An Introduction To SDR

Mike Richards G4WNC, takes a new direction this month in his *Data Modes* column looking at radio with a digital i.f. path!

Welcome to *Data Modes* (*DM*) where we're setting off in a new direction this month with an introduction to Software Defined Radio (SDR) As I'm sure you are all aware, SDR is creeping into rad o from all angles so, some basic understanding of the technologies and resulting benefits w uld seem to be in order

Th subject is a large one so I'm planning to deal with this over a couple of months. There's a lot to learn – and it does involved data modes!

What Is SDR?

Firstly, I suppose I should answer the question 'What is SDR?" Well the answer s that, according to the International Wireless Innovation Forum, Software Defined Radio s "Radio in which some, or all of the physical layer funct ons, are software defined".

Perhaps it might be easier to say more simply, SDR is where we replace one or mor stages of conve tional radio circuitry with computer hardware (PC) and software. This could be as simple as a computer controlled local oscillator or at the other end of the spectrum, digitisation of the entire h f. bands for subsequent processing in the PC.

Early examples of Amateur SDR dealt simply with the latter stages of the receive chain providing a low frequency digital final i f. and demodulation Technology is moving fast in this area and there are many affordable devices around that can digitise the entire h f spectrum! The SDR technique can bring with it, incredible levels of performance that are not possible with conventional circuitry.

One of the more obvious benefits is the quality of digital filtering that can be achieved. Not nly can the filters be incredibly steep sided but they can be adjusted ov r a wide ange simply by dragging the shape of the filter on a computer sc een.

Modulation and demodulation of just about any mode can also be handled in software and new modulation systems can be added with a simple software upgrade Programs such as **Simon Brown's** excellent *SDR Radio* include demodulation of most of the Amateur data modes from within the receiver software, thus creating a completely integrated solution

Another major benefit of SDR is the ext nsive use of spectrum ana ysis and waterfall displays to show show an enti e band Such system displays are usually accompanied with 'point-and-click' tuning, where you just put your mouse pointer over the required frequency and click! The system 'tunes' to that incoming signal.

From a wi e commerci l iewpoint, SDR is critically important and has revolutionised military radio systems. The requirement to interwork between the armed forces of different countries brings about the need for common operating modes and security systems. With SDR based radios, the same equipment can be used for multiple roles simply by updating the software.

The SDR Technologies

So, let's begin this introduction to SDR and to properly understand the subject we need to get to grips with a few fundamental technologies that are found in just about all SDR systems. These are: Analogue to Digital Conversion (ADC), In-phase and Quadrature data (IQ) Fast Fourier Transforms (FFTs), Digital Down Conversion (DDC) and decimation.

Note: *This isn't destruction down to one tenth part, although it is sometimes a large reduction in data* ***Editor***)

Please don't panic – this is not going to be a mathematical v ew of these systems, just sufficient detail so you can see how they work together to create Software Defined Radio. I've shown a block diagram of a modern SDR receiver in **Fig. 1**, so you can see how it a fits tog the .

Analogue To Digital Conversion

As the name analogue to digital, sugg sts, this is where we make that initial conversion from analogue (i.e. constantly and infinitely variable) signals to the discrete steps of the digital world The unit that makes the conversion is called an analogue to digital converter (ADC).

Although some of the devices and associated theory can be quite complex, the basic workings are very simple. As you'll probably know, in the digital world everything has to be

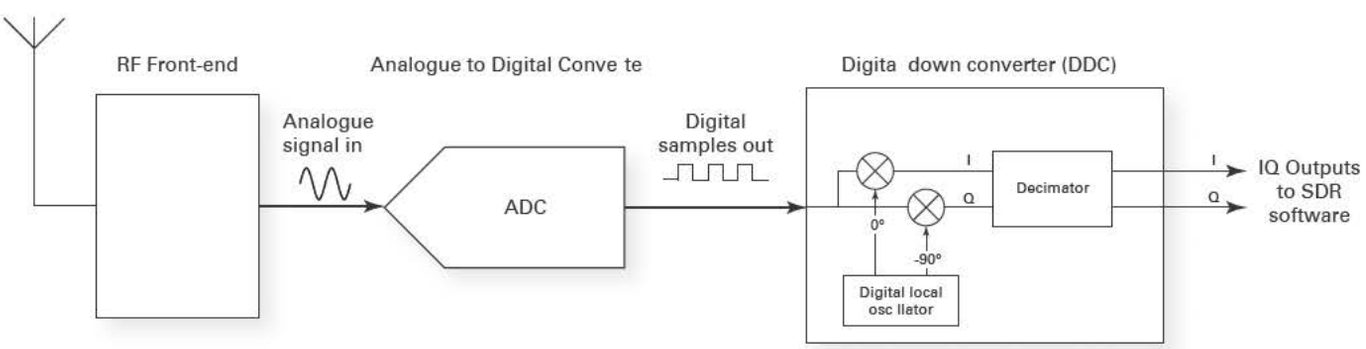


Fig. 1: Block diagram of a modern SDR receiver

represented as numbers and because computers only respond to on or off signals the numbers need to be in the binary system.

