

The Lentz Receiver: Taylor Evolved

This design is purpose-built to provide high image rejection.

In an effort to design my own pocket-sized transceiver, I've been working with different receiver designs for a couple years now, and I'm ready to share my latest design. Unlike current direct-conversion receivers utilizing "Taylor Detectors," this design is purpose-built for high image rejection of >70 dB, see **Figure 1**, with a low MDS (minimum discernible signal) sensitivity across all the HF bands, **Figure 2**. Technically, my receiver design can be characterized as a direct-conversion receiver (i.e., homodyne, first developed in 1932). It uses Hartley [1] image rejection architecture and builds on the work of Taylor [2] and Campbell [3]. There are three new things here: 1) a method of sampling I and Q that is slightly different from Taylor, 2) a finished amplification stage, and 3) the incorporation of a DSP (Digital Signal Processing) integrated circuit for the phase shifting to perform image cancelation as suggested by Campbell and Youngblood [4].

Before going into the specifics, let's take a step back and review why image rejection matters. With a direct-conversion receiver, there is no IF (intermediate frequency) – the oscillator in the receiver operates at or near the frequency of interest. So for example, if you want to receive a CW transmission at 14.100 MHz with a 500 Hz tone, you would program (or tune) your oscillator to 500 Hz below that frequency, i.e. 14.099500 MHz. Unfortunately, with a direct-conversion receiver, the problem comes from transmissions that are 500 Hz below your oscillator frequency, at 14.099000 MHz. Commonly called the opposite sideband, it is techni-

cally also the undesired image in a direct-conversion receiver, and without image rejection, you receive this signal just as strongly as the frequency of interest. For CW operations, this can result in confusing the receive frequency and then transmitting back a kilohertz or more away from

whom you're trying to QSO.

How does image rejection help? When I say that there is >70 dB of image rejection with this new receiver design, it means that the signal of the image must be >70 dB, or 10 million times, stronger than the desired frequency to have the same

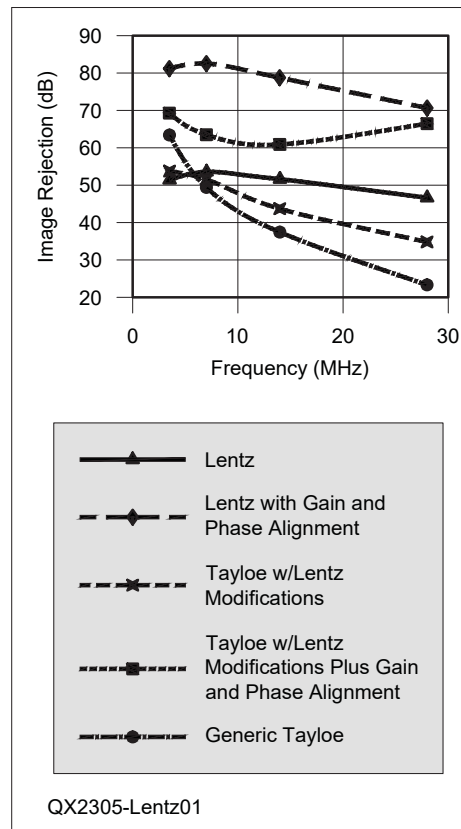


Figure 1 — Image rejection of the Lentz and modified Taylor receivers using either (a) DSP with a simple Hilbert Transform of 90° for image rejection, or (b) DSP precision phase and gain alignment for image rejection. Higher values are better.

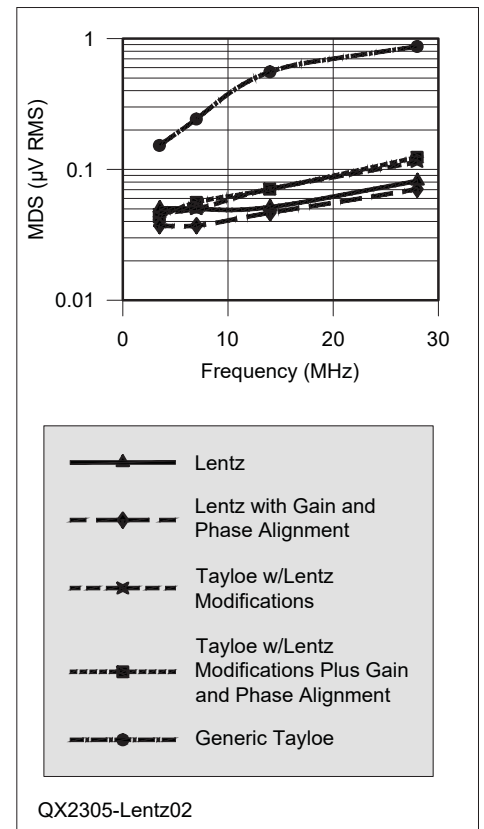


Figure 2 — The minimum discernible signal by frequency using the ARRL's 3 dB-over-noise CW method with a 500 Hz filter. Lower values are better.

