FM stereo MPX test transmitter. Design and build.

I do occasional repairs on stereo receivers and needed a way to test the FM stereo reception in various devices. Obviously, the air is full of FM signals that you can use to test any receiver, but it seemed like an interesting project to work on, and there is always much to be learned when developing your own test devices.

In the May 1993 issue of Elektor magazine, there is an article on that specific subject. The article ("FM stereo signal generator") can be found online. My design is partly based on the information found in that article. Another article on the subject appeared in the January 2007 issue. I also used information found in that one.

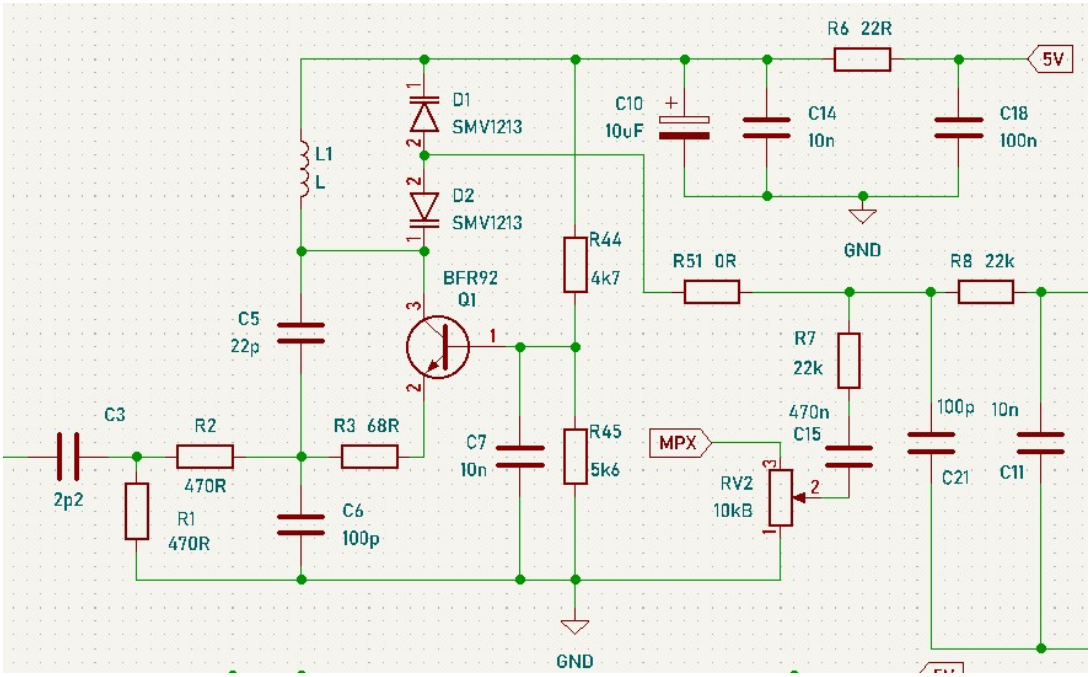
There are three building blocks for this stereo FM transmitter:

- the RF voltage-controlled oscillator
- the subcarrier and pilot signal generator
- the audio multiplexing section

1. RF oscillator

a. The oscillator circuit

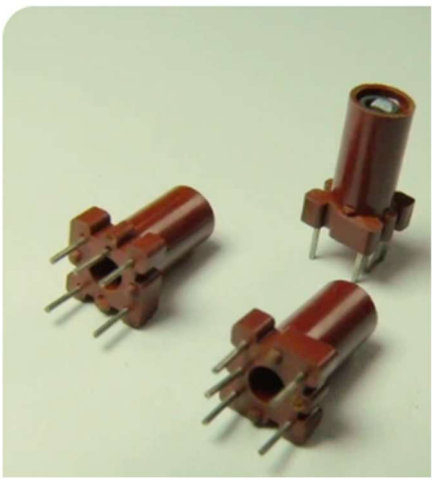
The RF section of my project is based on the oscillator found in the January 2007 issue of Elektor. I chose that one because of its simplicity. It requires only one coil and tuning is done with two varactor diodes. Since it is not my intention to transmit beyond my backyard, there is no need for power devices. The RF oscillator in question requires only one RF transistor.



- The heart of the oscillator is transistor Q1, inductor L1 and the two varactor diodes in series, and feedback capacitor C5. The transistor is BFR92 in a SOT-23 package, a high-frequency NPN device. The varactors are SMV1213's, which are tiny. Microscope required for soldering… The feedback capacitor is a 22pf NP0 device in a 1206 package.

© Jygsawaudio 2026

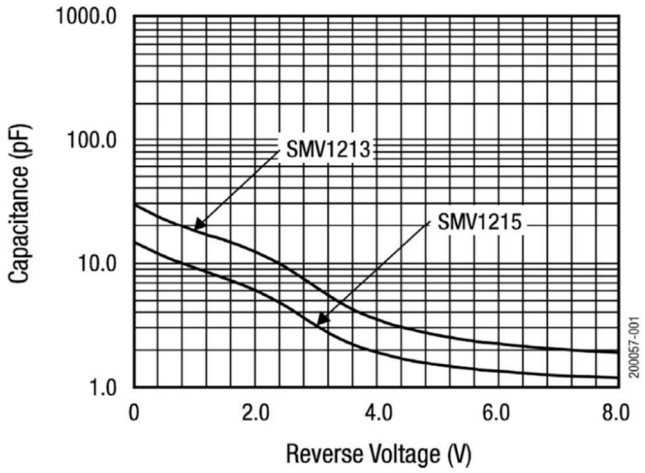
- The coil: this required a quite a lot of experimentation. It needed not only to work from roughly 88Mhz to 108Mhz, it also had to be relatively easy to adjust, and hopefully easy to mount on the pcb, close to a feedback coil (see later). I eventually used a small ferrite core inductor as a 'former'. This was easy to place on the pcb and thanks to the variable core I could roughly adjust the oscillator frequency range. Here is a picture of the former:



The former size is 5mm, 12mm high. The base is 7x7mm. There are 5 mounting pins.

By experiment I found that 2 turns of 0.5mm diameter enameled copper wire, spaced closely together, gave me the desired frequency range.

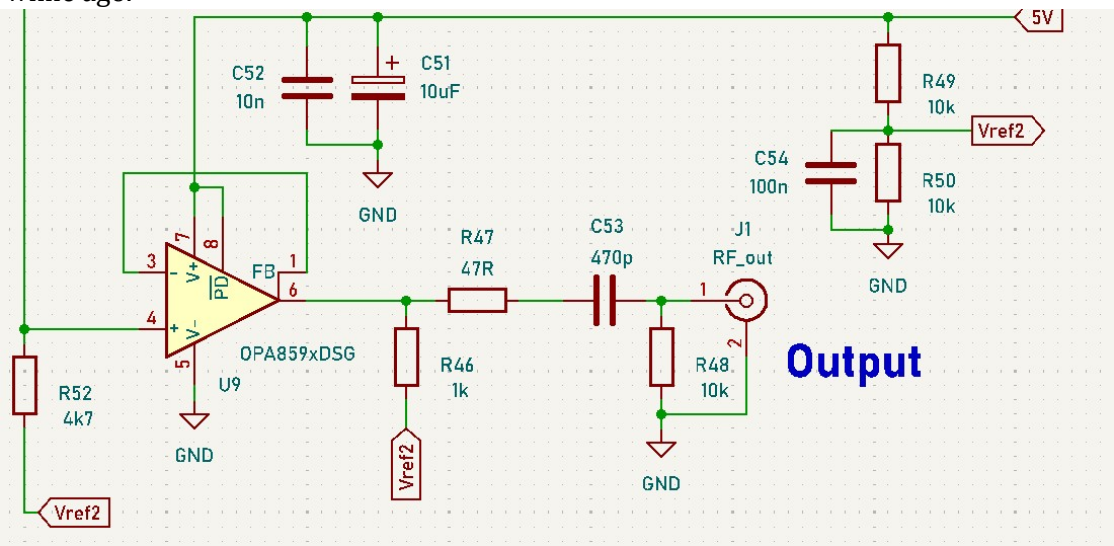
- Tuning: by applying a DC voltage to the junction of the two SMV1213 varactor diodes, you can vary the capacitance in parallel with the coil and thus vary the oscillator frequency. The SMV1213 datasheet gives the following information:
    - maximum reverse voltage 12V. That is more than enough since the power supply to the RF circuit is only 5V. Varactors with larger voltage ranges are less suitable for this circuit. These SMV1213's are so-called *hyperabrupt* devices, meaning that they are very sensitive to changes in bias voltage. The upside is that the change in frequency per ΔV is large (I measured roughly 5MHz per volt…). The downside is that that makes them very sensitive to noise or bias drift.
    - capacitance range: from around 30pF at 0V V_{rev} to around 2pF at 5V V_{rev}. Bear in mind that two of these devices are in series, so the coil 'sees' two capacitances in series. Hence the available parallel capacitance is 1pF-15pF.



There is 5Vdc applied to the cathodes of the diodes, and a variable bias voltage is applied to the anodes. So decreasing the tuning voltage to the anodes decreases the capacitance, increasing the oscillating frequency.

© Jygsawaudio 2026

    - Exact tuning by hand is very fiddly and unpractical, unless I would only choose to have this circuit operate on one frequency, and even then there would be temperature and bias drift issues to deal with. I looked for a way to tune the capacitance so I could have the entire range of FM frequencies available, in 100kHz steps. More on that later.
- Q1's DC base current is set with a voltage divider. I used a trimpot at first to dial in maximum signal amplitude, noted the values and replaced the pot by the closest possible fixed resistor values.
- From the original circuit in the Elektor article, it can be seen that the output is taken from Q1's emitter circuit directly. I chose to use a VHF opamp buffer to isolate the emitter circuit from the output. The opamp in question is the OPA859, with 1.8GHz unity gain BW, and it is unity gain stable. The only disadvantage is that it is tiny (WSOP package) and you really need a microscope to mount it on the pcb. The output impedance is made to be close to 50Ω with R47 and C53. I experimented with a potentiometer first to the input of the opamp in order to control output level, knowing well it was a naïve idea. It didn't work very well, due to the significant stray capacitances inherent to such a pot, but anyway. It does work more or less, but gives unreliable results, in that the output does vary, but gets fuzzy and varies very non-linearly. I bet it would give noise and unpredictable loading of the VCO. I removed it from the circuit electrically, but let it sit there for mechanical support for the pcb. If attenuation is necessary, I will use a separate stepped attenuator circuit at the output, which I built a while ago.



The VCO output is coupled capacitively to the input resistor to the opamp. The capacitor value is very small: 2.2 picofarad (NP0 or C0G). It isolates the VCO from the opamp's input and prevents low-frequency noise or phase noise from reaching the opamp's high-impedance input. It also makes for a more stable tank Q and easier for the PLL (see later) to lock into and stay locked into the intended frequency.

