broadcast stations and the narrowband receiver to receive the Colorado Springs national weather service (NOAA) station, and lab broadcast frequency shift keyed (FSK) digital modulation.

Wideband FM Receiver

The block diagram for the wideband receiver is given in Figure 1. The low–noise amplifier (LNA) is not implemented at this time, nor is the front–end bandpass filter (BPF). A short wire (clip lead) will serve as the antenna in the experiment.

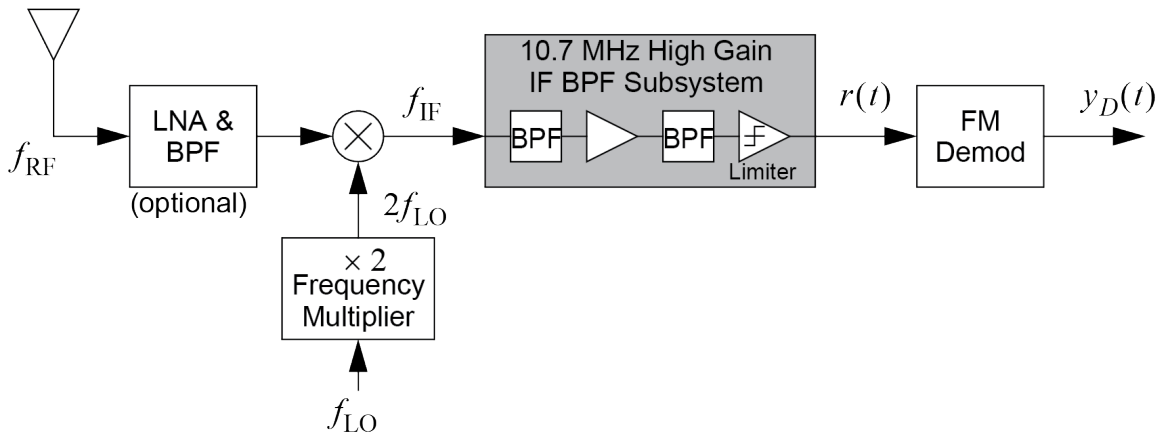


Figure 1: Wideband FM receiver block diagram.

The receiver requires and external local oscillator (LO). In order to receive VHF signals, in particular FM broadcast which covers 87.5––108 MHz, the output of the Agilent 33250, which will serve as the LO, needs to be frequency doubled. The Agilent 33250 has a maximum frequency of 80 MHz, but with the doubler the effective maximum LO frequency is 160 MHz. With low–side tuning for the LO, this means that carrier frequencies up 160 + 10.7 = 170.7 MHz can down–converted. The doubler, Minicircuits AMK–2–13+, is a passive circuit, the same part number as found on the RF board , which in simple terms acts as a full–wave rectifier, which has a strong second harmonic component.

The mixer output is processed with a multistage IF amplifier, with the 10.7 MHz IF passband shaping formed using ceramic filters. The nominal bandwidth of each filter is 280 kHz. Note from the schematic of Figure 2, the ceramic filters are external to the NXP SA636 RFIC.

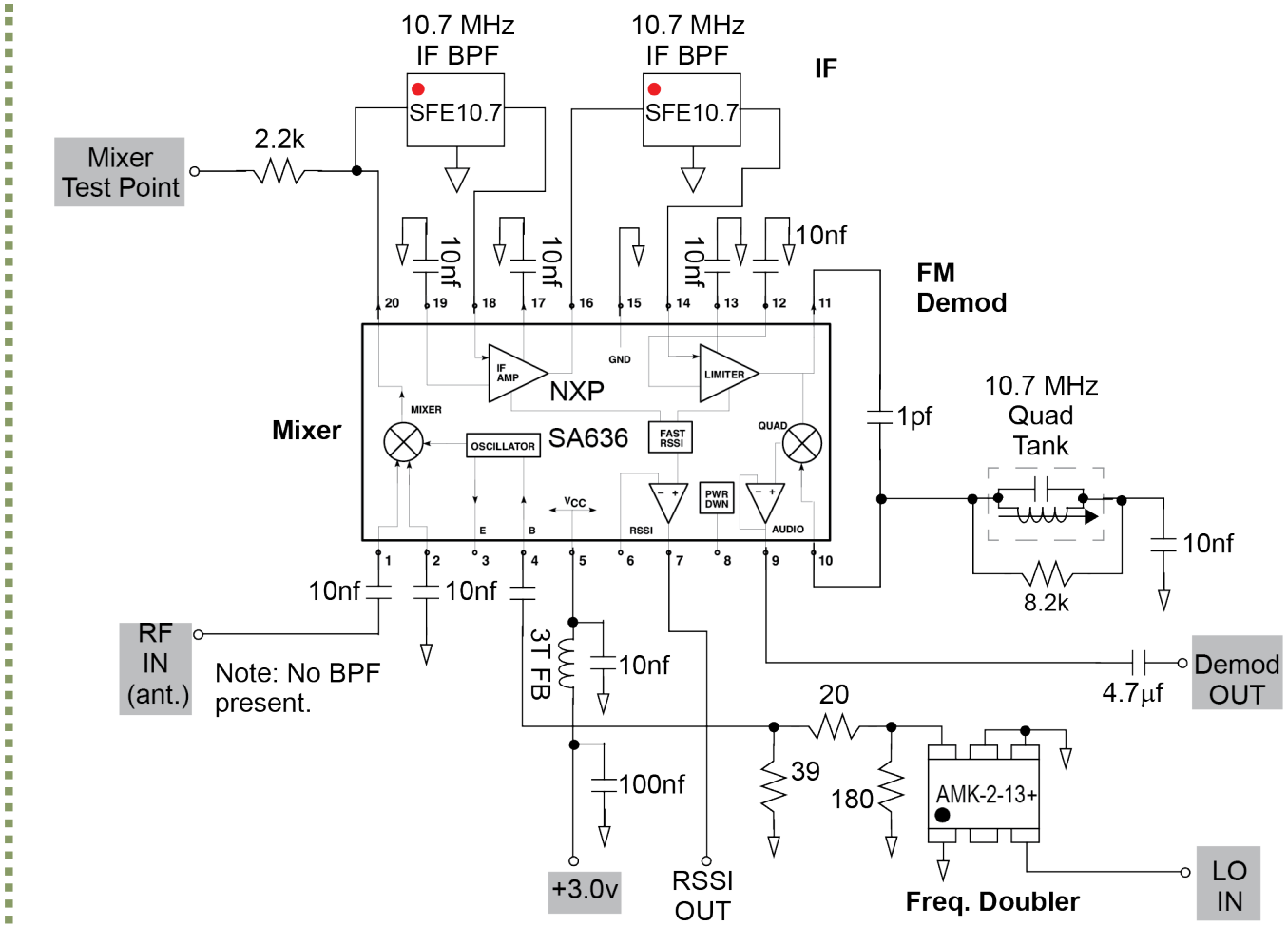


Figure 2: Single conversion VHF FM receiver schematic based on the NXP SA636 IF subsystem.

The final stage is the FM demodulator, which in this design uses a quadrature detector. The theoretical basis of the quadrature detector will be described in a later section.

IF Filters

An RF receiver needs to have high gain in order process weak signals arriving from a transmitter located many miles away. High gain over a wide bandwidth is hard to manage from a stability standpoint. Sensitive radio receivers also need to be very selective, that is supply high gain over just a relatively narrow band of frequencies. For the case of an FM receiver the needed bandwidth is the bandwidth of the modulated message signal following a Carson's rule analysis. In the FM broadcast band stations are spaced every 200 kHz at odd 100 kHz spacings, e.g., 101.1 MHz, 99.3 MHz, etc. From Wikipedia FM broadcasting the one–sided baseband spectral occupancy is around 100 KHz before being frequency modulated onto the carrier, as shown in Figure 3. The dominant baseband signals are the $L+R$ and $L-R$ audio signals.

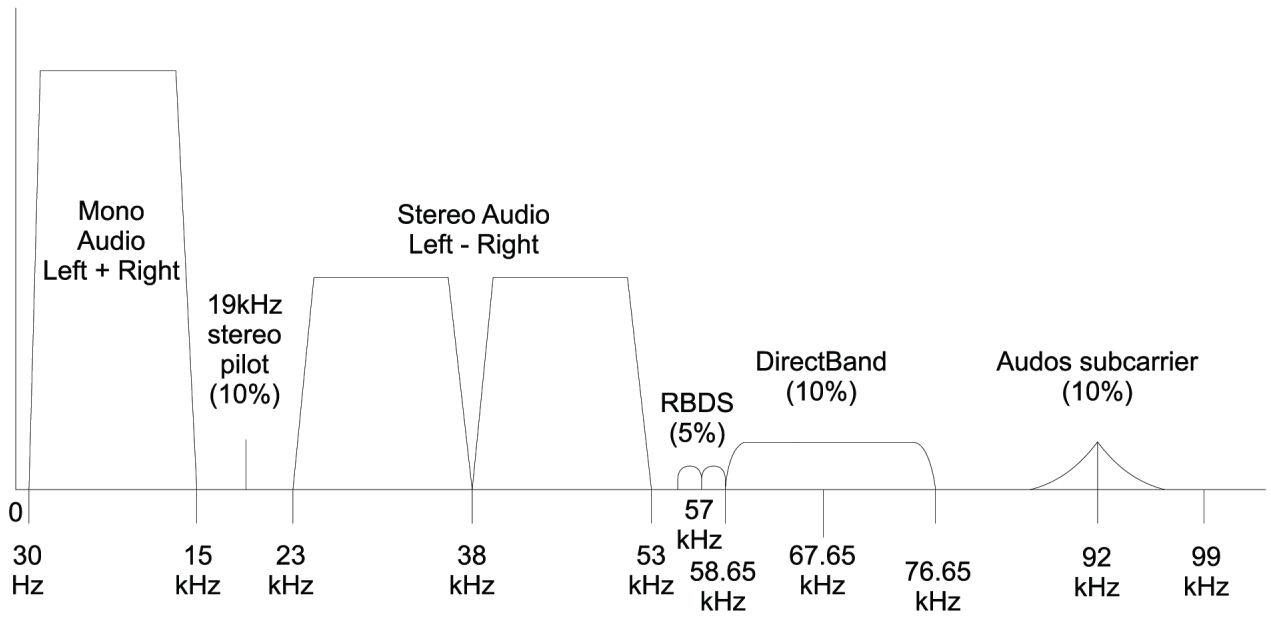


Figure 3: One–sided FM broadcast spectrum including the various subcarrier services.

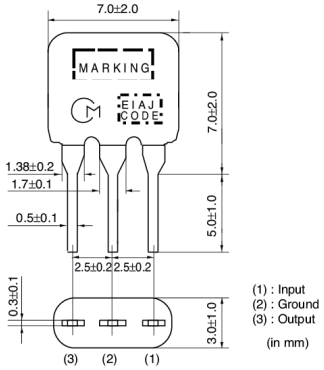
The peak frequency deviation is 75 kHz. If we apply Carson's rule to just the $L+R$ signal, which has a nominal bandwidth of 15 KHz, we arrive at a modulated bandwidth of

$$B_{\mathrm{RF}} = 2(D+1)W = 2\left(\frac{75}{15}+1\right)15 = 180\ \mathrm{kHz}. \tag{1}$$

When the $L-R$ for FM stereo is included along with the other services, e.g. radio data service (RDS) at 57 kHz, the occupied spectrum must still be within a 200 kHz wide footprint.

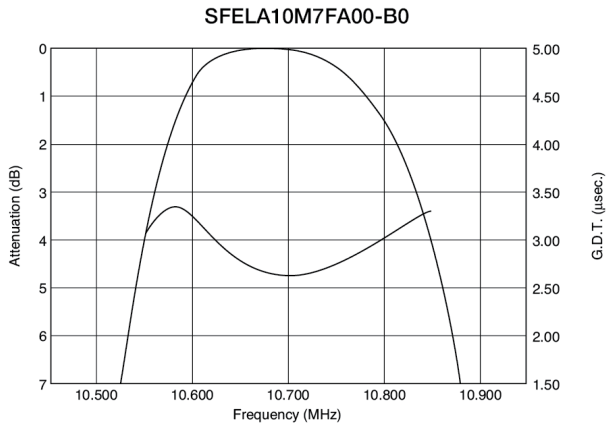
We desire an IF filter characteristic which passes only the signal/station of interest, rejecting all adjacent signals. A realizable IF filter will transition from passband to stopband over a finite band of frequencies. The ceramic filters employed in the wideband receiver design are wider than needed, being that thy have a 3 dB bandwidth of 280 kHz. The filter specifications and plots of the magnitude and group delay response are shown in Figure 4.

SFELA10M7FA00-B0

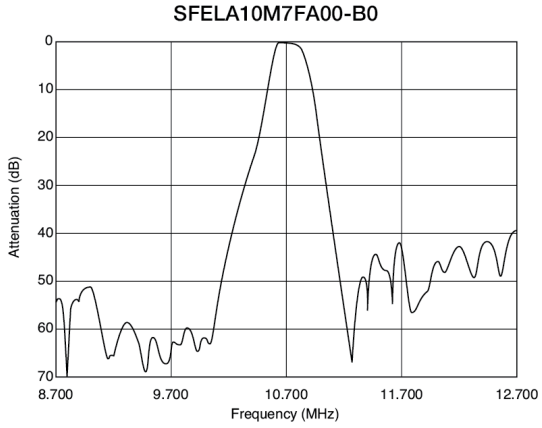


Part Number	Center Frequency (fo) (MHz)	3dB Bandwidth (kHz)	Attenuation (kHz)	Insertion Loss (dB)	Spurious Attenuation (dB)	Input/Output Impedance (ohm)
SFELA10M7HA00-B0	10.700 ±30kHz	180 ±40kHz	520 max.	7.0 max.	40 min.	330
SFELA10M7GA00-B0	10.700 ±30kHz	230 ±50kHz	570 max.	4.0 ±2.0dB	40 min.	330
SFELA10M7FA00-B0	10.700 ±30kHz	280 ±50kHz	650 max.	4.0 ±2.0dB	30 min.	330

(a)



(b)



(c)

Figure 4: Murata SFE10M7 ceramic IF filter specifications: (a) spec table, (b) passband detail, and (c) full band response.