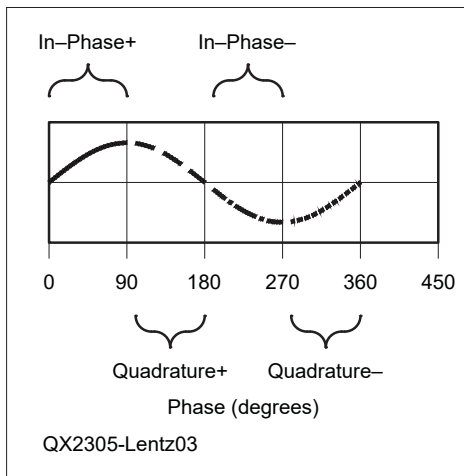


Figure 3 — The 90° sampling of a Tayloe detector.

signal strength as the desired signal. This amount of image rejection is similar to superheterodyne receivers that are much more complex; see the specifications for the popular Yaesu FT-991A.

How does image rejection work? Put simply, you sample or mix your received signal with in-phase and quadrature clocks, typically just called I-Q. The quadrature clock is delayed 90°, or in other words, one-fourth the time period of the frequency of interest. Now that we have sampled I and Q, we have baseband (audio) in two channels, which are also 90° shifted. By post-sample shifting (delaying) the audio quadrature channel another 90° and adding the signals back together, the signals either double (for the desired signal, i.e. the proper sideband), or the signals cancel (if the image, i.e. the wrong sideband). One way that has been common to do the phase shift is with operational amplifiers using tuned phase delays — see Campbell [3] and also the QCX line of transceivers by QRP Labs. Youngblood [4] proposed some complex PC based manipulation, but modern DSP integrated circuits allow us to simply add a single design block called a Hilbert Transform, which will do the 90° rotation for us. As described later, we will do some fine tuning of the phase and gain, but we'll use DSP for that too.

Now that you know the final trick, let's walk through the design to see how I optimized for it. I spent two years working with the Tayloe detector with which most of you will already be familiar. The Tayloe detector is an I-Q sampler patented more than 20 years ago and is included in the current *ARRL Handbook*. Tayloe functions by chopping the incoming signal at the

receive frequency into four segments with a multiplexer, see **Figure 3**. In other words, it takes 90° samples of the incoming RF wave, and stores the sampled voltage in capacitors before forwarding the differences on to amplifiers, either operation amplifiers, like OpAmps, or instrumentation amplifiers, like InAmps. Using a generic Tayloe design and a 90° Hilbert Transform, I get 60 dB of image rejection at 3.5 MHz, but rejection drops off to only about 25 dB at 28 MHz. Again, see **Figure 1**.

I realized that what I really needed was to have better phase and gain matching between I and Q. I can't emphasize this enough. Phase matching and gain matching are the keys to success here. If you do something that influences either of those, you must control the error it induces, with higher precision components.

In my attempts to optimize for image rejection, I have made several modifications to the Tayloe detector. The results of these are shown as "Tayloe with Lentz modifications" in **Figures 1** and **2**. Note that the "Generic Tayloe" is for general reference only — these results are from an earlier experiment of mine, which does not contain any of the "Lentz modifications" nor the INA849 InAmp. Your results may vary.

- 1) – I do not bias the receive signal.
Because I felt that adding a source of common-mode noise into the signal was counter-productive, I used a capacitively-coupled RF signal throughout the design.
- 2) – In his example circuit, Tayloe uses four relatively large (0.22 μ F) sampling capacitors that are each connected to ground. Unfortunately, I found that I get a significant dc offset on my sampling capacitors with this configuration, especially at higher HF frequencies (>14 MHz). I'm not sure if this is due to charge injection pumping from the multiplexer, but it definitely occurs. Further, I found that mismatching of capacitors leads to larger phase errors, especially at the higher values recommended by Tayloe. The fix here is to cross-connect *one* capacitor *between* the positive and negative samples instead of using *two* to ground for each signal path (I or Q). Reducing the number of capacitors from four to two increases stability of the amplifier and reduces this

source of phase and gain errors.

- 3) – I reduced the value of the capacitors considerably. I found that I got flattest performance in MDS across the HF bands, and best stability of the InAmps, with a cross-connected capacitor of only 100 to 1,000 pF. By using 1% COG rated caps, you don't have to worry about heat from your transmitter throwing the receiver out of alignment.

This is the point where we are ready to amplify the signals, and where Tayloe's design ends its specifications. To finish the design:

- 1) – We need to match the gain between the two signals, and this is where InAmps are better than OpAmps. Unlike OpAmps, InAmps require only a *single* resistor to set their gain, and by selecting high precision (0.1%) resistors with low temperature coefficients (25 ppm/°C or less), we can get high gain *and* very good gain matching between the two channels. I have had the most success with one particular low-noise, high-speed InAmp, the Texas Instrument INA849. The Analog Devices version of this, the AD8429, might be a suitable alternative, but I haven't tried it yet. I've successfully pushed the INA849 to provide 1000 V/V gain, so you get 1 mV out from 1 μ V in up to 30 MHz. With the INA849, only one chip is required for each signal chain (in-phase and quadrature), and no other hardware amplification is required. Since digital decimation filters are built in to the DSP chip low-pass filters aren't needed; just feed the outputs through dc-blocking capacitors into the DSP ADC (Analog to Digital Converter) inputs. Other InAmps I've tried worked fine at lower frequencies or lower gains. If you use a different InAmp, you'll know when the gain is too high because the output of one or both of the InAmps gets pegged to a power rail, e.g. 4 V or -4 V, in my case.
- 2) – As required by the InAmp datasheet, I experimented with various current-return resistors and determined that very low values of 50 to 100 Ω on the InAmp inputs worked best to prevent high noise floors.
- 3) – I used an easily programmed DSP

