Coherent ADC

A 1kHz sampling A/D converter for low rate LF data modes and frequency stability testing

Introduction

When the fully-coherent slow datamode EbNaut [1] started to appear on the LF bands a few years ago, users had to be able to receive and record off-air signals that would remain stable in phase to less than a quarter of a cycle over a duration of an hour or more. At 137kHz, a quarter of a cycle over two hours requires a frequency accurate to 34μHz with a long term accuracy better than 2.5x10^{10}. The latter requirement is easily met by locking the receiver and all frequency converters to a common-or-garden GPS-disciplined oscillator or a rubidium standard. The absolute accuracy can be sorted out in software using FFT techniques on a wider bandwidth. The weak point is capturing and storing the signal – typically as an audio tone.

A PC soundcard seems the obvious choice to record such waveforms out of a receiver, but now everything falls apart. The sampling clock in most soundcards is usually defined by a low-cost crystal oscillator, with a typical accuracy of perhaps 20ppm (parts-per-million), which translates directly to a sampling rate error. An error of 20ppm on a 1kHz tone is 0.02Hz – that's 20,000μHz – and while this would normally go unnoticed by most users, here it makes a complete mockery of the micro-hertz accuracy needed for EbNaut.

The users of EbNaut devised a workaround to cope with this issue. They used a customised version of the SpecLab software [2] that phase-locked, purely in software, the one pulse-per-second (PPS) signal from a GPS receiver fed into one channel of a stereo soundcard. The frequency-converted receive waveform went to the other channel. SpecLab could now generate

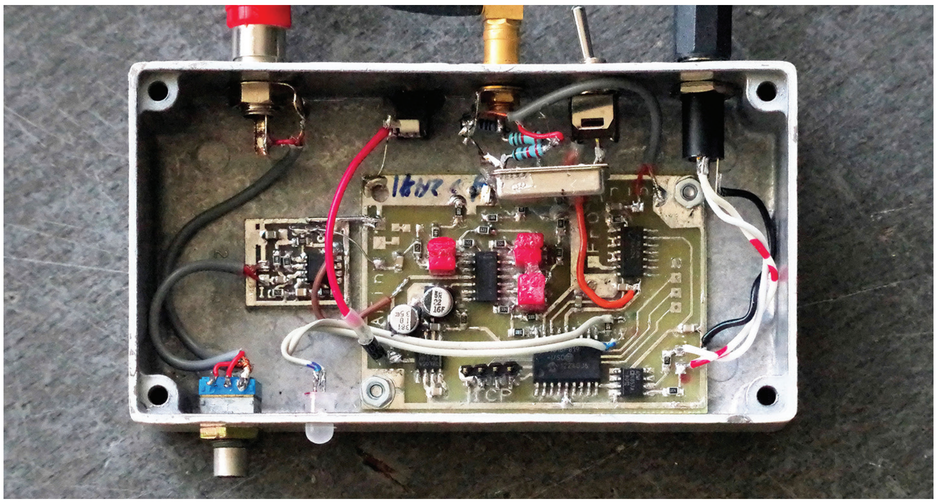
PHOTO 1: 1kHz sampling hardware. There's an extra opamp gain stage in this version (on the small PCB). An internal, switchable, 10MHz oscillator has also been added for non-critical tests when an external reference is not available.

