

Modern QRP Rigs and the Development of the QCX CW Transceiver Kit

Introduction

I present an attempt to describe some of the more modern techniques into QRP radio design. The majority of QRP radio receivers are based on the same superhet architecture that has dominated our homebrew, kit and even commercial projects for three decades, with some notable exceptions. More recent components and designs allow creation of transceivers which are simpler, higher performance and lower cost, all at the same time!

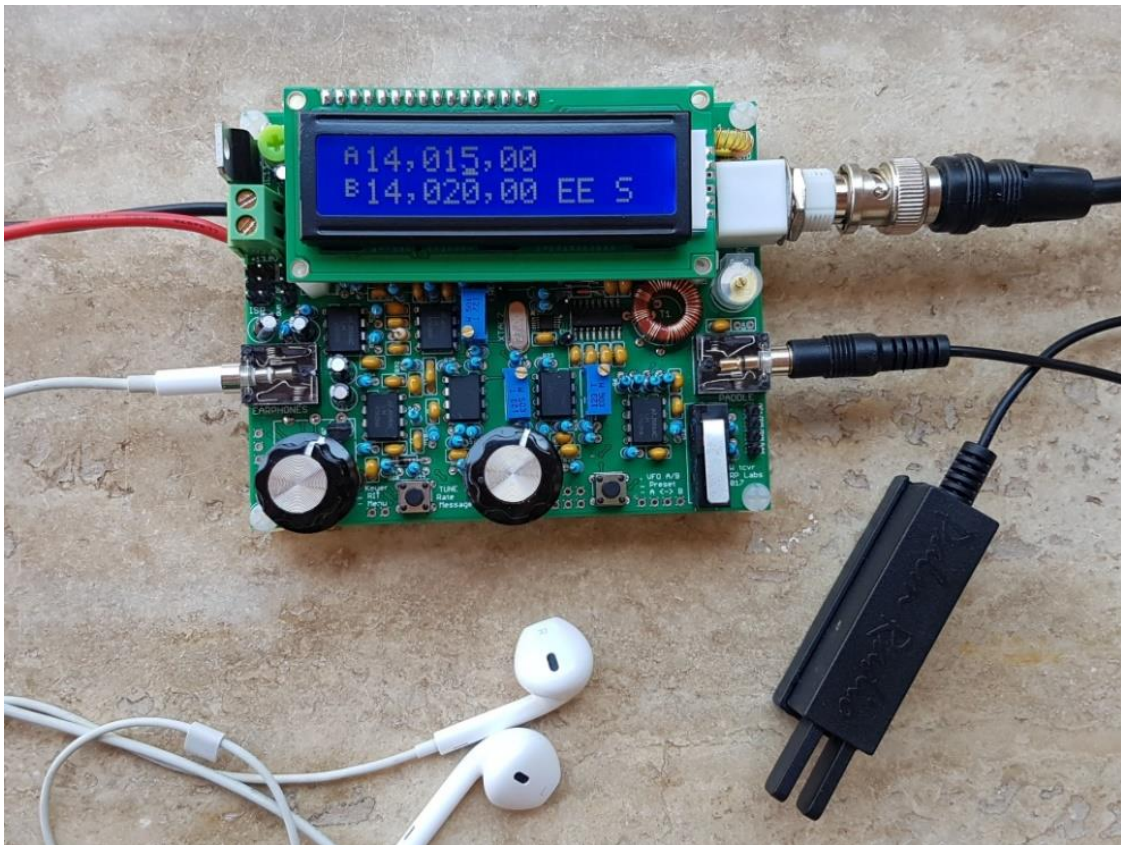
The three main topics presented here are:

- Escaping three decades of the classic SA602 superhet receiver
- Drift-free, flexible, accurate, easy and cheap: the new generation synthesizer chip
- Introducing microcontrollers to your projects

These topics and various other smaller features that may be new to you, are presented in the context of the QCX 5W CW transceiver kit designed for QRP Labs in 2017, which is a good example of how to design a practical, low cost, high performance radio using these techniques.

We'll stick with CW, and won't go as far as Digital Signal Processing, Software Defined Radio, etc. The concepts presented here are modern and high performance and yet won't be too daunting for the majority of us!

The QCX 5W CW transceiver kit



Let us start with a brief description of the QCX 5W CW transceiver kit, so that we know where the following discussion is leading to.

This 5W CW transceiver kit, named “QCX” (for QRP Labs CW Transceiver), was designed for the 2017 Youth On The Air (YOTA) summer camp “buildathon” hosted in UK in August 2017 by the RSGB. It is a mono-band CW transceiver available as a kit for bands 80, 60, 40, 30, 20 or 17m. The YOTA attendees built the 17m version (after all, the year was 2017... why miss an opportunity like that!).

The design process was very interesting, due to the constraints. The kit needed to be simple enough to build in a relatively short time, by a group of young people with potentially a wide range of construction experience... some with none at all. The cost had to be low to match the limited budget available. At the same time, I was not interested in designing a toy radio. As far as I was concerned, it had to be a really high-performance radio packed with useful features and a useful 3 to 5W output power. A rig that would also be of interest to radio amateurs all around the world. I wanted to create something really special for each young attendee of YOTA 2017 to take home with them and operate, something that would set a new standard in amateur radio kits.

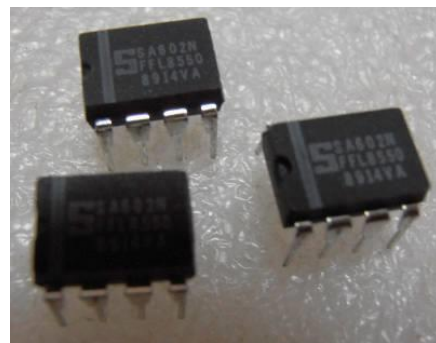
The result is a low-cost \$49 kit using all through-hole parts (except two SMD ICs which are pre-soldered to the PCB). It has very high performance, which competes well with top-of-the range commercial radio costing 100x as much. The kit also contains all its own test equipment for alignment and debugging (DVM, power meter, signal generator, RF power meter), so it can literally be built without any additional test equipment.

It's a lot of performance and value for \$49 and has been greatly appreciated by the QRP community. At time of writing (March 2018) over 4,200 of these kits have been shipped.

The traditional SA602 superhet

First a bit of history. The SA602 is a famous IC among QRPers, introduced by Signetics around 1985. The chip is known equivalently as NE602, NE612 and SA612, all of which are essentially the same device rebranded when Signetics was acquired by Philips (now NXP Semiconductors). It's hard to go back that far and discover the exact release date but Google found me an old 1985 Signetics datasheet listing the NE602.

It's a remarkable little 8-pin chip that contains an oscillator and a Gilbert Cell mixer. It has low noise, around 17 or 18dB of gain and will operate up to 200MHz and above. The third-order intercept (IP3) of -13dBm is much less spectacular, to say the least. The chip was intended for cellular radio applications, as the 45MHz IF and demodulator. It is available in both a Surface Mount Device (SMD) package and a conventional 0.1-inch pitch 8-pin through-hole package (DIP).



It wasn't long before the chip became known among radio amateurs. The first reference, at least as far as I could find, to what became the classic SA602 line-up, is Rick K1BQT's design in Ham Radio June 1987. Two SA602 chips operate as both the mixers of a superhet, and also both the oscillators: Variable Frequency Oscillator (VFO) and Beat Frequency Oscillator (BFO). So easy to use, easily available, and convenient: hardly surprising therefore that within a couple of years the SA602 was firmly rooted in amateur homebrew culture.

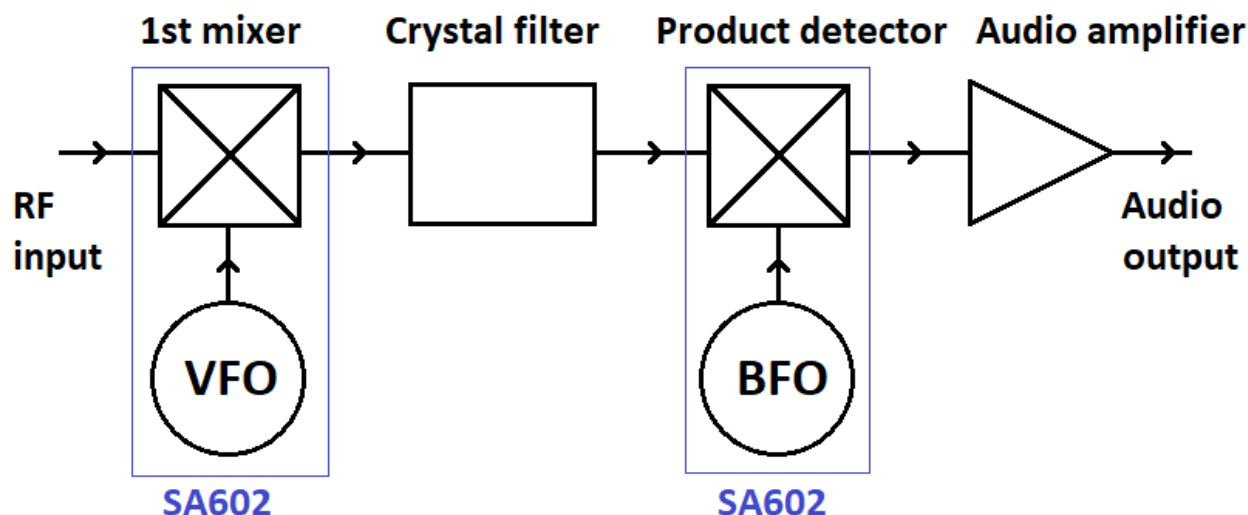
The typical receiver design consists of:

- SA602 as the VFO and first mixer
- Quartz crystal intermediate frequency (IF) ladder filter
- SA602 a BFO and product detector
- Audio amplifier IC

More often than not the audio amplifier is an LM386 IC, which also dates back as many decades.

The use of SA602 ICs in a superhet like this produces a receiver which is convenient and easy to design and build.

An astonishing number of QRP radios use this chipset for their receiver sections, in homebrew, kit and even commercial transceivers still designed and produced today, over three decades later!



So what's so wrong with the SA602-based superhet anyway?

Use of SA602s in this way in a superhet receiver produces a compact and convenient design, but it is not without its shortcomings.

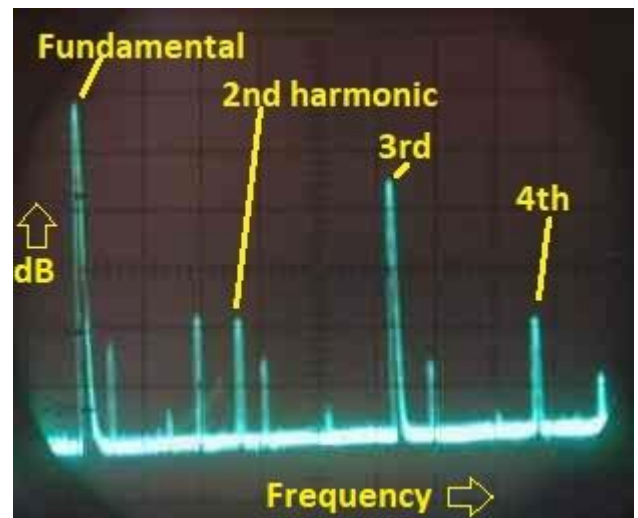
The first and foremost problem is a consequence of using the SA602 in ways for which it was not intended. It's a low power Gilbert Cell mixer and has a relatively poor IP3 of -13dBm (according to the datasheet), and a low dynamic range. This makes it highly vulnerable to strong signal overload.

There's an enormous difference in power arriving at your receiver, between the weak QRP station on the other side of the planet, and someone across town transmitting with his kilowatt on a nearby frequency. Or, for that matter, the local shortwave broadcast station with 10's of kW or more! When a receiver front end has a low third-order intercept, all of these strong signals can cross-modulate with each other due to the poor linearity of the mixer. The result is interfering signals, and/or a raised noise floor.

Nowadays the proliferation of household consumer electronic devices built to poor standards means many of us suffer high levels of interference; all of these signals are also prone to being added to the mix too.

The first time I learned this harsh reality about dynamic range, the importance of IP3, and what intermodulation is, was a 0-100MHz spectrum analyzer I designed in around 2003, see <http://hanssummers.com/spectrumanalyser.html> .

The oscillator swept 145-245MHz and the mixer converted the incoming signals to a first intermediate frequency (IF) of 145MHz. Yes, that oscillator/mixer was an SA602 and subject to all of the problems mentioned. The second mixer/oscillator was an SA602 as well. If the input signal was too strong, harmonics would all mix up with each other and produce a bunch of signals that aren't really there!



In this photograph there's a signal generator set to 20MHz. You can see its 2nd, 3rd and 4th harmonics annotated. All the other signals shown are ghosts, intermodulation products caused by the SA602 mixer overload!

You can identify the ghosts easily as you tune the spectrum analyzer's center frequency, because the real signals move across the screen slowly, whereas the unwanted mixing products move at 2x, 3x, 4x etc. the speed. Who wants to be able to identify ghosts? Wouldn't it be better to not cause them in the first place?

If this is what a single signal at the input does, imagine the chaos that results when you have thousands of signals hitting your poor SA602 all simultaneously!

A band-pass filter up front will cut out some of the out-of-band signals preventing them from reaching the mixer, and is ALWAYS a good idea – but, band pass filters are relatively wide and still let a huge number of stations through to hit the mixer.

There isn't a solution to this problem, except to change the mixer.

Some other less serious disadvantages apply to superhets, which are not related to the SA602 chip per se.

Superhets generally can often have a somewhat mushy sound, this is due to the variable group delay through the crystal ladder filter. Unless very carefully designed, crystal filters just don't always sound nice. Crystal filters (good ones, anyway) can get very expensive. Low cost crystals can also limit the intermodulation performance of the receiver.

Alignment is another issue that can be daunting for a beginner constructor. The BFO needs to be adjusted so that the shape of the crystal filter response is placed correctly with respect to the desired sideband being received. It's not trivial to get the desired bandwidth, and the BFO frequency at just the right place to have the filter passband in the right place.

If you want to use a frequency counter to display the frequency, then you need a frequency counter that can apply the IF offset.

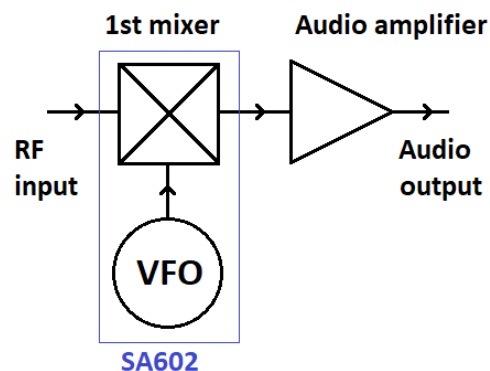
Finally, a superhet is prone to "birdies". Harmonics of the BFO can mix with harmonics of the VFO to create strong heterodyne "birdies" at certain frequencies. To avoid this one needs to make a careful choice of IF so that none of the harmonics of the BFO, will mix with any of the conceivable harmonics of the VFO across the tuning range of interest, to produce any mixing product that falls within the band and will be heard in the receiver audio. This gets harder and harder to do, if you add multiple bands to the radio! Shielding the modules of the radios into separate metal compartments is one solution but this introduces a much higher degree of mechanical complexity (and hence cost).