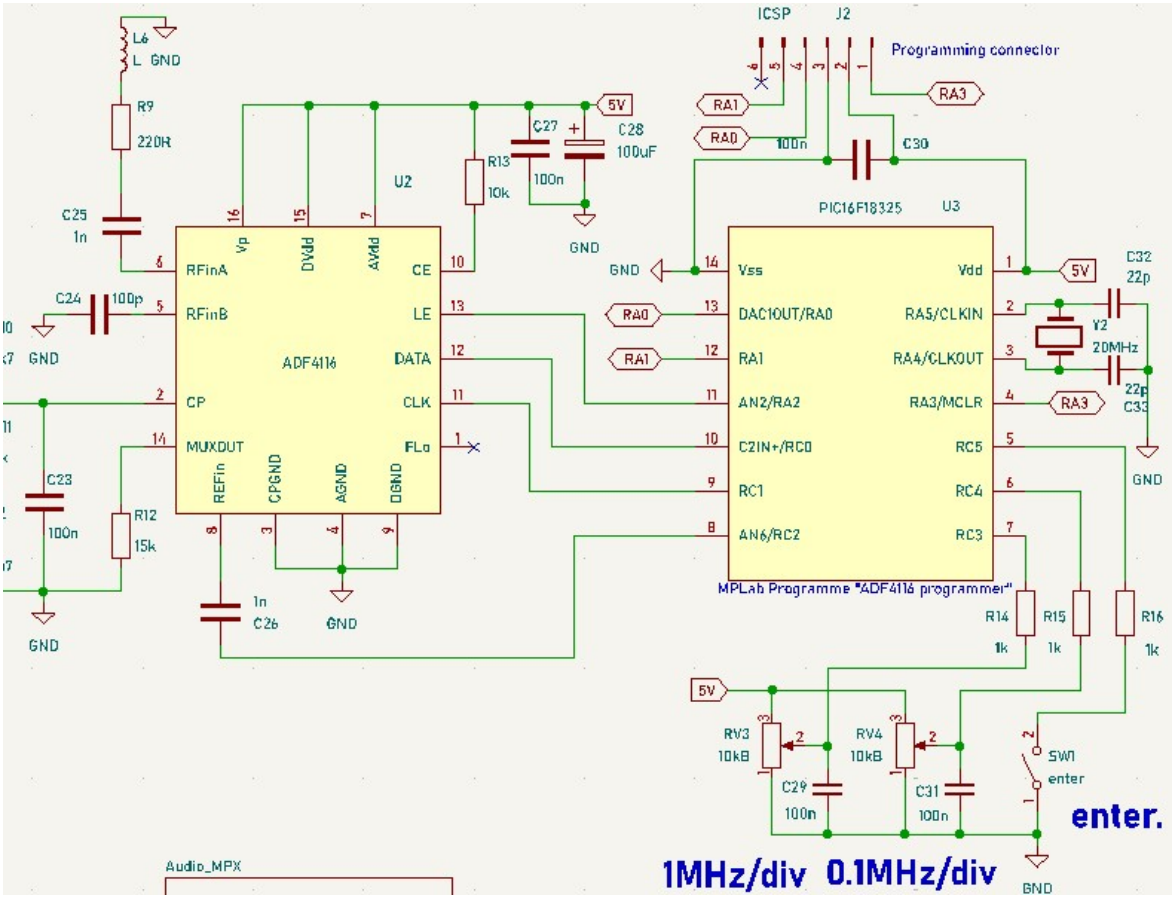
The 47Ω resistor serves to isolate the opamp's output from any stray capacitance that could cause instability. The 1kΩ resistor provides a leakage path for DC. The 470pf output cap isolates the antenna from the output of the opamp on a DC level, and provides an extra ±3Ω of impedance at frequencies in the FM range. That amounts to 50Ω total output impedance. The 10kΩ resistor to ground provides a leakage path to ground if any static voltage were present on the antenna. It protects the capacitor and the opamp output. The opamp feedback path is just a tiny copper trace. It may be advisable to use a

© Jygsawaudio 2026

small resistor there, but the trace is barely 1mm long, so the parasitic inductance is tiny and negligible. I have experienced no problems with that.

b. Fine-tuning the oscillator

- ADF4116: in order to be able to dial in precise frequencies in 100kHz steps from 88.0Mhz to 108.9Mhz, I resorted to the ADF4116 frequency synthesizer chip. It is a current-production version of the LMX2309, which is now obsolete. Pin-out is exactly the same, though.
  Getting this all to work properly required a fair amount of study and trail-and-error. It was a good day when everything finally worked as intended.



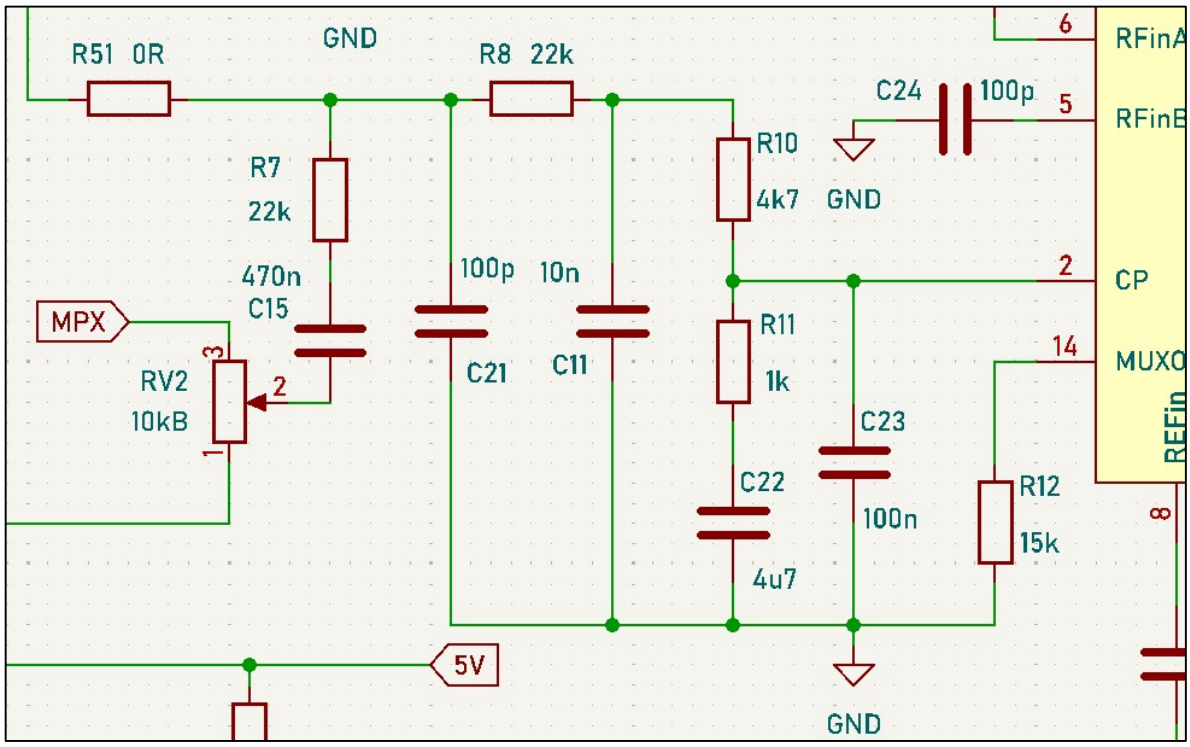
I refer to the datasheet for more detailed information, but basically the user can program the ADF4116 to monitor the RF oscillator frequency, compare it to any desired programmed frequency, and output a correction signal which is then applied through a low-pass filter to the tuning diodes. It's a programmable phase-locked loop chip.

- The **LE, DATA and CLK** pins are the inputs for programming the ADF4116. It basically requires a specific series of digital pulses to know what frequency the user wants. I programmed a PIC microcontroller to handle the ADF4116 programming routines (more on that later).
- The actual RF oscillator frequency is applied to input **RFinA**. That is achieved by means of a feedback coil on the same former as the RF coil. The feedback coil consists of only 1 turn of 0.5mm diameter wire. It turns out that it is important to keep the feedback coil at the top end of the former, and the RF oscillator coil at the bottom end. If the feedback wire loop is too close to the RF coil, it 'loads' the

© Jygsawaudio 2026

RF circuit and produces unexpected results. At least, that is what I think happens. Not 100% sure about that. Keeping both wire loops apart made things work well. 100% sure about that.
Only 1 turn was necessary since the signal amplitude in the RF oscillator coil exceeded the maximum allowed amplitude to the RFinA pin (0dBm in 50Ω, or around 320mVp). Even with just 1 turn, a dropping resistor was still necessary (R9).

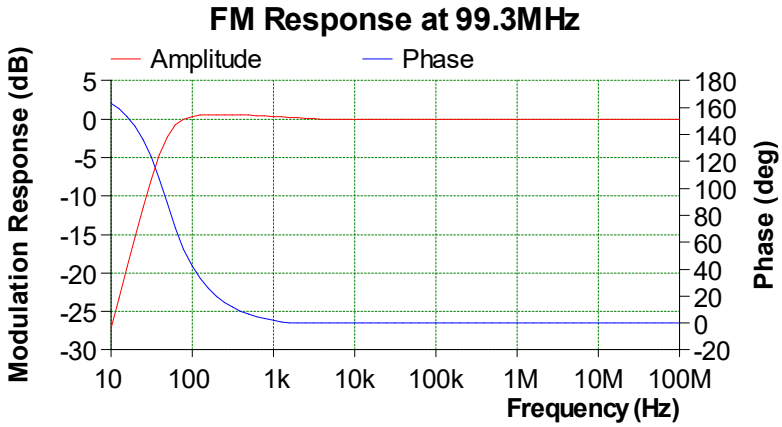
- The ADF4116 also needs a reference frequency as a clock to operate from. See datasheet again for details. In my project I used the PIC's reference clock output module. Basically the PIC's clock frequency determined by the crystal, 20MHz. That is applied to **REFin** at pin 8.
- The correction signal is output from pin 2, **CP** (charge pump). This signal is then filtered in three stages to result in a DC voltage which is then applied to the varactor tuning diodes through a 22k resistor.
- The loop filter: this was a tricky bit to get right. The phase detector as I have set up the device, runs at 25kHz, a certain subdivision of the 20MHz reference frequency, controlled by the programmable R register. The charge pump is a current source or high impedance output, pumping current (250uA) in 25kHz pulses. The error between reference and RF input determines the pulse widths. These pulses can then be filtered to obtain a DC voltage, biasing the varactors correctly. Also, you must take into account that the loop filter presents a load on its own to the MPX signal at the tuning node.



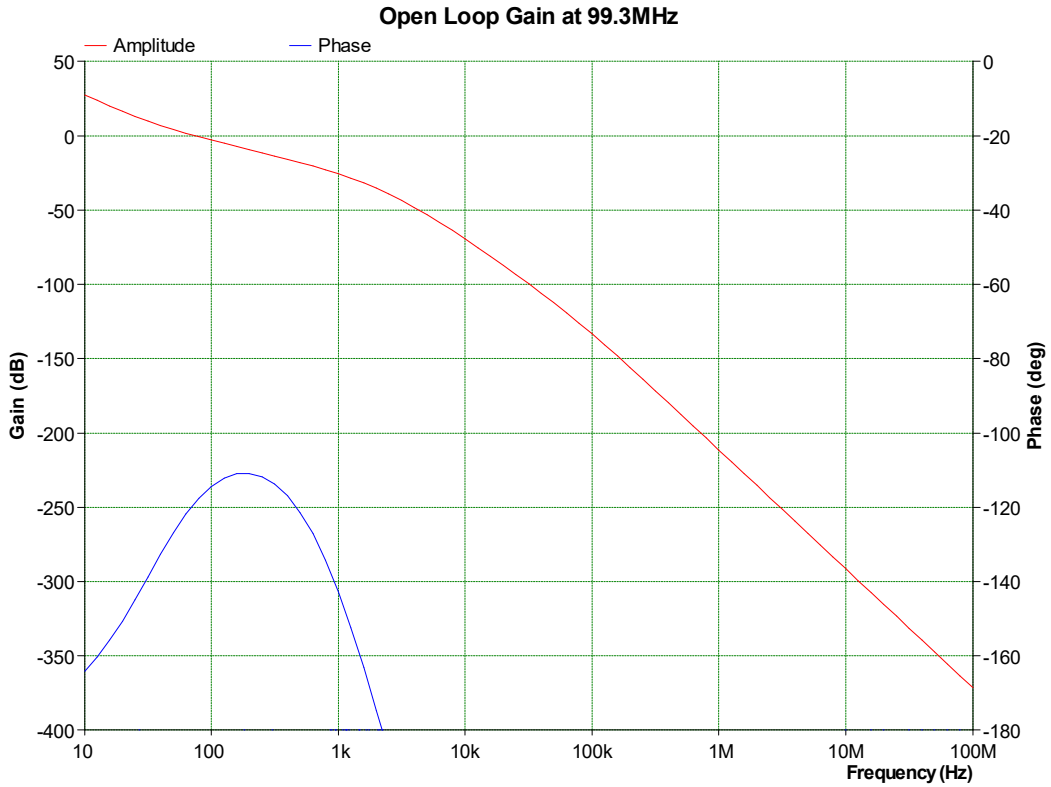
- I used the simulation tool from Analog Devices to simulate the loop filter (ADIsimPLL). This filter is a three-stage low-pass filter. Ideally the PLL is blind to the audio signals. Otherwise it would react to changes in VCO frequency, causing low-end attenuation of the mono L+R signal. On the other hand, you want the PLL to lock in to the desired frequency fast, without wobbling. With the values I finally settled on, the filter loop bandwidth is 74 Hz, meaning that signals below that frequency are being

© Jygsawaudio 2026

included in the loop and will (gradually) be corrected by the PLL. In any case, 74Hz is not a hard limit. Some bass attenuation may be expected, but only so in the mono signal (L+R) and it will be quite subtle anyway. If I wanted to transmit a 30Hz organ signal mono, it would be audibly attenuated. The stereo signal modulates a 38kHz carrier, so that lives way out of reach of the PLL loop. As does the 19kHz pilot. There will be no phase shifts there and good stereo separation is assured. This graph shows the response of the filter to modulating signals:



As can be seen, the amplitude curve rises up, reaches 74Hz at -3dB, followed by a minor hump of less than 1dB, indicating a stable system. The phase margin of the PLL loop is around 60°, which is also indicative of a stable system. In the figure below, the gain curve crosses the 0dB line when the phase angle is -120°, so there is a margin of 60° from -180°.