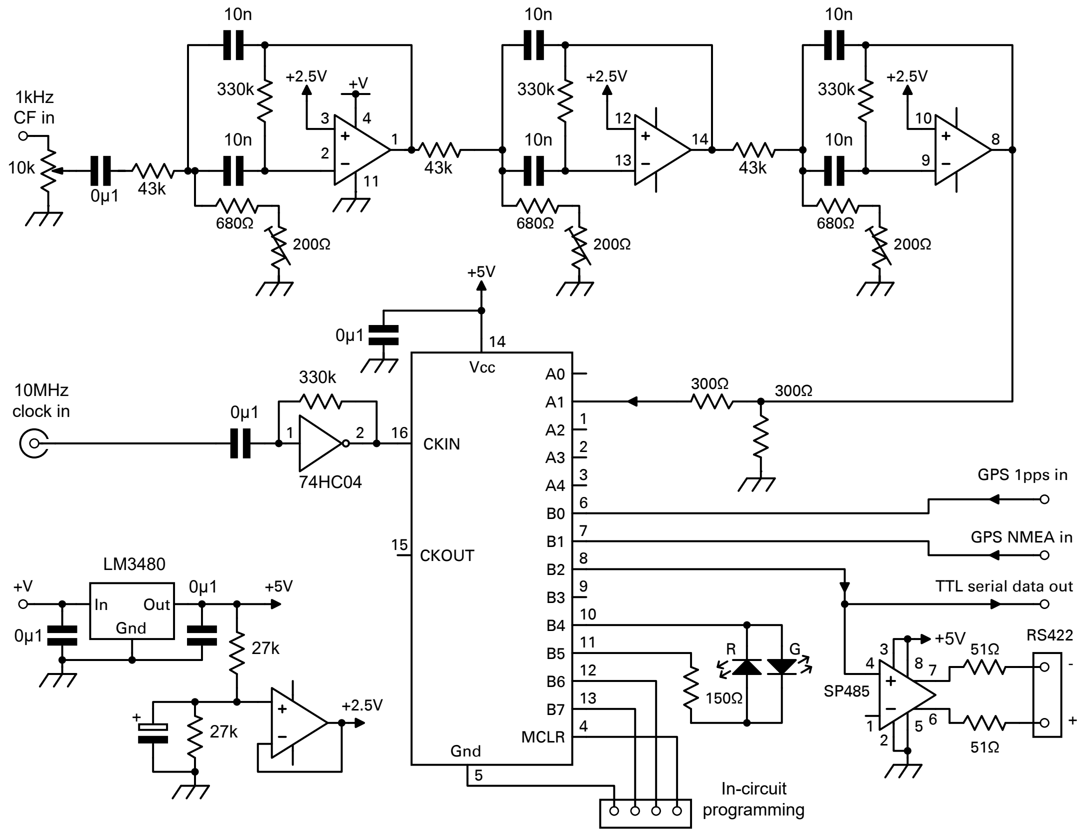

FIGURE 1: Circuit diagram of a 1kHz digitiser for high stability coherent communications and testing.

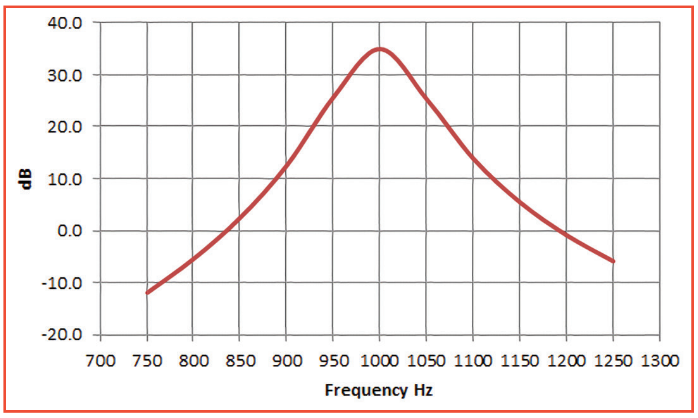

FIGURE 2: Frequency response of the input bandpass filter.

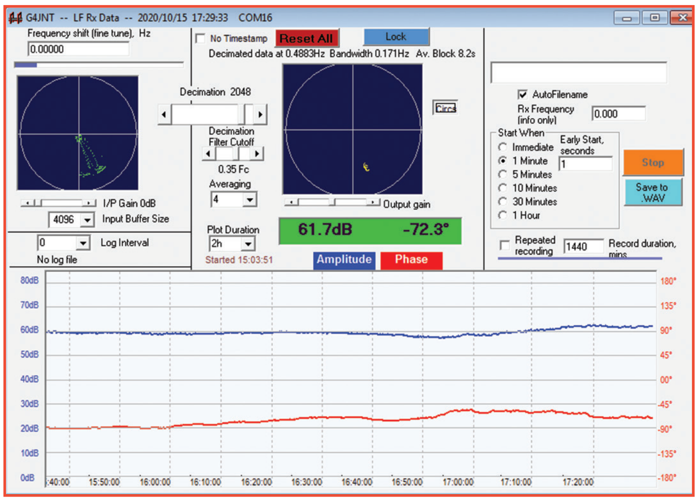

FIGURE 3: Off-air reception of the DCF77 time signal on 77.5kHz over the period of sunset. Propagation-induced phase and amplitude shift is clearly visible.