Direct Conversion

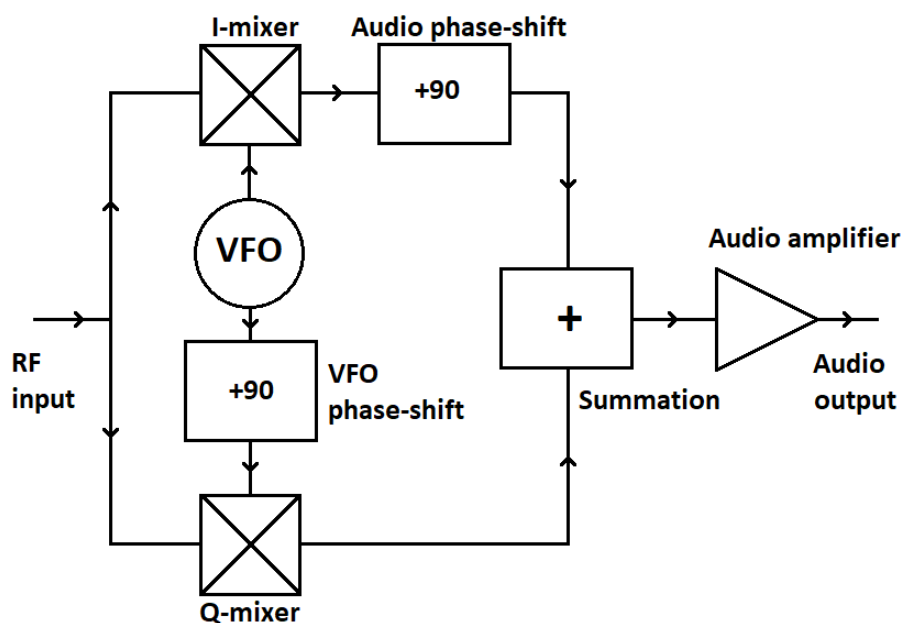
Direct conversion utilizes only a single mixer, straight from RF to baseband (audio). It eliminates a lot of the problems with superhets, such as alignment, the birdies, and the crystal filter. Many people are surprised at the nice clean sound of a direct conversion receiver, compared to the superhets they are used to.

The main and very serious disadvantage of a simple direct conversion receiver, is that the upper and lower sidebands are both received together at the same time.

The unwanted sideband can be eliminated by some clever mathematics, splitting the signal into two paths. Either the incoming signal or the oscillator paths must have a 90-degree phase relationship. It makes no difference whether you apply the 90-degree phase shift to the RF signal, or the VFO; but the VFO is normally easiest. A 90-degree phase shift applied at the audio output produces two outputs that when summed, magically reinforce the wanted sideband and cancel the unwanted one.



This scheme resolves the main drawback of direct conversion receivers by cancelling the unwanted sideband, leaving a clean SSB receiver. The system can also be operated in reverse to produce an SSB exciter for a transmitter. The complexities involved with matching the amplitude of the two paths and generating accurate phase shifts at RF and audio frequencies, are non-trivial.



There are several variations of implementing the audio phase shift, including use of operational amplifier all-pass networks, or generating the phase shift with a passive resistor-capacitor network known as a polyphase network.

This “phasing method” of direct conversion is not new, examples can be found dating back to the tube era. More recent components have made the design challenges a lot easier to deal with. With careful design a “phasing method” direct conversion receiver can have superb performance and avoid many drawbacks associated with superheterodyne architecture receivers. A good example of an early phasing receiver design is the R2 receiver by Rick KK7B, published in QST January 1993 and several subsequent editions of the ARRL handbook.

Other problems of direct conversion receivers that are often cited include microphonics and sensitivity to mains hum, since so much (or all) of the gain of the receiver is implemented at audio frequencies. It is also harder to apply good Automatic Gain Control (AGC) at audio, than at IF-derived (and applied) AGC.

Overall the direct conversion receiver has many benefits and is so well-suited to today’s available components, that over the last few decades the majority of radio receiver applications now use this technique, including Software Defined Radios (SDR), UHF and microwave receivers.

The Quadrature Sampling Detector (or “Tayloe detector”)

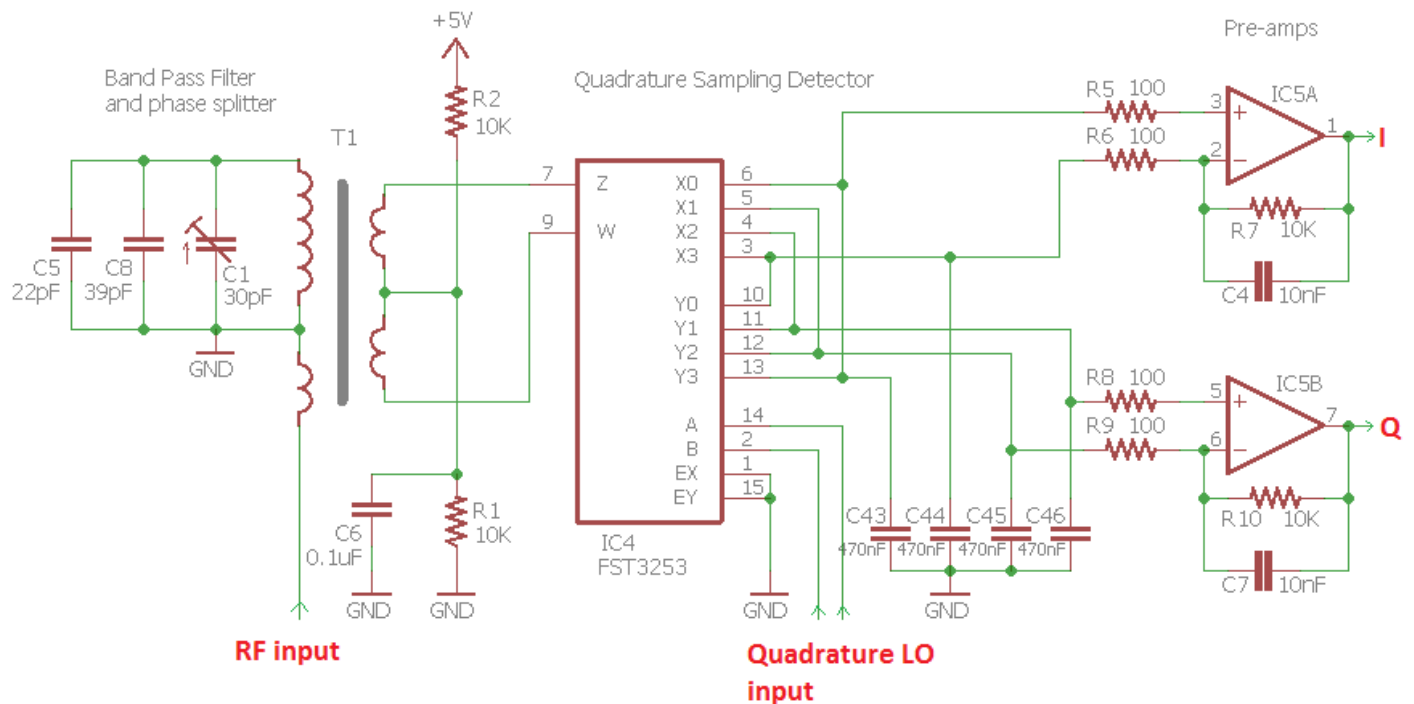
MOSFET switches have been used to create very high performance, “super linear” mixers for as long as the SA602 has been around.

But it was Dan Tayloe N7VE who really popularised the Quadrature Sampling Detector (QSD), also often known simply as “Tayloe detector”. I first became aware of this development in February 1999 when Pat Hawker G3VA reported it in his monthly RSGB RadCom journal column “Tech Topics”. Dan’s development uses inexpensive bus-switch ICs intended for computer applications, as a multiplexing detector. Most commonly the dual 1:4 bus switch type FST3253 is seen. These bus switches are not designed for RF use, but they have some very useful performance characteristics, including very low “ON” resistance (around 4-ohms), very fast switching speed, and excellent match between the switches.

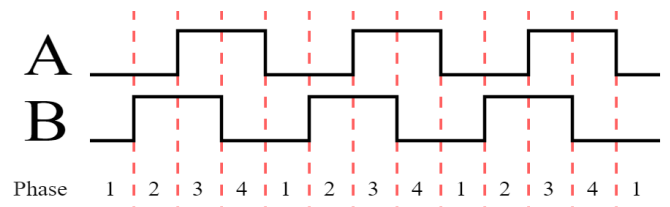
The QSD has an RF input and a LO input. The output is baseband (audio). This mixer (or detector) has very low loss, under 1dB – therefore on HF no RF amplifier is required ahead of the

detector. This further simplifies the receiver design and there is no need for an expensive highly linear, high dynamic range RF amplifier. The QSD has very high dynamic range and very high third order intercept (IP3). An additional benefit: the FST3253 is a very common and cheap chip, much cheaper and commoner than an SA602!

The schematic section below is from the QCX transceiver. This is a double-balanced quadrature sampling detector configuration. As such, the RF input to the two switches (labelled Z and W in the FST3253 in the diagram) is 180-degrees out of phase. This is simply generated by two windings on the transformer T1. The “center tap” creates a DC bias of half the supply voltage, by a simple potential divider.



The 1:4 multiplexer switch requires a quadrature local oscillator input from the VFO. That is two say, two drive signals at the same frequency (the reception frequency of the radio), but having a 90-degree phase relationship to each other. We'll discuss generation of such LO signals later.



The action of the 1:4 multiplexer is such that it spends ¼ of each cycle of the VFO routing the incoming RF to one of four same-value integrating (or “sampling”) capacitors. These are labelled C44-46 in the schematic. The voltage accumulated across each integrating capacitor is the audio frequency mixing product of the incoming wanted signal and the local oscillator drive signal. Crucially, the four outputs have different phase relationships to each other, which can be conveniently labelled 0, 90, 180 and 270 degrees.

The capacitors give a first order roll-off of the response with frequency, this behaves as a narrow band-pass filter further improving the performance by preventing large nearby signals reaching later stages of the receiver.

The QSD must be followed by high-performance audio pre-amps. The input noise characteristic of these op-amps determines the overall sensitivity of the receiver. In the schematic above, these are IC5A and IC5B. I used a dual op-amp type LM4562 in the QCX transceiver. This is a high-performance op-amp by Texas Instruments, and quite expensive. It has a low input noise of only 2.7nV/sqrt(Hz) and total harmonic distortion of 0.00003% (among other impressive

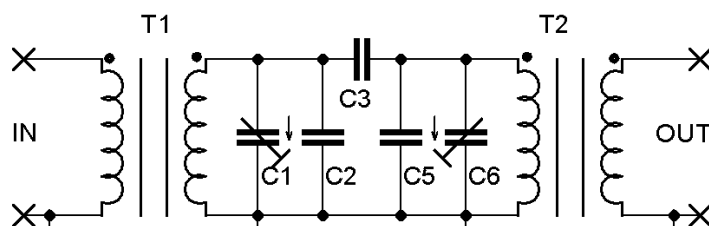
characteristics). The input noise characteristic is not the lowest available in an op-amp, but prices skyrocket very quickly so you'd need deep pockets to go much further.

One op-amp (IC5A) subtracts the 0 and 180-degree QSD outputs to produce what is conventionally known as the "I" output; the other op-amp (IC5B) subtracts the 90 and 270-degree QSD outputs to produce the "Q" output. This is essentially the front end of a simple "Software Defined Radio", in which the "I" and "Q" signals are fed to a PC with a stereo sound card, and software applies the necessary 90-degree audio shift to demodulate single sideband audio (cancel the unwanted sideband).

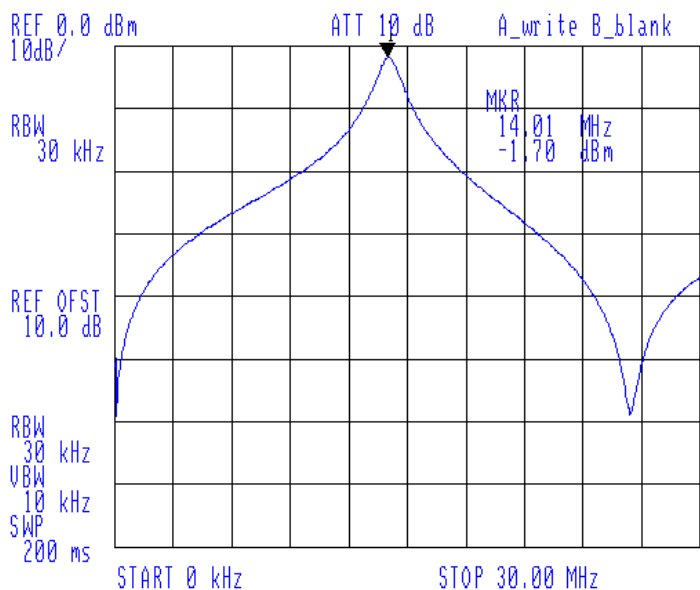
Whether you cancel the upper or lower sidebands can be arranged by swapping the inputs to one of the operational amplifier difference circuits e.g. IC5B. Or, more conveniently, simply by changing the phase relationship of the local oscillator signals! As we shall see later, the latter is very conveniently done in the QCX since it requires no hardware switching.

A final point to note on this front end, is the band pass filter. Conventionally I might have used a double-tuned circuit band pass filter, which would have required two trimmer capacitors and two toroidal transformers, AND then a trifilar phase splitter to feed 180-degree out of phase RF to the double-balanced QSD. That would have made three toroids and two trimmer capacitors. Instead, I came up with this T1 transformer arrangement shown to the left of the previous schematic. The three windings of what would have been the trifilar phase splitter transformer, are now separate small (and equal) windings on the toroid. There is then a large winding which is paralleled with a trimmer capacitor, to create an adjustable band pass filter.

A double-tuned circuit band pass filter



Since this is only a single tuned circuit, the bandpass filter characteristic is not as sharp as a double-tuned circuit would have been. Considering the high linearity of the Quadrature Sampling Detector, and the construction simplicity of the single toroidal transformer (and smaller PCB area, lower cost etc), it was felt to be an acceptable performance trade-off for this kit. The spectrum analyzer trace below shows the 20m version. Nevertheless, the T1 transformer is widely agreed by QCX constructors, to be the BEAR of the QCX assembly. I always say take it slow, don't rush it, and above all read the detailed step-by-step instructions in the manual!



The oscillator problem

The oscillator is often another problem in a radio design. For me, it always used to be the BIGGEST problem. In a superhet receiver, the BFO will typically be a simple crystal oscillator. You'll use one of the same crystal frequency used for the ladder filter and try to pull it to the right frequency to place the filter passband where you want it. The trouble is the LO. Sooner or later, unless you want to remain rockbound to the frequency of a particular crystal, or the narrow frequency range that you can pull it over – you will have to tackle the problem of variable frequency oscillators (VFO).

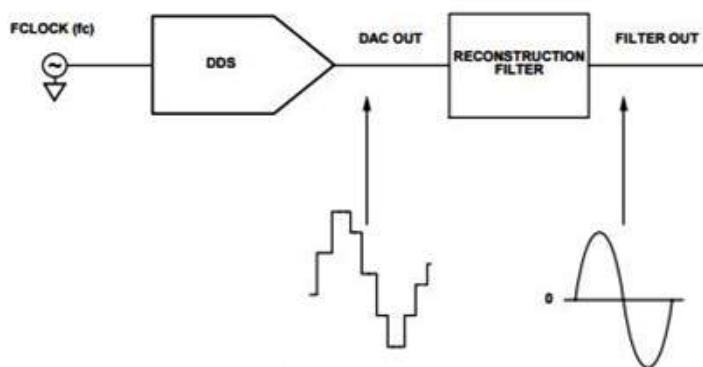
For the VFO, you would probably historically use a capacitor-inductor tank circuit. Most likely the capacitor is variable, though variable inductors are also a possibility. In either case, most people will struggle to achieve the frequency stability they desire. The capacitor and the inductor values change with temperature. To an extent, you can use the temperature coefficient properties of different capacitor materials (ceramic, polyester, silver-mica) to try to compensate for the temperature coefficients of the variable capacitor and the inductor. However, the active devices used will also have temperature coefficients, and particularly in the case of semiconductors, these are such non-linear dependencies that they are almost impossible to cancel out. Building a VFO whose frequency doesn't drift around during the course of your QSO, is often one of the biggest challenges of building your own radio!