Quadrature Demodulator

In analog integrated circuits used for FM radio receivers and the like, an FM demodulator known as a quadrature detector or quadrature discriminator, is quite popular. The schematic for the detector is shown in Figure 5.

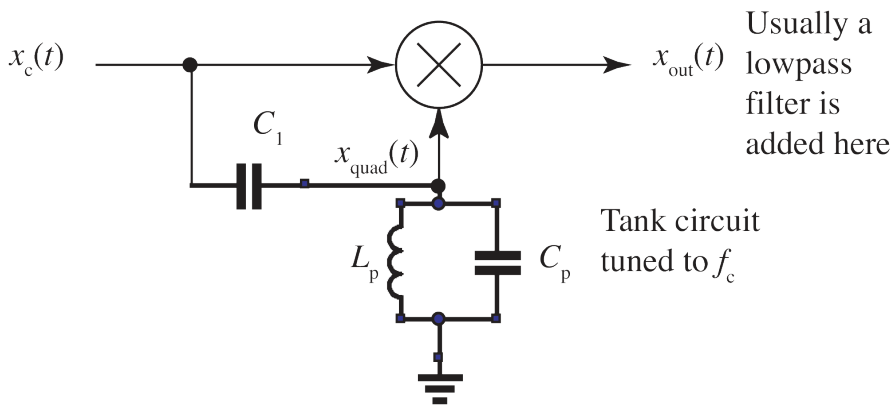


Figure 5: Quadrature detector schematic.

The input FM signal connects to one port of a multiplier (product device). A quadrature signal is formed by passing the input to a capacitor series connected to the other multiplier input and a parallel tank circuit resonant at the input carrier frequency. The quadrature circuit receives a phase shift from the capacitor and additional phase shift from the tank circuit. The phase shift produced by the tank circuit is time varying in proportion to the input frequency deviation. A mathematical model for the circuit begins with the FM input signal

$$x_c(t) = A_c \cos\left[\omega_c t + \phi(t)\right] \tag{2}$$

The quadrature signal is

$$x_{\text{quad}}(t) = K_1 A_c \sin\left[\omega_c t + \phi(t) + K_2 \frac{\phi(t)}{dt}\right] \tag{3}$$

where the constants K_1 and K_2 are determined by circuit parameters. The multiplier output, assuming a lowpass filter removes the sum terms, is

$$x_{\text{out}}(t) \frac{1}{2} K_1 A_c^2 \sin\left[K_2 \frac{d\phi(t)}{dt}\right] \tag{4}$$

By proper choice of K_2 the argument of the sin function is small, and a small angle approximation yields

$$x_{\text{out}}(t) \simeq \frac{1}{2} K_1 K_2 A_c^2 \frac{d\phi(t)}{dt} = \frac{1}{2} K_1 K_2 A_c^2 K_D m(t) \tag{5}$$

Laboratory Exercises – Wideband Receiver

The Colorado Springs market has a lot of FM broadcast stations. Several of them are likely familiar to you. Set–up the Agilent 4395A spectrum analyzer with a VHF flex antenna connected to a 50 ohm input, as shown in Figure 6. Set the span to 87.5 to 108 MHz. This will allow you to see the entire FM broadcast spectrum. At this resolution each FM station will appear as a spectral line.

Count the number of strong signals you see. For a few stations verify using the marker that the center frequency of each station falls at an odd multiple of 100 kHz and that the minimum spacing between stations is thus 200 kHz. Just from the spectrum analyzer display, and your knowledge of FM stations in the Colorado Springs market, clearly identify one of your favorite stations by setting the marker at the station center frequency.

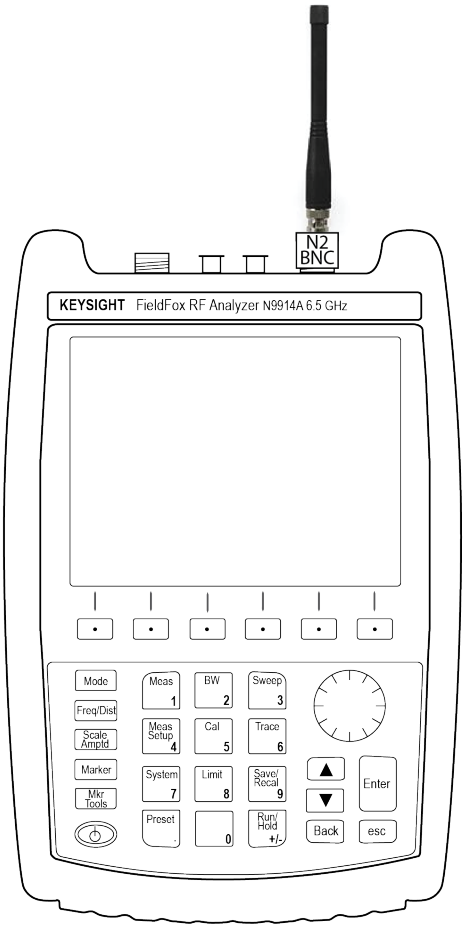


Figure 6: Keysight FieldFox N9914A with a VHF flex antenna mounted at an input port, to observe the FM broadcast signals present on the 87.5 – 108 MHz band.

Part a

Spectrally zoom into one of the stronger stations and measure the approximate occupied spectral bandwidth. Does it conform to needs of the 200 kHz spacing rule? The spectrum will be changes as the applied modulation, e.g. music or talking changes over time.

Part b

In the next step you will power up the wideband receiver. You will need to know how to set the LO frequency to receive FM broadcast stations. Develop two formulas for determining the required LO frequency to receive a station using either high–side or low–side LO tuning. Your *functions* should take as input the desired station frequency, e.g., 99.1 MHz and return the LO frequency. You need to take into account the fact that the LO frequency is doubled, as shown in Figure 1.

Part c

Finally, on to the receiver hardware. Configure the wideband receiver test set–up according to Figure 7. The photograph of Figure 8 will help you locate the power and signal input and output test points on the breadboard. Use a bench supply for the 3.3v. Be careful to not over–voltage the RFIC. Note that entire backplane of the board is ground. The LO signal needs to be connected to the board via an SMA–to–BNC adaptor. Take care with this connection so that you do not torque the SMA connector off of the breadboard. Set the LO level from the Agilent 33250 to about 600 mv pp, but not more than 1 v pp. Be sure to turn the RF output *on*. As the supply voltage is brought up to at least 5.3 volts (there is a 3.3 V regulator on the board) you should hear noise coming from the PC speakers. This indicates that the receiver is likely working, but is just not tuned to a station yet.

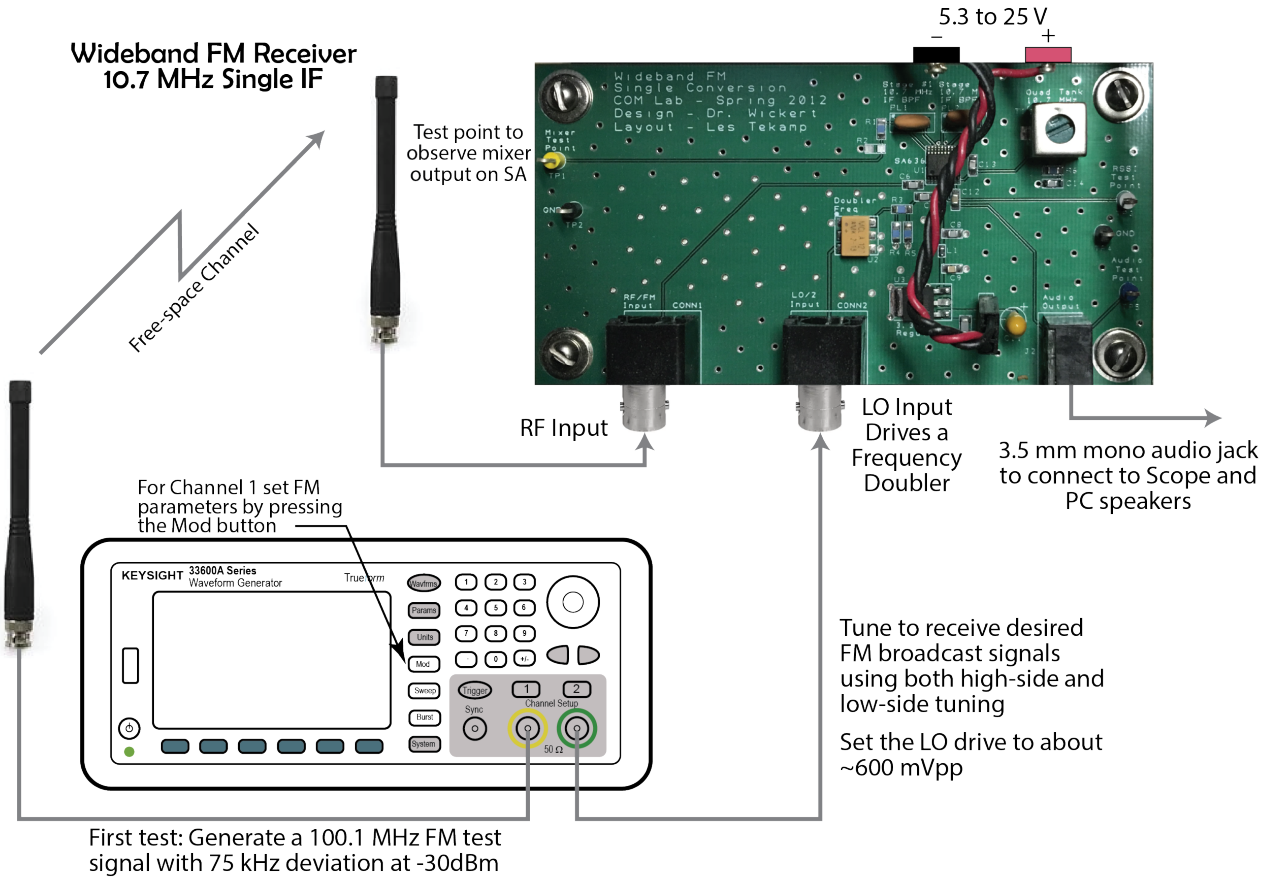


Figure 7: Wideband FM receiver test set–up.

Adjust the LO, using your formula, to tune in and demodulate one or more of the FM broadcast stations you earlier identified on the spectrum analyzer.

Part d

To actually see the superheterodyne principle in action place a probe using (1) a direct wire connection from an open SMA or BNC cable or (2) a newly developed active high impedance probe adaptor, or (3) the older Agilent 41802A active high impedance probe adapter, all connected as inputs to the N9914A spectrum analyzer to the mixer test point identified in the upper left corner of the board photo of Figure 7. Center the spectrum analyzer on 10.7 MHz with a span of about 200–400 kHz. As you tune the LO you should see the spectra of the FM stations, originally seen at 87.5–108 MHz, translated down to 10.7 MHz. Verify that the ordering of the stations, low frequency to high frequency, or high frequency to low frequency is dependent upon the LO tuning being either high–side or low–side to the carrier. The FieldFox N9914A test configuration is shown in Figure 8.

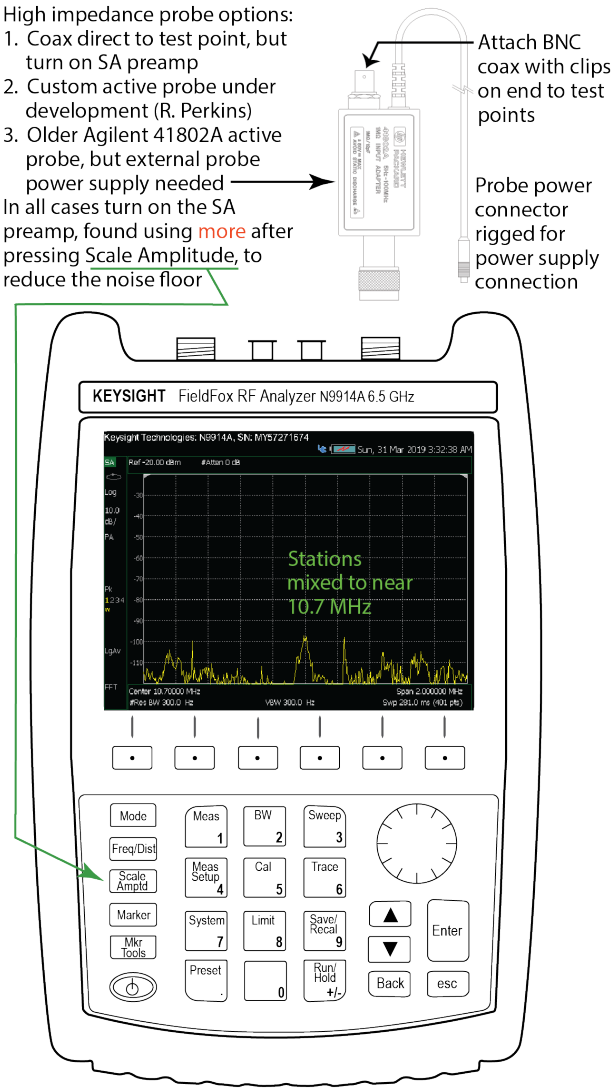


Figure 8: N9914A configuration for probing the mixer output test point of Figure 7.