a pseudo sampling clock locked to GPS and then coherently downconvert the audio input to a narrow baseband I/Q signal at a sampling rate in the region of a few Hz. Another novel quirk was added that allowed the user to timestamp each recording; in addition to recording the PPS signal on one soundcard channel, the NMEA data stream from the GPS receiver could be fed in as well. This somewhat unorthodox route required a special version of SpecLab to decode the stored digital stream as well as separate out the 1 PPS bits and do the phase locking – but it did allow a number of users to use the EbNaut suite without having to build any additional hardware.

The resulting I/Q samples were stored as floating point values in a customised file that, somewhat amusingly, was given the .WAV file extension and even – sort of – followed the rules for such files with regard to metadata and headers. The EbNaut receive software worked on these pre-stored low data rate samples to extract the really, really, ultra-weak signals buried in there. The time stamp was reflected in the file name – all automatically generated. I shall refer to these as 'pseudo-wav files'.

Alternative solution

I never got on with SpecLab (a purely personal issue, mainly down to its complexity and 'feel'; other people love it for its flexibility and utility) so I wondered what other alternatives were possible to generate a fully compatible pseudo-wav file. Back in the early days of 73kHz experimentation in the late 1990s, Peter, G3PLX had suggested to me a way that a PIC processor with internal A/D converter could be used to simultaneously digitise and downconvert a narrow received waveform centred on 1kHz. The output would be a 1kHz sampled baseband I/Q data stream sent to a PC using the serial or COM port. The only proviso was that the input frequency band must be filtered to keep it within the range 750 – 1250Hz to avoid alias products. This filtering requirement was easily met by using a 300Hz CW filter in the receiver and setting the BFO tone to 1kHz. Sampling rate and stability is now purely a function of the processor clock, which would be a quality crystal oscillator or master frequency standard. There was no need to rely on uncertain PC timing; the PC just takes serial words when they arrive.

In those days the only suitable PIC processor was the 16C71, which had an 8-bit A/D converter and a maximum processor clock speed of 5MHz (lower cost versions only offered a 1MHz clock), but either was more than adequate for the scheme Peter suggested. It works like this:

- In a software loop, digitise the input waveform at 4kHz, thus generating four samples for every one sent at 1kHz. Label these four samples S1, S2, S3 and S4, whilst moving the previous value of S4 to variable S4_Last
- Calculate two values, I = S1 + S2 – S3 – S4 and Q = S1 – S2 – S3 + S4_Last

Since each 4kHz sample gives an 8-bit result from the A/D conversion, adding these means the resulting I/Q pair each take on a 10-bit value. What we have done is to effectively multiply each block of four samples by one cycle of a quadrature square wave at 1kHz. Taking the signs of the samples in the sums above as ±, the following illustration makes it a bit clearer, showing four cycles of the effective quadrature local oscillator (LO):

```
++--++--++--++--    and
+--++--++--++--+
```

Multiplying an audio or RF input by a digitised I/Q LO to get an I/Q baseband output is a standard technique in any SDR to bring the centre frequency down to DC. Because we are generating the result at a 1kHz sampling rate means, from Nyquist, our maximum bandwidth can only be half this, or 500Hz. Hence the 750Hz – 1250Hz limit. Reducing four 8-bit samples to one single 10-bit one at a quarter the original sampling rate gives another 6dB of dynamic range over that available from the raw digitisation, a direct result of down-sampling from 4kHz to 1kHz.