To illustrate the system, I've shown an example of the binary numbering system in **Fig. 2**. (***Note:*** *Unlike the decimal version of the number, where leading '0's aren't normally written down, in the binary version they should be shown.* ***Editor***). As you can see in Fig. 2, any number can be represented by a sequence of '1's and '0's.

Converting an incoming analogue signal to a stream of digital numbers is done by taking voltage measurements of the analogue signal at regular intervals. I've shown a simple sine wave being sampled in **Fig. 3**.

You'll also see that there has to be some way of representing the negative portion of the wave. But don't worry about how we do that – it doesn't make any difference to understanding the basic techniques.

It's easy to see that a decent digital representation of the signal requires lots of measurements or samples as they're more commonly known. Like many seemingly modern developments, the theories for sampling were developed long ago and in this particular case we have Swedish engineer **Harry Nyquist** to thank.

Decimal Number	Weighting for each binary digit 128	64	32	16	8	4	2	1
29	0	0	0	1	1	1	0	1
136	1	0	0	0	1	0	0	0
229	1	1	1	0	0	1	0	1
186	1	0	1	1	1	0	1	0
93	0	1	0	1	1	1	0	1

So, the decimal number 29 = '00011101', 136 = '10001000', 229 = '11100101', 186 = '10111010' and 93 = '01011101'. It might help to think of the 'numbers' 1 & 0 as voltage and no-voltage levels.

Fig. 2: Some examples of using binary numbering to represent decimal equivalents.

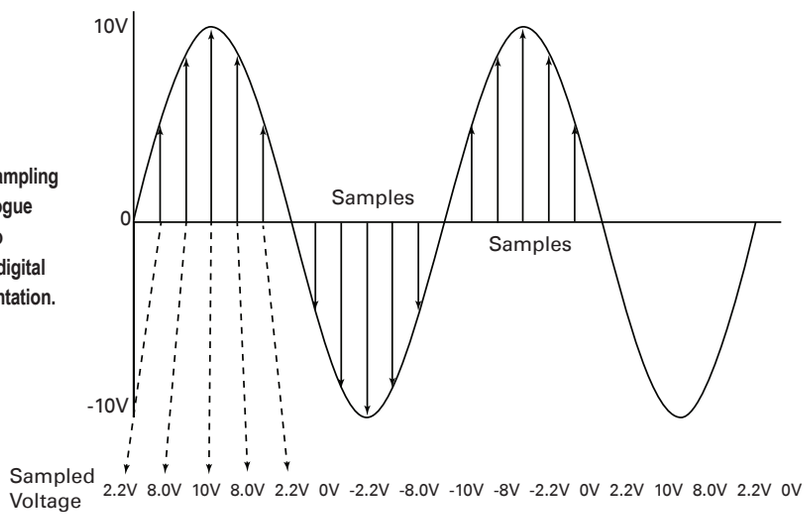


Fig. 3 Sampling an analogue signal to make a digital representation.

Accurate Representation

Harry Nyquist published a paper on the subject back in 1924. He showed, that an accurate representation of an analogue signal requires samples to be taken at a minimum of twice the rate of the highest frequency to be found within the signal.

So, for example, if we wanted to digitise a 455kHz i.f. signal, the analogue to digital converter would need to take measurements at 910kHz or higher. This would normally mean rounding up to 1MHz, or once every microsecond. This could be written as one million samples per second (1MS/s).

For computer sound-cards handling audio signals up to 20kHz the sample rate is usually set to 44kHz (44,000 samples per second or 44kS/s). Although some cards can sample at 96kHz (96kS/s) or even at 192kHz (192kS/s).

In addition to taking the regular measurements, we also need to consider how large a number we are going to use for each sample. This equates to the size of the measurement 'steps', the more steps, the more precise the representation of the signal level.