Part e

In this final wideband receiver task, you will investigate how a bandpass filter at the receiver input is used to suppress an interfering signal at the image frequency. Recall from the basic properties of a frequency translation system, the mixing operation has two possible solutions, *high–side* and *low–side* tuning. In the case of the wideband FM receiver the IF frequency is 10.7 MHz so with high–side tuning the local oscillator frequency at the mixer (regardless of the doubler) is set to $f_{\mathrm{LO}} = f_c + f_{\mathrm{IF}} = f_c + 10.7$ MHz. This LO frequency also is acting as a valid LO for low–side tuning at the second mixer input frequency of $f_{\mathrm{image}} = f_{\mathrm{LO}} + f_{\mathrm{IF}}$ or equivalently $f_{\mathrm{image}} = f_{\mathrm{c}} + 2f_{\mathrm{IF}} = f_c + 21.4$ MHz. In the end this means that if both f_c and f_{image} are present at the receiver input they will land on top of each other at the mixer output, resulting in strong interference. Placing a bandpass filter at the input to the receiver is a means to reject, or at reduce the power level of the image frequency signal. The system block diagram is shown in Figure 9.

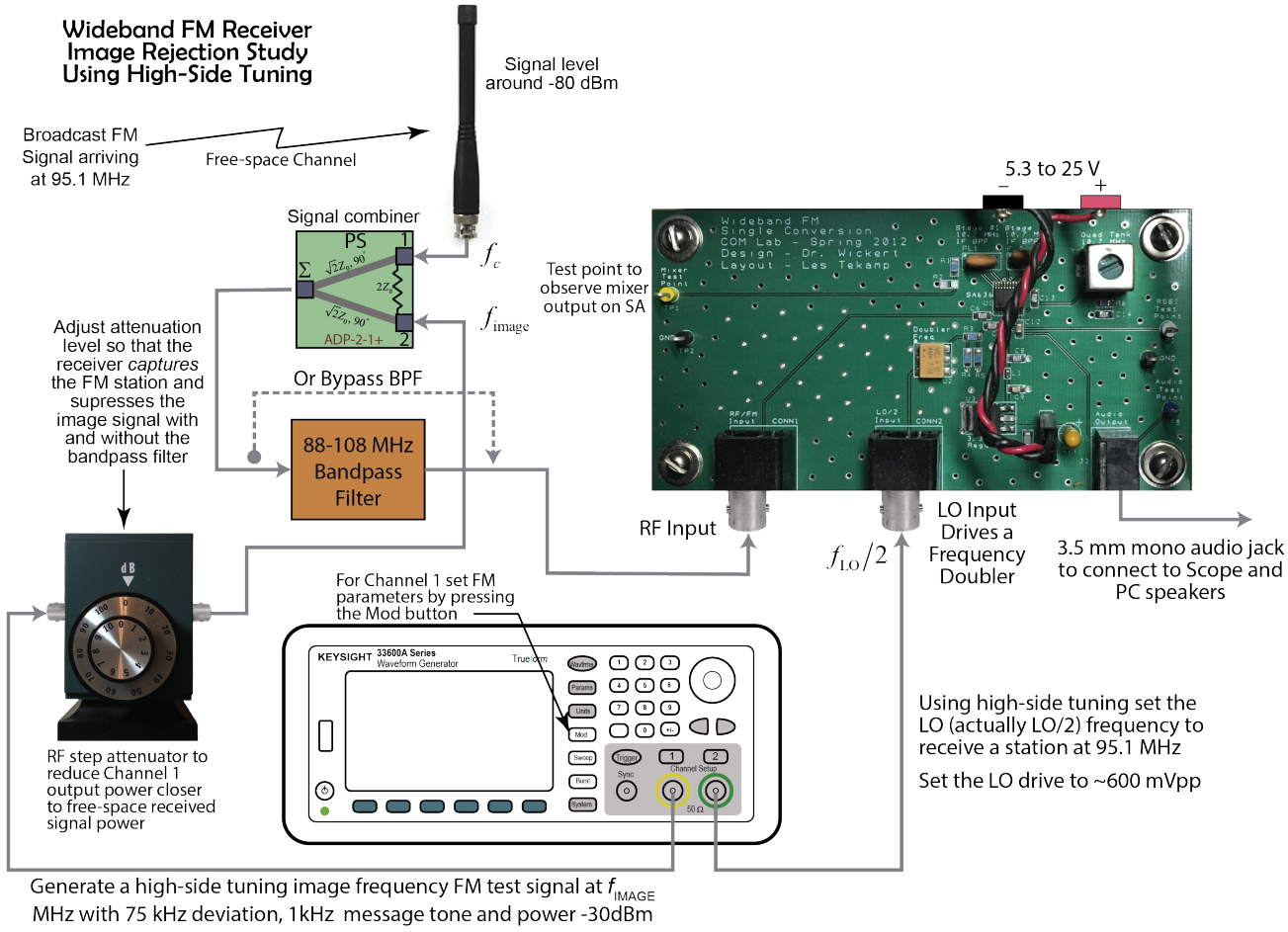


Figure 9: Image signal rejection using a receiver bandpass filter.

Configure the test equipment as shown in the block diagram, noting that two components are on the RF Board and a *step attenuator*, Weinchel Engineering Model 3051–110–88 with BNC connectors, is used to adjust the image signal power level at the input. The effective receiver input is at the power combiner (power splitter running backwards), where each input signal path receives

an additional attenuation of ~3dB.

1. Adjust the LO/2 signal from Channel 2 of the 33600A to receive the FM station at 95.1 MHz using high–side tuning. Use PC speakers to listen to the station and make sure it is clear. Start with the BPF that follows the power combiner *in circuit*. Turn on Channel 2 with the attenuator initially at 0 dB. You should find that the image signal *captures* the receiver demodulator and dominates what you hear through the speakers. Increase the attenuation until the interfering tone just disappears, that is you do not hear it in the audio output. Make note of the step attenuator setting.
2. Repeat (1) with the BPF bypassed, that is a straight coax connection between the power combiner and the receiver RF input. Initially turn Channel 2 off to be sure you have clear reception of the station on 95.1 MHz. Turn on Channel 2 and adjust the step attenuator until the interfering tone just disappears. Make note of the step attenuator setting.
3. Comment on additional image rejection in dB as a result of the using BPF.

To enhance your understanding of (1) and (2) use the spectrum analyzer in place of the receiver RF input. To see the weak signals use the *preamp* to drop the noise floor.

Narrowband FM Receiver

The block diagram for the narrowband receiver is given in Figure 10. The low–noise amplifier (LNA) is not implemented at this time, but a tank circuit is located at the input to provide some bandpass filtering. A short wire (clip lead) will serve as the antenna in the experiment.

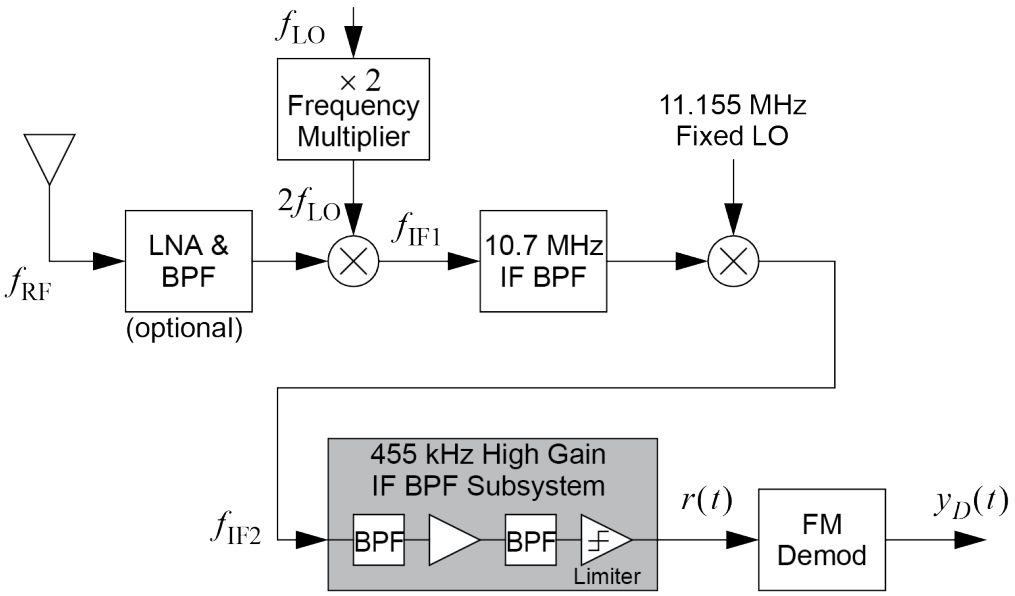


Figure 10: Narrowband dual conversion FM receiver block diagram.

This receiver also requires and external local oscillator (LO). In order to receive VHF signals, in particular narrowband FM broadcasts, such as the national weather service http://www.nws.noaa.gov/nwr/nwrbro.htm, which has carrier frequencies near 162.4 MHz, the output of the Agilent 33250, which will serve as the LO, again needs to be frequency doubled. Other VHF band services using narrowband FM include police and fire departments and Ham radio operators near 146.94 MHz.

The narrowband receiver utilizes a dual conversion superheterodyne approach. The first mixer output is processed with a single stage IF filter centered at 10.7 MHz. The 10.7 MHz IF signal then enters a second mixer with a fixed LO of 11.155 MHz. The difference frequency is at 455 KHz. The 455 kHz difference frequency is processed with a multistage IF amplifier, with the 455kHz IF passband shaping formed using ceramic filters. The nominal bandwidth of each filter is 10 kHz. Note from the schematic of Figure 11, the ceramic filters 10.7 MHz and 455 kHz, are external to the NXP SA602 and SA606 RFICs.

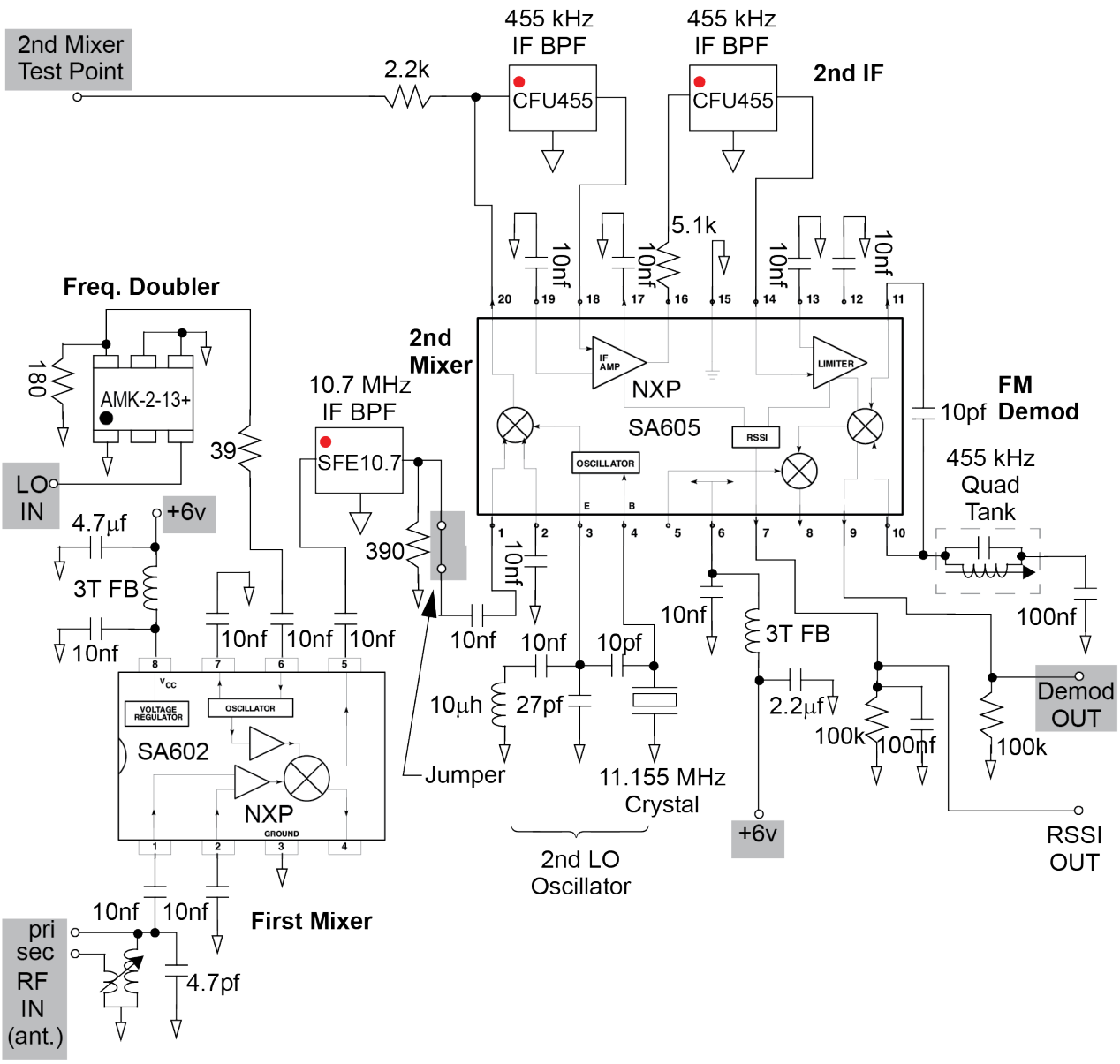


Figure 11: Dual conversion VHF FM receiver schematic based on the NXP SA605 IF subsystem plus the NXP SA602 mixer.

The final stage is the FM demodulator, which in this design is again a quadrature detector.

IF Filters

For the case of the narrowband FM receiver the signal bandwidth is much narrower than in the wideband case (10 kHz versus 200 kHz). The NOAA stations are spaced by 25 kHz from 162.400 – 162.550 MHz. The NOAA weather stations in Colorado are listed in Table 1.

Table 1: NOAA stations throughout colorado.