In those days internal COM ports were standard in PCs. The highest speed these could run at was 115200 baud – quite adequate to send several bytes of data 1000 times per second to a PC for further processing. The total frame length to send four bytes using stop-start signalling at 115200 baud is less than 400μs, so there was plenty of leeway at 1kHz sampling. 20 bits needed to be transmitted in a frame that would allow for 32, so each pair was split into five bits, allocated to each of the four bytes. The remaining three bits in each byte were set to different fixed header values so the receiving software could quickly identify each one and recombine them properly.

I built a digitiser using this technique and used it for a number of early DSP and coherent receiving experiments, but it was subsequently put on one side and forgotten when soundcards became ubiquitous.

An updated version

The need for a highly stable converter with time stamping for EbNaut was an ideal opportunity to bring this design up to date. Improved PIC devices now included 10-bit A/D converters on the chip and higher processor clock rates. 10-bit A/D conversion meant the I/Q output samples would now have 12-bit resolution, giving a 12dB dynamic range improvement over the old design. In spite of there now being a whole range of modern dsPICs and advanced 16-bit devices to choose from, all the requirements for an upgraded digitiser could be met with a basic 16F819 PIC processor. This is itself a bit long in the tooth now, but perfectly adequate; it's an 18 pin device costing less than £2 (of which I already had more than a few), with plenty of existing code written to just drop in and go.

Internal COM ports on PCs are rare nowadays, but USB serial interfaces such as the FTDI-Chip FT232 device are commonplace and can offer much

Andy Talbot, G4JNT
andy.g4jnt@gmail.com

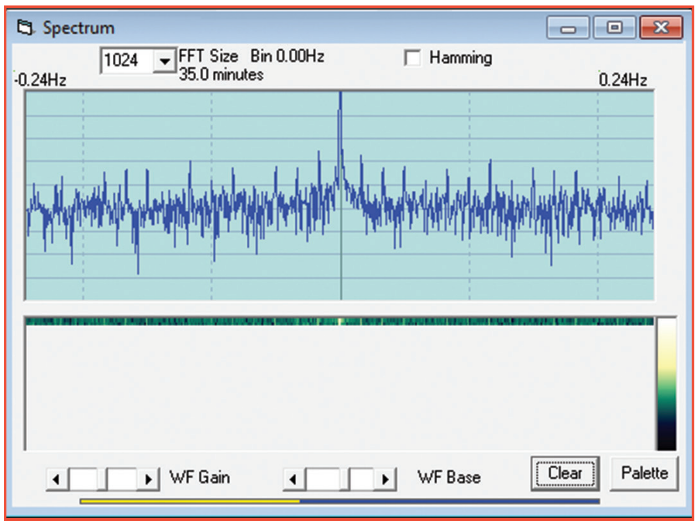

FIGURE 4: Narrowband frequency spectrum from the DCF77 transmitter showing spectral components due to the one-minute timeframe.

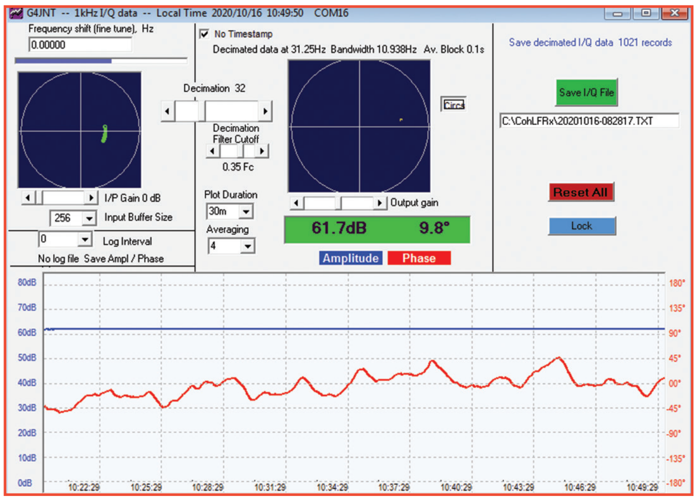
G4JNT -- 1kHz I/Q data -- Local Time 2020/10/16 10:49:50 COM16
Frequency shift (fine tune), Hz
0.00000
No Timestamp
Decimated data at 31.25Hz Bandwidth 10.938Hz Av. Block 0.1s
Save decimated I/Q data 1021 records
Decimation 32
Circ
Save I/Q File
C:\CohLFRx\20201016-082817.TXT
Decimation Filter Cutoff
0.35 Fc
I/P Gain 0 dB
256 Input Buffer Size
0 Log Interval
No log file Save Ampl / Phase
Plot Duration
30m
Averaging
4
Output gain
61.7dB
9.8°
Reset All
Lock
Amplitude
Phase