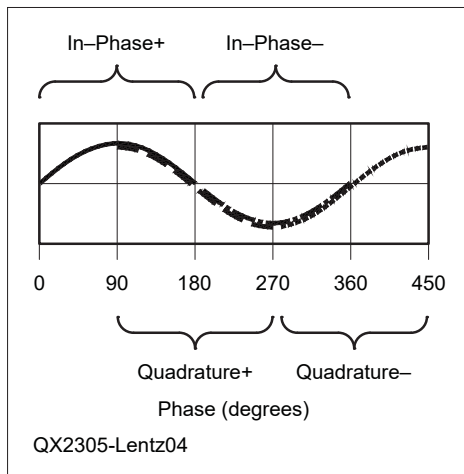


Figure 4 — The 180° sampling of the Lentz receiver.

chip, the Analog Devices ADAU1761, because it comes with the free Sigma Studio graphical programming interface (which also helps author the final C-code for a microcontroller) and because it comes with Hilbert Transform as a built-in design block. Analog Devices has other DSP chips that do the same thing if you search on “Audio Codecs” on their website.

At this point in the modifications, with only hardware changes and a simple 90° DSP Hilbert Transform rotation on the quadrature channel, image rejection is in

the 35 to 55 dB range, see **Figure 1**, and MDS is greatly improved compared to my previous experiments with Tayloe, see **Figure 2**. Using further DSP programming for gain (i.e. balance) and phase alignment, I was able to push the image rejection up to 60 to 70 dB, again, see **Figure 1**. This amount of rejection (60 to 70 dB) is pretty good, but my goal was to match a super-heterodyne with >70 dB of rejection.

So, I took a step back and, inspired by Campbell [3], I redesigned the Tayloe sampling scheme. Personally, I think that with this new scheme and all my modifications combined, the end result is different enough to warrant a distinct name. This is the Lentz Receiver. Through experimentation, I took what I’ve already described to improve Tayloe, and then determined that I could get better phase matching and better MDS by splitting the incoming RF signal in half using a simple “Y” in the PCB traces and sampling I and Q separately. Campbell had used a 50 Ω power splitter for this purpose, but a simple Y in the circuit board traces along with 4x100 Ω current-return resistors on the InAmp inputs, two of which are exposed to the incoming signal at any one time, leads to a 50 Ω match to the coax.

Like Tayloe, I chop the signal up and store-and-forward the differential results in capacitors — yes, I tried removing the capacitors completely — but instead of

taking a single signal and cutting it into fourths with a 1x4 multiplexer, see **Figure 3**, I use two clocks and two 1x2 analog switches — one precisely delayed by 90° — and cut each signal into halves, see **Figure 4**. This means I’m sampling 180° of the in-phase signal, which is forwarded to the amplifier positive input, and then I’m sampling the next 180° of the in-phase signal (the negative half), which is forwarded to the amplifier negative input. The same thing happens with the quadrature signal chain, except that it is sampled 90° out of phase compared to the in-phase clock. I fully expected my MDS level to be 3 dB worse with this split, but tests showed this design was more sensitive than my best modifications of Tayloe, see **Figure 2**, presumably due to the longer sample time.

One additional change from earlier designs, I had used two CMOS clocks to control the multiplexer. One at the frequency of interest, and one at two times that frequency, in order to match the truth table of the multiplexer. In my current design, I conditioned the clocks with flip-flops because I found a small technical specification in the oscillator chip data-sheet that explained some of the phase error I was getting: “Duty Cycle 48 – 52%.” In other words, the CMOS clock output wasn’t guaranteed to be a symmetrical square wave, it was only guaranteed for the rising edge. So, like several people