Many low frequency ADCs use 16-bit readings that give a total of 65,536 possible values for each measurement. However, in the case of very high speed ADCs (100 million samples or more per second) you will often find 14-bits in common use. This still gives 16,384 possible values per reading, which is still okay.

The final output from the ADC comprises a string of binary numbers flowing at the sample rate – that's a lot of information. There are two ways to get the information from the ADC to the next stage in the process – using parallel or serial data.

Parallel data uses a separate wire for each data bit, so in the examples here there would be 14 or 16 wires. On the completion of each measurement cycle all 14 or 16 bits would be transferred to the next stage.

However, there are a few complications. For reliable data communication from low voltage devices you really need to employ Low Voltage Differential Signaling (LVDS) or balanced outputs. These balanced outputs requires two wires per data line so we now need 28 or 32 tracks on the printed circuit board (p.c.b.) for our 14 or 16 lines, just to connect our ADC to the next stage.

Such a large number of tracks, would not only make printed circuit boards (p.c.b.s) design more complex but both devices would need to be physically large enough to contain an extra 28/32 pins just for the signaling. In practice, many of the latest ADCs have reverted to using serial data over a single LVDS connection thus reducing the signaling pin count to 2!

Serial data also has a snag and that is the very high data rate. Let's use a 14-bit ADC running at 100MS/s as an example. In this case, each sample produces 14-bits of data that are sent one at a time over the serial connection a bit like binary Morse code! As a result, the data rate becomes 100 million (samples) x 14(bits) = 1.4Gb/s – that's fast and demands special handling in the subsequent stages.

So, to summarise, the ADC takes rapid measurements or samples of the analogue signal. These numerical values produce an output that comprises a high-speed string of binary numbers representing the original signal.

What's IQ Data?

If you've already taken a peep at SDR radio you will doubtless have come across term 'IQ data' and perhaps noted that there aren't many friendly explanations out there. It's my turn to

have a stab at this so – let's see how I get on!

The fundamental operation of SDR is due to the stream of IQ Data, which comprises two signals called 'In-phase' (the 'I' part) and 'Quadrature' (the 'Q' part). These two streams can be used either to control (modulate) or to extract (demodulate) the amplitude and phase information from any signal.

Now my last sentence may seem like a pretty sweeping statement, but IQ information really is that powerful! If we can extract the frequency and phase information from a carrier that means we can see the amplitude, frequency and phase modulation plus any combinations of all three. The combination of amplitude, phase and frequency, will perhaps. explain why IQ data can be used to demodulate any signal and it's why it's at the heart of SDR.

The IQ process also works in reverse for transmit. So, by controlling the I and Q information into a transmitter, we can generate a.m., f.m., phase modulation (p.m.) or any combination thereof.

Extracting IQ Signals

While the theory behind IQ may be complex, extracting IQ signals itself isn't. To extract I and Q signals from an existing analogue source all we need to do is pass the signal into two identical mixers – but offset the local oscillator signal to the Q-mixer by 90° relative to the l.o. signal into the I-mixer. See **Fig. 4**.

The resulting I and Q signals from the mixers are still analogue but can be applied to a standard PC sound-card for digitisation and demodulation. As an alternative, the conversion to IQ data can also be performed digitally using digital mixers and local oscillators.

Streams of IQ data can also be used to generate modulated r.f. signals and in this case the process is simply 'run in reverse'. The IQ modulation information is generated in software (usually in the sound-card) and the two IQ baseband signals are applied to a pair of identical mixers as with the receive version. Again the r.f. carrier feeding the Q mixer is again offset by 90° – see **Fig. 5**.

The term baseband simply means a signal with its base at 0Hz, i.e. an audio signal before modulation is a baseband signal. So, if we were to put a speech signal through this IQ process the microphone output would go into the sound-card which would create two signals one with no phase shift (I signal) and one with a 90° phase shift (Q signal). Both signals would still be audio and so considered baseband.

Modern IQ modulator chips are readily available that can operate up into the GHz region and include a 90° phase shifter so you just connect a local oscillator and the IQ signals and you have a modulated r.f. signal ready for amplification and transmission!

Digital Down Conversion

The process of digital down conversion is used in SDR receivers to select a band of frequencies from the output of a wide-band ADC. If you recall in my analogue to digital conversion explanation, I explained how the data rate leaving the ADC can be extremely fast with 1.4Gb/s being typical for a 100MS/s ADC.

Handling this much data, 1.4GS/s, is beyond the capabilities of any of the standard computer serial interfaces and even if we could get it to the PC, the processor would struggle to deal with it. The solution used by just about all current SDR systems is to employ a dedicated Field Programmable Gate Array (FPGA) to deal with the high speed signal from the ADC.