The first pole in the filter is created by R11 and C23, around 1.5kHz. The second pole by R10 and C11 to ground (at 3.9kHz). R11 and C22 form a low-frequency zero at around 34Hz. This zero provides phase lead

© Jygsawaudio 2026

compensation; it pushes the phase back up towards +90°, protecting the system from instability. There is also a 100p cap to ground to filter out anything far above the MPX signal.

- The PLL loop filter output is applied to the varactor node through a 22k resistor. I used a 100k first, but that introduced too much low-frequency 'rumble'. The PLL not being able to lock in fast enough? Johnson noise from the large value resistor modulating the RF? In any case, lowering to 22k gave very acceptable results, and didn't load down the MPX output too much.
- Where possible (value-wise) I used NP0 capacitors for temperature stability.
- One last thing: the 'constant' *Kv* plays a major role in the loop response. Kv basically determines how much voltage is needed to produce a certain frequency swing. It appears that Kv is not so constant. It varies around 5Mhz per volt in this case, depending on the actual DC voltage level applied to the varactor node. Setting the DC voltage around midway between extremes gives the best reception results, so I measured that value and based all calculations on that figure. I can move the slug in the inductor to reach that midway point for any given frequency. You don't even need any measurement device. It can be tuned by ear, since the audio quality starts to degrade once the DC bias is approaching one of the extremes. In addition, although I can dial in any frequency I want with this circuit, I will probably mostly stick to only 1 commercially unused frequency anyway.

- <u>PIC programming</u>:
  The question was then: how do I make the microcontroller load the correct series of pulses into the ADF4116, so the latter adjusts the tuning voltage to the diodes exactly right for 210 different frequencies?
  My basic idea was to use the PIC's ADC module to read the positions of 2 potentiometers, one for the 1Mhz/division, one for the 0.1Mhz/division. The ADC action occurs in an interrupt routine that is only accessed when a button is pressed. So when the button is pressed, the ADC reads the positions of both pots, and the results are stored in two dummy variables. This avoids repetitive button presses to skip through frequencies, or unwieldy rotary switches, etc. I'm sure there are other, maybe more compact ways to achieve the same goal, but that is what I went for. Plus the microcontroller hardly does anything most of the time, except when the button is pressed.
  Now what sequence of data pulses corresponds to one desired frequency? That question is answered in the ADF4116 datasheet, but you will spend a lot of time calculating. Luckily, the internet exists. And people exist who write programmes that do just that. The programme can be found here:
  https://www.rfcandy.biz/communication/lmx2306.html. It is written for the obsolete LMX2306 chip, but works just as well for the ADF4116. You just input the reference frequency (20Mhz in my case), the desired RF frequency, and the programme spits out the required sequences in hexadecimal format.
  If you check the ADF4116 datasheet, you see that there are 4 registers to be filled in order for the chip to have enough information. One of those is an 'initialization' register and needs to be written only once. The other three are the function register, the R register and the N register.

© Jygsawaudio 2026

As it turns out, the LMX2306 calculator presents a list of results per desired RF frequency. There are many possible combinations. For each of the 210 different frequencies in my case, the R register value could remain the same throughout. The only one that needed adjustment for every frequency was the N register. That was good news, because it meant I would not need to worry about not having enough memory in the PIC to store all the hex numbers. The registers contain 21 bits, so an 'int' variable (16-bit variable) wouldn't do. I had to use a 'long' variable to store the values, which takes up more memory.
So one hex number for the R register, 210 hex numbers for the N register. I stored all the N values in a look-up table in the micro:

```
long preN[210] = {0xDC01,0xDC11,0xDC81,0xDC91,0xDD01,0xDD11,0xDD81,0xDD91,
0xDE01,0xDE11,0xDE81,0xDE91,0xDF01,0xDF11,0xDF81,0xDF91,0xE001,0xE011,0xE081,
0xE091,0xE101,0xE111,0xE181,0xE191,0xE201,0xE211,0xE281,0xE291,0xE301,0xE311,
0xE381,0xE391,0xE401,0xE411,0xE481,0xE491,0xE501,0xE511,0xE581,0xE591,0xE601,
0xE611,0xE681,0xE691,0xE701,0xE711,0xE781,0xE791,0xE801,0xE811,0xE881,0xE891,
0xE901,0xE911,0xE981,0xE991,0xEA01,0xEA11,0xEA81,0xEA91,0xEB01,0xEB11,0xEB81,
0xEB91,0xEC01,0xEC11,0xEC81,0xEC91,0xED01,0xED11,0xED81,0xED91,0xEE01,0xEE11,
0xEE81,0xEE91,0xEF01,0xEF11,0xEF81,0xEF91,0xF001,0xF011,0xF081,0xF091,0xF101,
0xF111,0xF181,0xF191,0xF201,0xF211,0xF281,0xF291,0xF301,0xF311,0xF381,0xF391,
0xF401,0xF411,0xF481,0xF491,0xF501,0xF511,0xF581,0xF591,0xF601,0xF611,0xF681,
0xF691,0xF701,0xF711,0xF781,0xF791,0xF801,0xF811,0xF881,0xF891,0xF901,0xF911,
0xF981,0xF991,0xFA01,0xFA11,0xFA81,0xFA91,0xFB01,0xFB11,0xFB81,0xFB91,0xFC01,
0xFC11,0xFC81,0xFC91,0xFD01,0xFD11,0xFD81,0xFD91,0xFE01,0xFE11,0xFE81,0xFE91,
0xFF01,0xFF11,0xFF81,0xFF91,0x10001,0x10011,0x10081,0x10091,0x10101,0x10111,
0x10181,0x10191,0x10201,0x10211,0x10281,0x10291,0x10301,0x10311,0x10381,0x10391,
0x10401,0x10411,0x10481,0x10491,0x10501,0x10511,0x10581,0x10591,0x10601,0x10611,
0x10681,0x10691,0x10701,0x10711,0x10781,0x10791,0x10801,0x10811,0x10881,0x10891,
0x10901,0x10911,0x10981,0x10991,0x10A01,0x10A11,0x10A81,0x10A91,0x10B01,0x10B11,
0x10B81,0x10B91,0x10C01,0x10C11,0x10C81,0x10C91,0x10D01,0x10D11,0x10D81,0x10D91,
0x10E01,0x10E11,0x10E81,0x10E91,0x10F01,0x10F11,0x10F81,0x10F91,0x11001,0x11011
};          // all values for N: Vco frequency from 88Mhz to 108.9Mhz in 0.1Mhz increments
```

Loading the required sequence of bits into the register was achieved with a simple 'for' loop:

```
/* ADF4116 CHIP N REGISTER DATA LOAD */

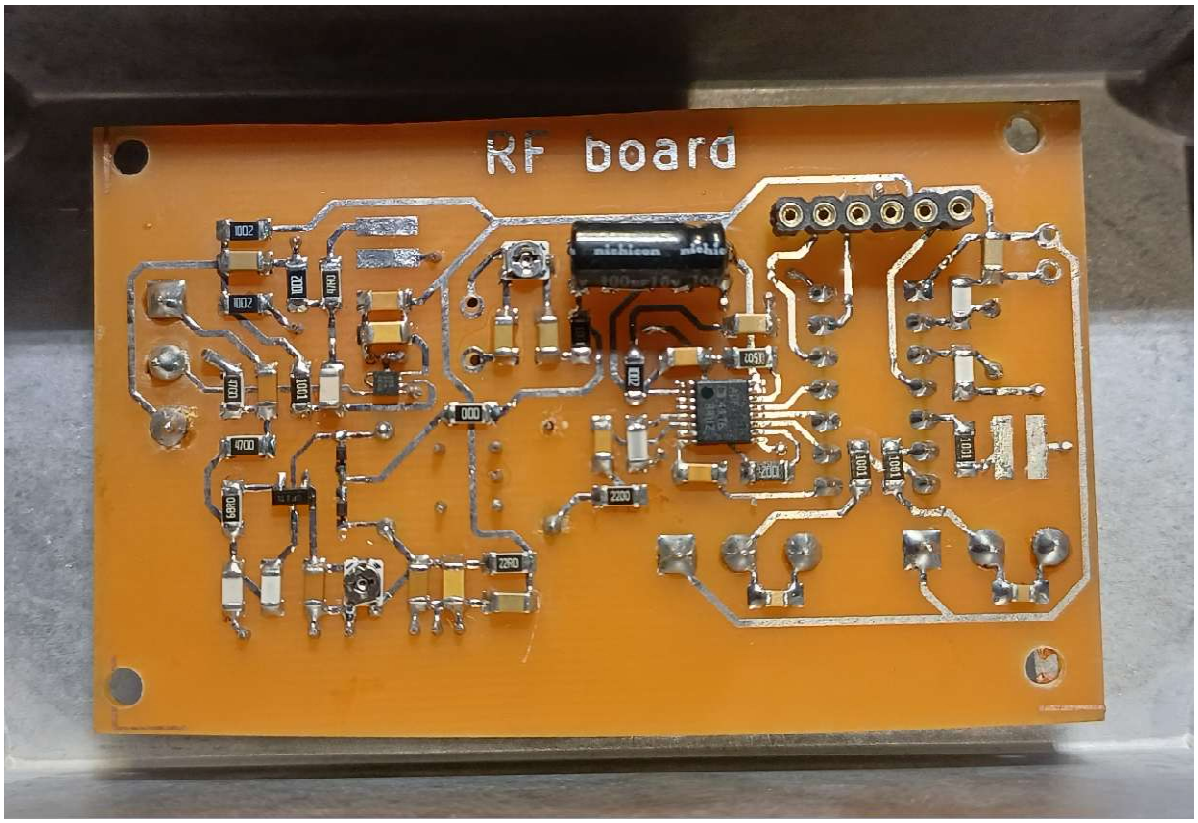
for(i=20; i >= 0 ;--i){     // loop is executed as long is i is not less than zero and counts down

    LATCbits.LATC0 = ((N >> i) & 1);  // load into DATA per bit, from MSB to LSB.
    __delay_us(1);
    LATCbits.LATC1 = 1;     // set CLK high
    __delay_us(1);
    LATCbits.LATC1 = 0;     // set CLK low again
}
__delay_us(1);
LATAbits.LATA2 = 1;         // LE high to load DATA into N latch
__delay_us(1);
LATAbits.LATA2 = 0;         // LE low for DATA input
```

After each loaded DATA bit, the CLK pin must be set high, then low again for the next bit. When the 21-bit sequence is completed, the LE bit must be set high, at which point the register is transferred into the chip's N latch, completing the process.
The dummy variable 'N' here is just a number, derived from the ADC result, and corresponds to the number of a hex value in the lookup table above.
A very similar routine is used for the R and F registers. The rest of the programme is just switch debouncing, ADC action and converting ADC results into usable numerical values.