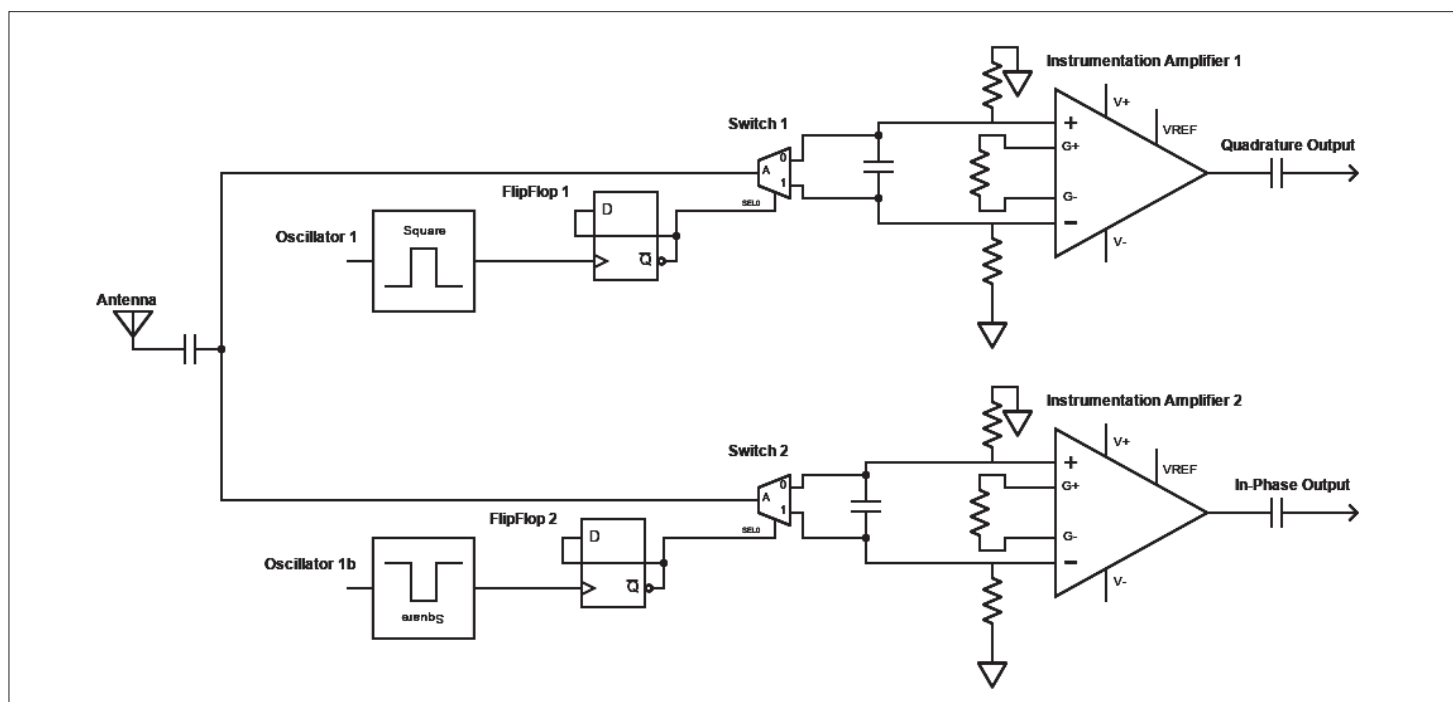


Figure 5 — The basic design of the Lentz receiver minus the power supply tree. Oscillator 1b is the complement, or inverse, of Oscillator 1. The two flip-flops are rising edge triggered.

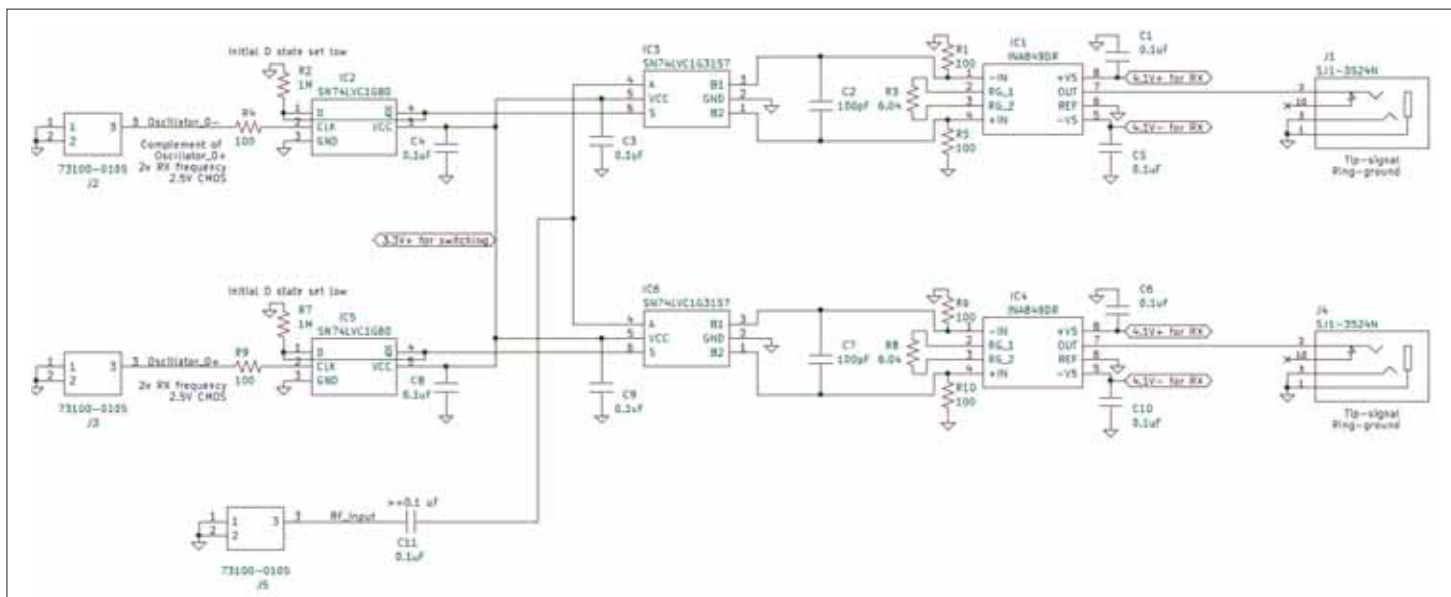


Figure 6 — Specific part selection and connections that I used for the Lentz Receiver in all tests.