Site Name	Transmitter Name	Call Sign	Frequency	Power	WFO

Denver	Glendale	KEC76	162.550	300	Boulder, CO
Canon City	Canon City	KJY81	162.500	300	Pueblo, CO
Anton	Anton	KJY84	162.425	300	Boulder, CO
Fort Morgan	Fort Morgan	KWN40	162.525	300	Boulder, CO
Durango	Durango	KWN54	162.425	300	Grand Junction, CO
Steamboat Springs	Walton Peak	KWN56	162.525	300	Grand Junction, CO
Lamar	Prowers County	KWN60	162.525	1000	Pueblo, CO
Montrose	Buckhorn Lakes	KXI90	162.450	300	Grand Junction, CO
Franktown	Franktown	WNG550	162.450	300	Boulder, CO
Walsenburg	Walsenburg	WNG579	162.450	300	Pueblo, CO
Springfield	Baca County	WNG664	162.400	1000	Pueblo, CO
Dillon	Dillon	WNG737	162.400	300	Boulder, CO
Bethune	Bethune	WWF77	162.525	100	Goodland, KS
La Junta	Miller Mesa	WWG23	162.500	100	Pueblo, CO
Glenwood Springs	Sunlight Peak	WWG43	162.500	100	Grand Junction, CO
Fowler	Olney Spring	WWG44	162.425	100	Pueblo, CO
Boyero	Boyero	WWH32	162.450	100	Boulder, CO
Deer Trail	Deer Trail	WXJ45	162.500	100	Boulder, CO
Greeley	Point of Rocks	WXM50	162.400	300	Boulder, CO
Mead/Longmont	Highland	WXM51	162.475	300	Boulder, CO
Pueblo	Pueblo	WXM52	162.400	300	Pueblo, CO
Sterling	Sterling	WXM53	162.400	300	Boulder, CO
Alamosa	Agua Ramon Mountain	WXM54	162.475	300	Pueblo, CO
Grand Junction	Rapid Creek	WXM55	162.550	300	Grand Junction, CO

Colorado Springs	Cheyenne Mountain	WXM56	162.475	100	Pueblo, CO
Wray	Wray	WXM87	162.475	300	Goodland, KS
Fort Collins/Ault	Black Hollow	WXM92	162.450	300	Boulder, CO
Eagle	Eagle	WZ2518	162.450	300	Grand Junction, CO
North Cottonwood	Grand County	WZ2544	162.500	300	Boulder, CO

Number of Stations in Colorado = 29

The 10.7 MHz first-conversion IF filter is the same 280 kHz bandwidth SFU 10.7, as used in the wideband design. For the second IF, the ceramic filters have a 6 dB bandwidth of 10 kHz. The filter specifications and plots of the magnitude and group delay response are shown in Figure 12.

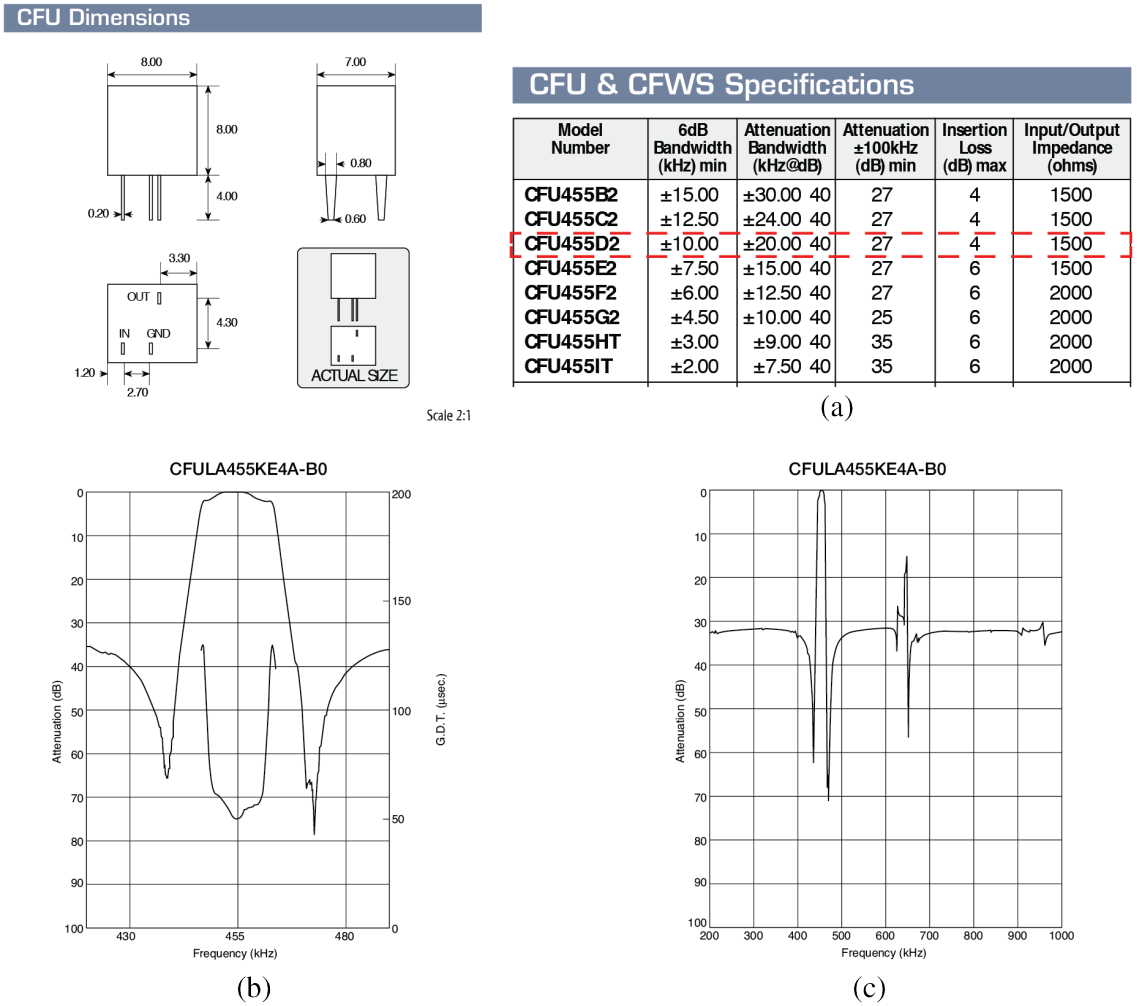


Figure 12: Murata CFU 455D2 ceramic IF filter specifications: (a) spec table, (b) passband detail of a similar design, and (c) full band response of a similar design.

Laboratory Exercises: Narrowband Receiver

In the Colorado Springs area the NOAA station easiest to receive is WXM56 from Colorado Springs. Looking at Table 1, we see that this station has its transmitting antenna on Cheyenne Mountain. The transmission power is small, only 100 W, compared with broadcast FM which is over 1000 W.

Part a

Using the VHF flex antenna, as shown in Figure 6, try to find WXM56 on the FieldFox N9914A spectrum analyzer. You have to reduce the attenuation on the analyzer front–end to zero dB and maybe turn in the analyzer *preamp*, to find the signal.

Part b

In the next step you will power up the narrowband receiver. You will need to know how to set the LO frequency to receive narrowband FM signals, like NOAA stations. Develop two formulas for determining the required LO frequency to receive a station using either high–side or low–side LO tuning. Your *functions* should take as input the desired station frequency, e.g., 162.475 MHz and return the LO frequency. You need to take into account the fact that the LO frequency is doubled and the receiver is dual–conversion, as shown in Figure 10. Given that the Keysight 33600A only tunes to 140 MHz, will you be limited in practice?

Part c

Finally, on to the receiver hardware. Configure the narrowband receiver test set–up according to Figure 13, but don't connect the LO just yet. The photograph contained in Figure 13 will help you locate the power and signal input and output test points on the PCB. Use a bench supply for the 6 V. A voltage regulator on the board insures that 6 volts in maintained for the circuitry. As the supply voltage is brought up to 6 volts you should hear noise coming from the PC speakers. This indicates that the receiver is likely working, but is just not tuned to a station yet.

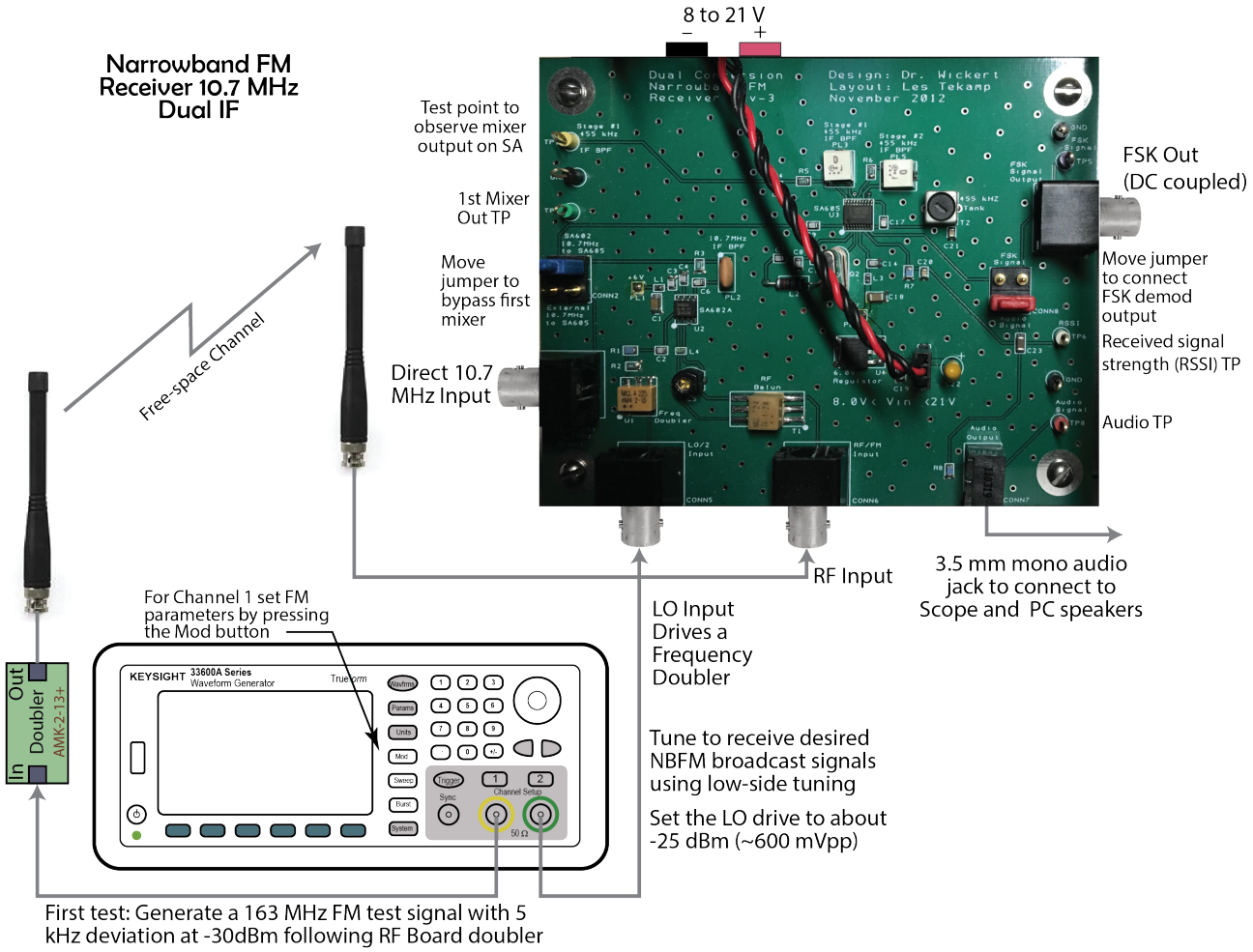


Figure 13: Narrowband FM receiver test set–up.

Part d

Before connecting the LO to the frequency doubler you will test that the receiver functions properly from the second mixer input to the demodulated output. This requires you to move the blue jumper, found in the left middle of the board photo of Figure 13, to the *External 10.7 MHz* position. This is located about in the center of the board between the two RFICs. Configure the Keysight 33600 to produce a narrowband FM signal with $f_c = 10.7$ MHz and a deviation of about 2 kHz. Set the output to about 100 mV pp. Just brining the BNC cable lead end close to the 10.7 MHz IF input should quiet the noise in the PC speakers and you should hear the modulation tone. This verifies that the backend of the receiver is working properly.

Part e

Now connect the LO (Agilent 33250) to the board. Set the LO level from the Keysight 33600A to ~600 mV pp, but not more than 1 Vpp. Be sure to turn the RF output *on*. Place a clip–lead on one of the two receiver input test points in the lower left of the board photo in Figure 13. Reconnect the *blue* jumper between the first–mixer output and the 10.7 MHz IF input.

Adjust the LO, using the one valid formula, to tune in and demodulate NOAA station WXM56. The overall sound quality of this signal will be poorer than the broadcast FM stations you hear from the wideband receiver. The signal is also rather weak, so you may have to position the antenna carefully to get good signal strength.

Part f

To actually see the superheterodyne principle in action in this dual–conversion receiver, place a probe from a high impedance input on the spectrum analyzer to 2nd mixer test point identified in the upper center of the breadboard photo of Figure 12. Center the spectrum analyzer on 455 kHz with a span of about 50 kHz. As you tune the LO you should see the spectra of the WXM56 station, originally at 162.475 MHz, translated down to 455 kHz.

Digital Modulation

In this section you explore digital modulation/demodulation transmission system using a combination of the 33600A signal generator, the RF board frequency doubler, and a Jupyter notebook running `pyaudio_helper` code that outputs and receives a digital bit stream. Here you are doing what is known as *hardware–in–the–loop* testing. The top level system block diagram is shown in Figure 14.