© Jygsawaudio 2026

- This is the entire RF section on the pcb. Pots and oscillator coil on the back. The backside

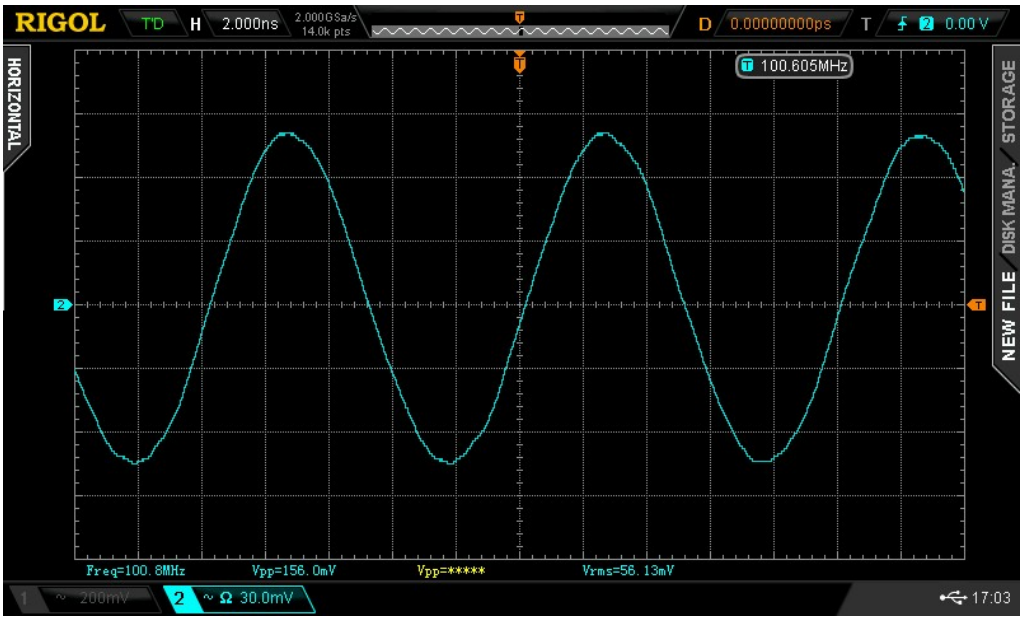


of the pcb is a copper plane. Check the size of that OPA859 to the middle-left. It is smaller than the 1206 capacitors.
Lay-out is important here. All 'sub-circuits' (the RF oscillator circuit, the PIC and ADF4116 section) run off the 5V supply, but they don't share power traces and they are all well-decoupled. There are numerous vias to the ground plane underneath to reduce stray inductances.



© Jygsawaudio 2026

This is the output of the transmitter, into the 50Ω oscilloscope input, set to 100.6MHz.



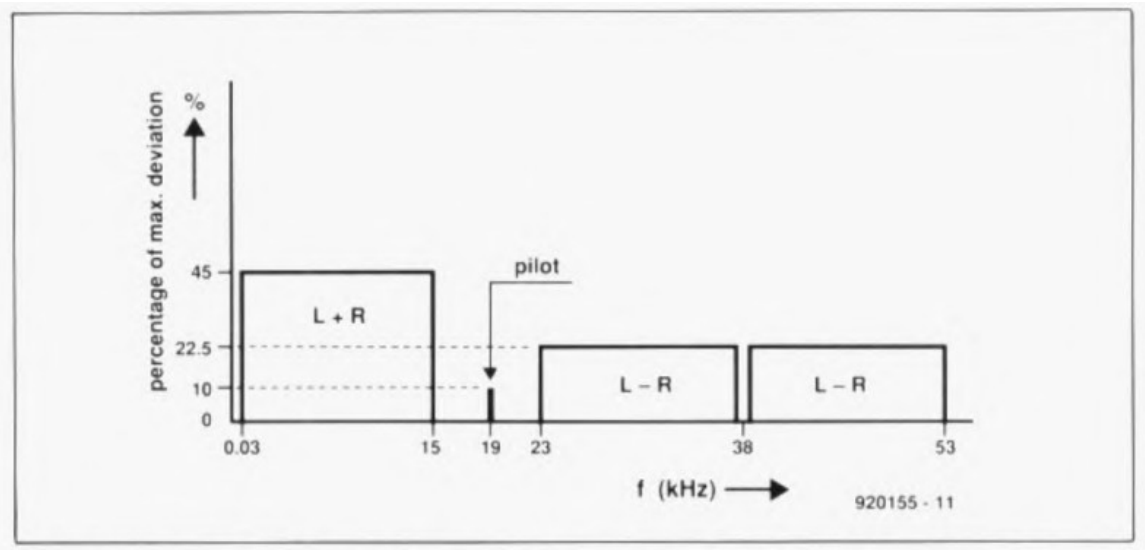
Output amplitude into 50Ω is 56mVrms, or around -12dBm. That is more than enough for an 'inside the house' transmitter.

2. Subcarrier and pilot generator

a. PIC again

When transmitting a simple audio signal, frequency-modulating a RF carrier wave, there is no way for the receiver to distinguish between the left and right channels. Therefore, there is more than just the audio signal to modulate the carrier. You need three 'information packages' in the modulating signal:

- the L+R signal or mono signal
- the L-R signal, amplitude-modulating a 38kHz sine wave subcarrier
- a pilot tone of exactly half the frequency of that 38kHz subcarrier.



© Jygsawaudio 2026

You can see from the figure above that the subcarrier itself is not included in the information package. *'Double sideband suppressed carrier modulated waveform'* is the term to describe the signal from 23kHz to 53kHz. Since the L-R audio information is supposed to be hi-fi, 30Hz is the lower limit, meaning that the audio information is contained within the left sideband from 23kHz to 37970Hz and in the right sideband from 38030Hz to 53Khz, if an audio bandwidth of 15kHz (minus the 30Hz) is to be achieved. Since the subcarrier itself is only useful for demodulation in the receiver, it would be painstaking to devise a filter to get rid of it. Living in a very crowded part of the spectrum, between the two sidebands, it could also cause unwanted effects such as aliasing and it would just be consuming extra power to be transmitted. Hence the so-called Zenith-GE system for mono + stereo broadcasts.

A weak 19kHz signal is sent along with the rest of the signals. 19kHz lies midway between the mono L+R block and the L-R left sideband. Its neighbours are 4kHz away on either side. 19kHz is also exactly half of 38kHz and as such can be used to recreate the 38kHz by frequency doubling in the receiver. It is important, however, that the 38kHz and 19kHz are exactly in phase in the total multiplex signal to be transmitted. Otherwise, the receiver will have troubling separating the stereo signals.

In the original Elektor magazine articles the authors use different methods to obtain a 38kHz subcarrier and 19kHz pilot tone. One makes use of a CD4060 binary ripple counter and a crystal oscillator producing a clock signal that is an exact multiple of 38k. The clock signal is divided down to 38k and 19k, after which the square waves are low-pass filtered to sine waves. Filtering causes phase shifts, so to assure that both are exactly in phase, an old PLL chip is used (the now obsolete XR2208CP). The other uses the same chip, but combined with a series of XOR gates for audio tone generation. I wanted to keep everything compact and decided to try to program a PIC16F18313 to produce the 38k and 19k and filter the square waves down to sine waves, as in the old Elektor article. I could make provisions in the software for programmable delays to compensate for the phase shifts after filtering. The only thing that matters is that both are in phase in the final MPX signal.

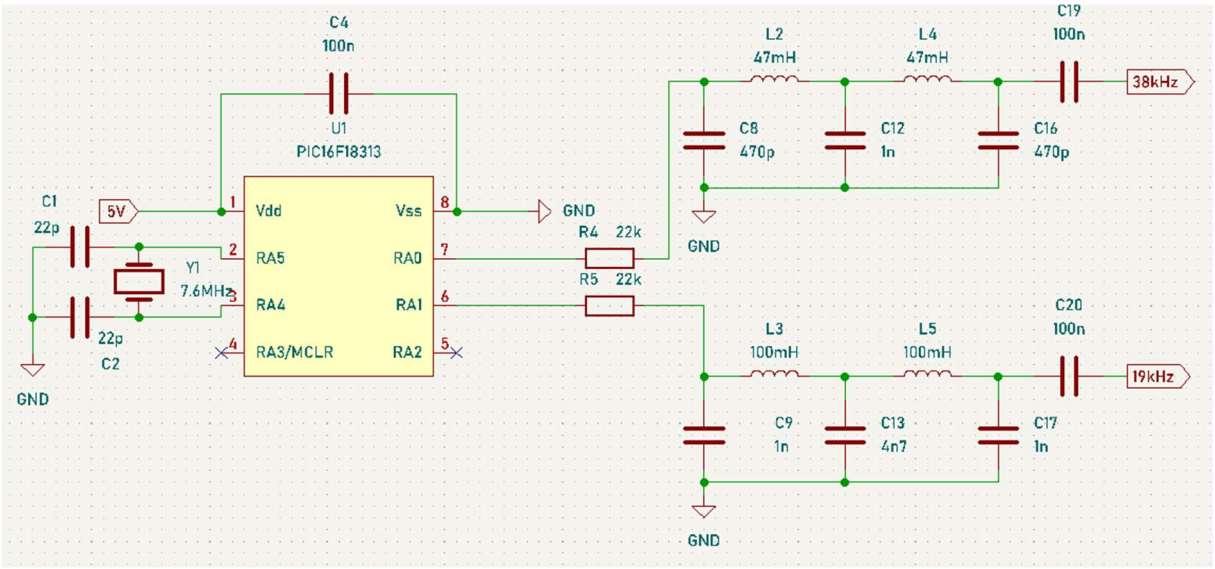
b. The programme

In order to get a precise, or at least very close to precise, 38kHz signal, I used a 7.6Mhz crystal as an external clock source. 7.6MHz divided by 200 is exactly 38kHz. The PIC I used is an 8-pin 16F18313 (which I use for many applications). It has a PWM module that runs off a timer (Timer2). Duty cycle is set to 50% and setting the PR or period register correctly gives you a PWM signal of 38kHz with fixed 50% duty cycle.

The next question is: how do I generate another signal of exactly half that frequency and be sure that it will be and stay in phase with the first one after filtering? Reactive components cause phase shifts that are not easy to predict (mathematically, yes, but in real life not so much). I had decided before to use the filtering used in the first Elektor article: for each of the two signals a fifth-order RC and LC filter, giving a steep roll-off and theoretically very good for rejecting the 3rd and higher harmonics (the PWM signals are square waves and contain

© Jygsawaudio 2026

odd harmonics only). The advantage of this approach is simplicity and fewer components. Disadvantages: the inductors are on the large side (100mH and 47mH). The 3rd harmonic would be -64dB in amplitude compared to the fundamental in the 19kHz signal.