The special FPGAs contain hundreds of logic blocks that can be programmed to take-on a wide range of functions. The FPGAs are also extremely fast and can happily deal with very high speed data streams. And as we don't actually need the entire spectrum for most receive purposes, we use the digital equivalent of a superhet receiver to select the band we want - this is known as Digital Down Conversion (DDC).

The DDC unit operates by applying the ADC output to two identical mixers and then applying an appropriate carrier to bring the required range of frequencies down to baseband. As we need IQ signals to achieve this translation, the signal to the Q mixer is again offset by 90°. At this point we have shifted our wanted band of frequencies down to baseband but the data rate has not changed, i.e. it is still running at 1.4Gb/s.

Decimation Stage

Let's now look at the Decimation Stage. To reduce the 1.4GS/s bit stream to a more manageable bit-rate, it's common practice to follow the DDC with a decimator, or decimation stage. In our example, we were trying to extract a 1MHz wide band of frequencies from a sample that covered 0-30MHz. But the 1.4GHz sample rate for a 1MHz bandwidth would be overkill.

For our 1MHz bandwidth, the decimator reduces the final sample rate to just over 2MHz by discarding unwanted samples. Once complete this more modest digital signal can easily be passed over a standard USB port to the PC for final processing.

So that was a quick review of the fundamental building blocks of Software Defined Radio. Next time, I'll take a look at some of the hardware and software and see how it all fits together. ●

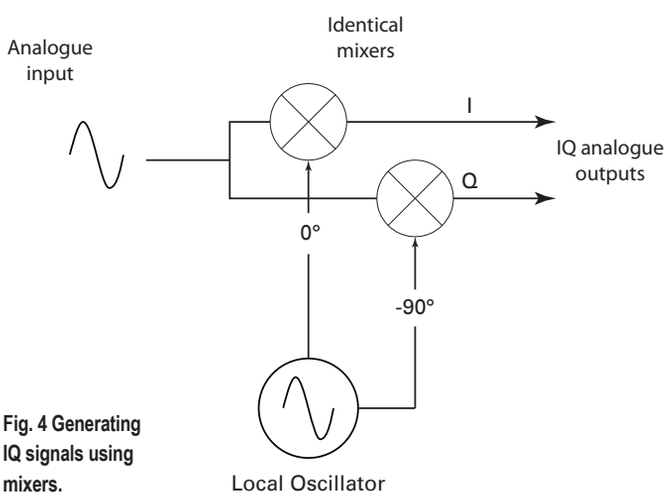


Fig. 4 Generating IQ signals using mixers.

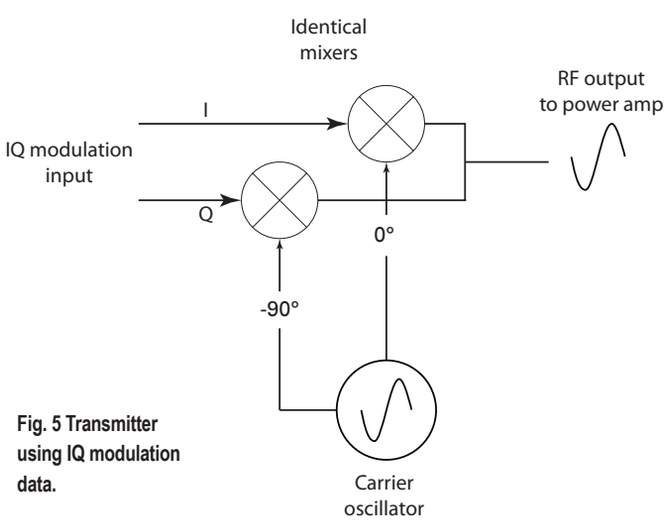


Fig. 5 Transmitter using IQ modulation data.