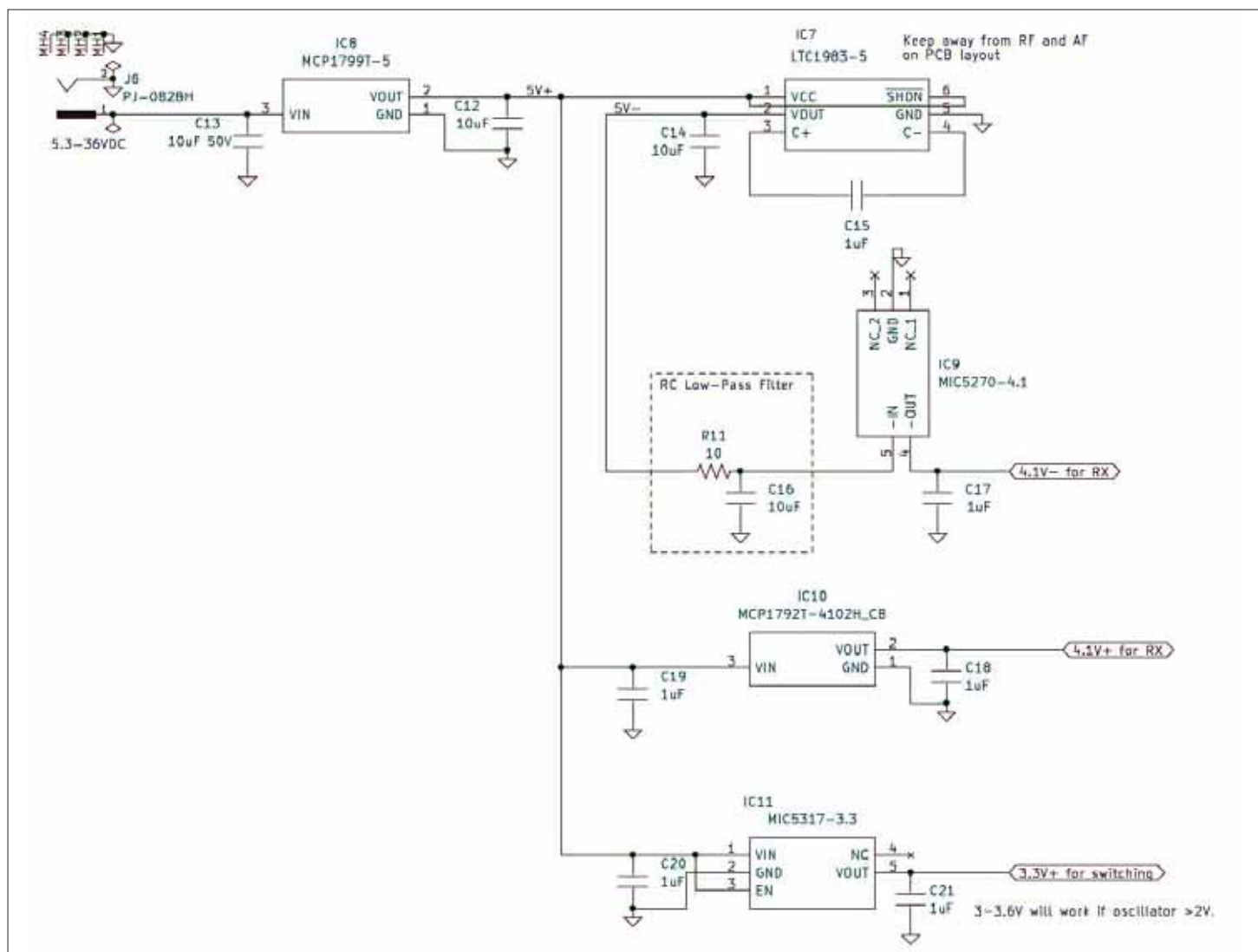


Figure 7 — The simplest power tree that I have come up with. Without the oscillator board or DSP board, the receiver draws 40 mA.

have already published online, I selected flip-flops that are rising-edge triggered to create the square wave CMOS clock needed at the two analog switches. As usual, to get a given frequency out of the flip-flop, the oscillator chip operates at two times the receive frequency and every rising edge of the clock represents a change from low-to-high or high-to-low CMOS level output from the flip-flop. For the quadrature clocking, fortunately, the oscillator chip I chose, the Skyworks Solutions Si5340D [5], can provide “complementary” clocks (180° out of phase on its rising edge) from a single programmed output. Because I’m already operating at twice the receive frequency, the base clock and the complementary clock combine for four times the receive frequency to equal a quadrature clock. So the complementary clock gets its own flip-flop and analog switch, and the two signals are 90° out of phase. This is simple and a much more precise phase shift than using the CMOS outputs of the Si5340D directly.

If you build the clock section and use the flip-flops that I did (without set/reset), add a high value (1 M Ω) resistor from the flip-flop D to ground, and then for proper startup: 1: Switch on the power to the oscillator first and “disable” but don’t “power down” the clock, 2: Switch on the power to the flip-flops (and to the rest of the board is okay too), and only then, 3: Enable the clock. This ensures that the flip-flops are properly initialized so that you always get the same sideband when you turn it on. After turning the clock on this way, you can manipulate the clock frequency without fear of selecting the alternate sideband because this clock IC has zero-crossing frequency changes (which also means, say good-bye to those annoying clicks when changing frequency!).

The end result is the design shown in simplified form in **Figure 5**, and completed schematics with the specific parts shown in **Figures 6** for the receiver signal chain, and **Figure 7** for the power tree. Note that the DSP and oscillator sections are not displayed because I used manufacturer-provided development boards. Please refer to the datasheets for those products to determine the ancillary components for a complete and final receiver design.