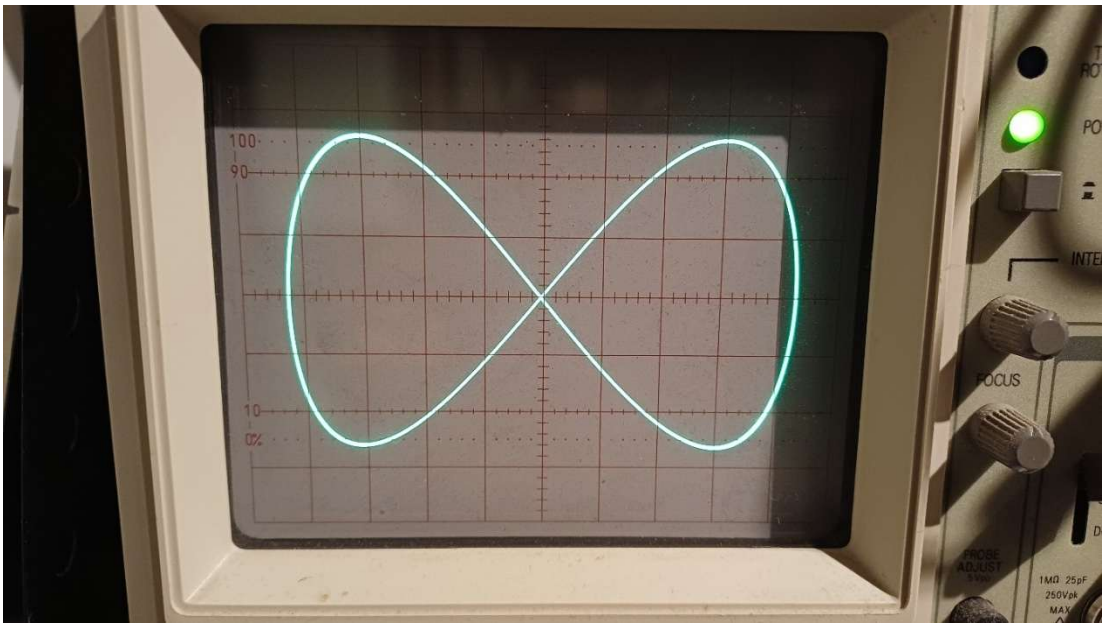
So after much trial-and-error with the 19kHz signal generation off the 38kHz, I landed on the following. A copy of the 38kHz signal, a second PWM module running off the same timer as the first, but with a separate duty cycle register, is fed into the CLK input of a J-K-flip-flop (using the Configurable Logic Cell peripheral). The J and K inputs are kept at a logic 1, the R input is kept at a logic 0. In that configuration, each time a rising edge[1] happens at the CLK input, the output of the flip-flop logic cell is toggled. That way, a 19kHz square wave is generated that is exactly in sync with the 38kHz. But:

- as said before, there is substantial delay caused by the analog filtering. I had to find some way to adjust the relative phase of both signals. That could be done in code, but as it turned out, relying on the PIC CPU for consistent and precise timing purposes is not such a good idea. There are several things the CPU has to do while your code is running, and some timing events come too late, or are missed due to the CPU not being able to truly multitask. The best option is to implement everything through the hardware inside the PIC, and bypass the CPU entirely for signal handling. So by adjusting the duty cycle of the second PWM signal and inverting, I had a way to control when the rising edge occurs on the second 38kHz signal, and thus on the 19kHz signal, with good enough resolution. One cycle of 38kHz equals 26.3µs and the duty cycle register for 50% needs to be 400. So I can shift the duty cycle, or the falling edge, say from 400 to 600 (50% to 75% duty cycle) and that would give a time difference of ¼ period between the falling edges of both signals. Hence I have a resolution of $26.3\mu s/800 = 32.89ns$ for precise adjustment.

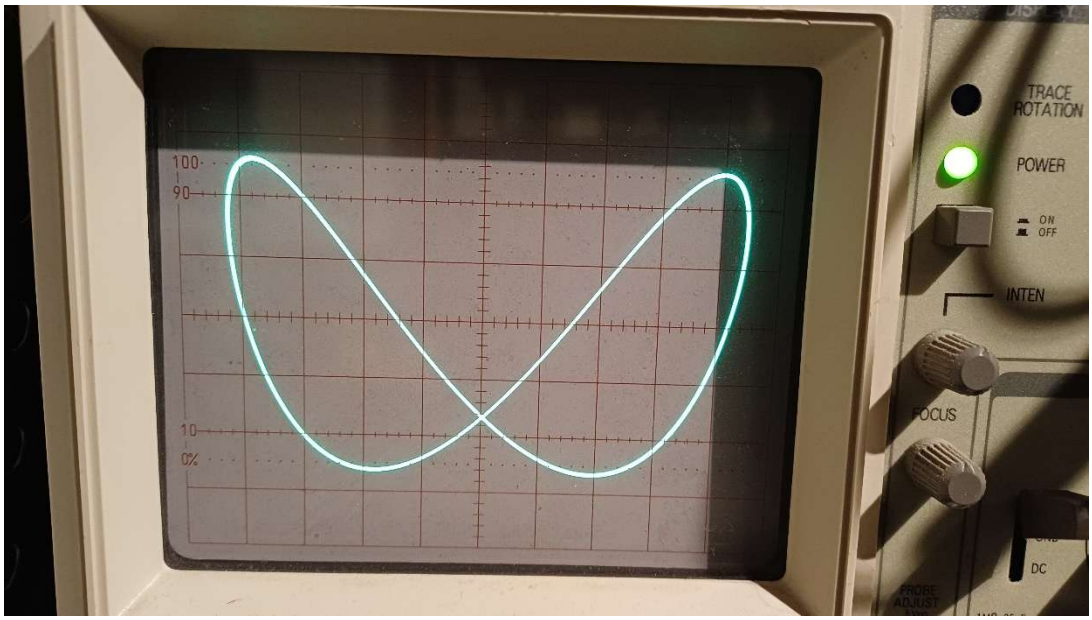
[1] Actually, the JK flip flop in the CLC module in the PICs are pulse-triggered, not edge-triggered, inducing some head-scratching… In effect, the output of the flip flop toggles on the falling edge of the clock pulse. Nothing in the datasheet about that fact. They could've at least mentioned it.

© Jygsawaudio 2026

Measuring the phase angle between two signals is easy enough on digital oscilloscopes, but a more pictorial way of checking the phase alignment is by using the XY mode on an oscilloscope. Analog scopes are great for displaying the resulting Lissajous figures. The 19k signal is sent to the X input, the 38k to the Y input of the scope. The resulting figure should look like a butterfly pattern, with perfectly symmetrical ‘wings’ and a crossing point exactly in the center of the screen. See the figure below.



If the phase alignment is off, you will see the horizontal symmetry degrade. The picture below shows a rather large phase difference of around 20°. That is not what you want to see.



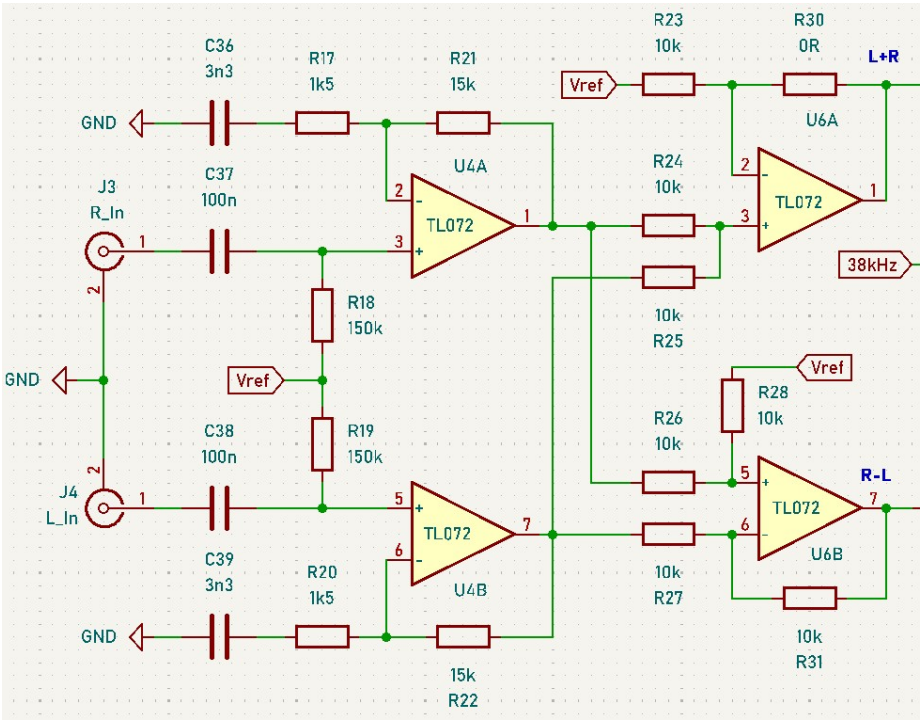
© Jygsawaudio 2026

An important note is that any reactance will introduce phase shifts, so when you measure the phase alignment of the two signals, you must make sure that both signals encounter no more capacitance or inductance when they are summed in the final summing amplifier. Both circuits should be electrically symmetrical as well. I used NP0/C0G capacitors in both filter circuits, so their capacitance stays stable even when ambient temperature changes.

Regarding signal levels. The 38kHz signal is meant to be modulated by the L-R signal. It should be large enough in amplitude, but not too large. Around 2Vpp is good. It will be multiplied in the analog multiplier IC with the audio signal, so if the amplitude is too high it will distort in the multiplier. The 19k signal should be summed at the summing amp with an amplitude of 9 to 10% of the amplitude of the total MPX signal. This is quite critical, actually, because too much will cause stereo degradation, too little and the receiver cannot lock into it and you'll have no stereo at all. I used a pot to dial in the correct final amplitude to the summer.

3. Audio and MPX circuit

a. Audio amplifiers

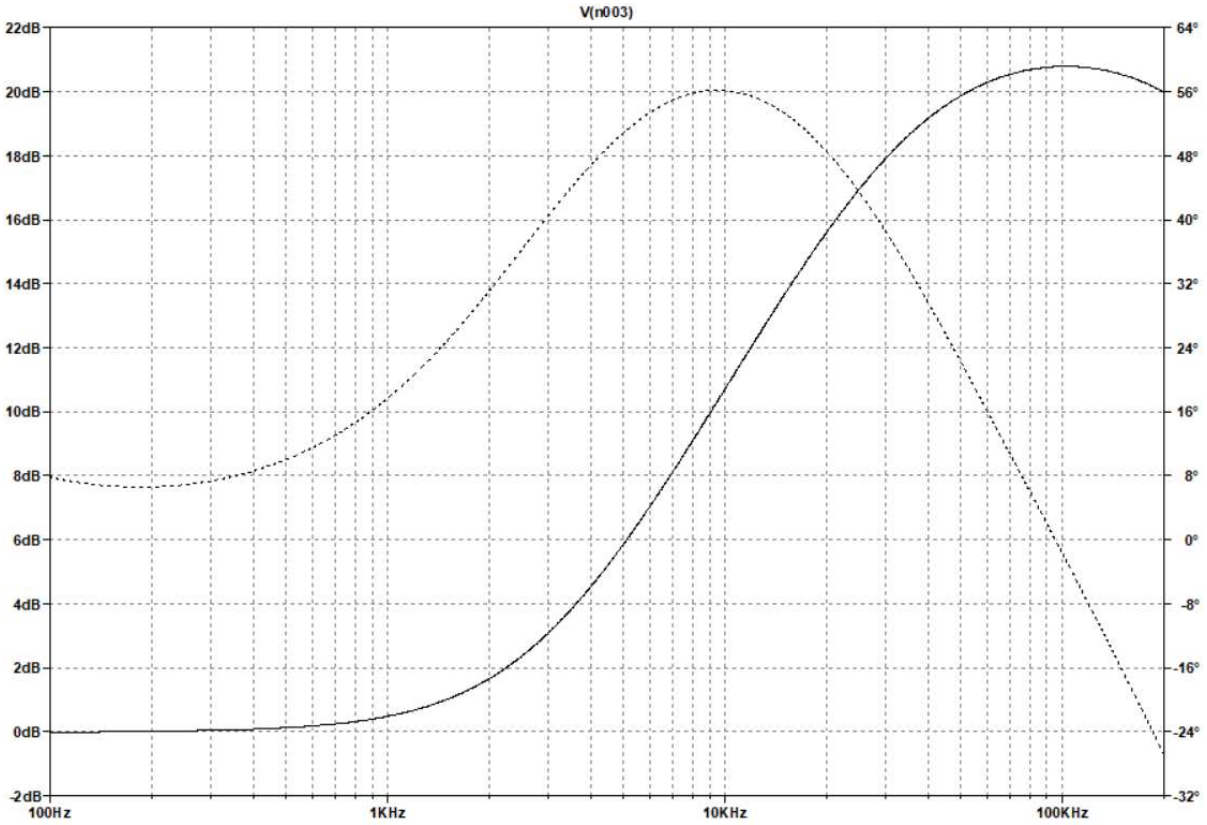


- Pre-emphasis

  Because of the way a receiver's discriminator works, FM creates more high-frequency noise or hiss (noise power increases with the square of the frequency). To combat this effect, audio signals are 'pre-emphasized' to make the high-frequency content larger. The receiver then incorporates a 'de-emphasis' circuit to balance that out again and crush the noise at the same time.