Direct Digital Synthesis

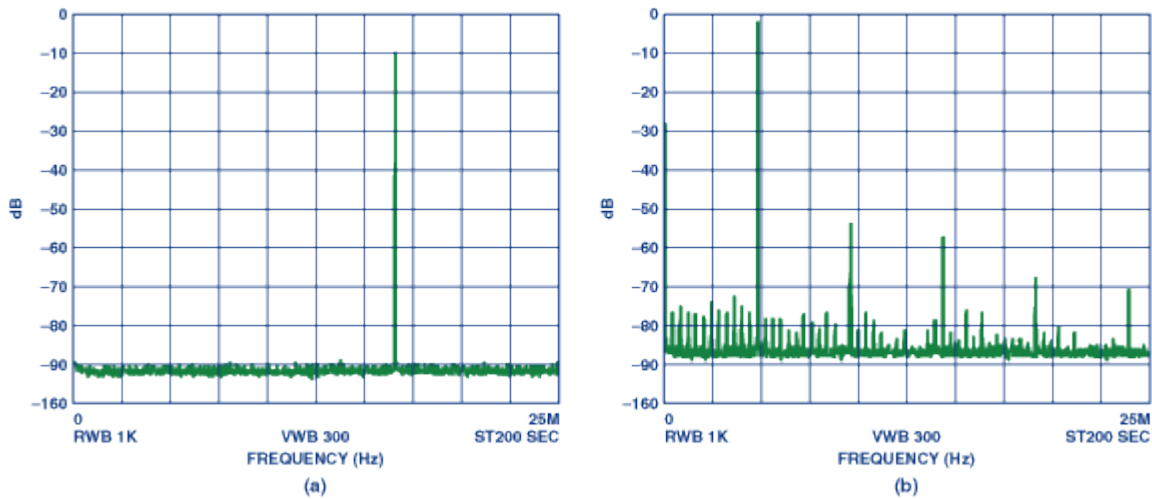
Direct Digital Synthesis (DDS) chips (market dominated by semiconductor manufacturer Analog Devices) provided a wonderful step forward.

The DDS chip requires a reference oscillator input, typically a crystal oscillator. A phase accumulator provides a digital representation of the desired sinewave at the target output frequency, updated at each cycle of the reference oscillator. This is converted to an analogue form by a Digital to Analogue Converter (DAC).

The output of a DDS is therefore a “staircase” which approximates the desired sinewave. A “reconstruction filter”, which is really just a normal low pass filter, always follows the DDS chip to smooth out the staircase and leave a precise sinewave. The DDS tuning word is normally programmed using a microcontroller. The frequency stability is that of the crystal reference oscillator you provide.



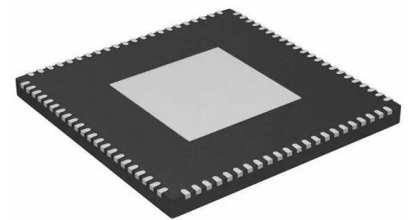
DDS chips have one prominent disadvantage: spurious output frequencies, which depend on the relationship between the desired output frequency and the reference clock input. These spurs cannot be removed by filtering. In a receiver they mix with unwanted signals and produce “birdies” in the receiver output.



Here for example, is Analog Devices own measurement of the AD9834 DDS chip, using a 50MHz reference clock. The left trace has the output frequency set to 16.6667MHz i.e. exactly 1/3rd the reference clock. The right trace is at 4.8MHz, which does not have a convenient integer numeric relationship with the reference clock frequency. Note the large number of spurious outputs. The spuria are the consequence of the staircase not repeating exactly from cycle to cycle, because the number of steps does not fit exactly into the period of the target output frequency.

It is apparent that the problem will be reduced if the reference frequency is a much higher multiple of the output frequency; and the problem will also be reduced by using a higher resolution Digital to Analogue Converter in the DDS. Hence a great deal of effort over the last two decades has resulted in DDS chips with higher and higher reference clocks, and DAC resolution.

For example, the AD9914 clock is up to 3.5GHz and the DAC has 12-bit resolution; or the AD9910 with 1GHz clock and 14-bit DAC. State of the art DDS is not cheap: the AD9914 and AD9910 chips cost \$199 and \$52 at Digikey respectively; so, you won't want to make a mistake soldering all the pins of that fiddly SMD package. It isn't cheap or easy to design the microwave reference clock, either!



It used to be possible to get free samples of some devices from AD, I am not sure if this is still possible; in any case that doesn't help kit producers or commercial manufacturers.

Chinese-manufactured modules using the AD9850 chip are available from eBay, AliExpress, BangGood etc. The AD9850 has a 125MHz reference clock and 10-bit DAC. It's not a bad compromise between price and performance. The module has a 20-pin DIP format, and the 125MHz reference oscillator already assembled. It therefore gets around the problems of designing the reference clock and soldering the tiny SMD DDS chip. In 2014 these modules cost under \$4 including shipping, but the price started rising and is now up at around \$14.



The SiLabs Si570 Digital Phase Locked Loop

The SiLabs Si570 chip is a programmable clock generator covering 10MHz to 1.4GHz (depending on the device family member). A complete synthesiser, including the reference oscillator) in a 5 x 7mm package. Phase Locked Loop (PLL) synthesisers swap spurious outputs (such as on a DDS) for phase noise (jitter); but these devices are very well designed and jitter is extremely low.



Programming the configuration of this chip takes place via a standard I2C serial interface, connected to a microcontroller. The configuration is a little complex (more than a DDS), but there are plenty of smart people on the internet who have already figured it all out and published their source code to follow.

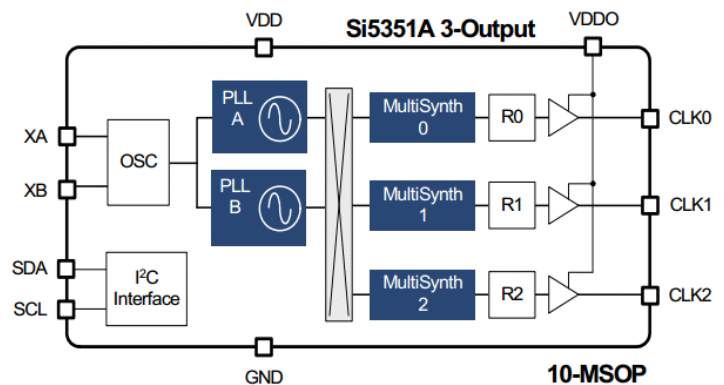
An important difference compared to DDS is that the Si570 output is a squarewave, not a sinewave. Many people's first reaction is concern, but in fact most mixer types work better with a squarewave LO drive switching them. Certainly, a digital bus switch like the FST3253 in a QSD (see previous section) requires a squarewave digital drive. But even traditional diode ring mixers prefer a squarewave drive.

Unfortunately, this chip isn't cheap either, Digikey currently lists the cheapest version (10-160MHz) at \$11.50. Nevertheless, it has been very popular with homebrewers.

The SiLabs Si5351A Digital Phase Locked Loop

Around 4 years ago, SiLabs released the Si5351A digital phase locked loop (DPLL): this remarkable little chip is a very low-cost device: Digikey currently lists it at \$0.92.

The chip contains two DPLL blocks and three output synthesiser blocks; it can therefore produce three independent outputs. The output frequency range is 3.5kHz to 200MHz. Unlike the Si570, the Si5351A does not include its own reference oscillator; but all that is necessary is to connect a 25-27MHz crystal. Like the Si570, it is programmed via I2C using a microcontroller.



The three outputs can be used for multiple purposes in a transceiver; if you wish to design a superhet for example, you could use one output for the LO and one for the BFO.

The phase noise (jitter) performance of the Si5351A is not as good as the Si570 but is still plenty adequate for use in a transceiver. In fact, Elecraft use the Si5351A as the synthesiser in their famous and highly regarded KX2 and KX3 transceivers.

The Si5351A is a tiny 3 x 3mm device (not much more than ¼ by ¼ inches) with 10 pins. It is not very pleasant to solder! Luckily, several modules are available on which the manufacturers have already soldered the Si5351A chip. I know of three examples though there may be more. All three have an onboard 3.3V regulator, and I2C level converters so that the modules may be used in 3.3V or 5V systems.

QRP Labs' Si5351A Synth kit costs \$7.75 and is the lowest-priced of these modules. It is a kit of through-hole parts (except the Si5351A which is pre-soldered). The board has pads for SMA connectors, but is also supplied with two 10-pin headers, and a pinout which largely matches the Chinese AD9850 DDS module described above, so it can in some circumstances be used as a plug-in replacement requiring only firmware changes.

Information and purchase (\$7.75):

<https://qrp-labs.com/synth.html>

Adafruit's Si5351A module is ready-assembled using all SMD components and costs \$7.95.

Information and purchase (\$7.95):

<https://www.adafruit.com/product/2045>

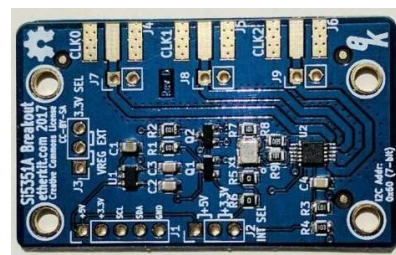
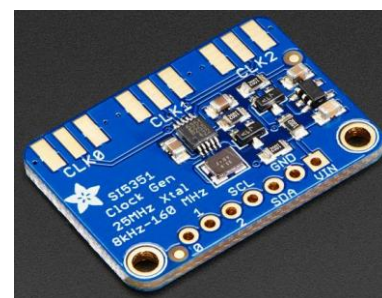
Etherkit's Si5351A module is also ready-assembled using all SMD components and costs \$10.

Information and purchase (\$10):

<https://www.etherkit.com/rf-modules/si5351a-breakout-board.html>

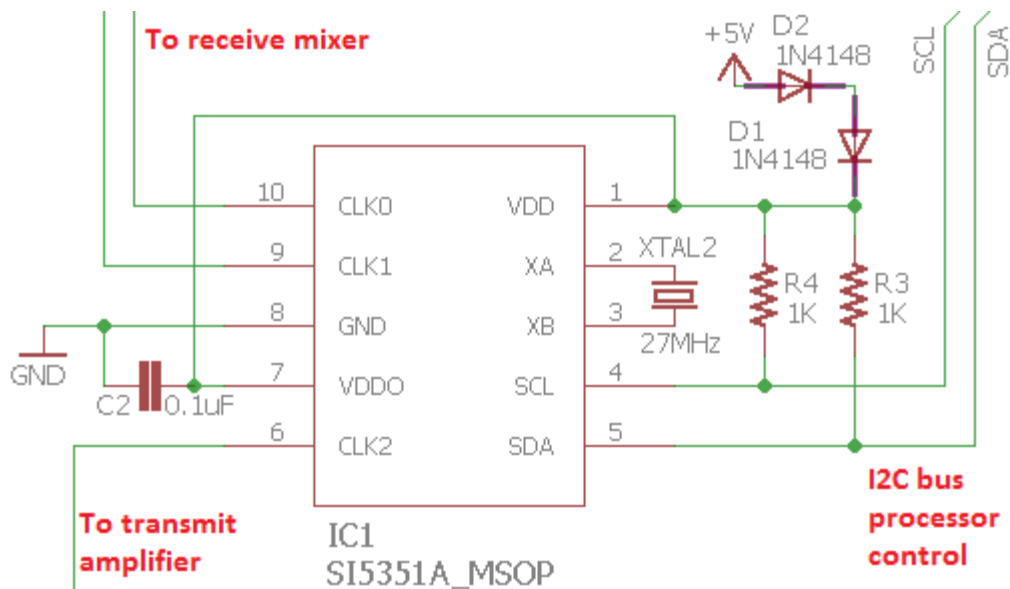
All three manufacturers also provide sample code. QRP Labs' samples cover AVR, PIC and Arduino platforms (see later).

The SiLabs Si5351A is also available in larger (more expensive) chip packages providing 8 outputs instead of three; and as the Si5351B variant (with internal voltage-controlled crystal oscillator) and Si5351C (synchronises to an external wide-range reference clock).



Using the Si5351A synth in the QCX 5W CW transceiver

This is the schematic section relating to the Si5351A in the QCX transceiver. The Si5351A was the natural choice due to its low cost and high performance. The Si5351A requires a 3.3V supply voltage and this was obtained, for the sake of low cost, using two 1N4148 diodes in series! Each provides a voltage drop of about 0.6V or 0.7V and the result is suitable for powering the Si5351A.

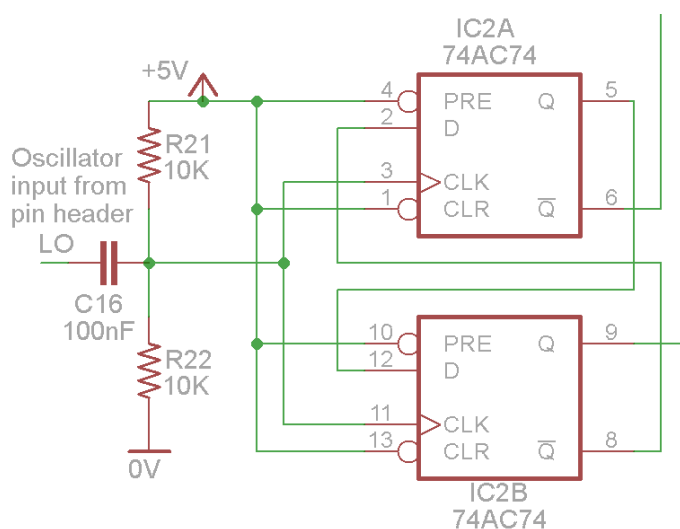


As mentioned previously, the Si5351A has three independent outputs. In the QCX CW transceiver, two of these are in quadrature, and drive the Quadrature Sampling Detector (receive mixer) directly. The third drives the transmit power amplifier and is also used as a signal generator when completing the receiver adjustments.

Generating a quadrature LO using the Si5351A

Referring to the previous section describing the Quadrature Sampling Detector (a.k.a. “Taylor Detector”) you will have noted that the QSD requires a quadrature LO at the reception frequency. That is to say, a two-phase LO drive: two signals at the desired reception frequency having a 90-degree phase offset to each other.

Conventionally generation of this quadrature LO has normally been done using a dual flip-flop divide-by-4 circuit and a single-phase LO at four times the reception frequency. This example (see right) is from the QRP Labs Receiver module <http://qrp-labs.com/receiver> (a different kit to the QCX).

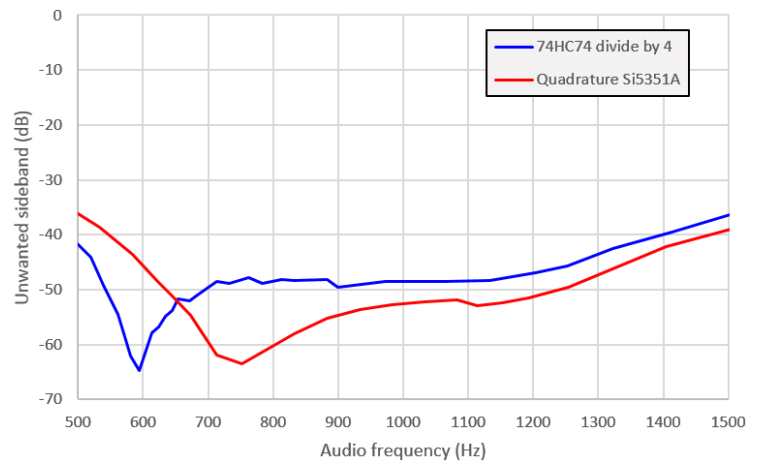


Elecraft’s KX2, KX3 using the Si5351A, Softrock, mcHF, etc transceivers (using the Si570), and transceivers using DDS chips, all use the divide-by-4 method. The disadvantages are:

- Uses one more chip: more board area and more components means higher cost
- Requires a 4x LO
- Limited by the maximum frequency of the divide-by-4 chip

Wouldn't it would be nice if we could use two of the Si5351A outputs and configure them with a 90-degree phase offset? In fact, we CAN... and this is exactly what is done in the QCX transceiver kit!

I also found by measurement of the unwanted sideband cancellation, that the direct Si5351A quadrature LO version produces a better performance than an 74AC74 divide-by-4 (the graph incorrectly labels it "74HC74"). I speculate that this is due to the fact that the Si5351A outputs are designed to operate up to 200MHz, significantly faster than the 74AC74's maximum 120MHz; therefore, the edge transitions of the Si5351A are faster resulting in better symmetry of the quadrature LO switching waveform.



To the best of my knowledge, the QCX is the first radio design to use the Si5351A to produce the quadrature LO directly at the reception frequency, with no divide-by-4 chip and quadruple-frequency oscillator. I was the first to understand how to generate a glitch-free, accurate 90-degree quadrature LO using the Si5351A, and implement it in a radio design. It is now also available in the QRP Labs VFO kit using the Si5351A Synth see <http://grp-labs.com/vfo>

The following description explains how to configure the Si5351A as a wide-band quadrature synthesiser. As far as I know, this is the first documented description of the procedure.