FIGURE 5: Comparison of the 10MHz from a commercial GPS Disciplined oscillator with that derived from a caesium beam frequency standard.

higher signalling speeds than the 115200 baud limit on the old RS232 ones. There is also no need for the messy ±12V signalling of RS232. The interface to the PC can be, to all intents and purposes, a USB one. Adopting the 16F819 did mean, though, there was no internal UART available to generate output data. So this would have to be generated by bit-banging in the same way as was done for the twenty-year-earlier design. It also meant a baud rate significantly greater than 115200 would not be possible; but that is more than fast enough.

As well as digitising the 1kHz-centred input and forming the I/Q data values, the new version also had to take a 1 PPS signal, together with NMEA time and date from a GPS receiver and merge this into the output stream along with the samples. All this meant some ingenuity would be needed in the format of the output data stream to ensure data slippages and errors in the interface could be easily recovered. The processor clock input would come from a 10MHz master frequency reference oscillator, resulting in a 2.5MHz internal processor clock rate; significantly faster than the original version. so allowing more processing to be done on each sample if needed.

Comms protocol and time tagging

The IQ Data appears every 1ms as two twelve bit numbers whose individual bits are labelled here:

`IIIIIiiiiiii` and `QQQQQqqqqqqq`

These are split into four 8-bit bytes, each pair containing respectively 7 and 5 bits of the I or Q value. Header bits are added to make up a complete byte in each case, and these transmitted on the serial interface sent in this format and order:

`0iiiiiii 100IIIII 0qqqqqqq 101QQQQQ`

The receiving software knows that any byte with a leading 0 contains the lower significant bits that will be appended to the next one(s). A byte starting '100' is defined as the high order bits of the I data, is merged with the previous one and forms that value. The same applies for the Q data when a byte beginning with the pattern '101' appears. Thus, if any data slippage or fail should occur, there is enough information in each frame of four bytes to recover immediately.

Time tagging is added to this in the form of extra bytes transmitted every second, ie once in every 1000 samples. As the PIC code already contains a timing function controlled from a high accuracy reference clock, it would be frivolous to waste time and effort continuously reading the GPS data stream every second. Instead, the GPS date and time is read once at turn-on and stored; this value is then updated every second in the PIC firmware by a clock / calendar routine synchronised at the start to the 1PPS signal from the GPS. Once internal timekeeping is synchronised the GPS receiver becomes redundant and could, if desired, be turned off to save power.

Every second, synchronised to the UTC seconds epoch, the binary values for the time and date are transmitted as additional bytes just after the Q data has completed. In total, a block of six additional bytes needs to be sent every second for the full epoch, but these have to be spread out to keep the total number of bytes within any 1ms frame down to 8 or fewer.

The first part of the additional data contains binary values for UTC seconds, minutes (shown as 'n'), hours and the day of the month, coded as:

`00ssssss 00nnnnnn 000hhhhh 111ddddd`

Once again, a byte beginning with a '0' has no meaning on its own. The '111' header on the final byte of the four indicates to the receiving software this is the first part of a time stamp and has to be combined with the preceding three. This block of four is sent immediately after the UTC seconds epoch. 32ms later two more bytes are appended at the end of a frame of Q data, containing binary values for the short-year and month in the format:

`0yyyyyyy 1100mmmm`

The '110' header is the unique label for this frame. So now the receiving software can determine each sample to a time resolution of 1ms from UTC / GPS time and can get a timestamp update within one second of acquiring the data stream. Any sample with the first part of the time tag (header '110') sits at the 0.000 seconds of the time/date epoch coded within the next few blocks. Knowing this, the exact timing of each sample can be determined to 1ms precision.