Without specific phase and gain alignment, just a simple 90° Hilbert Transform, I get 45 to 55 dB of image rejection and MDS levels — using ARRL’s CW method,

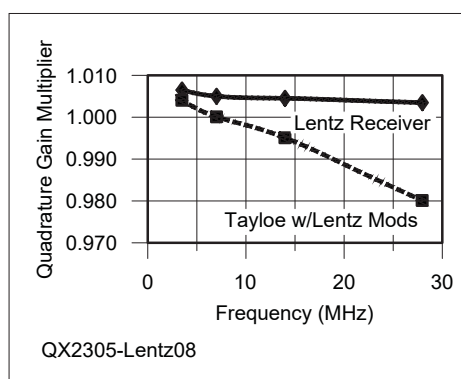


Figure 8 — The amount of gain applied to the quadrature channel to balance with the in-phase channel for the maximum image rejection values displayed in **Figure 1** for the “Taylor with Lentz modifications” and for the Lentz Receiver. The flatter line is better.

which is 3 dB rms over noise with a 500 Hz filter — of around 0.05 to 0.08 μ V rms at 50 Ω (–133 dBm to –129 dBm).

My testing method mimics the ARRL Lab. I use a Behringer UCA202 line level USB interface between the DSP headphone output and a laptop running True Audio TrueRTA Audio Spectrum Analyzer Level 4 Software. The RF signal is generated with a Siglent SDG1032x arbitrary waveform generator set for 50 Ω output of sine waves measured in rms voltage output levels in the low millivolt range. The generator has an SDR-Kits Mini Low-Jitter Precision GPSDO Reference Oscillator connected to it for accuracy. A 50 Ω cable connects to the antenna port of the receiver board through several 50 Ω RF fixed attenuators that sum to either –40 dB, –70 dB, or –100 dB. I turn on the radio under test, set the DSP for 500 Hz-wide filtering centered on 500 Hz (my preferred side-tone). I also use a peaking filter and headphone amplifier in the DSP, which adds 15 dB and 6 dB respectively to the gain of the InAmp in these measurements. Without any input signal, I record the rms dBm of the noise level shown in the software. I then turn on the signal generator and adjust the signal for approximately 3 to 4 dB higher rms to determine the MDS. I say approximately because these levels fluctuate with the noise up to about ± 0.5 dB, and my measurements are conservative, which means I make sure there is at least 3 dB difference. Next, I retune the signal generator to 500 Hz below the oscillator for the image (and usually remove 30 dB or 60 dB of attenuation) then increase voltage on the

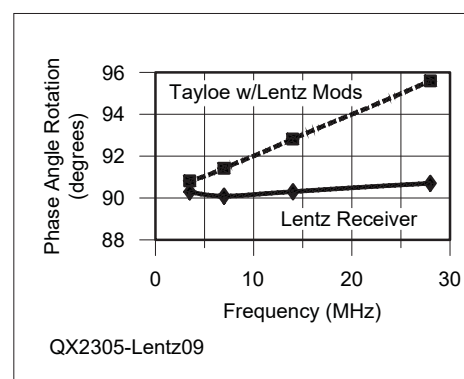


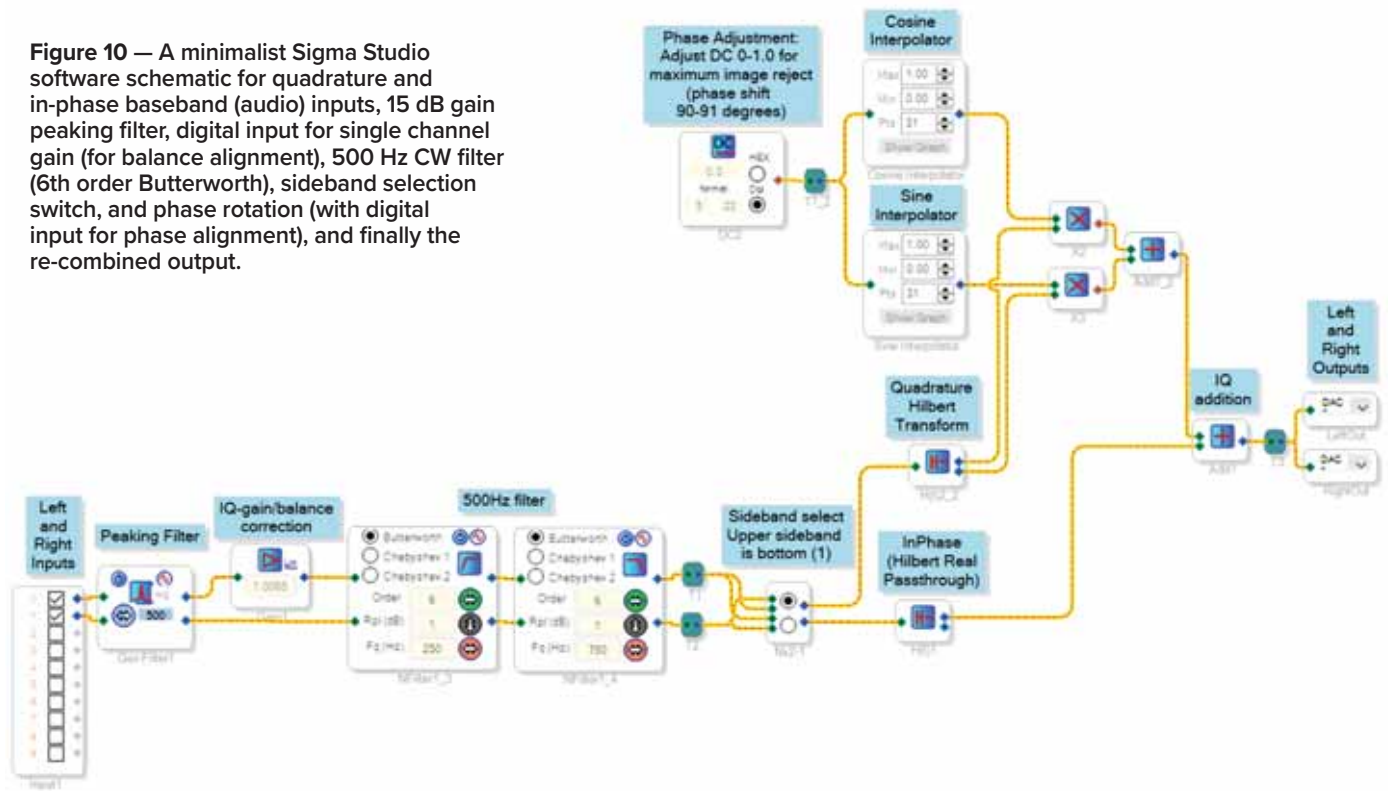
Figure 9 — The amount of phase rotation applied to the quadrature channel before adding to the in-phase channel for the maximum image rejection values displayed in **Figure 1** for the “Taylor with Lentz modifications” and for the Lentz Receiver. The flatter line is better.