The SiLabs documentation is not at all clear. SiLabs App Note AN619 "Manually generating an Si5351 Register Map" is useful even if it is often vague and contradictory, see <https://www.silabs.com/documents/public/application-notes/AN619.pdf> .

The configuration of the Si5351A is performed by writing values from a microcontroller, via the I2C bus, to a large number of internal 8-bit registers. I won't describe the entire configuration method here. A simplified summary is that there is an internal Voltage Controlled Oscillator (VCO), which should be configured in the range 600-900MHz. The multiplication (by the PLL) up from the 27MHz reference is configured via a fractional multiplier value (an integer plus 20-bit numerator and denominator). The VCO is then divided down by the "MultiSynth" division blocks, similarly also by a value which can be fractional. There is a final division stage which may be configured to divide by a power of two up to 128. This is used when you want an output frequency below 1MHz.

The Si5351A datasheet recommends for best low jitter (phase noise) performance, to use an even integer for the MultiSynth division block. This recommendation is implemented in all the QRP Labs products that use the Si5351A and in the QRP Labs Si5351A published source code examples.

The accuracy of the phase offset (LO and audio phase shift) in a phasing method direct conversion receiver limits the unwanted sideband cancellation that can be achieved. For example, a 1-degree phase shift error limits the unwanted sideband cancellation to no more than 40dB. In practice since there are phase shift errors on both LO and the audio phase shift, as well as amplitude imbalance and imperfect frequency response, the level of unwanted sideband cancellation is impaired by all of these factors. Clearly accurate phase shift is critical to the achieved performance.

Of particular interest are a group of Si5351A registers that define the phase offsets of the Si5351A outputs. The datasheet unhelpfully specifies an Output Phase Offset parameter "PSTEP" of

typically 333 picoseconds/step (with no minimum or maximum specified). This datasheet parameter specification misled many designers to interpret this to mean that the phase offset could be specified only in units of 333 picoseconds. This rather coarse granularity would severely limit the accuracy of the 90-degree phase difference between Si5351A outputs. Fortunately, this is not the way it works.

The real operation of these phase offset registers is found in section 6 on page 10 of SiLabs App Note AN619:

“Outputs 0-5 of the Si5351 can be programmed with an independent initial phase offset. The phase offset parameter is an unsigned integer where each LSB represents a phase difference of a quarter of the VCO period, $T_{VCO}/4$. Use the equation below to determine the register value.”

$$\text{CLKx_PHOFF}[4:0] = \text{Round}(\text{DesiredOffset}_{(\text{sec})} \times 4 \times F_{\text{VCO}})$$

Well, the equation is not very easy to understand either, as well as being WRONG... to start with, the [4:0] part specifies that the phase offset is a 5-bit number whereas in reality it is 7-bits.

The description of Register 165 (“Clk0 Initial Phase Offset”) on page 57 of AN619 is more helpful and tells us all we really needed to know:

“CLK0_PHOFF[6:0] is an unsigned integer with one LSB equivalent to a time delay of $T_{\text{vco}}/4$, where T_{vco} is the period of the VCO/PLL associated with this output.”

Similarly, there is Register 166, Register 167 for the phase offsets associated with the Clk1 and Clk2 outputs.

So here we can see that phase offset is a 7-bit value. And very simply, it is the number of quarter-cycles of the internal VCO that you want to apply as phase-offset to the desired output. No more, no less.

This example is typical of how difficult it is to make headway with the Si5351A using the SiLabs documentation. Two documents (datasheet, App note), and three different mentions of the phase offset; the first is ambiguous, the second is wrong and disagrees with the third, later in the same document. But the good news is that via head-scratching for some days and lots of experiments, I could figure it all out, and I’m explaining it here.

The Register 165 explanation also then makes it possible to understand the Si5351A datasheet document description of the 333ps typical phase offset step! This is because the internal VCO frequency specification (specified in App Note AN619, not in the datasheet!) is 600-900MHz. If a “typical” VCO frequency can be taken as mid-range, 750MHz, then the period of a 750MHz cycle is $1/750,000,000$ which is 1.3333 nano seconds. Ha! So, the 333ps is $\frac{1}{4}$ of a 750MHz cycle 1.333 nano seconds! The mystery is resolved.

Now we understand the phase offset register, it becomes easy. The phase offset is configurable from 0 to 127 quarter-cycles of the internal VCO. Recall that the internal VCO is divided down to produce the output frequency in the MultiSynth block. If we choose an even integer divider for the MultiSynth block and set the phase offset number to be the SAME as the divider, then that’s it! We get 90-degree phase offset of the output. It’s as easy as that.

For a quadrature (90-degree) LO outputs, just configure one output with 0 phase offset register (in other words, leave it at the default 0); configure the other output phase offset register to be the same as the even integer MultiSynth divider.

The MultiSynth division ratio has a minimum of 4 (for output frequencies over 150MHz). The phase offset register has a maximum value of 127. Since we are setting the phase offset register to the same value as the divider, and the divider is an even integer, this limits the maximum divider value to 126. It places a lower limit on the output frequency for which 90-degree quadrature can be obtained. Practically speaking, with a 27MHz reference oscillator crystal, the lowest frequency at which the Si5351A configuration supports quadrature LO on two outputs, is 3.2MHz. This means that this scheme cannot be used for 160m operation. This is why the QCX CW transceiver is available only for bands 80m and up!

One final but crucially important trick is required, in order to obtain 90-degree quadrature. Without it, the phase offset is just random (though stable), and you will be sorely disappointed, leading to a lot more head-scratching (I speak from bitter experience). The final trick is to do a “PLL Reset” using register 177, described thus in AN619 on page 60:

Register 177. PLL Reset

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	PLL_B_RST		PLL_A_RST					
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Reset value = 0000 0000

Bit	Name	Function
7	PLL_B_RST	PLL_B_Reset. Writing a 1 to this bit will reset PLLB. This is a self clearing bit.
6	Reserved	Leave as default.
5	PLL_A_RST	PLL_A_Reset. Writing a 1 to this bit will reset PLLA. This is a self clearing bit.
4:0	Reserved	Leave as default.

This register named “PLL Reset” is in fact very poorly named and explained, in my opinion. It has misled (and continues to mislead) many practitioners to believe that the operation of the Si5351A is similar to the Si570. In the Si570, the synthesiser can only be moved a relatively small percentage frequency change, then a “PLL Reset” action is required. People thought (and still think) that the Si5351A behaves the same way. It doesn't!

When the PLL Reset command is issued (by setting register 177 bits to 1), it causes an interruption to the synthesiser output frequency lasting around 10 milliseconds. This creates a horrible CLICK in the receiver audio. Some Si5351A libraries (and even early QRP Labs code) issued the reset command at every frequency change. When you tune the receiver, you get this nasty click-click-click as the frequency is swept. Horrible indeed!

The correct use of this register, which is now implemented in all QRP Labs kits, is to do a PLL reset only once at the beginning, and thereafter every time the MultiSynth “Divider” and phase offset parameters are set. The mis-named “PLL Reset” seems to set up the MultiSynth dividers and initialize the phase offset according to the values in the offset registers. It might have been more accurate to call it a “MultiSynth Reset” in my opinion.

Once you have done this “Reset” the phase offsets are magically exactly as you asked them to be. You may now tune the VFO by loading in new PLL feedback dividers (fractional). You can use as

large a frequency jump as you wish, as long as the PLL stays in the range 600-900MHz. If you stray outside this range then you need to alter the MultiSynth divider (and the phase offset register) values. 600-900MHz is a 1.5 to 1 tuning range. It is easy to choose a MultiSynth divider such that this tuning range covers the entire ham band of interest.

As you tune the VFO by updating the PLL feedback dividers in the Si5351A chip, the precise 90-degree phase offset is seamlessly maintained, the phase shift really IS a phase offset tracking frequency, it is not an offset implemented by a course discrete delay amounts. It's quite remarkable and very satisfying to see. Glitch-free tuning with a direct quadrature output: no divide-by-4 chip. The only thing you'll hear is what I describe as a "flutter" as the receiver frequency is tuned suddenly in steps. As you tune through a heterodyne you will hear it step down in 10Hz or 100Hz steps or whatever you have chosen to use for your VFO step size.

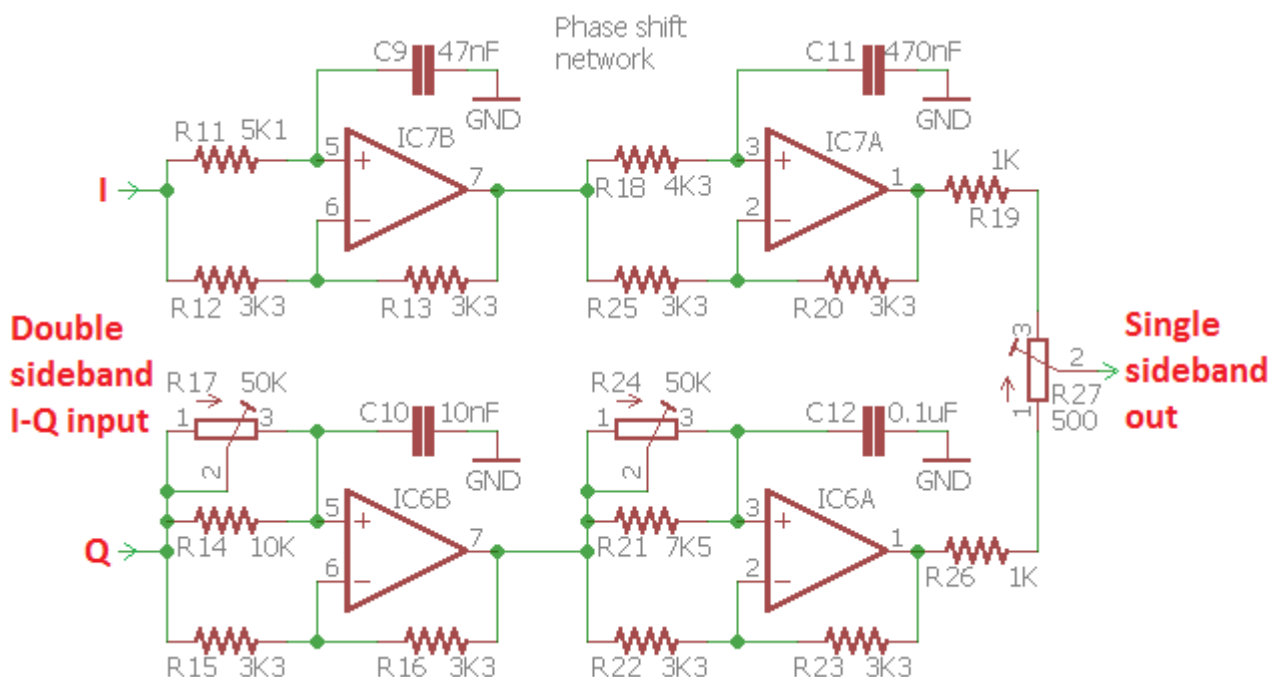
As I mentioned previously, obtaining the other sideband in the Quadrature Sampling Detector is just a matter of choosing a 270-degree offset instead of 90-degree offset. You won't want to set the phase offset register to a 3x higher number to get 270-degrees, because that will limit the lowest frequency to 3x 3.2MHz i.e. 9.6MHz. Instead, just set a 90-degree offset then set the "invert" bit in the Clk1 control register 17 (see AN619 page 20). Inverting the output is equivalent to a 180-degree phase offset so the total phase offset becomes $180 + 90 = 270$ -degrees, and you can still use it all the way down to 3.2MHz output frequency.

To summarize, these are the simple steps to get quadrature LO from your Si5351A, with glitch-free tuning and precise 90-degree phase offset seamlessly tracking the output frequency:

- 1) Use an even-integer MultiSynth divide ratio, in the range 4 to 126
- 2) Set the Offset parameter of Clk1 (register 166) to the same as the even integer divide
- 3) Calculate the PLL feedback fractional multiplier to get your desired output frequency
- 4) Do a PLL Reset ONLY when you change the MultiSynth divider/phase offset
- 5) If you want 270-degree offset, use 90-degrees and set the "invert" bit

Audio phase shift

The phasing method direct conversion receiver requires an audio phase shift as well as the RF phase shift (easily implemented as above, by configuration of the Si5351A synthesiser chip).



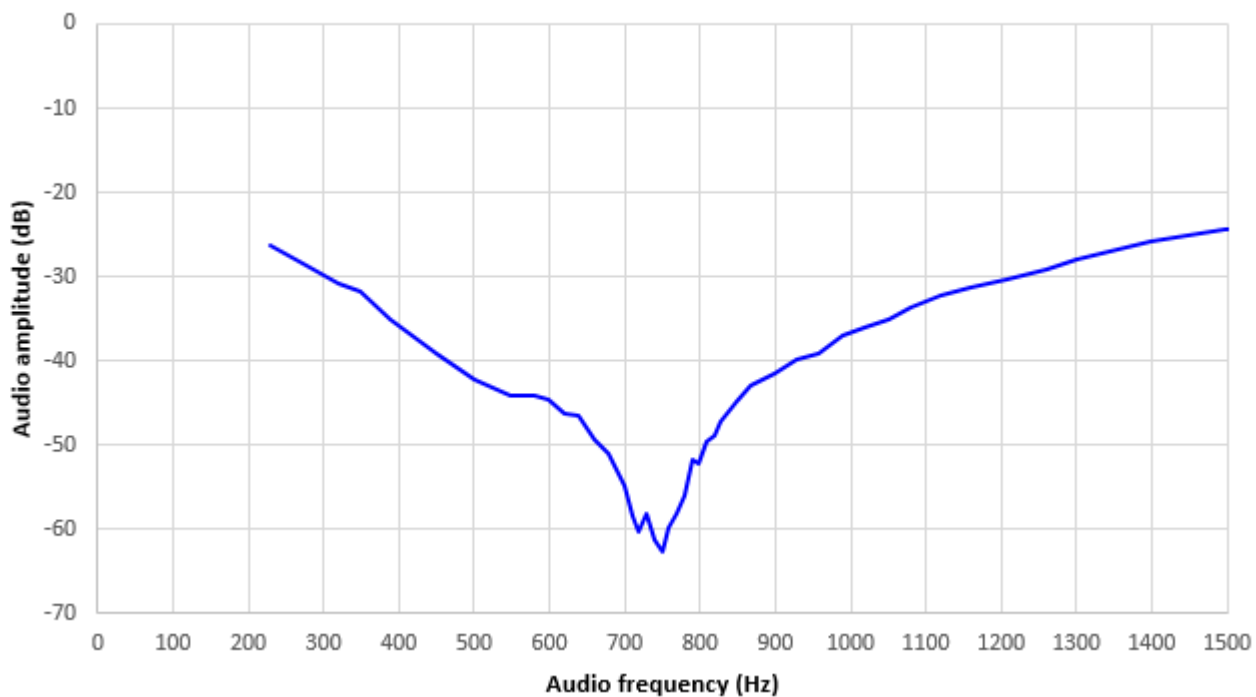
The schematic segment shown is how the audio phase shift is obtained in the QCX transceiver. It is an active two-path all-pass phase shift network using four operational amplifiers. The circuit is based on the same phase shift block as the Norcal NC2030 design by Dan Tayloe N7VE and elsewhere, see <http://www.norcalqrp.org/nc2030.htm>

Theoretically four poles give four nulls of "perfect" unwanted sideband cancellation. In the real world, nothing is perfect – there are component tolerances to think about. The unwanted sideband suppression is maximised when the amplitude of the two paths is equal, and the 90-degree phase shift is accurate.