Hardware

Figure 1 shows a circuit diagram of the complete digitiser shown in **Photo 1**. Apart from the PIC doing all the work, there is a bandpass filter to limit the audio bandwidth going into the A/D converter and a driver for RS422/485 twisted pair, described later. A dual colour LED driven from the PIC allows the input level to be set correctly, showing when overload or A/D clipping is about to occur. It also shows if the reference is not connected at turn on by lack of illumination when power is applied. A flash pattern indicates that no GPS data or PPS is being received. NMEA data and the 1PPS come from a separate GPS/GNSS module, not shown, which can be either external to this unit or built into the same hardware.

A three-stage opamp bandpass filter limits the amplitude of frequencies outside the alias band. This also provides a useful amount of audio gain, allowing input signals of a few hundred mV to get to full scale on the ADC. **Figure 2** shows the simulated response of this filter with all three stages tuned to exactly the same frequency. As the unit is designed for low bandwidth modes of perhaps just a few tens of Hz, the response does not have to be too carefully tailored and the three stages can simply be tweaked for something approaching what is needed. Stagger-tuning will widen the response if desired, at the expense of lower out-of-band rejection. As can be seen from the simulation, rejection at the edge of the alias band is around 40dB worst case, increasing monotonically beyond this. Since, in practice, only frequencies a few tens of hertz either side of 1kHz centre will be of interest for experiments or modes using this interface, alias products falling into this wanted region around 1kHz would have to be at 2kHz, 3kHz etc, where attenuation is considerably higher. If the audio input is derived from a receiver with a narrow CW filter, the opamps can be dispensed with and audio applied directly to the PIC input, biased to centre rail, via a gain-stage if necessary. The nature of the summation of the four A/D values means that any DC offset is automatically removed on the final value, so the mid rail voltage does not have to be trimmed too precisely.

Computer interfacing

The RS232 interface used on the original version with its need for ± voltage signalling is not being considered here. The simplest solution now is to connect the data output pin from the PIC directly to the serial input of an FT232R chip, or whatever USB COM port device is chosen. A direct connection, however, is not the best solution. USB connections are not all that electrically quiet and frequently conduct unwanted RF signals from the PC chassis to the target, either differentially along its unbalanced 5V DC supply, or by common mode injection.

A better solution, shown here, is to adopt differential signalling on twisted pair using RS422/485 levels. The USB connection to the PC (now a RS485 USB module) is plugged directly into the PC, keeping its interference at home. Having no common connection or ground, the twisted pair cannot introduce ground loops and is immune to common mode interference in either direction. Furthermore, it is easy to add filtering to reject common mode RF signals by wrapping several turns of the thin twisted pair around a suitable ferrite toroid.

For an even more interference-free connection, an optical fibre based solution could be adopted, or an RF one using low cost 432MHz modules. Such interfaces could be considered if the digitiser were used as a remote receiver in a quiet location, well-away from local electrical noise sources.

PC software

With a customised interface such as this, bespoke software is needed to receive and decode the serial data stream. COM ports are well supported in all computer languages and are usually easier to drive and set up than any other I/O mechanism. Software at **[3]** written in VB6 contains a package that will read the data stream and extract the 1kHz I/Q samples and time stamp. The signal is filtered and decimated by a user defined value between 4 and 4096 to give an I/Q steam at a sampling rate that can go down to 0.244Hz. A frequency offset (any arbitrary floating point value) can be added to the signal before decimation to allow, for example, a receiver limited to 1Hz tuning steps or the error introduced by a low resolution DDS used as a receiver's LO to be corrected. The resulting slow rate I/Q data can then be saved as time-scheduled pseudo-wav files, automatically named from the time stamp, in the format required by the EbNaut receive software. Vector displays of the raw input and the resampled waveforms are presented, along with a time averaged plot of amplitude and phase. Phase and amplitude readings at regular intervals can be saved to a log file. A spectrum and waterfall of the decimated waveform appear in a separate window. This can be set for an FFT size up to 4096 so when used at the maximum decimation of the input, could show a spectrum display to a resolution of 59.6μHz (equivalent to 4.66 hours per plot).