Mike Richards G3WNC's Data Modes

SDR Introduction Continued

Fast Fourier Transforms

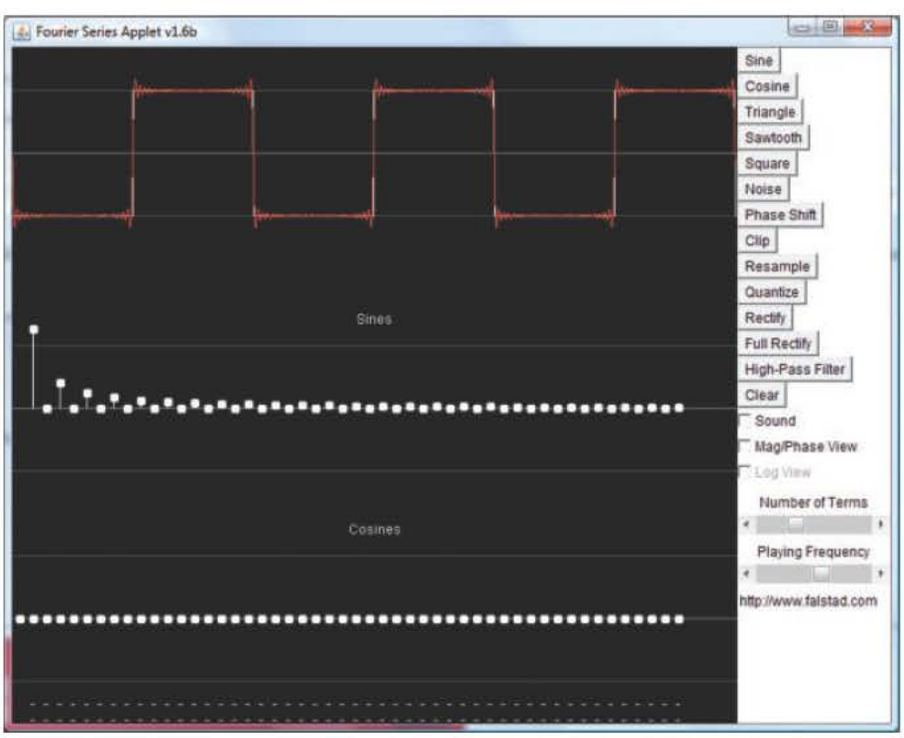
Having covered most of the basic theory In last month's *Data Modes* column, Mike Richards G4WNC, looks at Jean Fourier's legacy and the SDR techniques themselves.

Welcome to this month's *Data Modes* (*DM*) where I'm continuing to look at Software Defined Radio (SDR). Those of you with an interest in software defined radio (SDR) will doubtless have heard the Fast Fourier Transforms (FFT) term mentioned countless times and been baffled by it!

Hopefully, via *DM* I can give you an insight into what these are and why they're so important to SDR. The concepts and mathematics of Fourier transforms were first shown in a paper by French mathematician Jean Fourier way back in 1822.

Jean Fourier's work formed part of a paper on heat flow, where he observed that some functions can be represented by a series of sines at multiples of the function. Putting that into simpler terms, this means that a complex waveform can be created from a number of sine waves. If you would like to see a practical demonstration of a complex waveform and it's components, take a look at the following web applet to be found at: **www.falstad.com/fourier/**

I've shown a screen grab in **Fig. 1**. Choose a square or a triangle waveform and then play around with the levels of the component sine waves to see the results. And the quotation *"One picture paints a thousand words"* il prove itself!



F g. 1: Java applet i lustrat ng the Fourier series.

The examples on the website show very clearly how you can construct a complex waveform by combining sine waves in exactly the right proportion. This process is reversible, so you can extract the sine wave components of a complex waveform by examining the Fourier series. This is known as a discrete Fourier transform but is generally too slow for communications work.

The solution to the lack of speed of discrete Fourier transforms, is to use a different algorithm known as a Fast Fourier Transform (FFT) The use of FFTs is now standardised throughout the signal processing business and many designers use the well proven FFT routines that are built into the Intel IPP software library.

Why Are FFTs Important?

If you recall from last month's *DM* the information coming from the digitisation process in the SDR is a stream of in-phase and quadrature (IQ) data that represents the signal we want to receive One of the first useful things would be a spectrum display so we have a visual representation of our signal This is an ideal task for an FFT as we can use this routine to split the incoming complex signal into its separate frequency components and show them on the display.

The spectrum display routine works by splitting each signal sample into a number of very narrow-band 'bins' and then measuring the contents of each bin These measurements can be used to feed the spectrum display and another type of display called a 'waterfall' display which represents the spectrum display over a period of time.

The FFT bins are rather like banks of very narrow band filters and are often used to create the SDR receiver filters. By using reverse FFTs it's possible to convert a bandwidth shape into a real high performance filter. You can play with a digital filter in another web applet to be found at **www.falstad com/ dfilter/** I've shown a screengrab from the website in **Fig. 2** That's enough theory for this time!