signal generator until the 500 Hz sound level reads the same value of 3 to 4 dB over noise rms. I record the voltage level of the input in millivolts rms, and do the attenuation and logarithm math in a spreadsheet for the reject level.

With DSP band-specific gain and phase alignment, I get image rejection numbers approximately 25 dB better (70 to 82 dB) and MDS levels a couple hundredths of a microvolt lower (–136 dBm to –130 dBm), see **Figures 1** and **2**. Although you can use DSP gain and phase alignment of I and Q for any quadrature receiver, the Lentz Receiver requires much less alignment than Taylor (with Lentz modifications) for maximum image rejection, see **Figures 8** and **9**.

So that’s all the good stuff; what are the downsides to my design? The biggest drawback is also its biggest benefit. The design obviously requires a DSP chip, which in turn requires software on a microcontroller to program and adjust the DSP. You will also need to write the microcontroller code for the oscillator and probably a display, but with a modern radio, you are probably doing this already. This is a software intensive design; one that some would call a “software defined radio.” I’m at the stage where I’ve finished the receiver design using development boards from the manufacturers for the oscillator and DSP chips plugged into a laptop to control them over USB (that’s the bus not the sideband) using manufacturer provided graphical interfaces. I’ve spent two years getting to this point, and I’ll probably spend a month on the C-code software for a microcon-

Figure 10 — A minimalist Sigma Studio software schematic for quadrature and in-phase baseband (audio) inputs, 15 dB gain peaking filter, digital input for single channel gain (for balance alignment), 500 Hz CW filter (6th order Butterworth), sideband selection switch, and phase rotation (with digital input for phase alignment), and finally the re-combined output.



troller to control and display my final transceiver design.

Note that whether you incorporate some or all of my modifications, or adopt my final design, I want to make it clear that I have no interest in patenting these developments. I intend them to be “Open Source Hardware” for the good of all, and upon receipt by the public of this article, should be considered “prior art.”

Here are a few more notes that I’d like to share on programming the DSP chip. Please refer to **Figure 10** for the DSP software schematic, which is also available on the www.arrl.org/QEXfiles web page, as a complete Sigma Studio file download to program your DSP chip. If you wish to mimic my simple 90° Hilbert phase shift results without gain and phase alignment (the solid lines in **Figures 1** and **2**), delete the gain block and then simply connect the imaginary output from the quadrature Hilbert Transform block to the I-Q addition block, and skip the sine and cosine table multiplications and addition.

First, a software trick is needed, which is not well documented. In order to get the Hilbert Transform to work correctly for this usage, you need two Hilbert Transform blocks — one that has the input from the in-phase channel and one that has the input from the quadrature channel. For the in-

phase channel, you are just passing one channel through the “Real” side to apply the appropriate time delay so that both channels match properly. It won’t work if you don’t do this.

For phase alignment, the configuration is a bit complex with multiplication by sine and cosine lookup tables on the quadrature output from the Hilbert Transform block, but the actual usage is to enter a single number for phase rotation. I followed the design example published online (<https://ez.analog.com/dsp/sigmadsp/f/q-a/65849/how-to-use-sigmastudio-software-to-realize-an-external-potentiometer-to-control-phase-the-phase-of-the-range-is-0-to-180>), and I modified it to include a dc input block (0 to 1 V in decimal form) instead of a potentiometer and reduced the range of phase adjustment of 90° to 100° degrees for Tayloe or 90° to 91° for the Lentz Receiver. My lookup table multiplier values (31 values each) for the cosine and sine interpolators are shown in **Table 1**, also on the **QEXfiles** web page.

Next for gain (balance) alignment: place a gain block on the quadrature audio line before the Hilbert Transform, and adjust as necessary for maximum image reject. This is a multiplier block, so you set it to 1.0 for no change, or greater than 1.0 for gain on the channel, or less than 1.0 for

attenuation of the channel. In tweaking this value with the Lentz Receiver, note that 0.0005 steps actually make a difference.