As an example, **Figure 3** and **Figure 4** show the 77.5kHz DCF77 time signal transmitted from Mainflingen in Germany, 735km due East from me. The capture period covers the transition from daylight to just after sunset, which at the time of this plot occurred at 1712UTC here and 1630UTC at the transmitter. The signal was captured off-air, decimated to 0.488Hz sampling rate and filtered to ±0.17Hz, sufficient to just hide its one-second pulsing. The change in propagation during the day-night transition is clearly visible in the plots of amplitude and phase after 1600UTC.

The spectrum plot in Figure 4 is at 0.00048Hz resolution (35 minutes per FFT) and clearly shows the structure in the MSF signal caused by repeated bit patterns in the timecode common to each one-minute interval. The resulting spectral components are at 1/60s = 0.017Hz apart; six per graticule mark.

Other uses

Apart from viewing narrowband LF signals this digitiser has uses in the laboratory environment, especially in the measurement of variations of frequency and stability of oscillators and references. While a frequency counter with a long-enough gate period will show what happens in the long term, the way a frequency reference varies over a few seconds or minutes can be more important if it is to be used to derive a local oscillator. An alternative version of the software that was originally written for EbNaut on LF was generated, geared more towards use for test and measurement. It allows for, and can ignore, a signal that does not have valid time stamp information and file saving has been altered to make it more test equipment friendly. As an example, a 10MHz output from a GPS disciplined oscillator needs to be compared with a reference caesium source and we are interested in its stability over a few seconds or tens of seconds. This is expected to be within a few parts per billion. The obvious way to compare the two would be to apply both 10MHz signals to a mixer, look at the DC component from the IF port and see how it varies over time. At a few parts-per-billion frequency variation, any significant drift over the short term would be small and vary only slowly. Digitising this for subsequent analysis would be beset with DC zero / shift problems, as well as the 0/180° ambiguity within any single-ended mixer, meaning the direction of relative frequency shift is unknown.

By coherently converting one of the signals that are to be compared to a new frequency 1kHz higher (or lower) than the other, this DC offset and phase ambiguity can be resolved. The frequency conversion can be done with a direct digital synthesiser (DDS) with integral clock multiplier. To maintain sufficient accuracy a DDS with a 48-bit accumulator is needed. Assuming a clock multiplication factor of 20 to get a 200MHz DDS clock, any frequency generated by a 48-bit device such as the AD9852 will have a maximum uncertainty of 200MHz / 248 = 0.7μHz (one cycle every 16 days). If a 32-bit DDS like the AD9851 were to be used, clocked at 60MHz from its internal multiplier, the uncertainty could be as high as 0.014Hz (although this is completely deterministic and could be catered for **[4]**). Applying both these signals to a mixer, the output at 1kHz contains all the information needed to extract stability data, with no DC offset and no phase ambiguity. The 1kHz digitiser and supporting software described does the rest. The plot shown in **Figure 5** shows the result over a 30-minute period for the output from the GPSDO, converted to 10.001MHz as described and mixed with the output from a caesium frequency standard at 10MHz.

Although the average phase is stable over the 30-minute plot, it does show some rapid excursions over short periods. One example, at 10:45, shows around 45° of phase shift over a period of something like 30 seconds. This equates to a short term frequency shift of 45°/360° / 30s = 0.004Hz. At 10MHz this corresponds to a frequency error of 0.4 parts per billion (PPB) for this time interval, a short term wander typical of that seen in many commercial GPSDOs. If required, the decimated waveform can be saved and subsequently used in a calculation of other parameters such as the Allen Variance.

Websearch

[1] EbNaut coherent data mode for low frequencies: http://abelian.org/ebnaut/

[2] SpecLab spectral analysis toolkit: https://www.qsl.net/dl4yhf/spectra1.html

[3] LF receiver / EbNaut control software: www.g4jnt.com then follow the links for 'DSP Software'

[4] If a 32-bit DDS were to be used as the reference frequency converter, the setting uncertainty can be determined precisely, using the register values sent to the DDS to calculate the exact frequency generated. The error from the wanted value can then be entered into the fine tune box.