Practical SDR

Now we'll get on with the practicalities of SDR. The first thing you'll notice

about SDR systems is that the hardware itself has few, if any, controls. This is because all the control is handled in the specialised SDR software.

The good news is that most of the software is free and works with an assortment of receivers. This means you can experiment to find the software that suits your needs and upgrade to add more features.

Software control is the big advantage of SDR over conventional receiver technology. Whereas many conventional receivers present the listener with a single frequency, all SDRs provide a panoramic view of a band of frequencies and often use 'mouse clicks' to tune. This is a very different approach to the familiar silky smooth rotation of a tuning knob so, as a technique it can take some getting used to by most of us!

Direct Conversion SDR

One of the simplest ways to create an SDR receiver is to use the direct conversion (DC) technique. This is where the incoming r.f. signal is mixed with a local oscillator of the same frequency to produce what's known as a baseband signal. You could quite rightly observe that this is the same technique as a normal c.w. or s.s.b. direct conversion receiver.

However, there is an important difference as we need to produce IQ outputs as opposed to a single audio signal. To do this, we apply the r.f. signal to two identical mixers fed by the same local oscillator, except that the local oscillator feed to the Q mixer is delayed by 90° (see **Fig. 3**).

The output from the DC mixer combination is a base-band IQ signal. By base-band I mean a signal the signal originates from 0Hz – so let's next look at an example. Suppose we want to receive a signal of 14.2MHz. The local oscillator would be set to 14.2MHz and the output from the mixer would be the sum and difference signals. So a 100Hz tone on the 14.2MHz carrier would appear at +100Hz (14.2001-14.2) and 28.4001MHz (14.2001+14.2).

While it's possible to create a DC IQ signal using conventional mixers, there's one system that is particularly effective and can be found in most direct conversion SDRs. This is the Tayloe switching mixer – or product detector – designed by **Dan Tayloe N7VE**.

The Tayloe product detector is extremely simple and features a conversion loss of less than 1dB. It also offers a very high 3rd-order intercept point. Because it's not a conventional mixer the detector only outputs the difference signal so, there is no sum signal to filter out. The frequency range can extend to at least 10GHz so you'll realise that the Tayloe mixer is an impressive design.

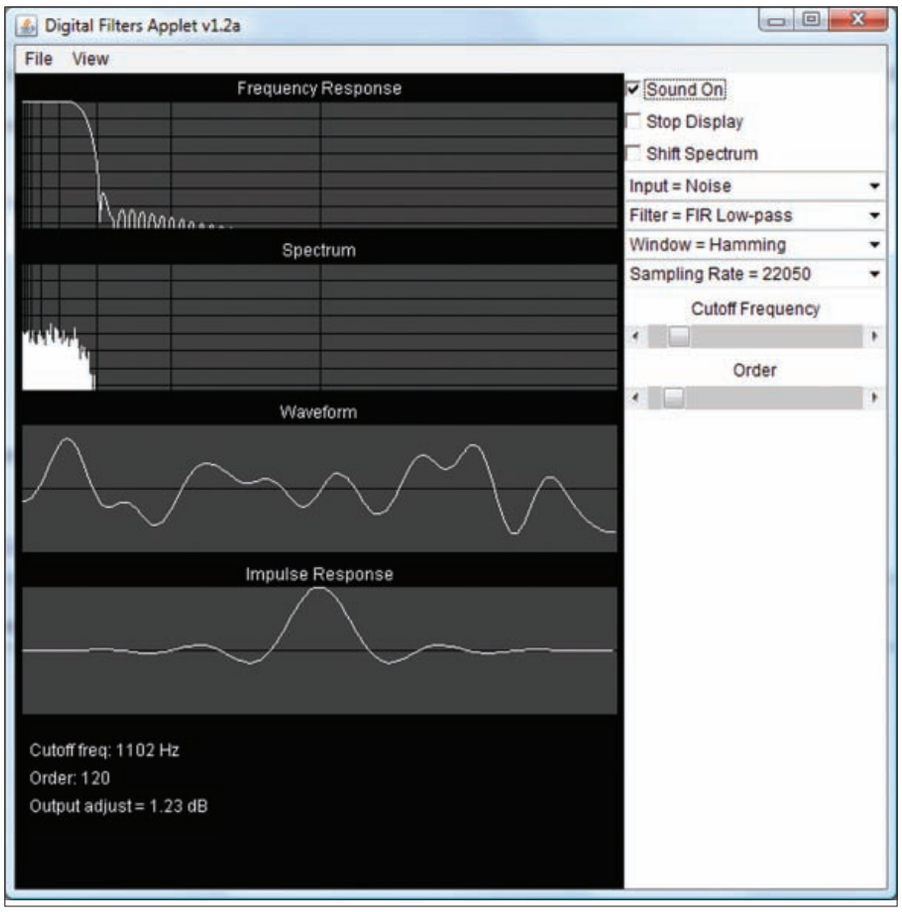


Fig. 2: Java applet where you can experiment with a digital filter.