© Jygsawaudio 2026

The pre-emphasis circuit above is an almost exact copy of the circuit described in the 1993 Elektor article. Beware though that there is a mistake in the original article (33n instead of 3n3; later corrected).



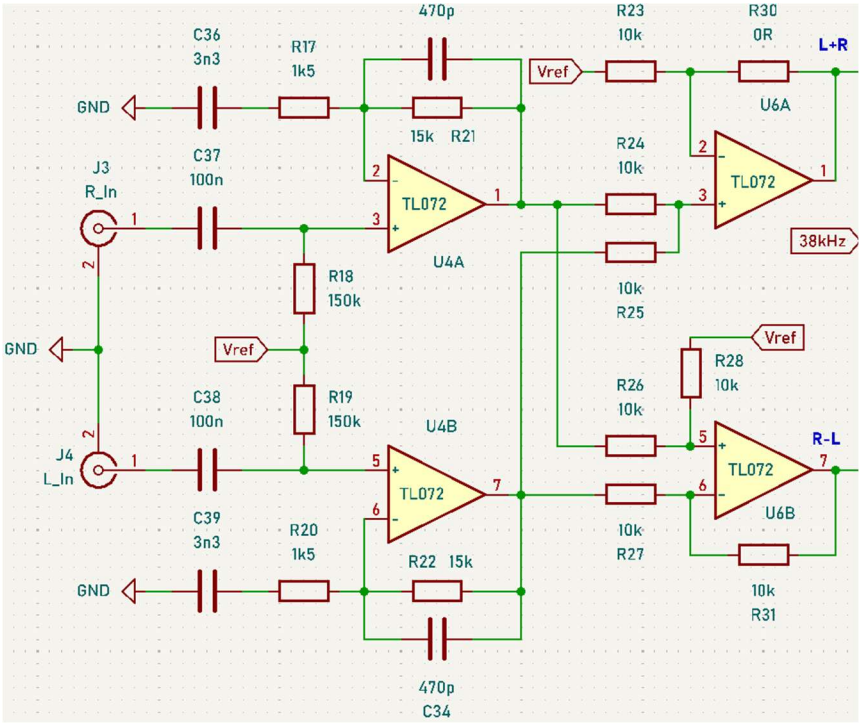
The output of the pre-emphasis circuit looks like the above. There is a gain slope starting from 0dB, reaching +3dB at 3.2kHz and rising at 6dB per octave until a maximum of ±20dB is reached at ±100kHz.

The slope itself is a standard requirement for FM broadcasting. The required RC time constant τ is 50us. In the circuit above, this τ is achieved by a non-inverting amplifier configuration. At DC the gain is 1 or 0dB. At 3.2kHz the gain is 2 or +3dB relative to the DC level gain of 1, since the 3n3 capacitor's reactance is then around 15kΩ. As frequency is increased and there is only one reactive element in the circuit, the gain increases by 6dB per octave. The 1.5kΩ resistor creates a shelf at +20.8dB for very high frequencies. So the τ is defined by 15kΩ×3.3nF ≈ 50us.

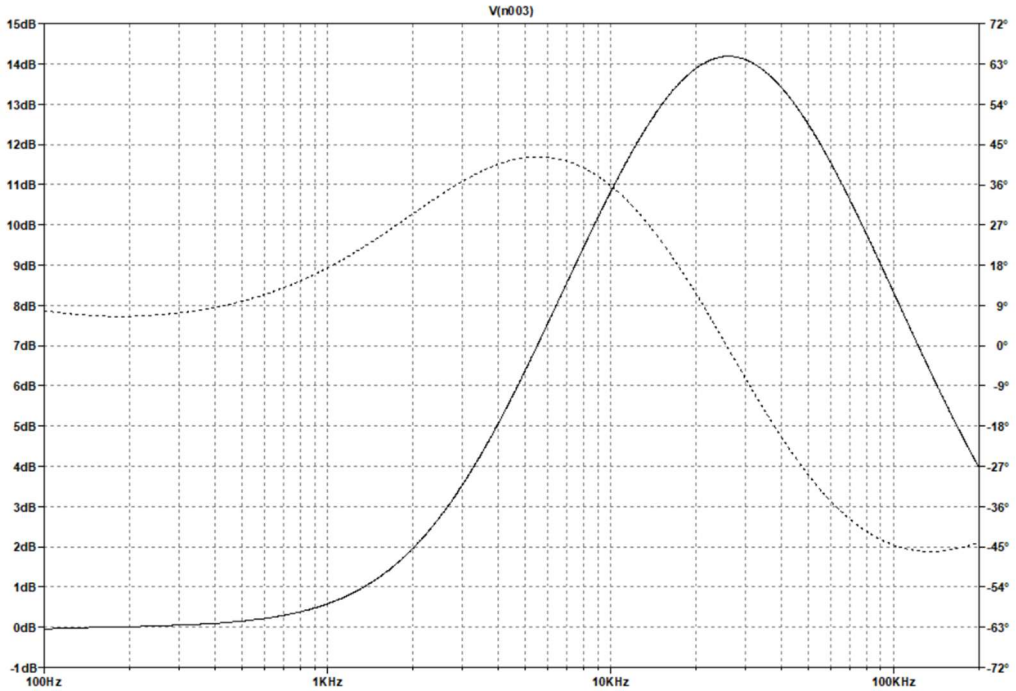
However, in more advanced transmitters this would not be enough. A second more complicated filter would be required for a very sharp roll-off for anything higher than 15kHz. The reason for this is that the audio signal may well contain frequencies above that and may even contain the 19kHz that we have as our pilot signal. That may lead to undesirable effects: the audio frequency interfering with the pilot signal, confusing the receiver. It might think that the 19kHz audio signal is the pilot.

© Jygsawaudio 2026

I do not intend to build an extensive 5th or 7th order filter to get ≤-1dB at 15kHz and ≥-30dB at 19kHz. Instead I opted for a simple addition to the above circuit: a 470p capacitor in parallel with the feedback resistor. Looks like this:



And the resulting gain curve:



The result is fine for me: a steady +6db/octave with a slightly higher 'knee' at 3.2kHz of +3.8dB, and a slightly reduced amplitude at 15kHz of +13dB instead of +13.6dB before. It knocked around 2dB off the 19kHz signal.

© Jygsawaudio 2026

There is hardly any audible difference between the two versions. I can't hear it anyway. In addition, most audio signals from computers or smartphones don't contain any frequencies above 15kHz anyway.

For completeness' sake, the 100n capacitor acts as a DC blocking capacitor and forms a high-pass filter with the input resistor of 150kΩ. The -3dB roll-off towards the low end happens at 10.6Hz, far below the audio spectrum.

- Adding and subtracting the L and R audio signal

The L and R pre-emphasized audio signal then goes into a non-inverting summing amp U6A, which produces the L+R mono signal at its output. In the original schematic, U6A has a gain of 2 (symmetrical with the difference amp for the L-R), but that did not appear to be necessary. A unity gain amplifier produces plenty of amplitude for feeding the mono signal current into the final summing amp.

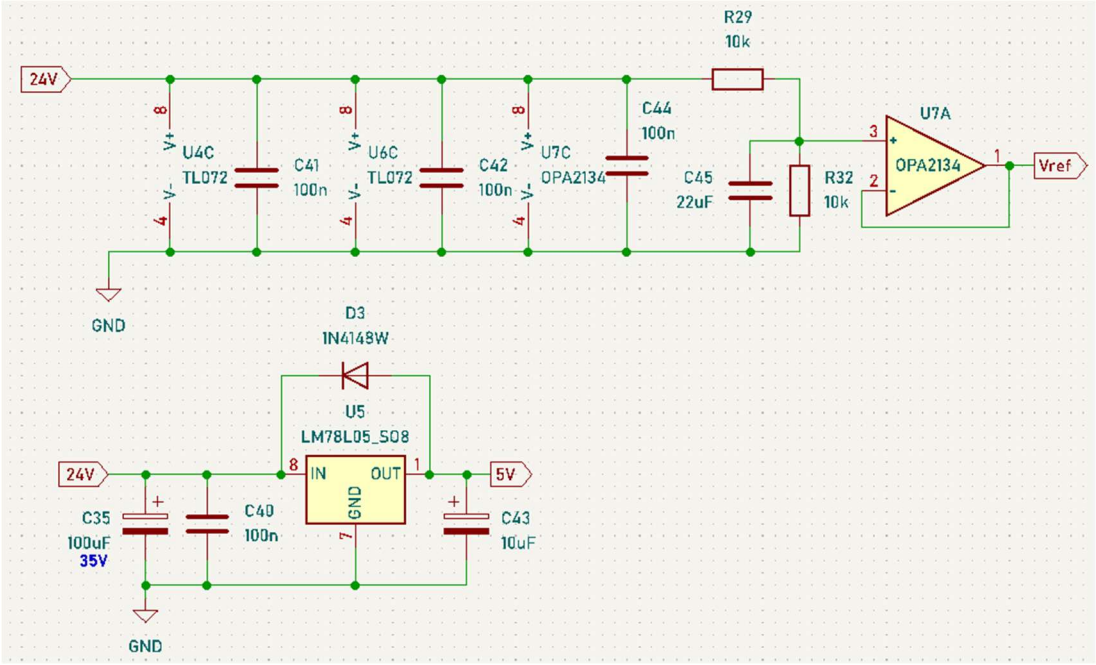
U6B is the difference amplifier. In the original schematic, the L signal is subtracted from the R signal (R-L). The reason for that is that the original XR2208 IC X-input is inverting. So the result after multiplication with the subcarrier is a 38k modulated by –(R-L) = L-R. In my design I had a choice since both the – and + inputs for both X and Y are available. I chose the same configuration as the original, so the difference amplifier produces R-L, which is then fed into the inverting Y-input.

An important fact here is that the L+R, R-L and 38kHz signals are all DC coupled to their respective nodes. As said before, the circuit must remain symmetrical to avoid phase shifts. AC coupling with capacitors would introduce phase shifts and mess things up (unless very large caps were used).

The power supply for this part of the circuit is a 24V single supply, though it may run off a lower supply voltage, provided it is high enough for the opamps and the multiplier to handle the signals without distorting. Suppose a signal voltage of 1Vp is applied to the inputs, the first amplification stage will output +14dB or roughly ×5, so 5Vp at 20kHz (see the Bode plot above). To be safe you'd best have a supply voltage of 10V plus additional margin of 2V either way because the opamps' inputs and outputs don't go rail-to-rail. I'd say 15V is the minimum. The original circuit operates from 12V, which seems a bit low for peace of mind.