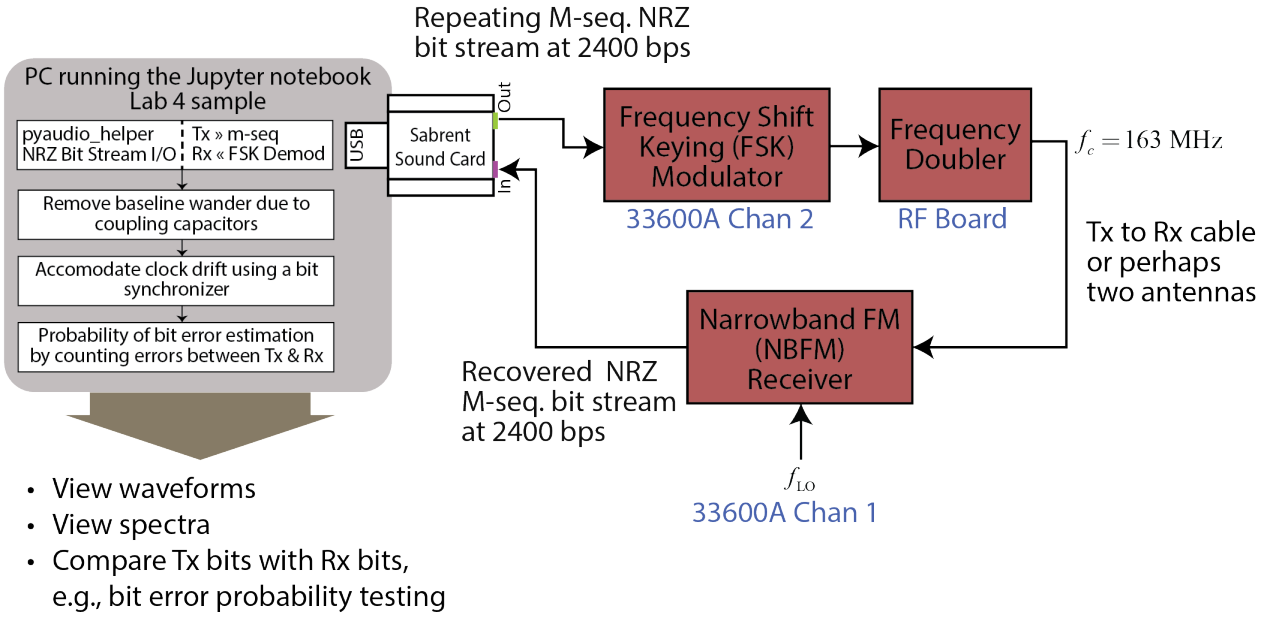


Figure 14: FSK transceiver top–level block diagram

Here we use `pyaudio_helper` in combination with the Keysight 33600A generators to generate frequency–shift keyed (FSK) modulation on a carrier frequency $f_c = 163$ MHz, data rate of $R_b = 2400$ bps with 2 kHz peak deviation before the doubler, hence 4 kHz peak deviation after the doubler. A PN test sequence waveform is generated in the Jupyter notebook sample and converted to an output waveform using a USB sound card. The demodulated output from the narrow band FM receiver in taken from the FSK output by moving the red jumper to the *FSK Signal* position (more on this later). The FSK signal is fed into the USB sound card mic input for further processing inside the Jupyter notebook sample.

The sample notebook contains both Tx and Rx processing algorithms. In particular bit synchronization is to process the receiver/demodulated bit stream before bit error probability (BEP) measurements can be performed. Along the way *baseline wander* [2] must be dealt with due to a DC block on the PC audio card mic input.

Starting with a Simple Audio Loopback

Before configuring the entire transceiver system, consider a simple audio loopback from the USB audio card output directly into the mic input of the Sabrent device. A block diagram that depicts this configuration, including a scope screen shot, is given in Figure 15.

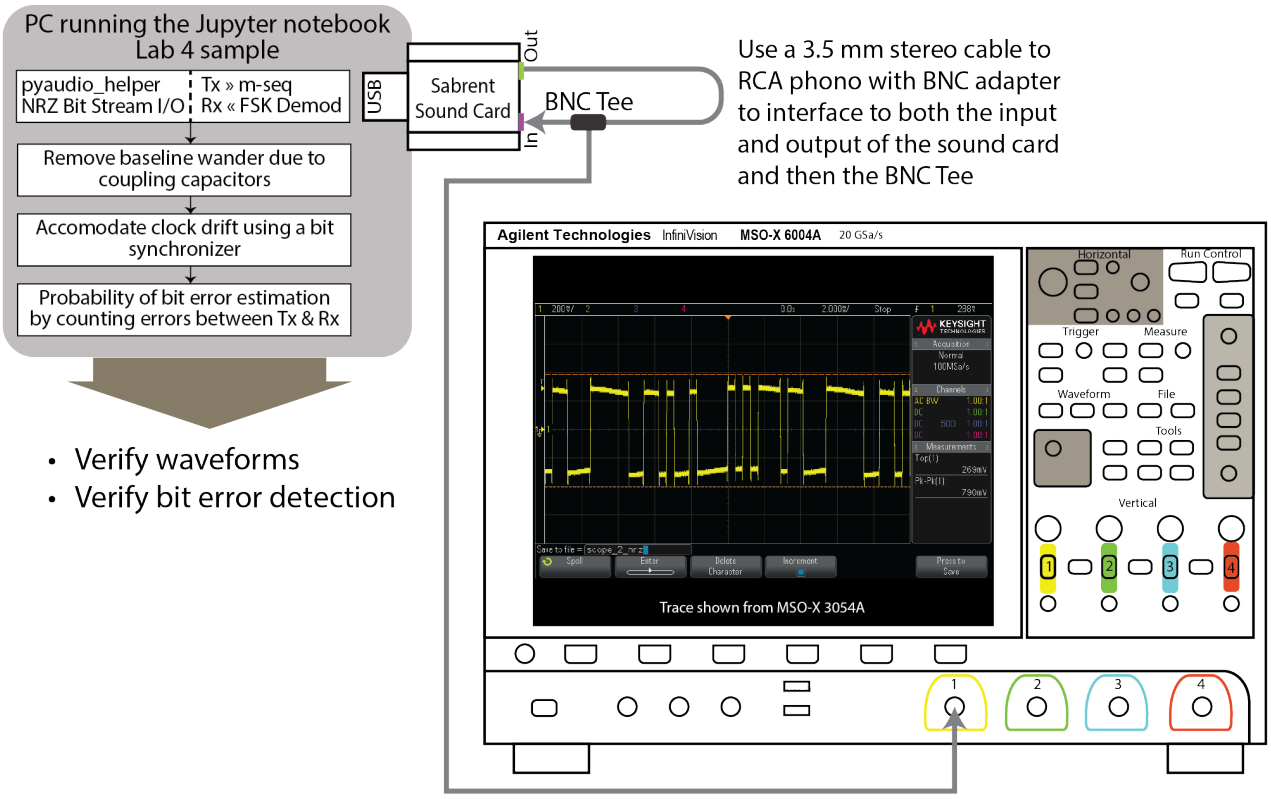


Figure 15: Simple loopback NRZ data stream test from `pyaudio_helper` output to `pyaudio_helper` input.

Inside the Jupyter notebook sample you run the opening notebook cells and then see what audio devices are available on your system.

```
1  pah.available_devices()
```

```
{0: {'name': 'Microsoft Sound Mapper – Input', 'inputs': 2, 'outputs': 0},
 1: {'name': 'Microphone (3– USB Audio Device', 'inputs': 2, 'outputs': 0},
 2: {'name': 'Microphone (Realtek Audio)', 'inputs': 2, 'outputs': 0},
 3: {'name': 'Microsoft Sound Mapper – Output', 'inputs': 0, 'outputs': 2},
 4: {'name': 'Speakers / Headphones (Realtek ', 'inputs': 0, 'outputs': 2},
 5: {'name': 'Speakers (3– USB Audio Device)', 'inputs': 0, 'outputs': 2}}
```

It is important that the USB audio device be plugged in before you start the notebook kernel. If you do not do this PyAudio will not find the device and properly enumerate the desired device index. The device name you are looking for is *USB Audio Device*, both as a microphone input and a speaker output. If the device is not found you pull down the *Kernel* menu in Jupyter and select *Restart Kernel...*, and then run the initialization cells once again, working you way down to the cells that follow: *Stream NRZ Data Bits over the Link as an M–Sequence*.

Generate One *M*–Sequence Period

```
M = 5
N_bits = 2**M - 1
data = ss.PN_gen(N_bits,M)
x_NRZ, b_pulse = ss.NRZ_bits2(data,20,pulse='rect')
#x_NRZ *= 0.2
```

Provide Gain Sliders on the Two Output Streams

For this application these controls are best left at the default settings of 1.0/1.0. Changing the left gain does change the amplitude of the *M*–sequence.

```
L_gain = widgets.FloatSlider(description = 'L Gain',
                continuous_update = True,
                value = 1.0,
                min = 0.0,
                max = 2.0,
                step = 0.01,
                orientation = 'vertical')
R_gain = widgets.FloatSlider(description = 'R Gain',
                continuous_update = True,
                value = 1.0,
                min = 0.0,
                max = 2.0,
                step = 0.01,
                orientation = 'vertical')

#widgets.HBox([L_gain, R_gain])
```

PyAudio Call Back Function

This function contains the actual DSP code that produces the non–return–to–zero (NRZ) waveform output on the left channel, with gain slider, via an object from the *contiguous loop class* defined at the top of the notebook, and on the right channel applies the right channel gain from the slider. Both channels are output, but the left channel is the only signal of interest.

```
# L and Right Gain Sliders
def callback(in_data, frame_count, time_info, status):
    global DSP_IO, L_gain, R_gain, x_loop_mono #x_loop_stereo
    DSP_IO.DSP_callback_tic()
    # convert byte data to ndarray
```

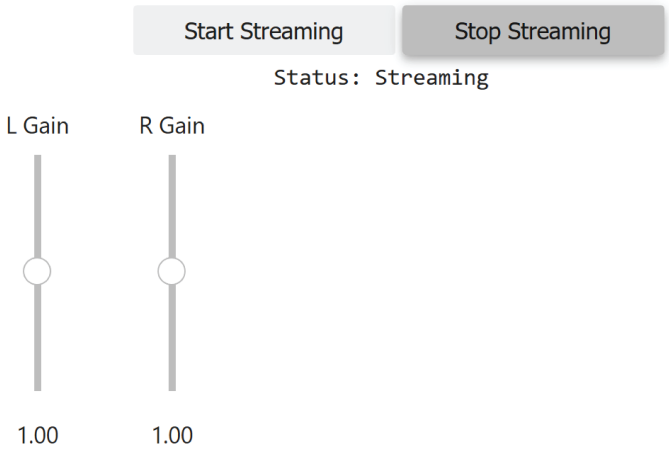
```
    in_data_nda = np.frombuffer(in_data, dtype=np.int16)
    # separate left and right data
    # The right samples will contain the input from the FSK demod output
    x_left,x_right = DSP_IO.get_LR(in_data_nda.astype(float32))
    # Use a loop object as a source of mono NRZ waveform contiguous samples
    # Note since wave files are scaled to [-1,1] we are scaling to
    # rescale to the dynamic range of int16
    #new_frame = x_loop_stereo.get_samples(frame_count)
    new_frame = x_loop_mono.get_samples(frame_count)
    x_left = 20000*new_frame
    #x_right = 20000*new_frame[:,1]
    #************************************************
    # DSP operations here
    y_left = x_left*L_gain.value # The Tx NRZ bit stream
    y_right = x_right*R_gain.value # The Rx NRZ bit stream

    #************************************************
    # Pack left and right data together
    y = DSP_IO.pack_LR(y_left,y_right)
    # Typically more DSP code here
    #************************************************
    # Save data for later analysis
    # accumulate a new frame of samples
    DSP_IO.DSP_capture_add_samples_stereo(y_left,y_right)
    #************************************************
    # Convert from float back to int16
    y = y.astype(int16)
    DSP_IO.DSP_callback_toc()
    # Convert ndarray back to bytes
    #return (in_data_nda.tobytes(), pyaudio.paContinue)
    return y.tobytes(), pah.pyaudio.paContinue
```

Instantiating a `DSP_io_stream` Object

With GUI controls defined and the callback function defined we are finally ready to do input/output streaming. The variable `T_record` set the record and playback time for the waveforms input and output from the USB audio interface. Setting `T_record` to 30s is reasonable. Capturing for longer time periods can be problem if your system has limited RAM to temporarily store the capture. For initial testing, where all you want to do is evaluate the analog waveforms, then you can set `T_capture = 0` which allows the stream to run forever, but no capture buffer is filled. The scope screen shot in Figure 15 was created in this way.

The object `x_loop_mono` takes the one full period of the *M*–sequence, at 20 sample per bit, to be repeated over and over again as the `DSP_IO` object streams audio through the PC audio subsystem. The third line is creates the `DSP_IO` object to take it input from device `1` and output to device `5`, at 48 ksps. The fourth line kicks off audio streaming with two channels. The GUI width are displayed below to allow interactivity while streaming is active.

```
T_record = 0 # in s; 0 <==> infinite, but no capture, typical 5 to 30s
x_loop_mono = loop_audio_contig(x_NRZ)
DSP_IO = pah.DSP_io_stream(callback,1,5,fs=48000,Tcapture=2*T_record)
DSP_IO.interactive_stream(2*T_record,2)
widgets.HBox([L_gain, R_gain])
```



During the loopback the audio level is higher than what is fed back from the narrow band FM receiver. The USB audio input has high gain for working with microphones. To reduce the gain and avoid overload of the PC audio subsystem, you need to *right click* over the speaker icon in the lower right *task bar* and then choose Sounds from the menu. From the dialog box that appears click the *Recording* tab and then click the microphone corresponding to the *USB Audio Device*. Then in the lower right click the *Properties* button and finally click the *Levels* tab. You should now see something similar to Figure 16 below.