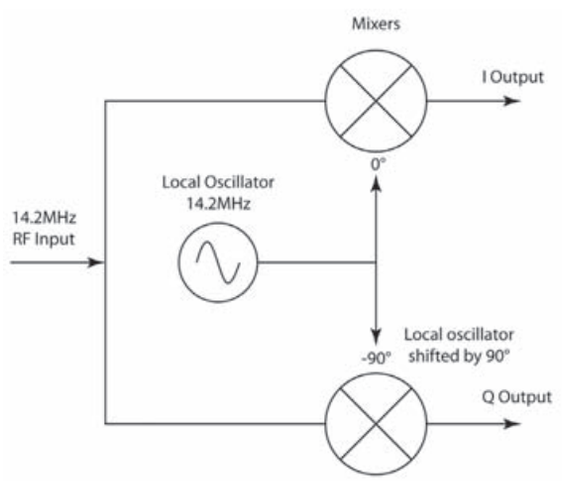


Fig. 3: Basic direct conversion SDR configuration.

The theory of operation of a Tayloe mixer is also remarkably simple and the best way to understand it is to think of the detector as a four-pole rotary switch with the r.f. signal connected to the wiper. The switch is then set to complete one revolution for each cycle of the incoming r.f. signal.

Each pole of the switch has a capacitor connected to it, and this is charged by the incoming signal but only whilst the rotating wiper is in contact. As a result, the capacitors connected to the four poles contain a sample of the signal at 0°, 90°, 180° and 270°.

By adding the 0° and 180° connections we can create the 'I' signal and by combining the 90° and 270° samples we create the 'Q' signal. I've shown an illustration of the process in **Fig. 4**. Clearly, it's not practical to achieve this with a mechanical switch, so electronic switches are employed and many modern implementations use the 4066 quad bilateral c.m.o.s. analogue switch integrated circuit (i.c.).

In order to make the switches operate at the correct time, the local oscillator (l.o.) has to runs at four times the required carrier frequency and is applied to the switch via a 4-stage Johnson counter as shown in the timing diagram of **Fig. 5**. A Johnson counter is a specially configured shift register where the first l.o. cycle sets output A to 1 the second sets output B to 1 and so

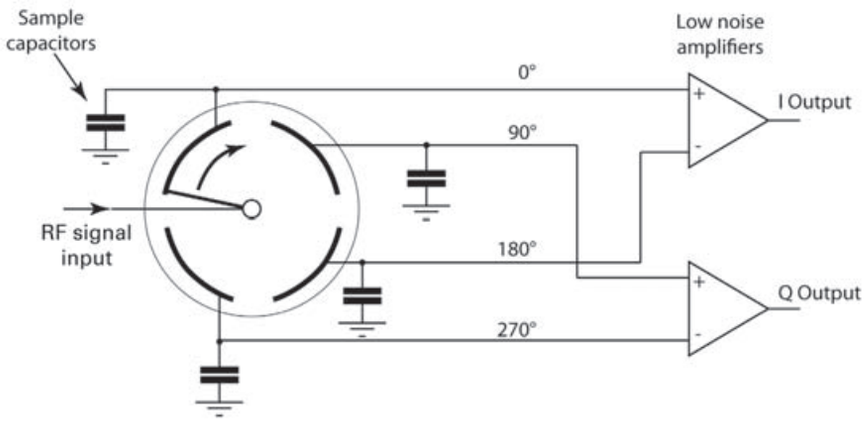


Fig. 4: Illustration of the ingenious Tayloe Product Detector.