During my experiments, I found that I needed to go back and forth between phase alignment and gain alignment 2 or 3 times to get the best image rejection for that band. Now that I see the final values graphed (**Figures 8** and **9**), a bit of linear interpolation between the alignment of 7 MHz and 28 MHz will probably get me pretty close to the bands in between, and I may be able to let the microcontroller do the math.

Obviously the DSP chip can do a lot more than just our image rejection. This DSP chip has 2-channel (i.e., stereo) ADCs (Analog to Digital Converters) and DACs (Digital to Analog Converters) built in. It has a headphone amplifier, greatly simplifying output circuitry. In software we can swap the input channels to select upper or lower sideband by swapping which channels get the quadrature processing. We can easily create a very helpful peaking filter. We can drop in a 3000 Hz voice filter, or a 500 Hz CW filter or narrow it down even further. Obviously, we have attenuation and volume control, etc.

Looking ahead, I’m sure it is only a matter of time until some company releases a flip-flop or clock buffer that

Table 1 – Lookup table multiplier values for the cosine and sine interpolators.

| Degrees | Cosine lookup | Sine lookup |
|----------|---------------|-------------|
| 90.00000 | 0.000000000 | 1.000000000 |
| 90.03333 | -0.000581776 | 0.999999831 |
| 90.06667 | -0.001163553 | 0.999999323 |
| 90.10000 | -0.001745328 | 0.999998477 |
| 90.13333 | -0.002327104 | 0.999997292 |
| 90.16667 | -0.002908878 | 0.999995769 |
| 90.20000 | -0.003490651 | 0.999993908 |
| 90.23333 | -0.004072424 | 0.999991708 |
| 90.26667 | -0.004654195 | 0.999989169 |
| 90.30000 | -0.005235964 | 0.999986292 |
| 90.33333 | -0.005817731 | 0.999983077 |
| 90.36667 | -0.006399497 | 0.999979523 |
| 90.40000 | -0.006981260 | 0.999975631 |
| 90.43333 | -0.007563021 | 0.999971400 |
| 90.46667 | -0.008144780 | 0.999966831 |
| 90.50000 | -0.008726535 | 0.999961923 |
| 90.53333 | -0.009308288 | 0.999956677 |
| 90.56667 | -0.009890038 | 0.999951092 |
| 90.60000 | -0.010471784 | 0.999945169 |
| 90.63333 | -0.011053527 | 0.999938908 |
| 90.66667 | -0.011635266 | 0.999932308 |
| 90.70000 | -0.012217001 | 0.999925370 |
| 90.73333 | -0.012798732 | 0.999918093 |
| 90.76667 | -0.013380458 | 0.999910478 |
| 90.80000 | -0.013962180 | 0.999902524 |
| 90.83333 | -0.014543898 | 0.999894232 |
| 90.86667 | -0.015125610 | 0.999885601 |
| 90.90000 | -0.015707317 | 0.999876632 |
| 90.93333 | -0.016289019 | 0.999867325 |
| 90.96667 | -0.016870716 | 0.999857679 |
| 91.00000 | -0.017452406 | 0.999847695 |

utilizes one of the better differential clocking methods, e.g. LVDS (Low Voltage Differential Signaling) with the CMOS outputs necessary for the analog switch, or even LVDS-controlled analog switches. Better yet, someone will probably release a single quadrature sampling chip that includes the oscillator and switches. Also, watch the market for even lower noise, high-speed InAmps and DSP chips with more memory and MIPS than currently produced.

Software

FINAL_DSP_program_for_Lentz_Receiver_article.dsproj is the DSP software needed by the Lentz Receiver. It is available from www.arrl.org/QEXfiles web page, and requires the user to download and install the free SigmaStudio software from Analog Designs in order to program their DSP development board.

H. Scott Lentz, AG7FF, holds an Amateur Extra class license. He was first licensed in 2003. His ham radio interests shift

every few years, and most recently he has been working on designing a CW transceiver from scratch. Scott retired early from the US Forest Service where he was a fisheries biologist for his regular job and a communications technician for his fire job. He holds a BS in Aquatic Wildlife Biology and an MS in Organismal Biology and Ecology, both from the University of Montana.

Notes

- [1] R. Hartley, "Modulation System," US patent 1,666,206, 1928.
- [2] D. Tayloe, "Product detector and method therefore," US patent 6,230,000, 1998.
- [3] Campbell, R. KK7B, "High-Performance, Single-Signal Direct-Conversion Receivers," QST Jan. 1993, pp. 32-40.
- [4] G. Youngblood, AC5OG, "A software-defined radio for the masses, Part 1," QEX July/Aug., 2002, pp. 13-21.
- [5] The oscillator development board comes with capacitive coupling of the clocks. These capacitors must be removed and replaced with 0604 sized resistors or jumpers to use the CMOS clocks. See the development board datasheet for more information.

Errata

- In S. Geers, KA8BUW, "DSP CW Filter," QEX Jan./Feb. 2023, in Figure 1 the + and – inputs on op-amps U1 and U2 should be reversed. Thanks to Bob Liesenfeld, WBØPOQ, for pointing out the error.
- In Dr. U. L. Rohde, N1UL, "AM and FM Noise in Oscillators," QEX Mar./Apr. 2023, p. 5, Eqn (8-11) should be:

$$L \frac{di(t)}{dt} + (R_L - R_N(t))i(t) + \frac{1}{C} \int i(t) dt = e_N(t)$$

Thanks to James McNamee, KEØNRE, for spotting the error.