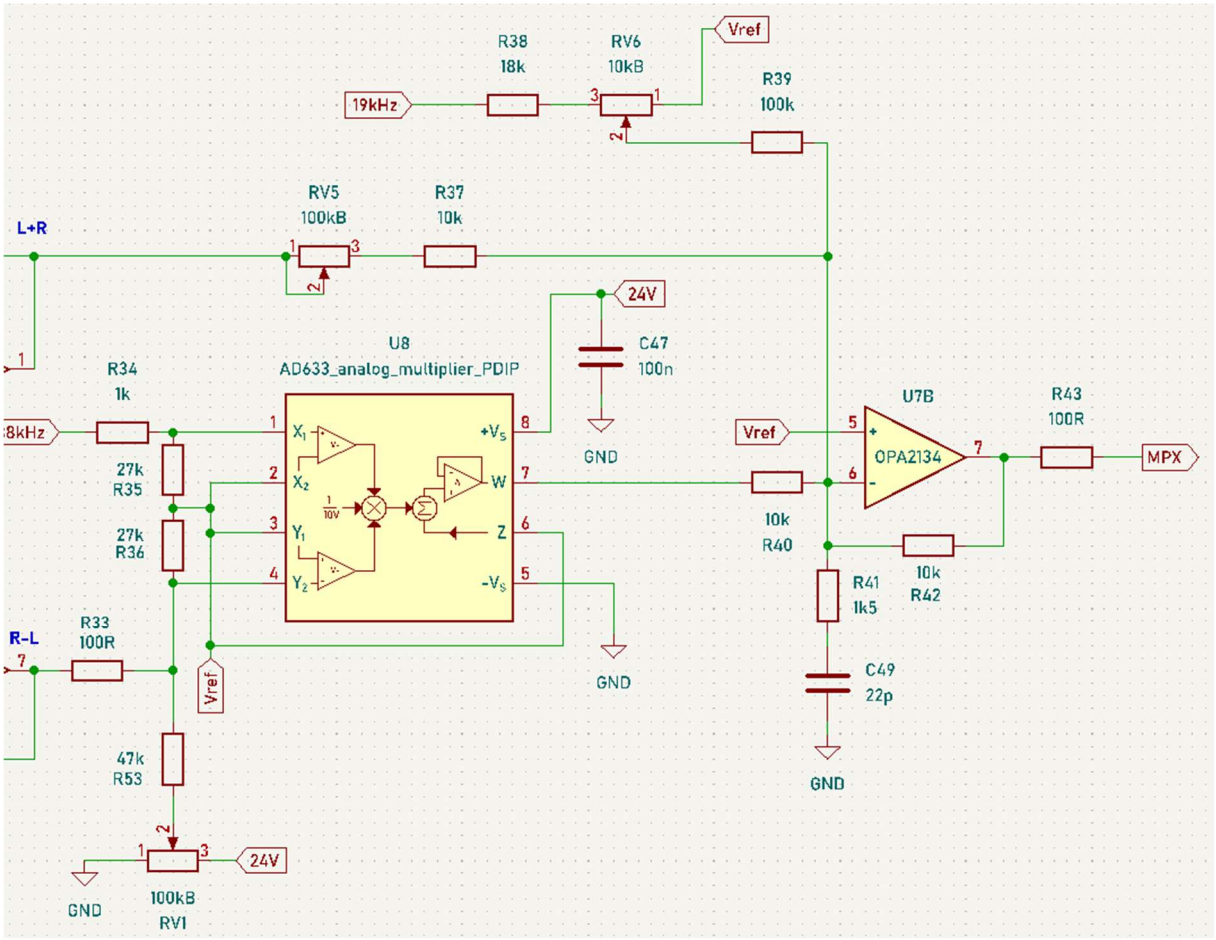
Opamp U7A provides the reference voltage of Vcc/2. The other half of this OPA2134 is the final summing amplifier. The 5V supply is for the RF board.

© Jygsawaudio 2026



- <u>The AD633 analog multiplier</u>

The old XR2208 chip is obsolete. In my design I replaced it with an AD633 IC; rather expensive devices, unfortunately. The AD633 is quite simple to use. It has four input pins X1, X2 and Y1, Y2. These are high-impedance



© Jygsawaudio 2026

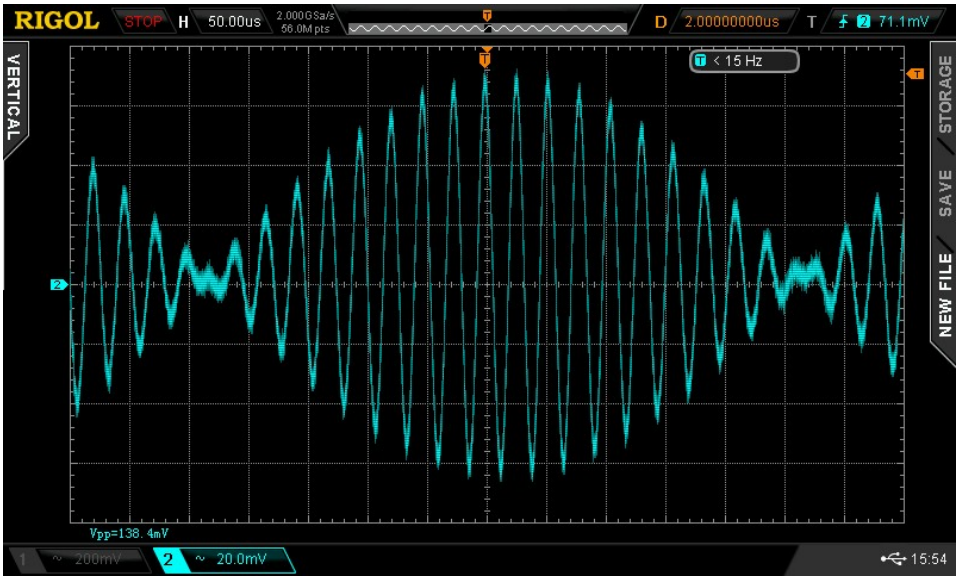
inputs for the signals to be multiplied. The '1' inputs are the non-inverting inputs, the '2' pins are the inverting inputs.
The Z-input is a summing input, as can be seen from the component symbol. In this design, it is kept at Vref. The multiplication result is buffered internally and available at the W-output pin.

So the 38kHz subcarrier is applied to the X1 input, the R-L to the Y2 input. The output is then $W = \frac{(X_1 - X_2)\cdot(Y_1 - Y_2)}{10} + Z$. The X2 and Y1 inputs in my circuit are AC grounded, DC-referenced to Vref, as is the Z input. These are all 0, resulting in an output voltage that equals $\frac{(38\ )\cdot(-(R-L))}{10}$, which is an amplitude-modulated 38kHz sinewave, modulated by the L-R signal. The result is divided by 10 internally to keep voltage levels reasonable after multiplication. That last fact proved to be a good thing, since the varactor tuning diodes are very sensitive devices and don't need much modulation voltage.

Looking at the equation above, if the (Y1-Y2) term is not exactly 0, some of the carrier will appear at the output. That is, even if a very small amount of DC offset is present at the Y1/Y2 input, that term will not be 0. In a dual sideband suppressed carrier signal none of the carrier is supposed be in the MPX signal. The AD633 is not internally compensated, and there is no compensation pin (as in the larger and more expensive AD632). So I had to add a DC offset compensation circuit. Looking on a spectrum analyser at the output, a weak 38kHz signal was still present even without audio input. Then it was just a matter of turning RV1 until a null was reached and the output W was truly 0.

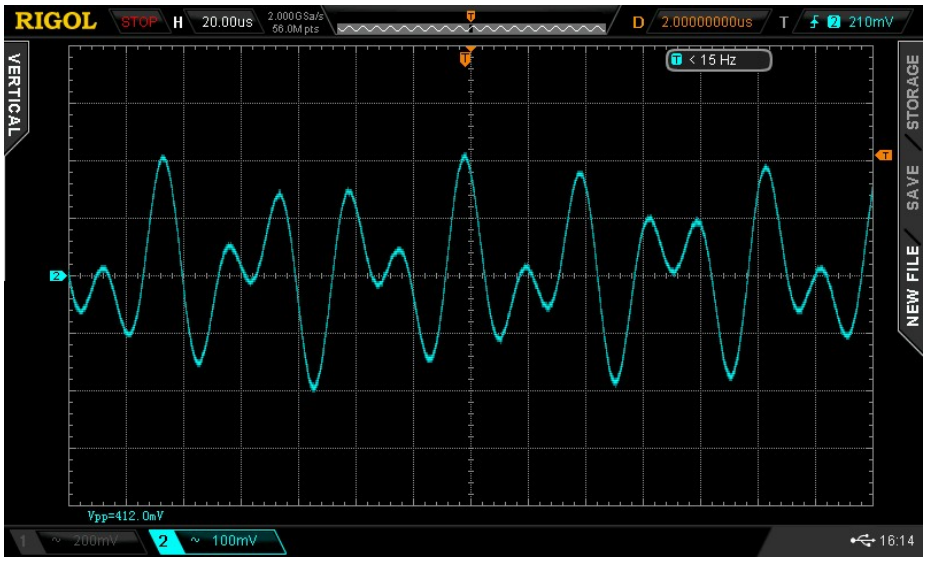
- <u>Mixing the signals</u>

Feeding a 1kHz 1Vpp signal into the left input, no signal into the right input, results in the following waveform at the output W of the AD633:

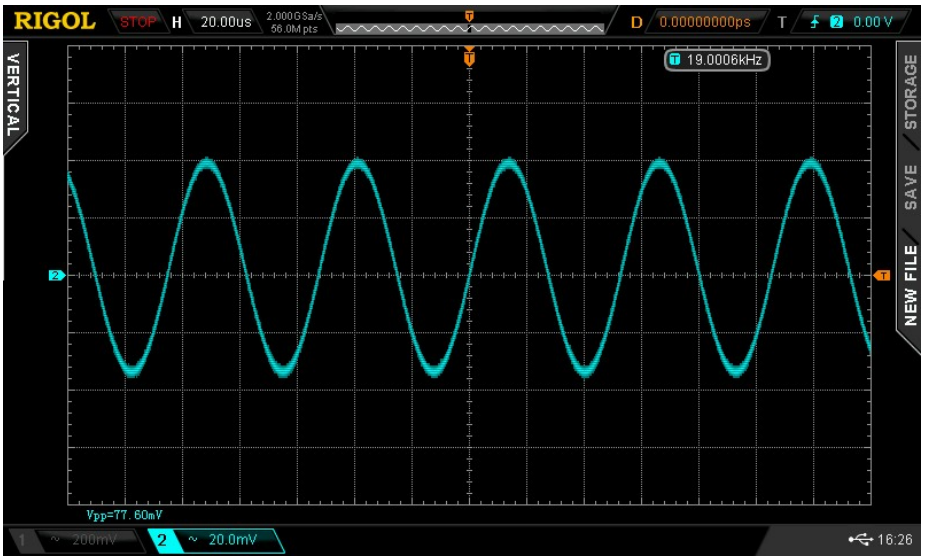


© Jygsawaudio 2026

A 10kHz 1Vpp signal input gives this result:



The pilot then:



And the L+R signal is just a 1kHz sine wave. All the signals are summed at the summing node of U7B, pin 6 (inverting input). This is a standard inverting summing amp. Pin 6 is at virtual ground and looks like a very low impedance point to all the signals, which means that the signals don't 'see' each other and there is no one signal interfering with the other.