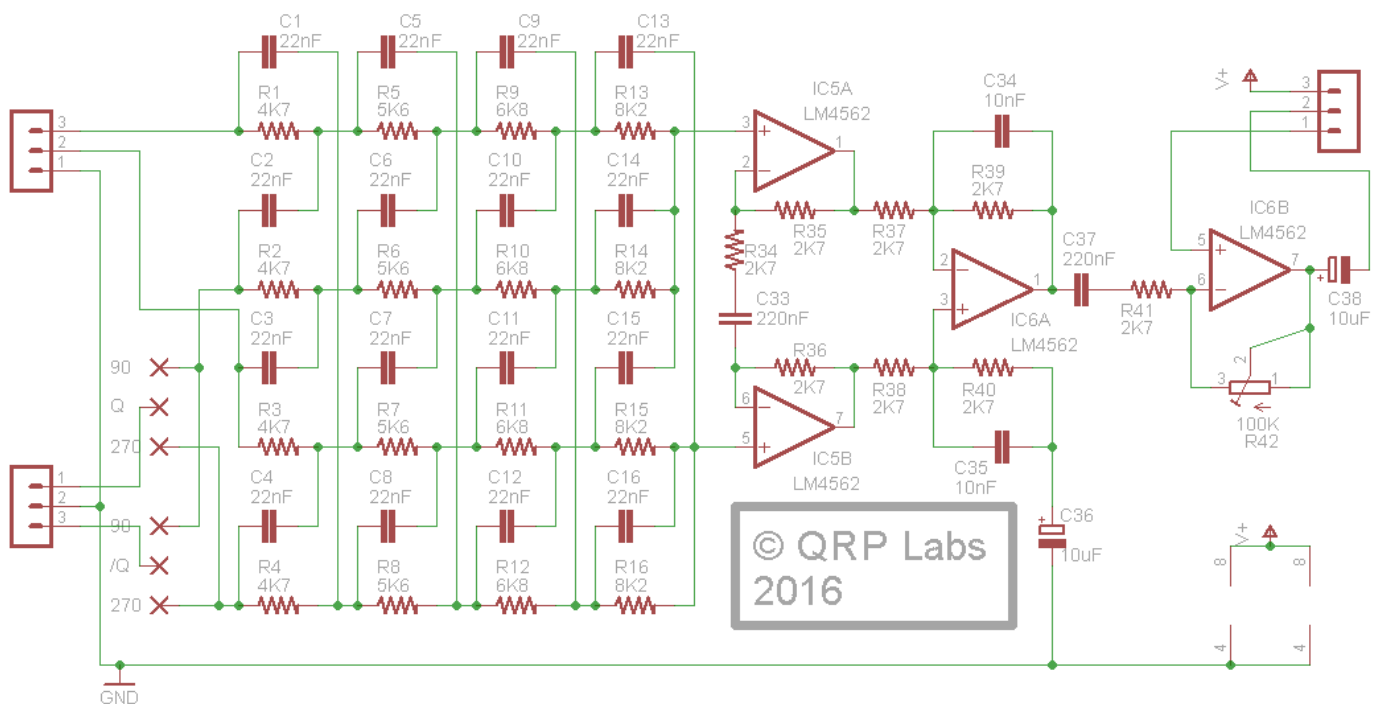
To improve the accuracy of the 90-degree phase shift, R17 and R24 multi-turn trimmer potentiometers allow adjustment of the phase shift at higher and lower audio frequencies respectively. R27 allows adjustment of the balance between the I and Q channels, to equalise the amplitude from each path.

The QCX kit includes built in alignment and test equipment, with a signal generator that can inject a test signal into the receiver input. It makes it easy to perform these adjustments, which will be described later. The adjustments make it possible to achieve better than 50dB of unwanted sideband cancellation across the narrow CW bandwidth of interest, as shown by the following QCX receiver measurement.

Unwanted sideband relative level



Another audio phase shift method is called “Polyphase” and uses a 4-path passive network of resistors and capacitors. This is used in the QRP Labs polyphase module kit <http://qrp-labs.com/polyphase> (see schematic below).



There are advantages and disadvantages of the polyphase approach. Theoretically it is less sensitive to component tolerances than the above 2-path active all-pass method; in other words, for any given component tolerance level, the polyphase method would provide a more accurate audio phase shift (and hence better unwanted sideband rejection). But the polyphase method is hard to adjust, and unless carefully designed can exhibit a non-flat frequency response.

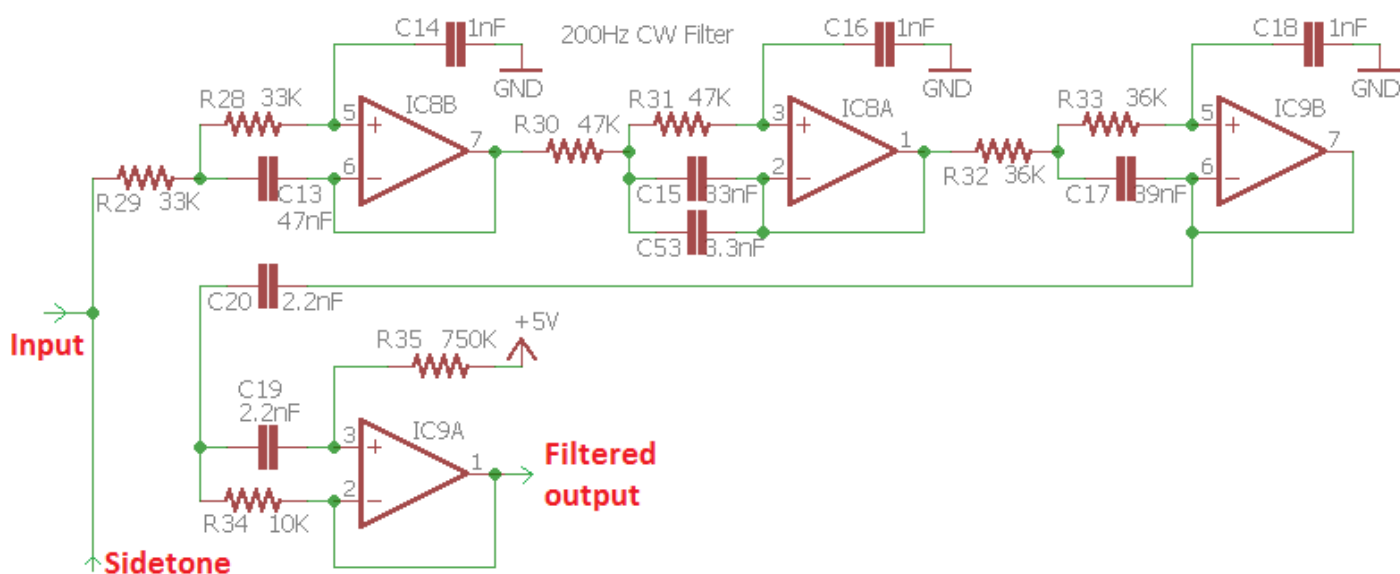
For some reason, the two-path active all-pass circuit has been more popular in US, and the polyphase network has been more popular in Europe.

In my opinion, after having experimented a lot with both methods over the years, my feeling is that for a CW receiver, an active all-path phase shift is preferable since the phase shift nulls can be adjusted; whereas in an SSB receiver which has a wider audio bandwidth, an adjustment-less polyphase network is preferable.

Yet another topology of phasing method direct conversion receivers for unwanted sideband cancellation is the “Weaver” or third-method. Interested readers are referred to <http://hanssummers.com/weaver> for articles and projects using the Weaver method. Many SDR implementations now use Digital Signal Processing versions of the Weaver method.

CW Filter and Audio amplifier

The CW filter section of the QCX CW transceiver is a well-proven design. It is not an area of the transceiver which I consider to be a modern technique; therefore, I do not intend to spend a great deal of space describing it. It is a high performance, solid design by David Cripe NM0Sm and is available as a kit from the Four State QRP group: <http://www.4sgrp.com/HiPerMite.php> (thanks David for permission to use it here). It has a 700Hz center frequency (which may be changed by substitution of component values) and 200Hz bandwidth; it is specifically designed to avoid objectionable ringing. The version used here has a few modifications to suit the DC biasing arrangement in the receiver. I think in use, it sounds wonderful.

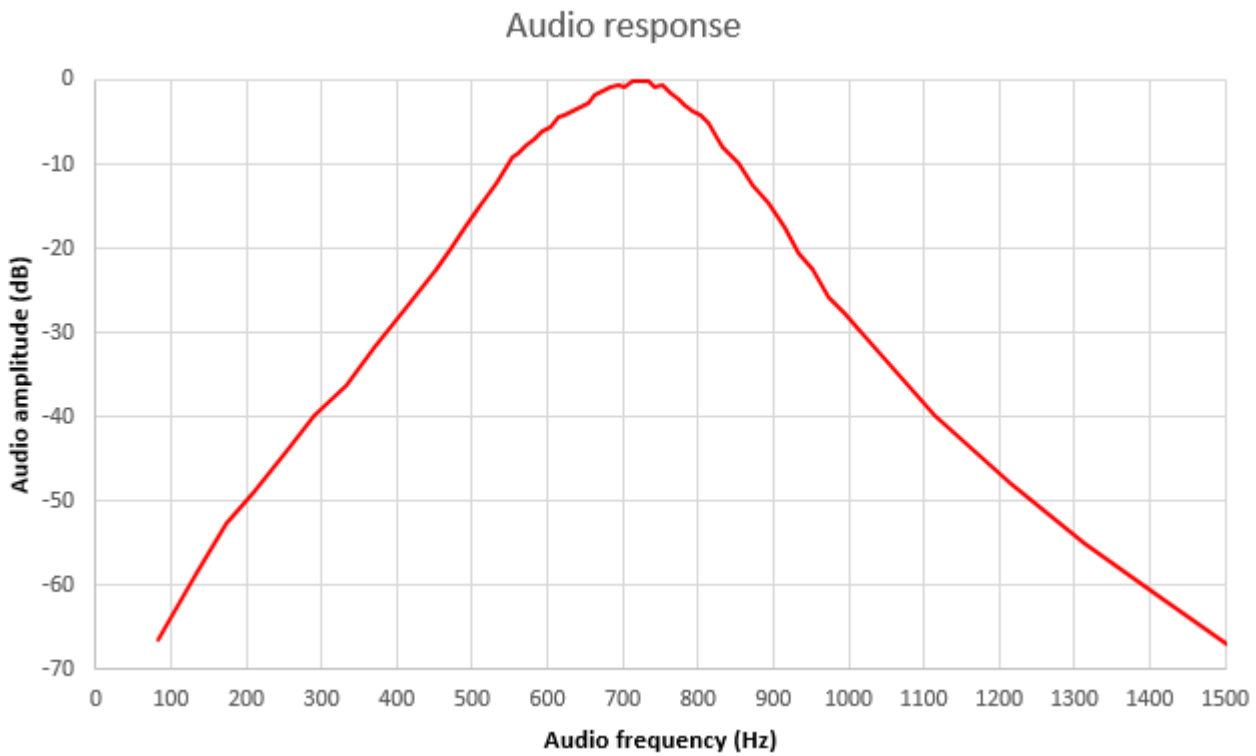


There are three stages of low-pass filtering and one stage of high-pass filtering. The first three stages retain the 2.5V “midrail” bias all the way through from the input transformer T1. The final stage IC9A is biased using the 5V supply (avoiding a few extra components to create a real 6V midrail at half the supply).

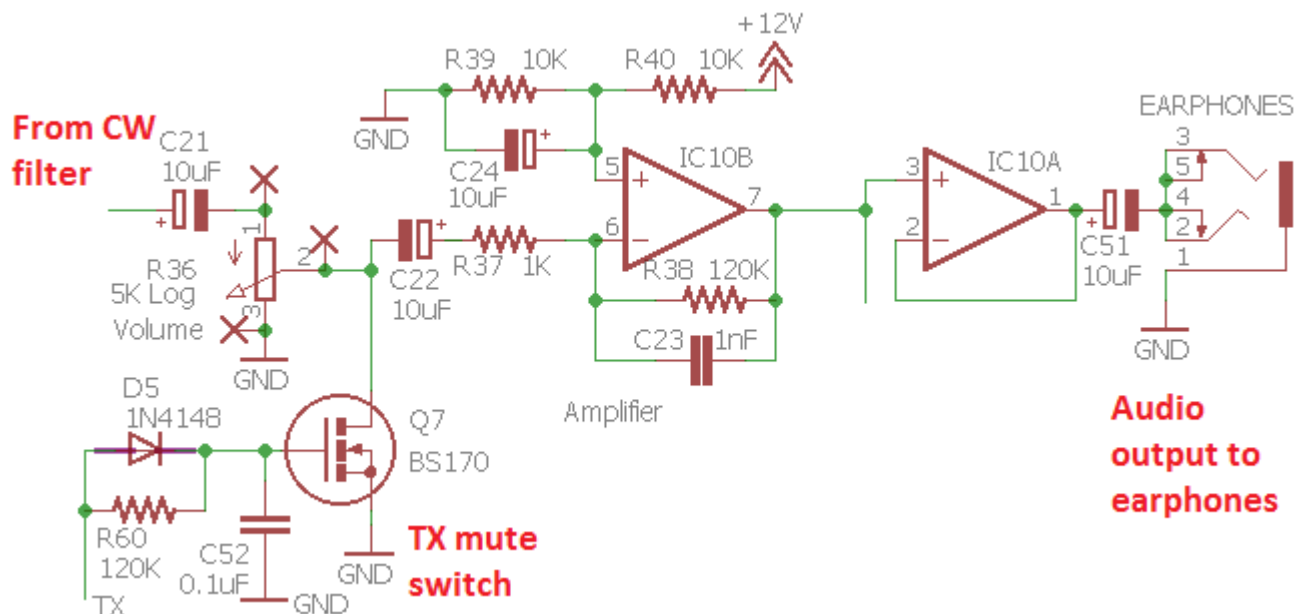
Sidetone is generated by the microcontroller and is a squarewave. To my ears, a 700Hz squarewave sounds horrible. So, to make it sound nice and clean, the sidetone injection is done at the input to the 200Hz CW filter, which cuts off any of the squarewave harmonics leaving a clean 700Hz sinewave sidetone.

The HI-PER-MITE CW filter also provides a measured 18dB of gain.

This is the measured response of the CW filter, normalized to 0dB at the center frequency.



The audio amplifier (see schematic section below) is preceded by a MOSFET switch which shorts the audio to ground during TX. A simple delay circuit at the MOSFET gate gives several milliseconds for the Rx/Tx “Pop” to subside before switching back to Receive. The gain stage IC10B has about 41dB of gain, and the final stage IC10A has unity gain (for stability driving the earphones).



There is no “real” audio amplifier IC in the QCX receiver. An important revelation for QRPers is that for headphone use with popular 32-ohm earphones for example, an op-amp is capable of providing sufficient drive by itself. There is no need for a “proper” audio amplifier IC.

If you wish to drive low-impedance headphones or a loudspeaker, then use an audio amplifier IC – but I recommend avoiding the LM386! It is another 3-decades old device (at least) which has been superseded by more recent devices which perform better, as well as being cheaper.

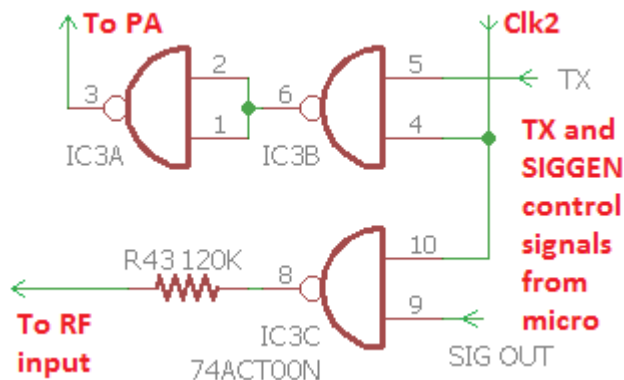
Transmit drive switching

Now we come to the transceiver's transmit circuits.

The schematic fragment shows the QCX transmit drive circuit. As we shall see in the next section, the Class-E final power amplifier requires a sharp 5V squarewave drive. The output of the Si5351A is a 3.3V peak-peak squarewave and is inadequate to drive a MOSFET into Class-E.

The gate IC3A (74ACT00 quad NAND logic gate) converts the 3.3V peak-peak from the Si5351A to 5V peak-peak. The "ACT" family is important in this instance, because the "T" in the part number indicates the TTL-logic compatible voltage thresholds. A 74AC00 has a binary "1" threshold of 70% of Vcc, which is 3.5V. The Si5351A will not reliably switch the input of a 74AC00. TTL logic's "1" threshold is defined as 2.4V historically. Using the 74ACT00 ensures that the 3.3V peak-peak signal from the Si5351A will switch the gate.