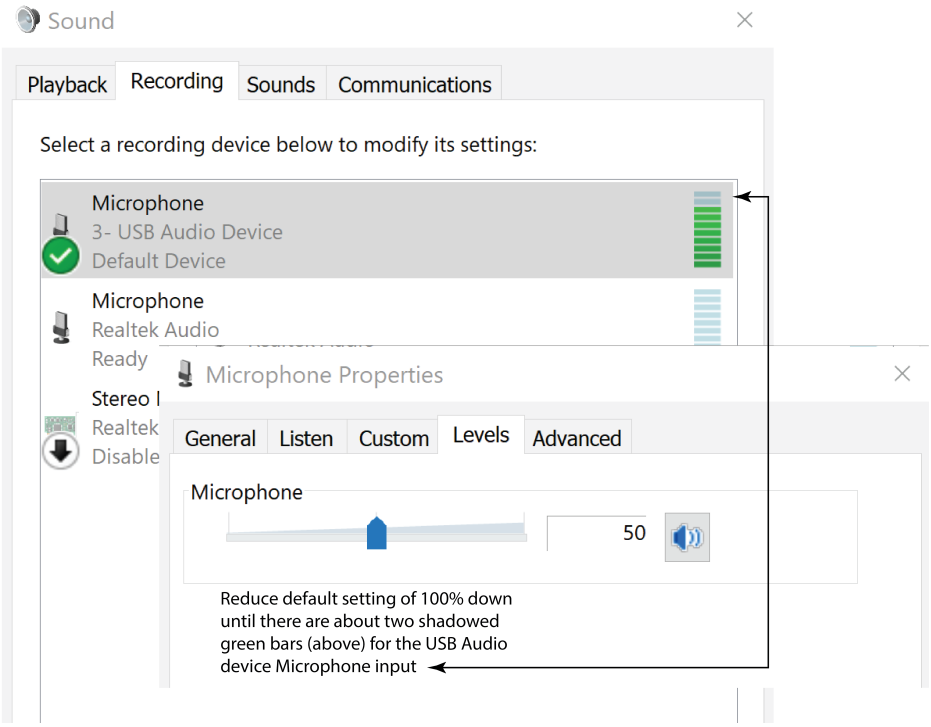


Figure 16: Adjusting gain levels in the PC sound system to avoid overloading the mic input.

With infinite streaming coming from `pyaudio_helper` in the Jupyter notebook, finally adjust the microphone gain level until you see two grayed out bars as shown in Figure 16. You can also view the waveform on the scope and should see a waveform similar to Figure 15. With the gain level reduced now move on and attempt a 30s capture, and then post process the capture in the Jupyter notebook. To do this simply change `T_capture = 30`, reload the cell, then click *Start Streaming*. Streaming will stop after 30s.

When streaming stops you can immediately view the results using the cell that follows the heading *Take a Quick Look at the Capture*. The corresponding plots of the left (Tx) and right (Rx) channels is shown in Figure 17

```
fs = 48000
Nstart = 10000 # Wait about 163 ms for the Rx signal to arrive in the
data_capture buffer
Nspan = 2000
t_capture = arange(0,len(DSP_IO.data_capture_left))/fs*1000 # ms
plot(t_capture[Nstart:Nstart+Nspan],DSP_IO.data_capture_left[Nstart:Nstart+N
span])
plot(t_capture[Nstart:Nstart+Nspan],DSP_IO.data_capture_right[Nstart:Nstart+
Nspan])
title(r'Raw Capture Held in the DSP_IO Object')
ylabel(r'int16 Scaled Amplitude')
xlabel(r'Time (ms)')
legend((r'Tx PN Code Signal',r'FSK Demod Signal'),loc='upper right')
grid();
```

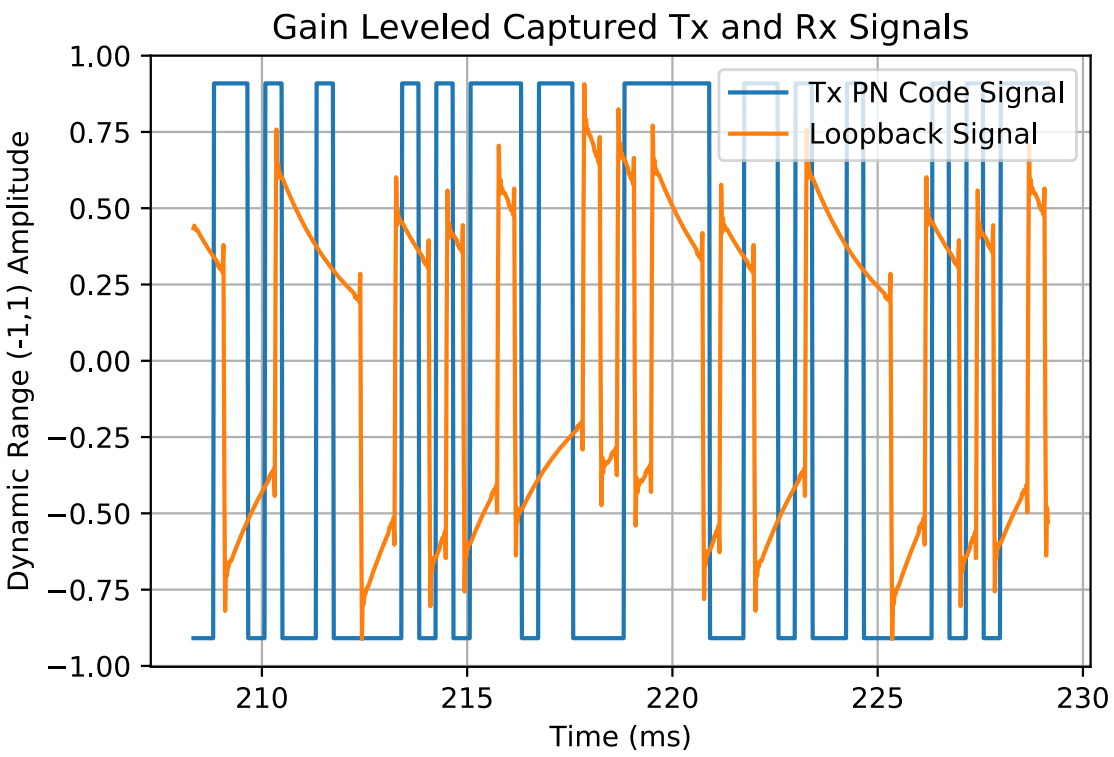


Figure 17: Quick look at a portion of the captured left (Tx waveform) and right (Rx waveform) channels.

Gain Normalizing the Capture

Notice that the raw amplitude levels from the PC audio subsystem are signed integers of type `int_16`, so before you archive a capture you scale to a $(-1, 1)$ range and stack horizontally the left and right channels for saving using `ss.to_wav(filename, rate, x)`:

```
left_right_2400bps =
hstack((array([DSP_IO.data_capture_left]).T/(1.1*max(DSP_IO.data_capture_left
)),

array([DSP_IO.data_capture_right]).T/(1.1*max(DSP_IO.data_capture_right))))
# Need to be scaled to (-1,1) for wave
ss.to_wav('left_right_2400bps_lcap_LoopBack.wav',48000,left_right_2400bps)
```

For the remainder of the DSP employed by the Jupyter notebook sample you work with the normalized $(-1, 1)$ amplitude capture array, `left_right_2400bps`:

```
fs, left_right_2400bps = ss.from_wav('left_right_2400bps_lcap_LoopBack.wav')
print('Capture period from sample count = %4.2f s' %
(left_right_2400bps.shape[0]/48000,))
```

```
Capture period from sample count = 20.03 s
```

Baseline Wander Correction Due to AC Coupling

Note the drastic difference between the input waveform (scope screenshot of Figure 15) and the corresponding capture in discrete–time form as shown in Figure 17. The coupling capacitor of the USB audio device output introduces some *droop* as seen in long strings of 1's and 0's. The coupling capacitor of the mic input has an even higher *highpass* cutoff frequency, hence in Figure 17 we a more serious form of *droop* known as *baseline wander* (BW). With DC coupling this would not be an issue, but the USB audio device is designed for speech, not an NRZ data stream. The signal is clean and relatively free of noise, so the baseline wander correction system of Figure 18 can be applied. In the sample notebook see the code cell following the heading *Baseline Wander Correction*. See [Baseline wander – EECS: www–inst.eecs.berkeley.edu](#) for more detail.

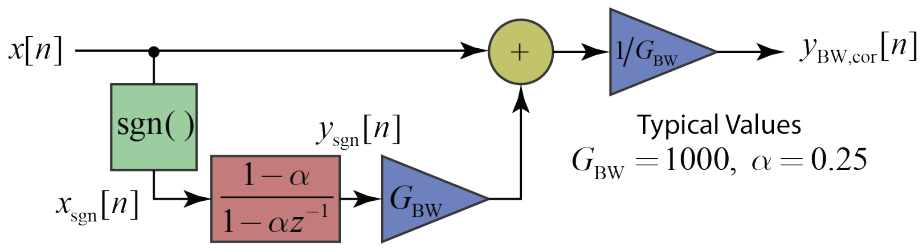


Figure 18: Baseline wander (BW) correction system block diagram, as implemented in the Jupyter notebook sample.

Note $\mathrm{sgn}()$ is the signum function (`sign()` in Python). The filter is a lowpass filter, so the low frequency content of a *hard–decision* version of $x[n]$ is gained boosted by G_{BW} to restore the DC blocked by the coupling capacitor. With significant receiver noise present on $x[n]$, the DC restoration property of the system fail to work as intended. Here, with a high quality signal, the correction system works well as shown in Figure 19.

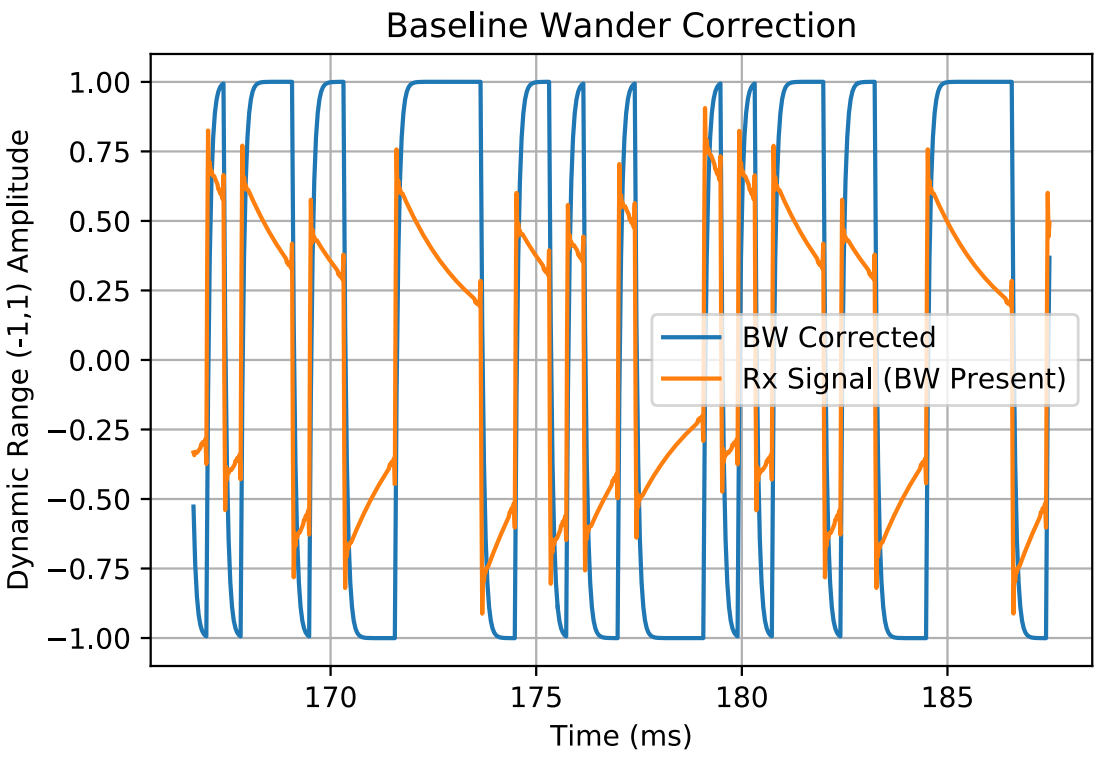


Figure 19: Baseline wander correction system block diagram, as implemented in the Jupyter notebook sample.

Managing Sampling Clock Drift

Even with the simple loopback configuration the action of ADC followed by DAC, with ~165ms latency, the sampling clock phase drifts enough that the 20 samples per bit are no longer perfectly aligned on the bit transitions. In a most digital communication systems transmit and receive sampling clocks are asynchronous. Yes, they are based on crystal oscillators, but drift is inevitable. A *bit synchronizer* resolves this issue and effectively derives a clock to match the clock phase of the received signal, and hence allow the bit stream waveform to be converted by to 1's and 0's sampled *mid–bit* or in the case of a noisy signal at the *optimal* sampling instant, once per bit.

In this lab a simple *pure digital* bit synch, known as SCCS [3] is included in the Jupyter notebook sample. Additional algorithms can be found in the module `sk_dsp_comm.synchronization` . The input to the SCCS bit synch is converted to ± 1 values before processing:

```
rx_symb_d, clk, trac = sccs_bit_sync(sign(y_BW_cor),20)
```

The array `trac` contains the optimum sampling instants, modulo `Ns`, for each recovered bit. The results are shown in Figure 20.