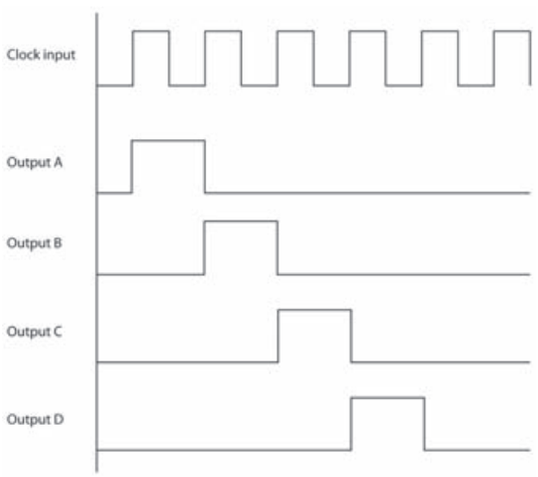


Fig. 5: Chart showing how a Johnson Counter generates control signals for the Tayloe switches.

on. Each output is a single pulse at one quarter of the l.o. signal.

The outputs of the Johnson counter are connected to the 4066 switch to control the sampling switches, directing the incoming r.f. signal. Finally, the outputs from the switches are combined and amplified by a pair of low noise operational amplifier i.c.s (opamps) to produce the required IQ output.

Note: Despite the use of digital circuitry to help process the r.f. signal, the IQ outputs are still analogue signals at this stage.

In simple designs these outputs can be sent to a computer sound card for digitising and processing by the SDR software. The only snag with this approach is that there are lots of different sound-cards out there so the results can vary from computer to computer. In some of the more modern SDR implementations a sound-card and USB driver are included on the board.

Sample Rate Confusion & DC Bandwidth

One area that often confuses people new to SDR, is the bandwidth of the DC receiver. So, I'll try and explain it here. Remember, the IQ signal from our DC receiver is analogue and has to be applied to a sound-card for digitisation.

All sound-cards have a range of sample rates that they can use. The basic rule (Nyquist) is that the sample rate has to be set at slightly more than twice the highest frequency that you want to process. For audio signal that means setting the sample rate at a minimum of 44kHz for Hi-Fi sound. However, as computers prefer to work in 8-bit multiples, 48kHz and multiples, such as 96kHz and 192kHz are often employed.

Getting back to our SDR example, if we apply our direct converted IQ signals (using 14.2MHz l.o.) to the left and right channels of the sound-card using a sample rate of 48kHz we could expect to be able to receive signals between 14.2MHz and 14.224MHz, i.e. up to 24kHz above 14.2MHz.

By manipulating the IQ signals it's also possible to receive the band that stretches 24kHz below 14.2MHz. This manipulation of the signals means the total bandwidth available is 48kHz wide centred on 14.2MHz. Within the SDR software, the IQ signals are further processed to minimise images between the upper and lower ranges and so effectively provide continuous tuning of the full 48kHz bandwidth.

In operation, the software can often be 'tweaked' for improvement, as the image rejection works best when both I and Q signals have absolutely identical level and phase characteristics. One common side effect of using sound-cards for IQ processing is a central spur in the tuning display. This is due to a combination of noise and the sound-card filters that cut off signals below about 10Hz. These filters can cause phase and amplitude differences between the I and Q channels that result in a visible spur.

In the better systems this spur can be reduced to levels close to the noise floor on the h.f. bands. So to summarise, you can expect the tunable bandwidth of a direct conversion SDR receiver to be approximately the same as the sound-card's sample rate.

Local Oscillators For DC Receivers

For single band operation the simplest way to provide the local oscillator is to use a crystal oscillator. This is the approach used by the very successful SoftRock and Soft66 series of SDR systems. The downside of this approach is that you have to choose your crystal carefully so that the limited receiver bandwidth covers your area of interest.

A better solution for modern designs is to make use of the excellent programmable crystal oscillator by Silicon Labs, the Si570. This is a pretty incredible, self-contained, 8-pin chip that can generate accurate frequencies in 1Hz steps between 10MHz and 945MHz!

Programming is handled over a simple two-wire interface and the output is a square wave – just what we need to drive the Johnson counter in a DC SDR. Prices for the Si570 start at around £14 for the 200MHz version – so they are a very economical solution.

Spectrum Digitisation

The development of high speed analogue to digital converters has meant that it is now possible to digitise the entire h.f. spectrum in one hit! However, as I explained in last month's *Data Modes* the result is a very high speed data stream that has to be processed in a specialist Field Programmable Gate Array (FPGA).

As a result, the FGPA is generally confined to commercial implementations of SDR with the WiNRADiO range being a good example. Some of the more advanced systems use a combination of superhet front ends and wide-band ADCs to deliver incredible performance.

I've run out of space this time, but next month, I'll continue with a look at SDR software and how to use it. But in the mean time, if you'd like to try out this method, have a look at the two receiver kits assembled by **Phil Chiotti G3XBZ** that appear elsewhere in this magazine. ●