The NAND gate has the additional function of switching the Si5351A's Clk2 signal, routing it either to the Power Amplifier, or as a signal generator injecting it into the RF input of the receiver for alignment purposes.

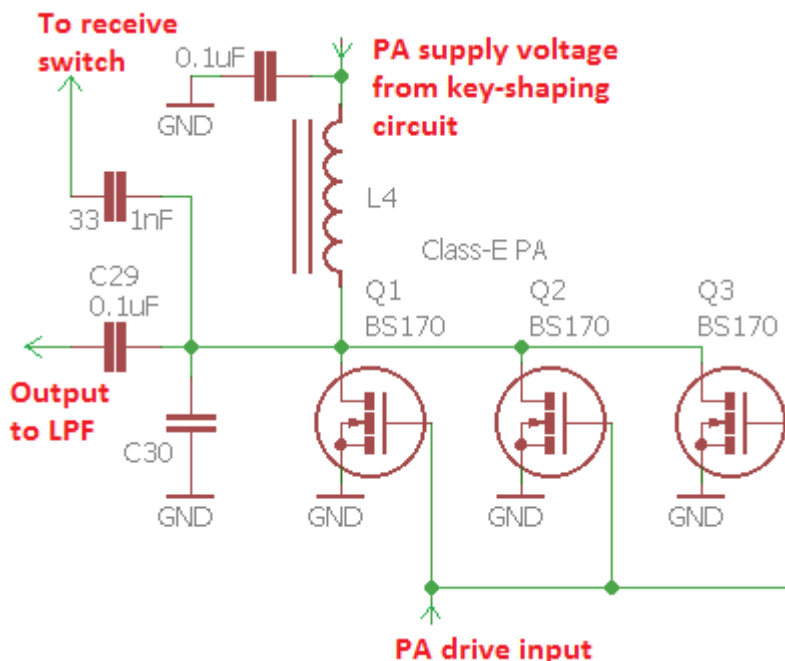


Class-E Power Amplifier

A Class-E power amplifier is a wonderful thing. It has a very high efficiency, sometimes over 90%. It should certainly be considered an important "modern technique" for the QRPer.

The high efficiency has several important benefits:

- Not much power is dissipated, we can use smaller (and cheaper) transistors
- So little power is wasted as heat that the requirement for a heatsink is reduced or eliminated
- During transmit the radio requires less current, so the drain on a battery is less – important for people who want to operate portable.



A Class-E Power Amplifier contains a resonant circuit at the frequency of operation, so it is only suitable for single-band use. It is also non-linear, so you can only use it in a CW transmitter, not SSB. Once you have mastered Class-E, you will want to use it EVERY time you build a monoband CW transceiver!

A lot has been written about Class-E, much of it is very technical and mathematical. Unfortunately, this has tended to make it a scary subject that many of us mere mortals are afraid to approach.

Some excellent background reading are two papers by Paul Harden NA5N:

<http://www.aoc.nrao.edu/~pharden/hobby/ClassDEF1.pdf> and
<http://www.aoc.nrao.edu/~pharden/hobby/ClassDEF2.pdf>

Paul NA5N describes two defining features of Class-E:

- 1) Use of a square-wave drive to reduce switching losses: the transistors are either on, or off... no lossy region in between
- 2) Reducing the effects of the transistor capacitances. Class-E has a resonant tuned circuit. The capacitance of the transistors, normally an unpleasant lossy aspect, is now a part of the tuned circuit.

Class-E also has a reputation for being difficult to achieve. All those intense mathematics Google will help you find, don't help. In reality, once you realise the secret – it is not so difficult. My “ghetto” design process for a Class-E amplifier is simple. Perhaps it is not totally optimal and a few more percentage points of theoretical efficiency could be squeezed out by the more advanced mathematical treatment. But for the average ham, my method produces excellent results with a minimum of fuss! I had previously attempted more complex methods and had always failed.

Calculation of the impedance of a resonant circuit is simple, and there are many online calculators which will do the job for you. For example, <http://toroids.info/T50-2.php> which allows you to type in the operating frequency, and the desired resonant circuit impedance. Then the calculator computes the required inductance, capacitance, and the number of turns required for a certain toroid (in the QCX, a T37-2 is used).

In the schematic fragment from the QCX transceiver, the “Class-E inductor” is L4 and the “Class-E capacitor” is C30.

First choose the output impedance. We usually choose 50-ohms, because this is the input impedance of the Low Pass Filter we will use. The online calculator will tell you what inductance is needed, and how many turns to wind on the toroid.

The online calculator also tells you the required capacitance to bring it to resonance at the operating frequency. Here we resort to experiment, because it is a little difficult to know what the output capacitance of the transistor is. The device capacitance varies depending on supply voltage and whether it is on or off. A simple experiment is required, adding different small capacitances to the circuit, and measuring the efficiency (measure supply voltage and supply current to calculate power input; then measure RF power output. Divide one by the other to get the efficiency). It is easy to find what additional capacitance is required to peak the efficiency. The resonance is quite broad and non-critical.

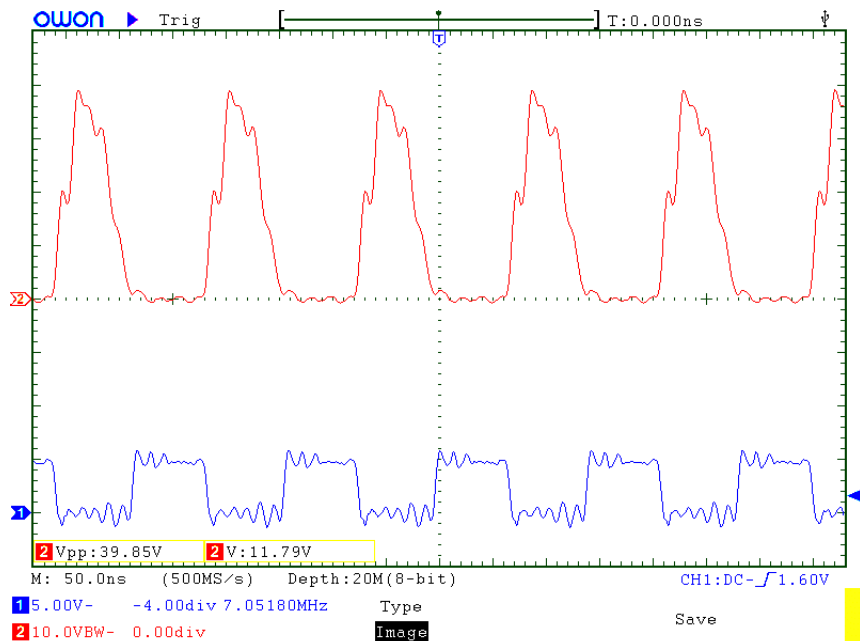
In the QCX transceiver, three BS170 transistors are used in parallel. The BS170 is inexpensive and small but is rated for 500mA drain current and up to 830mW of dissipation. Per device! Three in parallel provides plenty of capability to achieve a 5W “full QRP power” output on a single band.



An important note about MOSFET transistors: as any other component, there are always minor variations in device characteristics from one transistor to the next. If these were bipolar NPN transistors, we would not be able to parallel them in this way. If one NPN transistor takes more of the load and starts to heat up, its resistance further decreases and this causes it to get even hotter. This process is known as “thermal runaway” and results (quickly) in destruction of the transistor. Emitter resistors are used to help balance the load, and they also lower the efficiency. But with MOSFETs, their resistance INCREASES as the

temperature goes up – so there is an inherent self-balancing when multiple devices are used in parallel, without any need for additional balancing resistors which would increase component count and waste some power. MOSFETs are a really useful component for a QRPer! Three cheap BS170s can be paralleled like this to get you full 5W. No heatsink is required for CW mode which is not continuous key-down. This helps reduce board area and cost!

This oscilloscope screenshot shows the classic Class-E waveform. (Ignore the ringing due to poor set-up of the 'scope probes etc). The lower (blue) trace is the 5V squarewave at the gate of the BS170 transistors. The upper (red) trace is the voltage as the BS170 drain. It peaks at approximately 40V in this example. This measurement was done with 12V supply and on 40m (7MHz).



The important point to note is that when the BS170 transistors are switched ON (the gate voltage is 5V), the drain voltage is zero. When the BS170 is OFF the gate voltage pulses nicely to a large amplitude. This is what Class-E is supposed to look like!

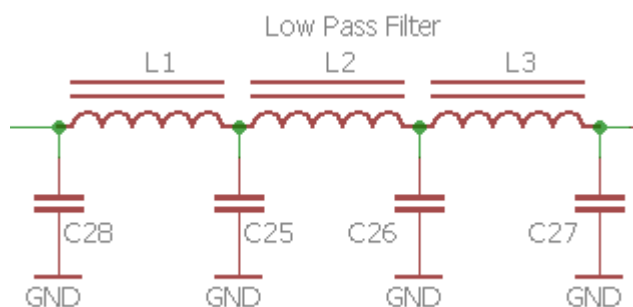
The summary: Class-E is actually quite easy to achieve in practice! Perhaps all the complicated mathematics might help to squeeze out another % or two of efficiency. But for practical purposes, it's a wonderful building block to use in a single-band CW transceiver, and just use these three simple steps to choose the component values:

- 1) Decide the output impedance: normally 50-ohms to match the Low Pass Filter without hassle
- 2) Use an online resonance impedance calculator to tell you the inductance to use for the desired operating frequency
- 3) Experiment (measure efficiency by measuring power output and DC power input) with different Class-E capacitors, to find the value which gives the best efficiency.

Output harmonic filter

Now it's important to note that the output of a Class-E amplifier is far from a nice clean sine wave. A good Low Pass Filter (LPF) is required to clean up the harmonics and keep the FCC from knocking on your door.

The design used in the QCX is nothing revolutionary, it is a 7-element filter design originally by Ed W3NQN then published for many years on the G-QRP Club web site's technical pages.



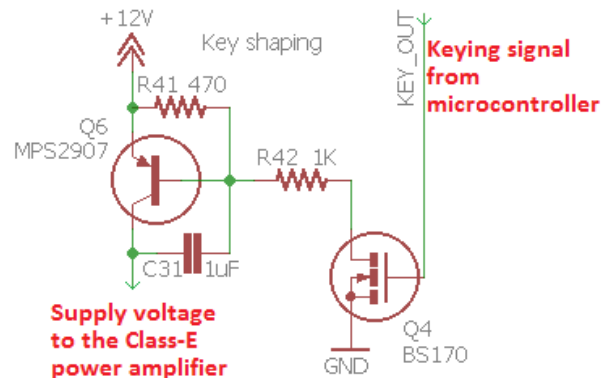
Key-shaping circuit

Talking about clean output signals, it is important to avoid “key clicks”. A hard-keyed CW transmitter generates spurious instantaneous energy many hundreds of Hz away from the transmitted signal that can annoy users of adjacent frequencies, sounding like a “Click” on key-down and key-up. This is purely a consequence of the mathematics of the Fourier transform and is unavoidable. Any time you switch a signal instantly on or off, you WILL splatter energy onto unwanted nearby frequencies.

It’s a worse crime for a QRO operator to commit, but even a QRPer should try to emit a clean signal with good keying characteristics. Any good CW transmitter should include an RF envelope shaping circuit to soften the key-down and key-up transitions. The ideal envelope shape is a raised cosine, but this is difficult to implement in simple circuits, without significantly increasing the complexity.

The simple key-shaping circuit used here uses only a few components, including a PNP transistor, but produces good results.

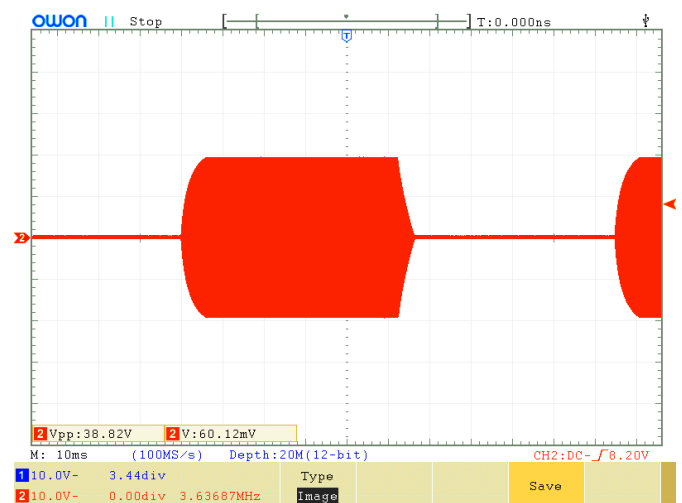
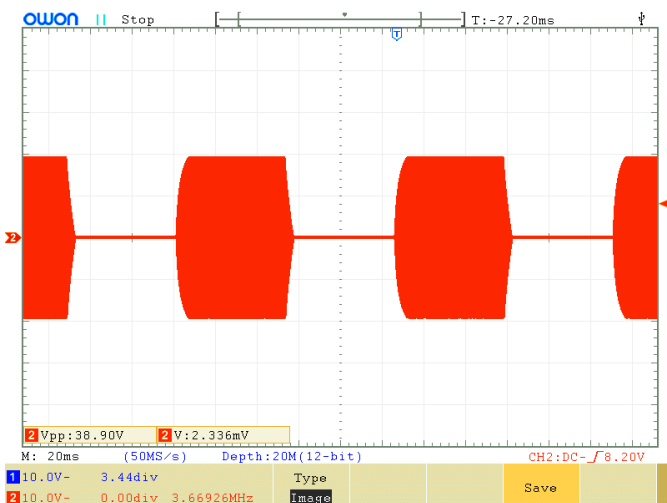
This circuit was derived by one published by Don Huff W6JL, who I first met through my interview on the QSO Today podcast <https://www.qsotoday.com/>. Don’s QRZ page has some fascinating details of his homebrew station, see <https://www.qrz.com/db/W6JL/>. As Don says, “*this integrator-type keying circuit is found in many published homebrew designs over the past 40 years or so, so it is nothing new*”. It uses a PNP transistor (Q6) and R-C integrator circuit. Don W6JL uses this key-shaping circuit to drive a 600W Power Amplifier!



On key down the Q4 switch is “closed” by the keying signal. In the QCX transceiver kit, this signal is generated by the microcontroller. In a really simple transmitter, Q4 could just be replaced by a straight Morse key to ground!

The component values set the rise and fall time. With the components shown, the rise and fall time is about 5 milliseconds.

The following oscilloscope screenshots show a 40m band (7MHz) transmission, keyed with a continuous series of CW dits at approximately 24 words per minute. The amplitude is approximately 3.8W into a 50-ohm dummy load (with 12V power supply).



It's not raised cosine (theoretically ideal); but it's a big improvement over hard on/off keying, and such a simple circuit that it is feasible to include it in all your QRP transmitter projects.

Solid state transmit/receive switch

In my past I have used a toggle switch as a transmit/receive switch, to select between two separate pieces of equipment. Sometimes people use a relay. But relays are expensive, and unreliable (a mechanical device with limited lifespan), and you can't use QSK (break-in) unless you can tolerate all the clattering. For QRP purposes, a relay is overkill.

Some QRP designs use a small value capacitance to couple the RF connector to the receiver, and back-to-back diodes to limit the peak voltage delivered to the receiver.

This is not a great design, because the small value capacitance has significant impedance at the operating frequency, which decreases the sensitivity of the receiver; and the back-to-back diodes create intermodulation products in the presence of strong signals, effectively degrading the third-order intercept and dynamic range of the receiver.

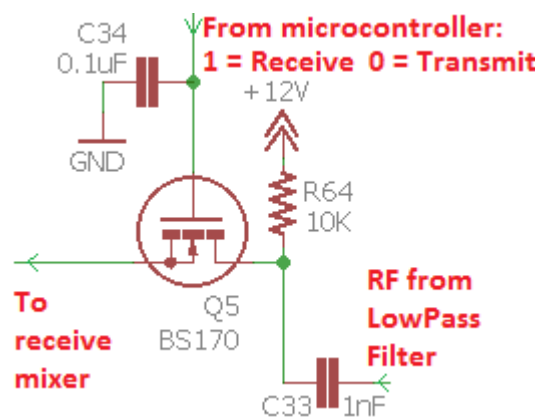
The single-transistor circuit used in the QCX transceiver is effective and simple to implement. The source is at DC ground via the primary of the receiver input transformer T1. The drain is pulled to +12V via a 10K resistor, this was found to be important to prevent instability due to leakage. The MOSFET functions as a very effective on/off switch, letting the signals pass through with very little attenuation when ON, but blocking most of the RF when OFF.