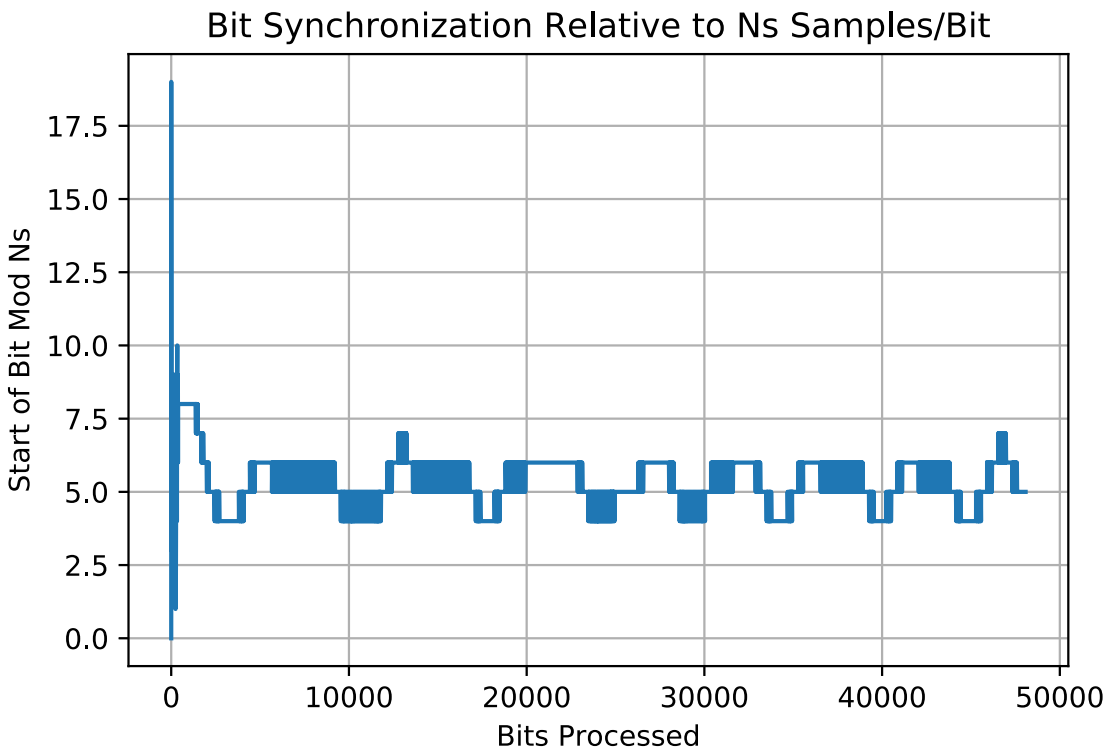


Figure 20: The SCCS bit synchronizer tracking signal corresponding to the optimum sampling instant modulo `Ns`.

Note: Tracking is not *locked* until a signal is actually present and the algorithm has acquired the optimum sampling instant. Once the SCCS is tracking it tends to *hunt* around the optimal timing instant modulo the number of samples per bit, `Ns`. It generally varies over three values, say 4,5,6, which for the case of `Ns=20` may straddle the wrapping point, i.e., 18,19,0 in a modulo `Ns` sense.

Bit Error Checking

The transmitted NRZ bits at `Ns` samples per bit are first down sampled to just one sample per bit. The bits returned from `sccs_bit_sync()` in `rx_symb_d` then compared with `tx_bits`. The time delay due to PyAudio and other analog processing the Tx–Rx link means that the two bit patterns need to first be brought into alignment. The function `sk_dsp_comm.digitalcom.bit_errors()` takes care of the alignment problem using cross correlation. This assumes that the bit errors are not so numerous so as to eliminate a correlation peak. Since the transmit bit stream is a repeating *m*–sequence, the bit streams may not be that far out of alignment, modulo the sequence period. To reduce the Tx waveform into one sample per bit you *downsample* and also transform ± 1 bit levels to $0/1$:

```
tx_bits = int64(ss.downsample(sign(left_right_2400bps[:,0]),20)+1)//2
```

On the Rx waveform you trim away nominally 500 bits, which corresponds to the delayed arrival of the received signal, and again transform ± 1 levels to $0/1$. Because of the relatively high signal–to–noise ratio (SNR) of the received signal, no errors occur:

```
N_bits, N_errors = dc.bit_errors(tx_bits,int16((rx_symb_d[500:]+1)/2))
print('N_bits = %d, N_errors = %d, BEP = %1.2e' % (N_bits,N_errors,
N_errors/N_bits))
```

```
kmax =  0, taumax = 6
N_bits = 47569, N_errors = 0, BEP = 0.00e+00
```

The `bit_errors` function automatically aligns the tx & rx patterns and the value `taumax` tells you the alignment skew found via cross correlation.

Laboratory Exercises: Digital Modulation

Now its time implement the full system and take some measurements of the system originally described at a high level in Figure 14. Figure 21 shows the full transceiver hardware configuration for transmitting at 163 MHz using a frequency doubler. A synopsis of the transceiver operation is given at the bottom of the figure.

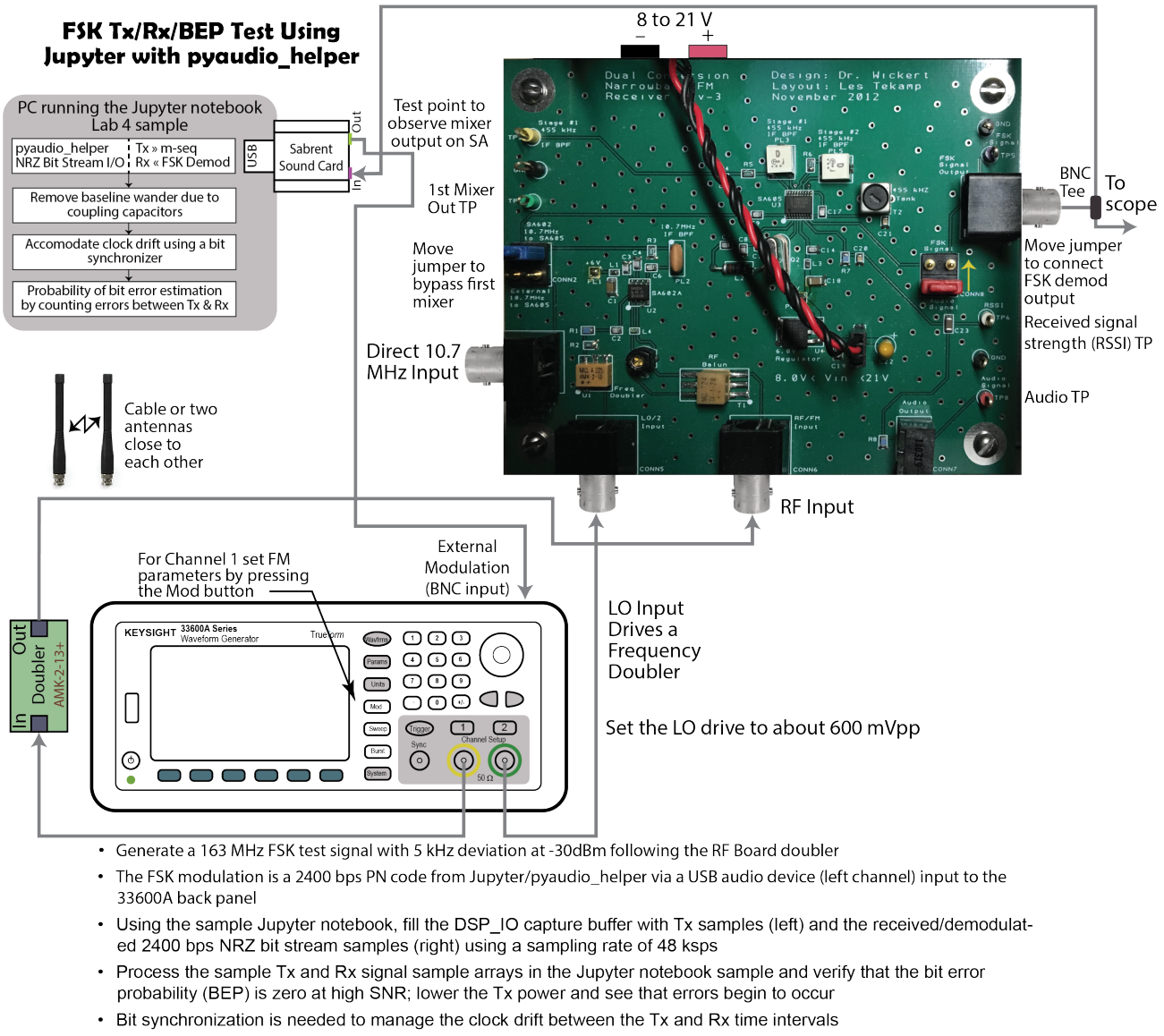


Figure 21: Narrowband FSK transceiver test set–up, including the USB audio card interface to Jupyter notebook to provide an NRZ bit stream for the transmitter and stream capture and process the FSK demodulator output from the narrow band FM receiver in Python.

To get a DC coupled output from the FM discriminator use *FSK Out* BNC connector seen on the upper right side of the board photo in Figure 21. You then need to move the position of the red jumper. Unfortunately the DC coupled output from the radio board is lost when it passes through the coupling capacitor of the mic input. *Baseline wander* is introduced by the coupling capacitor which ultimately is corrected in Python code. Furthermore, because the sampling clock drifts over time a bit synchronization algorithm is also needed to allow bit comparison between the transmitted PN bits and the received bits.

Part a

Set up the generators of the 33600A as follows:

1. Use Channel 2 as the local oscillator, which due to doubler in the narrow band receiver, you input $f_{\mathrm{LO}}/2$. Set the amplitude to 600mV pp.
2. Use Channel 1 as the transmitted RF signal that will use the the RF board AMK–2–13+ doubler to produce a modulated frequency modulation (FM) carrier at $f_{\mathrm{RF}} = 163$ MHz. Set the power level to –15 dBm when directly cabling to the narrow band receiver RF Input. A higher power level will be required if interfacing with antennas. Press the *Modulate* button on the front panel of the 33600 and choose FM modulation with a frequency deviation of 2 kHz. For the modulation source choose *External* with *Mod In* equal to 1V.
3. In the Jupyter notebook sample start the audio stream with `T_record = 0` so the output waveform runs continuously and no capture is made. You may need to adjust the mic gain closer to 100% in Figure 16.
4. Initially place the doubler output into the spectrum analyzer to verify that you have a signal centered on 163 MHz. Save a screenshot or csv file for display in the Jupyter notebook.

As a follow up to step 3 verify on the scope that the FSK demodulator output, when the cable to the USB audio device input is pulled off is similar to Figure 22 shown below.

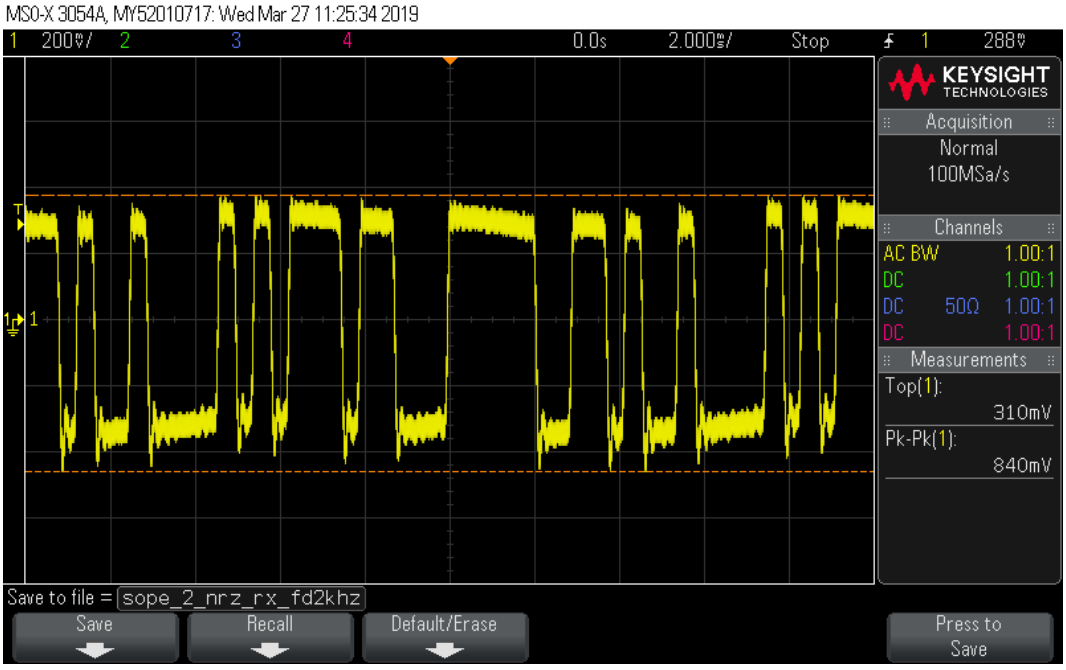


Figure 22: Expected FSK demodulator output waveform when the cable to the USB audio device input is temporarily pulled and the power into the transmit power into the doubler is –15 dBm (direct cabling).