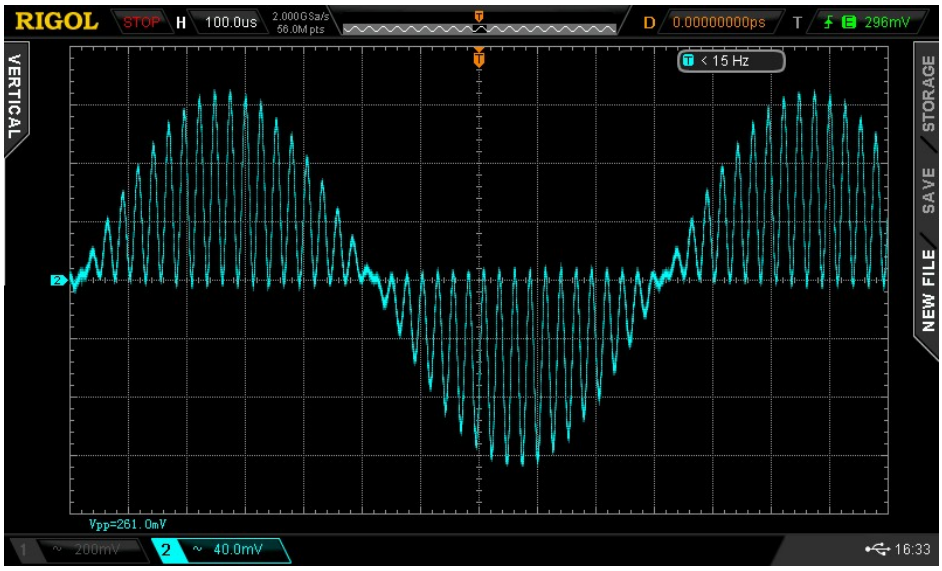
The feedback resistor is 10k. Any signal from a low-impedance output fed into the inverting input through an input resistor R_{in} will appear at the output inverted and weighed (multiplied) by a factor of $^{10k}/_{R_{in}}$.

In this circuit the L-R modulated signal is summed at unity gain (-1) through a 10k resistor. The pilot is summed at -1/10 through a 100k resistor. The level of the pilot at the pot wiper should be set to approximately to the same level as the total MPX output level, since that

© Jygsawaudio 2026

means that it constitutes 10% of the total, because of the 100k feed resistor.

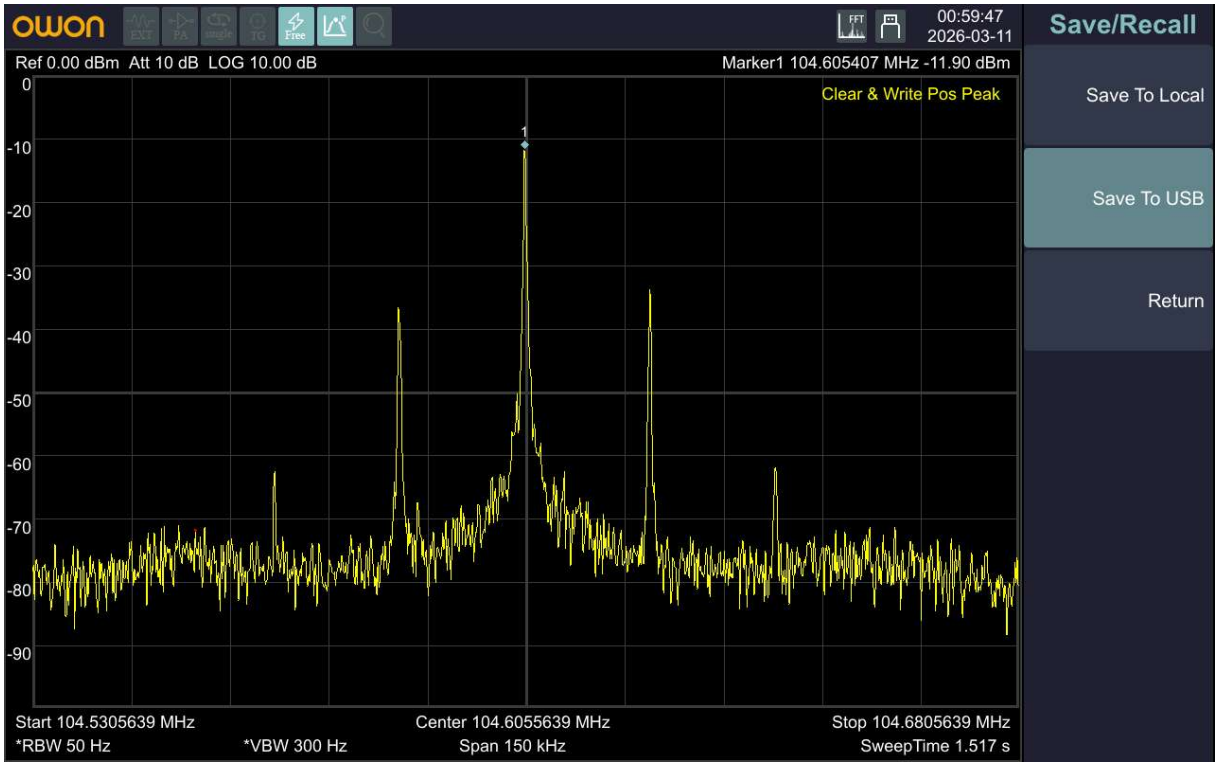
In order to set the L+R level, you can feed a 1kHz 1Vpp signal into the left or right input only, ground the other input, null the pilot, and check the output of the summer with a scope. Set the trimpot (RV5 in my circuit) so the resulting waveform looks like this:



This is an indication of a good balance between L-R and L+R. Another way to achieve this is to transmit, and monitor the receiver output. Adjust the trimpot for a null, or lowest possible signal, in the grounded input channel's output from the receiver. You can also calculate the stereo separation in dB from these results. I measured 840mVpp left versus 10mVpp right, giving a stereo separation of $20\log\left(\frac{840}{10}\right) = 38.5dB$, which is a good result for a homebrew device. Interestingly, lowering the pilot level gives better results. Increasing pilot level towards 250mV actually degrades the stereo separation, so I set it a fair bit lower, but still enough to avoid confusing the receiver.

© Jygsawaudio 2026

The output of the transmitter, when fed into the 50ohm input of a spectrum analyser, looks like this:

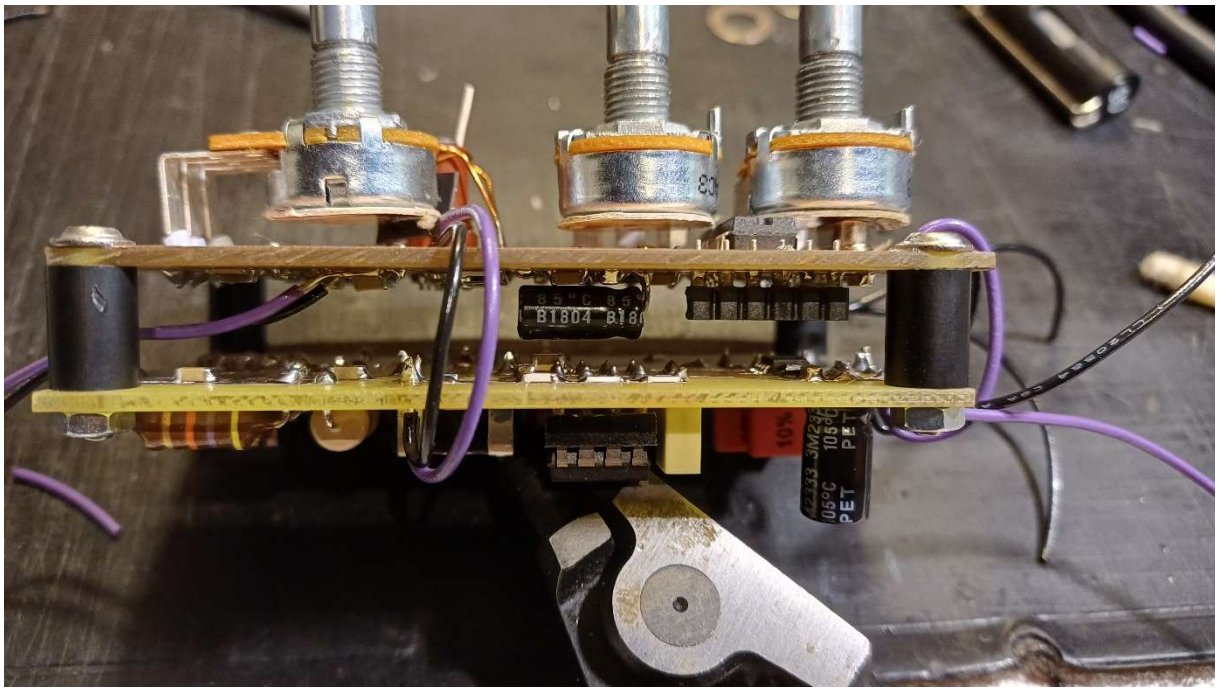


The transmitter is set to 104.6Mhz. The SA shows 104.605, which is pretty close. Output is -12dBm. The 19khz pilot can be seen in both sidebands. The 38khz is a harmonic of the 19khz that is a result of the frequency modulation process. It certainly doesn't come from the AD633, since I nulled the DC offset and can confirm this 38khz disappears into the noise floor when I turn off the pilot.

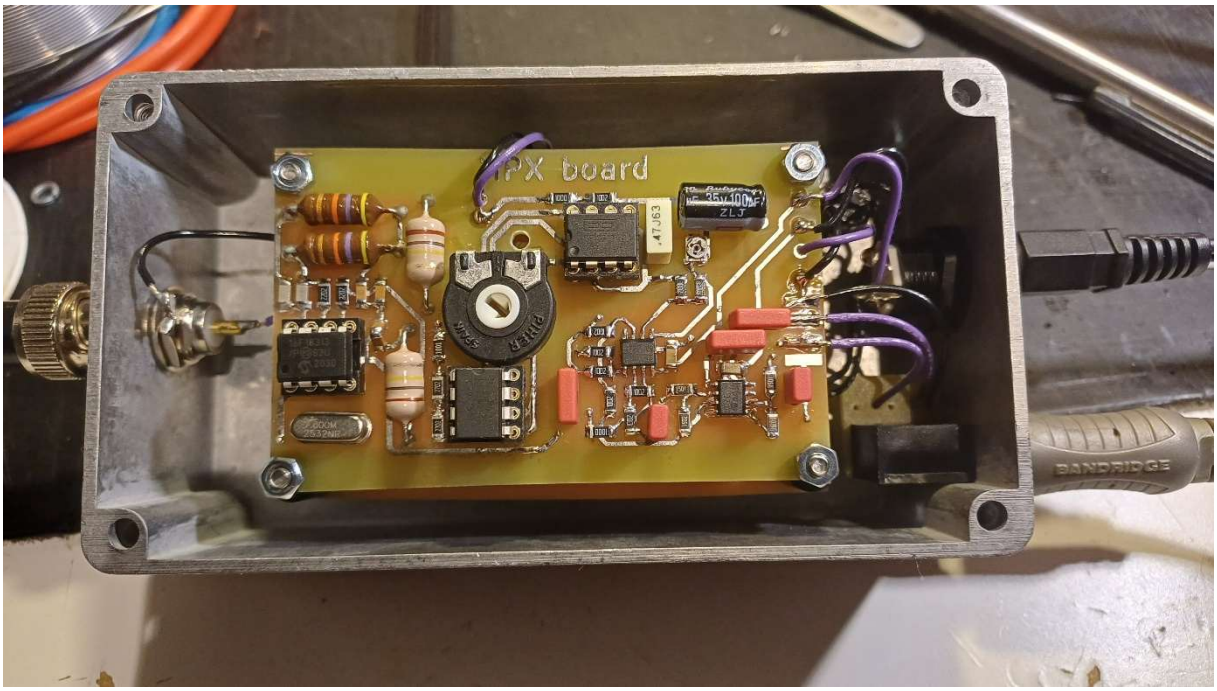
The 25kHz reference spurs from the ADF4116 are buried in the noise floor entirely, so the loop filter is doing its job well.

© Jygsawaudio 2026

This is a picture of the two boards (MPX and RF) stacked.



This is how everything looks, assembled in the enclosure:



© Jygsawaudio 2026