The solid state transmit/receive switch allows full break-in (QSK) operation – the ability to switch back to receive and listen to the band in between your key-down dits. Not everyone likes this; personally after many years of not having QSK, when I started to use it (on the QCX transceiver) I quickly came to love it and would not wish to live without it now!

Microcontrollers

Throughout the text so far, I keep referring to “the microcontroller”. A microcontroller is a single IC which integrates a microprocessor, some memory, and some peripherals all on one chip. It's like a miniature computer-in-a-chip. We have GOT to have a microcontroller, because if nothing else, we need something to control the Si5351A synthesizer chip (or DDS or Si570, if you prefer those). For those people who have not used microcontrollers, and perhaps secretly or openly FEAR them... you really should give it a try! Today there are easier and lower cost ways to get started than ever before. There is a wealth of information and advice out there.

Anyone can do it! My Grandfather introduced me to his ZX Spectrum 8-bit home-computer in the 1980's and continued with subsequent machines until his passing (aged 84). These days, my daughter (age 4) takes a table to her pre-school for elementary coding classes. If you're aged 4 to 84, I know you can do it. And I bet outside that range too.



Popular low-cost 8-bit microcontroller families are the PIC (by Microchip) and AVR (by Atmel) families, both of which have been around for many years. These families provide a range of devices with different pin count, memory size, peripherals and so on. You can choose the appropriate device for your application and yet retain all the features you are familiar with. For many years, many hobbyists have divided themselves into semi-religious camps debating the superior features of AVR or PIC... well in 2015 Microchip acquired Atmel so we're all brothers now.

A very popular choice for newcomers to microcontrollers is the Arduino platform. Here's a picture of the original "Arduino Uno" but they are now available in a huge variety of shapes, colours, different devices (more power, more memory), different manufacturers. You can purchase an Arduino Nano from China for as little as \$2.50 including shipping!



A free PC development environment (IDE) makes programming the Arduino incredibly simple. A huge number of libraries and examples exist which will make almost any task easy. Learning to turn this into a powerful tool in your radios will be easy and fun. It's an ideal way to introduce yourself to microcontrollers. The official Arduino page is <https://www.arduino.cc/> and is the place to start for tutorials, examples, and the Arduino language and function reference.

A vast array of plug-in "shields" are available that provide many different peripheral functions, such as LCD screens, WiFi, Bluetooth connectivity, relays, sensors, you name it.

Perhaps the best aspect of the Arduino is the huge number of helpful people in forums and bloggers sharing their code. If you ever get stuck, people all over the world are ready to help.

The QCX transceiver does NOT use the Arduino as its microcontroller, though it does use the same Atmel AVR ATmega328 processor as the original Arduino Uno. This is an 8-bit processor with 32K of "Flash" memory for program storage, 2Kbytes of RAM for program variables, and 1Kbyte of EEPROM for non-volatile storage even when the power is switched off. A variety of peripherals in the chip include serial communications, timers and a 6-channel 10-bit Analogue to Digital Converter (ADC).



When using a mixture of digital circuits (microcontroller) and analogue (RF, audio) we have to take care that digital noise does not find its way into the analogue sections. Board layout can help reduce radiated noise. Try to keep digital signals short and locate relevant components near each other. Additional power line filtering is effective at preventing conducted noise through the power rails. In practice it turns out, in my experience, to be reasonably straightforward to keep digital noise out of analogue circuits and as you can see in the QCX transceiver, there is no special metal shielding etc.

Once we have a microcontroller onboard, many other advantages come to the fore, not just the ability to configure the Si5351A synthesizer chip. The microcontroller can handle many functions

which enhance the performance and usability of the radio! In the QCX the microcontroller is used for all of the following duties, some of which will be described in the following sections:

- Controlling the Si5351A synthesizer, and rotary encoder tuning
- LCD screen to show the frequency
- Iambic keyer
- Frequency and message memories
- VFO A/B/Split operation
- Transmit/receive switch sequencing
- Sidetone generation
- CW decoder
- Test equipment: DVM, Frequency counter, RF power meter, Signal generator
- Built-in alignment tools
- GPS interface for calibration and WSPR mode timing/frequency discipline
- CW and WSPR beacons

When you start to use a microcontroller to operate some of the complex functions of a project, and to add extra useful functionality, time and again you notice a pattern emerge. The microcontroller can replace some parts of the hardware and implement the same functions in the firmware. The component count of your project (and hence the cost) goes down. The flexibility increases, because now you can change the function just by altering the firmware a little bit.

Software? Firmware? Or “Sketch”?

These terms and more (“Program”, “Code”) are often mixed up, when describing Software. I’m not sure what the formal definition is, if there even is one, and who would be in charge of deciding it anyway. But my own understanding is roughly:

“Software” describes the big software applications which run on a general-purpose computer such as your PC.

“Firmware” is the program that runs on a microcontroller embedded in a small system designed to operate and manage one particular piece of hardware. This applies to our use of microcontrollers in radio transceivers.

“Sketch” is just what the Arduino people call their programs that run on the Arduino board. If you aren’t a programmer (yet), then “sketch” probably sounds less scary.

Rotary encoder tuning and the Si5351A

The Si5351A configuration has already been discussed in some depth, and reference to source code samples will provide more information on the exact register configurations needed.

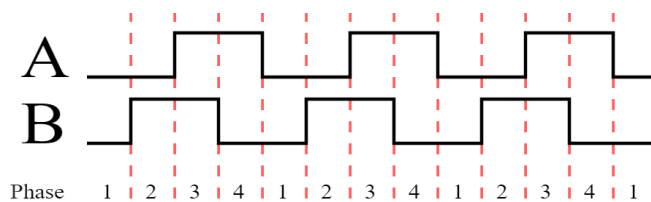
A convenient way to tune the synthesizer is with a rotary encoder. Expensive optical shaft encoders could easily cost upwards of \$50, that’s more than an entire QCX kit! But inexpensive 20- or 24-position mechanical encoders are readily available and work well. They have a push-button on the shaft too. Regardless of the type (mechanical or optical), they all operate on the same principle: they have a



quadrature output; a microcontroller observing the sequence of voltages on the two output signals can derive both the direction and speed of rotation.

All mechanical switches exhibit switch bounce, where the switch contacts generate multiple transitions for a short time when the switch is activated. It is common to see resistor-capacitor networks to debounce these switch transitions.

Simple debounce circuits involving a resistor and a capacitor inevitably involve a compromise when choosing the R-C time constant. It is easy to miscalculate and make the time constant too short (bounce noise still gets through) or too long (rapid switch closures are missed). In some cases, it is impossible to find the sweet spot in between these two extremes.



In my opinion, resistor/capacitor debouncing is a poor solution to the problem when the project contains a microcontroller. Why not get the software to do the work? This allows you to control time-constants or other debounce logic much more precisely. It also eliminates those additional resistors and capacitors, which helps reduce the cost and complexity of the project!

Another unnecessary component often seen is a pullup-resistor on a switch signal to a microcontroller, so that when the switch is open (not pressed) the microcontroller sees a “high” voltage. This is unnecessary because modern microcontrollers such as the ATmega328P all contain configurable internal pull-ups!

Many examples online show how to read a rotary encoder. In my opinion, there is a right way and a wrong way to do this, and unfortunately the majority of these examples fall into the latter category. These often use timing loops or interrupts to determine the direction and speed of rotation and prevent false triggers due to switch bounce noise.

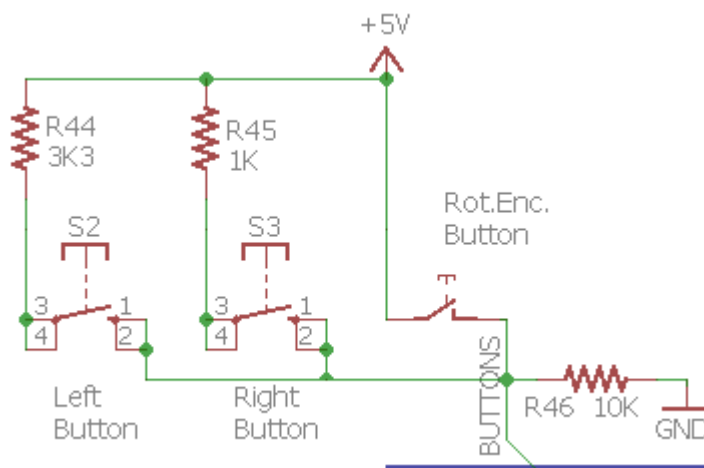
In the QCX transceiver, the rotary encoder is debounced using a state machine, rather than any timing loops. Simply put, the microcontroller knows what order of the four permutations (states) indicate travel in one direction, or the other. Upon the transition from one of the four states to the next, the microcontroller will ignore any subsequent transition between those two states, only attending to changes to the states on either side of it. In this way, if there is any switch bounce at a state transition, it is simply ignored. All the switch debounce work is taken care of, with no additional effort, and no time constant optimization.

What if I run out of I/O lines?

You may well find that there are more things you want to control with the processor, than input/output control pins (I/O) to do it with. All is not lost.

A simple trick to read multiple open/closed button switches is to use an Analog to Digital Converter input to the processor. Then you can read several switches on a single pin of the processor.

In the QCX transceiver kit, the left button, right button and center button (rotary



encoder shaft) are all read via the same ADC pin. Three resistors implement various potential dividers so that the input voltage depends on which button was pressed, as follows:

Button pressed	Voltage
None	0.00V
Left	3.76V
Center	5.00V
Right	4.55V

It's a reasonably straightforward procedure for the processor to repeatedly read this analogue input, and check if the measured voltage is somewhere near one of these; when it is, call further functions to effect the required actions.

Another example of shared use of an I/O pin in the QCX transceiver kit is the LCD module's RS input, which is shared with the SDA signal of the I2C serial data bus. The level of the RS input only makes sense to the LCD when it is clocked on its EN input. Meanwhile the level on the SDA signal of the I2C bus is only important to I2C peripherals (in this case the Si5351A) when the SCL (clock) signal transitions. Since these two activities never occur concurrently, the LCD RS and the Si5351A chip SDA can both be controlled in parallel by the same microcontroller pin.

There are often many opportunities such as these, where one microcontroller pin can be shared for many purposes. Microcontrollers present tremendous flexibility for QRP projects.

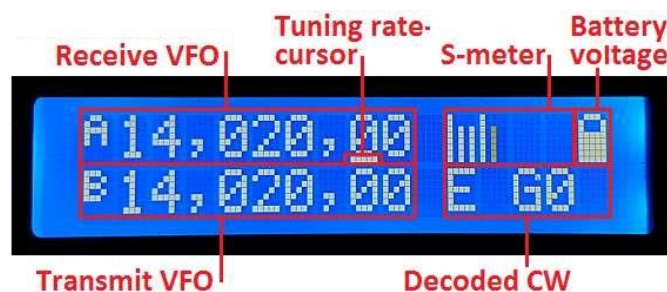
LCD Module

These 16-character 2-row LCD modules (known as 1602) are available very inexpensively. If you want a frequency display on your radio, it is no longer necessary to wire-wrap a bunch of discrete logic chips and 7-segment LED displays. The ubiquitous HD44780 controller standard is implemented on practically every LCD module you can find.



In the QCX transceiver kit, the microcontroller is already telling the Si5351A what frequency to produce, by setting its register configuration. It is a simple matter to write this information to the screen at the same time. This picture shows the information which can be shown on the QCX transceiver display.

These display modules can operate in 4-bit or 8-bit mode. The 4-bit mode is preferable because it uses fewer processor I/O pins. Display modules with an I2C interface are also available, albeit slightly more expensively. These can generally be wired in parallel with other I2C devices such as the Si5351A because in general each device will have a different 7-bit I2C address. Therefore, they require NO additional I/O pins at all!

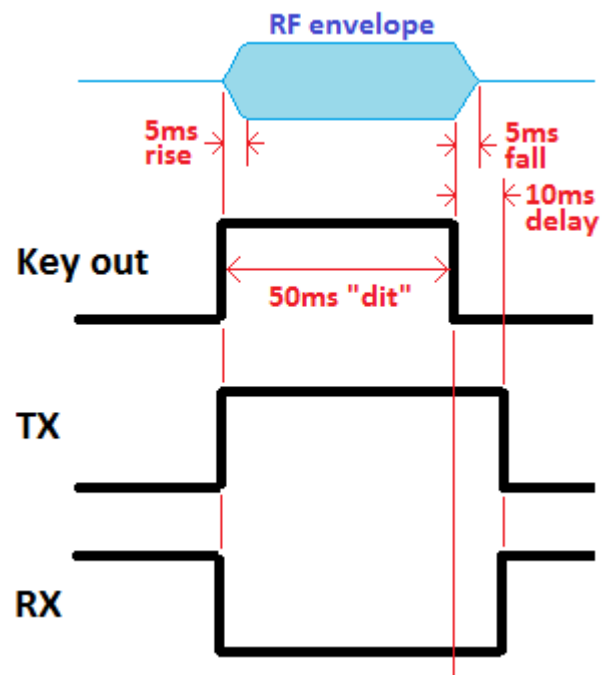


Iambic Keyer, and Tx/Rx sequencing

Several QRP rig designs have a separate keyer chip. Once you have a microcontroller in charge of the radio, why not let it manage the keying too!

In the QCX transceiver, the microcontroller activates the keying signal which turns on the key-shaping transistor. The actual RF signal generation must continue for long enough after key-up, for the RF envelope to decay (as controlled by the key-shaping transistor). The Transmit/Receive switch and audio mute must also remain in "Transmit" mode during this period.

The decay time of the keying envelope is approximately 5ms and the microcontroller waits for 10ms after key-up before deactivating the Si5351A transmit signal generation and switching back to Receive mode. The diagram shows a 24wpm dit lasting 50ms, and the corresponding envelope decay and TX, RX switching signals.



Sidetone generation

All modern microcontrollers contain a number of timers, which can also be used as Pulse Width Modulation (PWM) generators to certain output pins of the microcontroller. Both the frequency and the pulse width can be varied by the microcontroller.

This presents a simple and ideal way to implement sidetone in a modern transceiver design. The actual frequency can be configured at the whim of the operator. The sidetone volume can also be adjusted, simply by altering the pulse width.

Any regular stream of pulses (rectangular waveform) can be expressed as the sum of an infinite series of sinewaves, at the fundamental frequency and all the harmonics (with varying amplitudes). In the QCX transceiver the sidetone generation is injected into the audio signal path BEFORE the 700Hz CW filter. This neatly removes all of the harmonic content of the rectangular waveform, leaving only the fundamental sinewave. This is pleasant to listen to, but it also provides a sidetone volume adjustment at the command of the microcontroller. Maximum volume occurs at 50% duty cycle. Reducing the duty cycle reduces the amplitude of the fundamental sinewave, and therefore the volume of the sidetone. In the QCX transceiver the sidetone volume is adjustable in 100 levels from 0 to 99. The pulse width must be calculated from the chosen volume via a logarithmic function because of the way the human ear perceives sound.