As a follow–up to step 4, the Jupyter notebook sample has a section on FSK modulation theory and contains a code cell for simulating the expected power spectral density with a repeating *M*–sequence bit stream and random bits. The exact shape of the spectrum depends upon amplitude swing of the NRZ waveform coming from the USB audio device and the frequency deviation set on Channel 1 of the 33600A generator. In the Python code I assume the rear modulation input on the 33600A is 790 mV pp. The resulting spectrum for random data bits is shown in Figure 23.

```
fs = 48000
fc = 0
N_bits = 10000
n = arange(0,20*N_bits)
fd = 2000 #Hz/v
data = ss.PN_gen(N_bits,M)
#x_NRZ, b_pulse = ss.NRZ_bits2(data,20,pulse='rect') # M-seq bits
x_NRZ, b_pulse, data = ss.NRZ_bits(N_bits,20,pulse='rect') # random bits
m = fd*.79/2*x_NRZ
xc = exp(1j*2*pi*2*cumsum(m)/fs)*exp(1j*2*pi*fc/fs*n) # 2 next to cumsum for
doubler
psd(xc,2**10,48000);
title(r'Complex Baseband Model of Binary FSK')
xlabel(r'Frequency (Hz)')
xlim([-10000,10000])
ylim([-90,-30]);
```

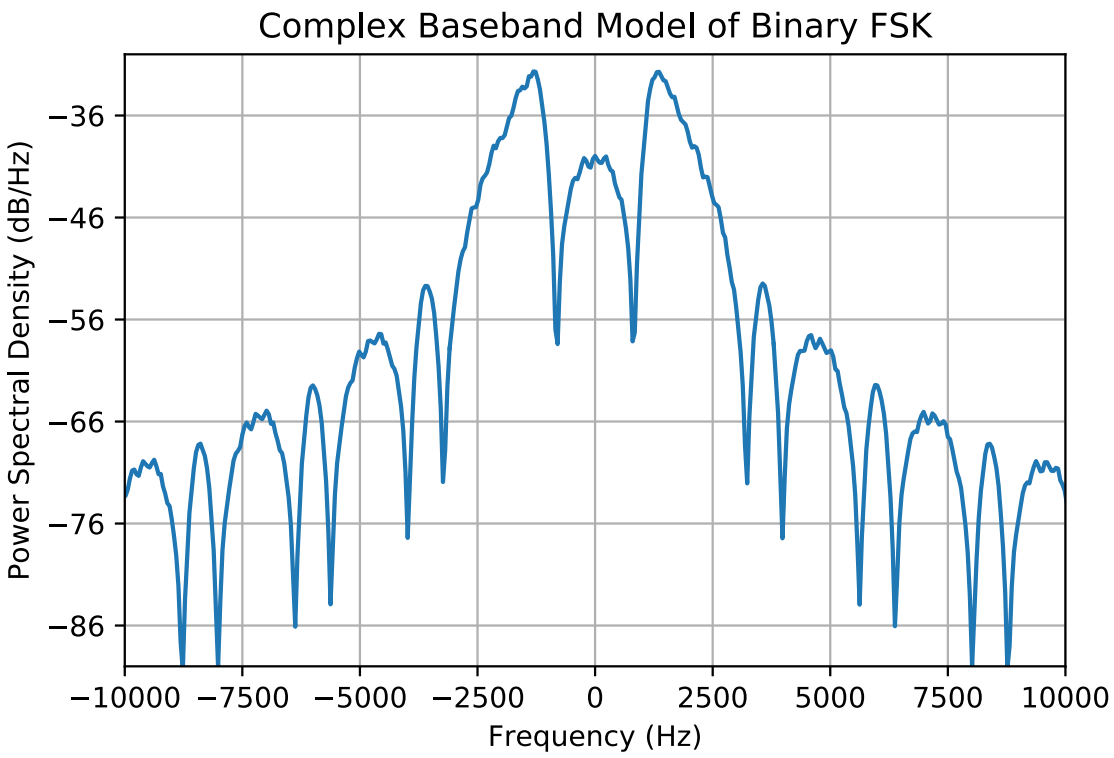


Figure 23: Python simulated complex baseband FSK spectrum when using random bits.

Note In communication system modeling the complex baseband (bb) view can be thought of as simply shifting the carrier frequency to zero, i.e., $f_c = 0$. When transmitting the complex baseband signal to the desired carrier you can use a DSP means or an analog means to effectively form $x_{c,\text{actual}}(t) = \text{Re}\{x_{c,bb}(t) \cdot e^{j2\pi f_{c,\text{actual}} t}\}$.

Part b

Perform a long `T_record = 30` s capture with transmit power, before the doubler of –15 dBm. Document the capture with plots similar to Figures 14, 16, and 17. Finally verify that the bit error probability (BEP) estimate is zero (`BEP = 0.00e+00`). Note if there are memory issues with `pyaudio_helper` on the PC you are using, reduce the capture to 20s.

Part c

Repeat part b except reduce the signal power before the doubler to –17 dBm. You may have to make some find adjustments on the LO (33600A Channel 1) to get the best demodulated NRZ bit stream waveform on the scope. The receiver is running very to close to its threshold of FM demodulation operation. The intent here is to see how quickly the BEP degrades. Don't be surprised if the BEP is now around 10^{-2}, that is on average one in 100 bits is in error.

In Lab 6 you will be using the *RTL–SDR* software defined radio as a receiver. The receiver front–end here has a lot more gain and thus we can run the received signal down into the noise floor and revisit BEP measurements when additive receiver noise is dominant, as opposed to receiver threshold due to the input signal being too small due to lack of receiver gain, regardless of noise.

References

1. Rodger Ziemer and William Tranter, Principles of Communications, 8th edition, Wiley, 2014.
2. Baseline wander – EECS: www–inst.eecs.berkeley.edu
3. K. Chen and J. Lee, “A Family of Pure Digital Signal Processing Bit Synchronizers,” IEEE Trans. on Commun., Vol. 45, No. 3, March 1997, pp. 289—292.

Appendix: NXP RF IC Data Sheet Highlights

Low voltage high performance mixer FM IF system with high-speed RSSI

SA636

DESCRIPTION

The SA636 is a low-voltage high performance monolithic FM IF system with high-speed RSSI incorporating a mixer/oscillator, two limiting intermediate frequency amplifiers, quadrature detector, logarithmic received signal strength indicator (RSSI), voltage regulator, wideband data output and fast RSSI op amps. The SA636 is available in 20-lead SSOP (shrink small outline package).

The SA636 was designed for high bandwidth portable communication applications and will function down to 2.7 V. The RF section is similar to the famous SA605. The data output has a minimum bandwidth of 600 kHz. This is designed to demodulate wideband data. The RSSI output is amplified. The RSSI output has access to the feedback pin. This enables the designer to adjust the level of the outputs or add filtering.

SA636 incorporates a power-down mode which powers down the device when Pin 8 is LOW. Power down logic levels are CMOS and TTL compatible with high input impedance.

FEATURES

- Wideband data output (600 kHz min.)
- Fast RSSI rise and fall times
- Low power consumption: 6.5 mA typ. at 3 V
- Mixer input to >500 MHz
- Mixer conversion power gain of 11 dB at 240 MHz
- Mixer noise figure of 12 dB at 240 MHz
- XTAL oscillator effective to 150 MHz (L.C. oscillator to 1 GHz local oscillator can be injected)
- 92 dB of IF Amp/Limiter gain
- 25 MHz limiter small signal bandwidth
- Temperature compensated logarithmic Received Signal Strength Indicator (RSSI) with a dynamic range in excess of 90 dB
- RSSI output internal op amp
- Internal op amps with rail-to-rail outputs
- Low external component count; suitable for crystal/ceramic/LC filters
- Excellent sensitivity: 0.54 μV into 50 Ω matching network for 12 dB SINAD (Signal to Noise and Distortion ratio) for 1 kHz tone with RF at 240 MHz and IF at 10.7 MHz
- ESD hardened
- 10.7 MHz filter matching (330 Ω)
- Power-down mode (I_{CC} = 200 μA)

PIN CONFIGURATION

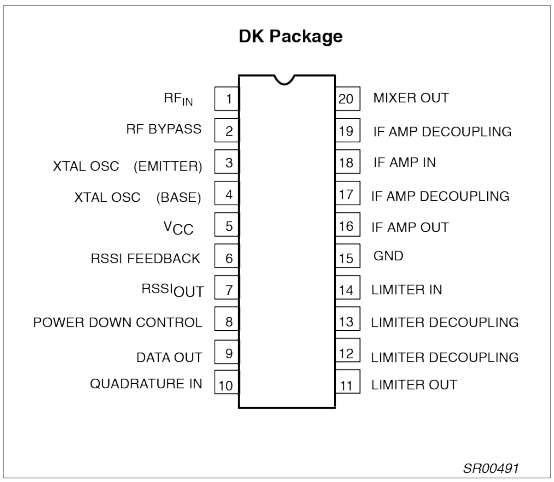


Figure 1. Pin configuration

APPLICATIONS

- DECT (Digital European Cordless Telephone)
- Digital cordless telephones
- Digital cellular telephones
- Portable high performance communications receivers
- Single conversion VHF/UHF receivers
- FSK and ASK data receivers
- Wireless LANs

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
20-Pin Plastic Shrink Small Outline Package (Surface-mount)	–40 °C to +85 °C	SA636DK	SOT266–1

Figure 23: A portion (first page) of the NXP SA636 data sheet.

Double-balanced mixer and oscillator SA602A

DESCRIPTION

The SA602A is a low-power VHF monolithic double-balanced mixer with input amplifier, on-board oscillator, and voltage regulator. It is intended for high performance, low power communication systems. The guaranteed parameters of the SA602A make this device particularly well suited for cellular radio applications. The mixer is a "Gilbert cell" multiplier configuration which typically provides 18dB of gain at 45MHz. The oscillator will operate to 200MHz. It can be configured as a crystal oscillator, a tuned tank oscillator, or a buffer for an external LO. For higher frequencies the LO input may be externally driven. The noise figure at 45MHz is typically less than 5dB. The gain, intercept performance, low-power and noise characteristics make the SA602A a superior choice for high-performance battery operated equipment. It is available in an 8-lead dual in-line plastic package and an 8-lead SO (surface-mount miniature package).

FEATURES

- Low current consumption: 2.4mA typical
- Excellent noise figure: <4.7dB typical at 45MHz
- High operating frequency
- Excellent gain, intercept and sensitivity
- Low external parts count; suitable for crystal/ceramic filters
- SA602A meets cellular radio specifications

PIN CONFIGURATION

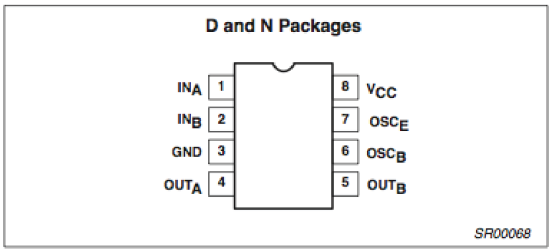


Figure 1. Pin Configuration

APPLICATIONS

- Cellular radio mixer/oscillator
- Portable radio
- VHF transceivers
- RF data links
- HF/VHF frequency conversion
- Instrumentation frequency conversion
- Broadband LANs

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
8-Pin Plastic Dual In-Line Plastic (DIP)	-40 to +85°C	SA602AN	SOT97-1
8-Pin Plastic Small Outline (SO) package (Surface-mount)	-40 to +85°C	SA602AD	SOT96-1

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER		RATING	UNITS
V_{CC}	Maximum operating voltage		9	V
T_{STG}	Storage temperature range		-65 to +150	°C
T_A	Operating ambient temperature range SA602A		-40 to +85	°C
θ_{JA}	Thermal impedance	D package	90	°C/W
		N package	75	°C/W

Figure 241: A portion (first page) of the NXP SA602 data sheet.

High performance low power mixer FM IF system SA605

DESCRIPTION

The SA605 is a high performance monolithic low-power FM IF system incorporating a mixer/oscillator, two limiting intermediate frequency amplifiers, quadrature detector, muting, logarithmic received signal strength indicator (RSSI), and voltage regulator. The SA605 combines the functions of Signetics' SA602 and SA604A, but features a higher mixer input intercept point, higher IF bandwidth (25MHz) and temperature compensated RSSI and limiters permitting higher performance application. The SA605 is available in 20-lead dual-in-line plastic, 20-lead SOL (surface-mounted miniature package) and 20-lead SSOP (shrink small outline package).

The SA605 and SA615 are functionally the same device types. The difference between the two devices lies in the guaranteed specifications. The SA615 has a higher I_{CC}, lower input third order intercept point, lower conversion mixer gain, lower limiter gain, lower AM rejection, lower SINAD, higher THD, and higher RSSI error than the SA605. Both the SA605 and SA615 devices will meet the EIA specifications for AMPS and TACS cellular radio applications.

For additional technical information please refer to application notes AN1994, 1995 and 1996, which include example application diagrams, a complete overview of the product, and artwork for reference.

APPLICATIONS

- Cellular radio FM IF
- High performance communications receivers
- Single conversion VHF/UHF receivers
- SCA receivers
- RF level meter
- Spectrum analyzer
- Instrumentation
- FSK and ASK data receivers
- Log amps
- Wideband low current amplification

PIN CONFIGURATION

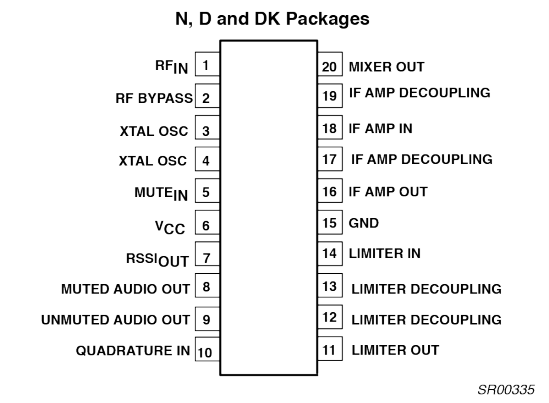


Figure 1. Pin Configuration

FEATURES

- Low power consumption: 5.7mA typical at 6V
- Mixer input to >500MHz
- Mixer conversion power gain of 13dB at 45MHz
- Mixer noise figure of 4.6dB at 45MHz
- XTAL oscillator effective to 150MHz (L.C. oscillator to 1GHz local oscillator can be injected)
- 102dB of IF Amp/Limiter gain
- 25MHz limiter small signal bandwidth
- Temperature compensated logarithmic Received Signal Strength Indicator (RSSI) with a dynamic range in excess of 90dB
- Two audio outputs - muted and unmuted
- Low external component count; suitable for crystal/ceramic/LC filters
- Excellent sensitivity: 0.22μV into 50Ω matching network for 12dB SINAD (Signal to Noise and Distortion ratio) for 1kHz tone with RF at 45MHz and IF at 455kHz
- SA605 meets cellular radio specifications
- ESD hardened

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
20-Pin Plastic Dual In-Line Package (DIP)	–40 to +85°C	SA605N	SOT146-1
20-Pin Plastic Small Outline Large (SOL) package	–40 to +85°C	SA605D	SOT163-1
20-Pin Plastic Shrink Small Outline Package (SSOP)	–40 to +85°C	SA605DK	SOT266-1

Figure 25: A portion (first page) of the NXP SA605 data sheet.