CW Decoder

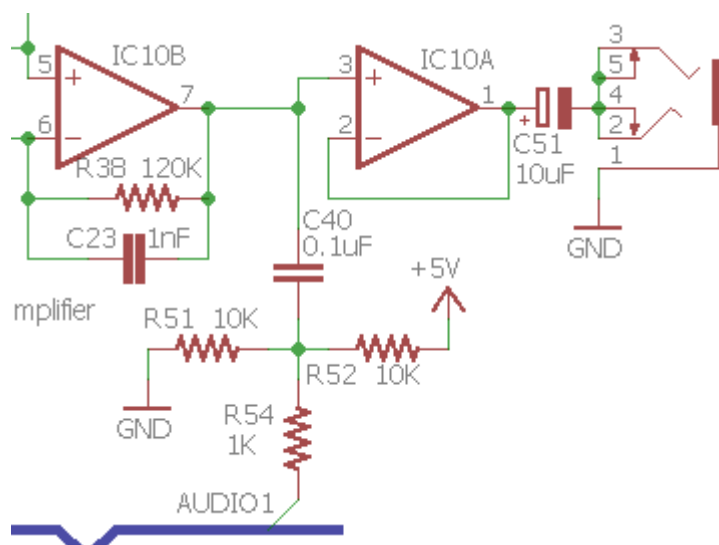
A CW decoder is a nice feature to have, for some people. Others hate it. If you fall into the former category, there are a large number of CW decoder examples available for the Arduino platform. Many of these are variations on a project by Hjalmar OZ1JHM <http://skovholm.com/cwdecoder> and indeed this project was my inspiration for the CW decoder in the QCX transceiver kit. In the end, I discovered some problems with this decoder (which I may call bugs), and several

improvements which I could make. My final code bears little resemblance to the original, once it is re-written to fit into the QCX firmware framework, fixes bugs, and applies some improvements.

In the QCX transceiver the CW decoder can be disabled on reception. It's useful though for keying in configuration parameters, much faster than pressing the buttons, particularly if you want to record a CW message into one of the memories!

Like the OZ1JHM decoder, I implemented the Goertzel algorithm Digital Signal Processing (DSP) in the ATmega328 microcontroller in the QCX transceiver kit. This uses the value from an Analogue to Digital Converter input, which is connected to the audio output of the receiver. The resistor/capacitor network applies a DC mid-rail bias to the audio signal.

The microcontroller samples the audio output at 12,019 samples/second, at 10-bit ADC resolution. It collects 48 samples for analysis by the Goertzel's algorithm, taking 4ms to collect 48 samples.



The Goertzel algorithm can be considered like a single bucket of a Fast Fourier Transform (FFT). It implements a DSP filter centered on 700Hz and with 250Hz bandwidth, additional to the operational amplifier analogue CW filter. The numeric output is an amplitude number every 4ms, which is then used for the CW decoder logic.

I was pleasantly surprised to discover that the humble 8-bit AVR processor has plenty of program space and processing power to manage some simple Digital Signal Processing, in this case the Goertzel algorithm implementing a 250Hz wide filter at 700Hz. The sample collection takes 4 milliseconds and the Goertzel algorithm calculation takes 1.5 milliseconds, so there is plenty of time left over to manage all the other tasks the firmware must keep track of.

I was even more surprised to discover that I could actually implement a full FFT, but that was rather complex and I put that to one side for some further work later.

So, 250 times per second, the algorithm produces a measurement of the amplitude of the 700Hz audio, representing it as a number in the range 0 to 16,383 (14-bits). The CW decoder algorithm applies a simple noise-blanker first, which ignores any pulses shorter than a specified duration (which is operator configurable).

Next the CW decoder must attempt to track both the amplitude (in case there is QSB) and the speed of the CW transmission, all automatically. To do this I coded an exponential moving average of the amplitude samples, and again the averaging duration is under the control of the operator. When the received sample amplitude is above the moving average for more than the noise-blanking ignore period, this is counted as a key-down; otherwise it is key-up.

The interval between key-down events is tracked and another exponential moving average applied to it, to try to automatically adjust to the CW speed. This interval is then used to decide whether the duration of a key-down event is to be interpreted as a CW "dit" or a "dah". Once we have collected a series of dits and dahs, and then hear a "space" which consists of a pause that is too long to be an inter-symbol pause, then the logged dit/dah pattern is matched against a table of Morse characters. When a match is found it is printed to the screen.

The result is quite pleasing; I have received a lot of feedback about the CW decoder. Many people have been surprised by how well it works. Others comment that it works somewhat but will never be as good as the proper analogue grey matter between their ears. Wetware beats firmware any day.

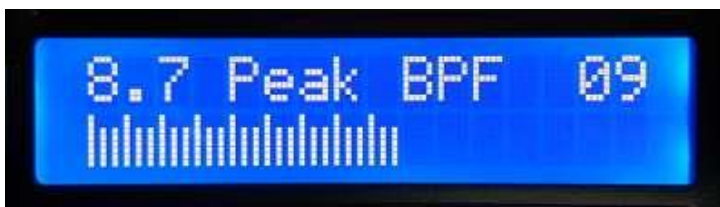
Built-in Test Equipment (BITE) and alignment tools

So far, we have a high-performance transceiver, with synthesised oscillator tuning and an intelligent control system providing a long list of features. But the combination of the three-channel Si5351A synthesiser and the microcontroller provides the opportunity for more!

With only a handful more components, we can include enough test equipment in the transceiver, to be able to align and debug the kit without any additional test equipment.

In the QCX transceiver kit, quick and easy alignment resulting in a well-adjusted radio is a very useful feature, both at the YOTA 2017 summercamp buildathon and for anyone else building the kit subsequently. “Built In Test Equipment” (BITE) is a feature not normally (if ever) seen in a low cost kit like this.

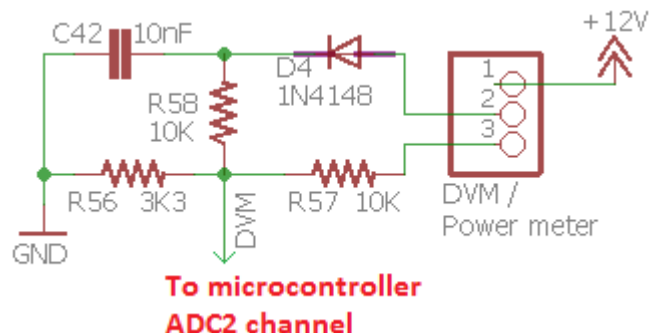
In ordinary use, the Si5351A’s Clk0 and Clk1 outputs drive the QSD during receive; and the Clk2 output drives the transmitter during transmit. Having three independent frequency generators means that we can also configure an alignment mode, where the 3rd (Clk2) output is used as a signal generator, fed back into the receiver’s input. It generates a test signal, which the microcontroller measures via its ADC and Goertzel algorithm DSP (see above) and displays on-screen as a signal strength bar. Peaking the Bandpass Filter (BPF) trimmer capacitor is just a matter of adjusting for highest signal strength!



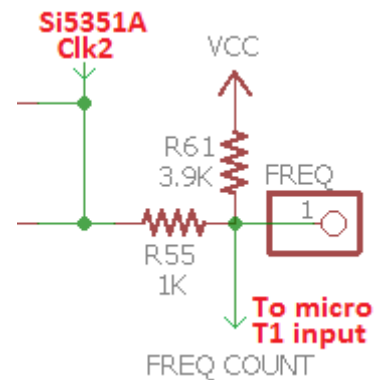
Other menu items cause the test signal to be positioned in the unwanted sideband. Three multi-turn trimmer potentiometers are used to fine-tune the I-Q amplitude balance and audio-phase shift circuits. They are adjusted for minimum amplitude shown the display (since we don’t want to hear the unwanted sideband). Within a few moments, it is easy to align the BPF and adjust the phase cancelation for better than 50dB of unwanted sideband suppression.

Adjustment of many QRP radios is not easy, and often requires significant test equipment. The QCX transceiver has in-built functions to completely align the four adjustments of the radio in a matter of moments, resulting in excellent unwanted sideband suppression.

Why stop there? The radio transceiver can even become useful in the shack for other purposes. The QCX transceiver implements a number of additional items of “test equipment” for general purpose use or for debugging the radio. They require no, or very few, additional components, and firmware is free!

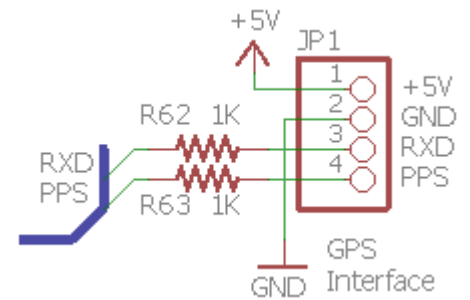


- 1) Digital Voltmeter (DVM): With the scaling potential divider made up of R56 and R57, and feeding to an ADC channel of the microcontroller, DC voltages from 0 to 20V can be measured and displayed on the screen. The DVM can be connected to the battery power (by connecting a jumper between pins 1 and 3 of the header, see above) and used to show a battery voltage icon on the screen.
- 2) RF Power meter: Addition of a diode detector (D4, C42) and scaling potential divider (R56, R58) creates a simple RF power meter measuring 0 to 5W. The firmware measures peak voltage, but must account for the diode voltage drop, and must then calculate the RF Power assuming a 50-ohm dummy load. Pin 2 of the header can be connected to the RF power output of the transmitter to make a reasonable measurement of the transmitter power.
- 3) Signal generator: In this function the Si5351A is just operated on its own as a free-standing signal generator, covering the entire range 3.5kHz to 200MHz. The Clk0 and Clk1 outputs are operated in quadrature (90-degree phase shift) while this is possible, i.e. while the frequency is above 3.2MHz. No buffering is included, just the raw Si5351A outputs. They are quite robust but not infinitely so. A buffer amplifier would be advisable too perhaps. The last thing you want to be doing is replacing that tiny 10-pin SMD chip (I speak from painful experience).
- 4) Frequency counter: The ATmega328 has a 16-bit timer counter input and in the QCX transceiver this has been connected to a pin header, and via a series resistor R55 and pull-up resistor R61, to the Si5351A Clk2 output. This gives the counter the opportunity to act as both a general-purpose frequency counter, and to count the frequency of the Si5351A's clock, to calibrate the 27MHz reference oscillator. The speed limit of the counter is 40% of the microcontroller's system clock (20MHz), this limits the range of the counter to 0-8MHz. The accuracy of the counter is limited to that of the 20MHz system clock – which itself can be calibrated (see next section).



GPS interface for calibration and WSPR mode timing/frequency discipline

The QCX transceiver also has a GPS interface. A GPS 1 pulse-per-second (1pps) signal and 9600-baud serial data can be connected to the paddle input which shares the same microcontroller pins. The I/O pins were all in use and so some sharing had to be done. It has the unfortunate side-effect of keying the transmitter, so the GPS must in this case, only be connected while calibrating the oscillators or when operating in beacon mode.



Two crystal oscillators run on the QCX transceiver board: a 27MHz crystal which is the reference of the Si5351A synthesiser IC, and a 20MHz crystal which is the system clock for the microcontroller. In both cases, to save component count (and board space, and cost, as usual), there is no trimmer capacitor to adjust the frequency to exactly 27.000000 and 20.000000MHz. A correction can be applied in the firmware! There is no need for extra hardware. The microcontroller only needs to know the actual crystal frequencies, then it can calculate compensating values for timing functions or for configuring the synthesizer output frequency.

These values can be measured by zero-beating on a calibrated receiver in your shack and entered manually in menu configuration items. But an alternative is to connect a GPS module and use the calibration menus. The microcontroller then uses the 1pps signal as a frequency counter gate. It counts the frequency over four consecutive seconds and takes an average. This can be done easily for the 20MHz system clock, which the microcontroller can count internally. In the case of the 27MHz synthesizer reference oscillator, the Si5351A is configured to pass through a buffered, divided-by-4 version of its 27MHz system clock to its Clk2 output. That 6.75Mhz frequency is low enough to be counted by the microcontroller's timer input.

When the QCX transceiver kit is used as a WSPR beacon (see next section), the GPS 1pps signal applies regular corrections to the synthesizer reference frequency by measuring it after every transmission; it also disciplines the real time clock. The latitude and longitude read from the GPS position data are converted into the 4-character Maidenhead gridsquare locator that is encoded into the WSPR message.

CW and WSPR beacon

It may be a cliché to say that there's no limit to what you can imagine doing once you have a microcontroller in your system. But to a certain extent it does start to feel like that. In the case of the QCX transceiver, at a late stage the CW and WSPR beacon was added to the firmware as a "bonus feature".

WSPR stands for "Weak Signal Propagation Reporter". It is a digital message format designed by Joe Taylor K1JT and filled with clever forward error correction. The message consists of three parts: the operator's callsign, Maidenhead locator (4-character, e.g. IO90) and two digits specifying the power. At the receiving station, messages are decoded and uploaded to a central internet database. At any time, you can go to WSPRnet <http://wspnnet.org> and click on the map, enter your callsign (and other filters if you wish), and see a map of where your signal is being heard. You could also undertake more in-depth propagation studies by downloading the database of reception reports.

WSPR seems well outside the remit of this article and is a whole detailed topic in its own right; so, I won't say any more about it here. You can read more details starting from <http://wspnnet.org> –

and suffice it to say that it can be a nice way to occupy your QRP transceiver while you aren't using it for CW QSOs!

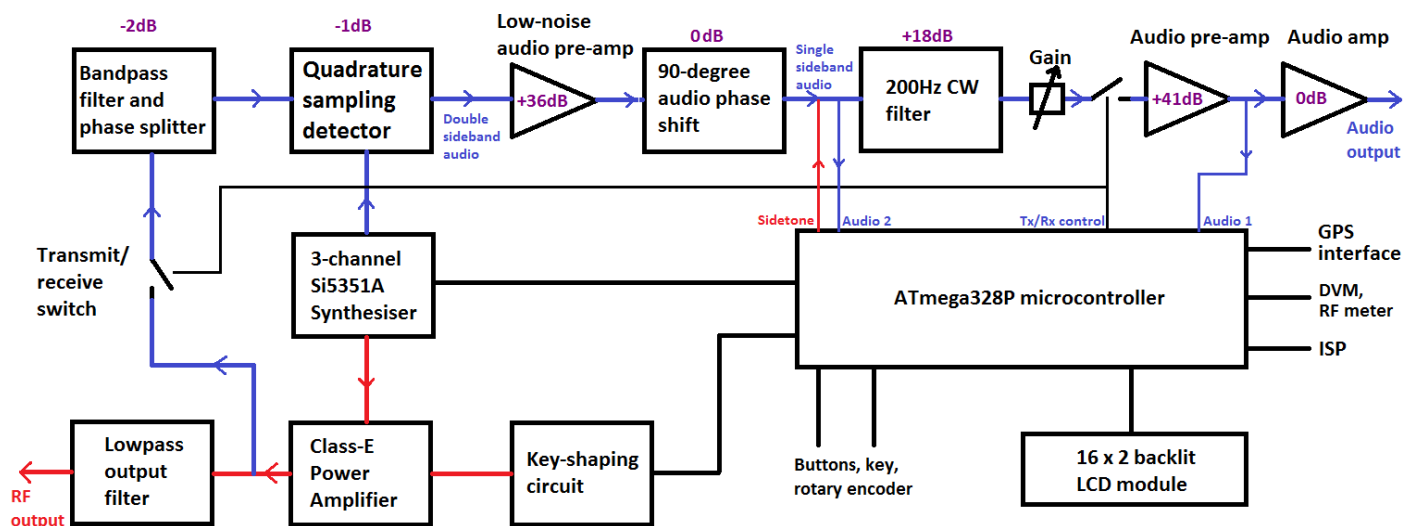
Using EEPROM for persistent storage

A final point to make about microcontrollers in your project is that you often want to store parameters even when the power supply is removed. The best way to do this is to write the parameters into EEPROM (Electrically Erasable Programmable Read Only Memory), which is one of the built-in features of many microcontrollers. 1kByte of EEPROM storage is available in the ATmega328. In the QCX transceiver for example, this memory is used to store:

- All the configuration parameters available in the firmware
- 16 preset frequency memories
- Four 100-character long message memories
- Eight 100-character long message memories

Using the EEPROM is easy, there are many code examples available and library functions in the case of the Arduino.

QCX CW transceiver more information



Above is the block diagram of the QCX CW transceiver. The full schematic is in the 141-page manual on the QCX web page <http://qrp-labs.com/qcx>. The QCX CW transceiver kit costs \$49 from QRP Labs.

Conclusion

I hope this article has generated some ideas for you to use in your own homebrew projects. Microcontrollers can be incredibly useful and you won't know what you're missing until you try it, then you'll never be without one again. The Quadrature Sampling Detector (a.k.a. "Tayloe Detector") offers high performance at low cost, and an escape from the classic SA602 superhet receiver. Modern synthesizer chips such as the Si5351A make building a flexible, accurate, drift-free and cheap oscillator 60dB easier than ever before.

Happy homebrewing and happy QRP!