A GPS-Locked Frequency Source for LF

by Andy Talbot, G4JNT *

The low-cost GPS-locked frequency source described here is designed specifically for low data-rate signalling on the LF bands. Phase noise and short-term frequency stability preclude its general use at HF or above; for these frequencies, other, more conventional, frequency standards are preferable, based around crystal oscillators of inherently higher stability.

For locking a frequency source to the 1 pulse per second (PPS) signal from a GPS receiver, a conventional synthesiser or phase-locked loop (PLL) approach is impractical. The only reference frequency is at 1Hz, so a PLL would need to have a bandwidth considerably less than this, meaning that the loop would only lock up after many tens of minutes or hours. Furthermore, the stability of the basic oscillator element - the Voltage Controlled Oscillator or VCO - would have to be such that is did not drift by more than 1Hz during a time comparable with the loop bandwidth, otherwise lock could never be achieved. A high-stability VCO is needed and, for a design operating at a few megahertz, the required stability would be less than 1 part per million, (PPM). Voltage-controlled crystal oscillators (VCXOs) to this accuracy, also usually temperature-controlled, are available, but are expensive and only occasionally appear on the second-hand surplus market as part of test equipment. Brooks Sheera, W5OJM, has designed an excellent GPS-locked high-stability frequency source using such an oscillator, which was described in *QST*, July 1998. This does require several hours to achieve lock up and is the sort of precision test equipment that should be left running continuously and not be turned off every day.

However, where short term stability is of less importance, another technique can be used. The source described here is based roughly on the old 'Huff and Puff' stabiliser, first described by Klaus Spaargaren, PA0KSB, in 1973. This design stabilised a 'good' LC-tuned VCO to give it crystal oscillator stability, in steps of 10Hz. In essence, the design used a 1-bit frequency counter - a single flip-flop clocked by pulses divided down to 10Hz from a crystal oscillator. Depending on whether the VCO frequency was high or low of the appropriate 10Hz step, the output of the flip-flop ends up at a 1 or 0 and is applied via a low pass filter to a varicap on the LC oscillator, which is ramped up or down to maintain the long-term frequency. The output would then be continuously hunting just above or just below the nearest 10Hz step.

The problem with this design was when the LC oscillator drifted by more than one step, ie >10Hz; the locked frequency would then slowly move in jumps of 10Hz. There were numerous improvements to the original design in the following years, several adding more bits and resolution to the frequency counter allowing a higher inherent drift, but Huff and Puff gradually died out as frequency synthesisers took over.

CIRCUIT DESCRIPTION

THIS DESIGN APPLIES the Huff and Puff technique to a simple un-ovened VCXO to maintain a source frequency that hunts either side of a specified 1Hz multiple, but uses an extended 8-bit frequency counter to avoid the possibility of the locked frequency drifting in jumps of 1Hz. The VCXO operates at 4.194304MHz (2^{22}Hz), for reasons that will be described later, but any frequency that is a multiple of 1Hz may be employed; it is also easier to understand the concept if a frequency that is an exact multiple of 256Hz is initially chosen. Refer to the circuit diagram, **Fig 1**.

A straightforward crystal oscillator is built with CMOS gates, with a varicap connected across the crystal to shift its frequency either side of nominal by a few tens of hertz. Choose the varicap and coupling capacitors to ensure that it is not possible to pull the frequency more than 128Hz over the full input voltage swing of 0 - 5V. This oscillator drives a synchronous 8-bit counter made from a pair of 74HC161 chips, the eight outputs from which are connected to a 74HC374, 8-bit D-type latch. This is triggered by the rising edge of the 1PPS (pulse per second) signal from a GPS receiver so that, each second, the output of the latch contains the lowest significant eight bits of the count. This counter overflows many times for each counting interval but, for any frequency that is an *exact* multiple of 256Hz, the counter will overflow to the same point each time, and the count latched by each seconds pulse will then be a constant number. If the frequency is not a multiple of 256Hz, but is still an exact 1Hz multiple, the count will change from one second to the next in a predictable manner. For example, a 5MHz signal would give successive counts (assuming a start at zero) of 00, 64, 128, 192 before the sequence repeats, which is completely predictable. However, for this description, we will continue with a frequency that is a multiple of 256Hz, so the ideal count stays a constant. If the frequency departs slightly from its correct value, the residual count will steadily increase or decrease by the magnitude of the frequency error and, if not corrected, the counter will eventually wrap round.

What we now do, is to read the count every second, then drive a charge pump circuit in a direction such as to correct the frequency. Software in a 16F84 PIC processor, using the trigger pulse from the GPS, reads the latched counter value and calculates an error value from the expected count every second. For multiples of 256Hz, this merely involves subtracting a constant, which can conveniently be a value of 128 - half the counter length. If other frequencies are wanted, the 'correct' value will increment by the frequency, modulo 256, every second. The sign and magnitude of the error value obtained is then a measure of the phase (and hence frequency) error and is used as follows :

Port A3 on the PIC is normally maintained at a high impedance, by setting it as an input. Every second, except for the single case where the error value is zero, this port is set as an output and strobed high or low, depending on the sign of the frequency / phase error. The duration of the strobe pulse is proportional to the magnitude of the error so that an error count of 128 leads to a strobe pulse of 640ms, either high or low, down to the minimum resolution of 5ms for an error count of one. This strobe pulse is then taken via an RC network to drive the varicap on the VCXO. The RC constants were determined by a combination of experience, trial and error, serendipity and luck to give an acceptable compromise between lock-up time, chirp and pull in range. To assist lock-up, the PIC software after turn-on applies a precise square-wave to the charge pump for 20 seconds, to force the capacitor to mid-voltage, and the frequency near to the nominal operating frequency. The loop then has only to make minor corrections to get the correct frequency and phase (the counter mid-point). The use of a three-state high-impedance port which is only briefly pulled high or low considerably reduces the residual chirp over that of a continuously changing low-impedance connection, as in the original PA0KSB concept.

* *15 Noble Road, Hedge End, Southampton SO30 0PH.
E-mail: actalbot@dstl.gov.uk*

As an additional aid to debugging, the error value is output from the PIC as an RS232 signal at 1200 baud. The format is a decimal number ranging from -128 to 127, followed by a space, for display on a standard ASCII terminal.

PERFORMANCE

WITH THE RC VALUES and varicap specified, and after the 20s initialisation period, the loop was fully-stabilised after running for about two minutes. The frequency as measured on a counter was exact, when using a 10s counter gating time. By monitoring the output on a vectorscope referenced to a high stability source, every second, on average, the phase would rapidly rotate by a value between 180 to 500° (0.5 to 1.5 cycles), then return more slowly. This appeared as a slight audible chirp of a few Hz. However, the mean phase stays constant, ie the vectorscope always averages out the clockwise and anticlockwise rotations to zero. The error value transmitted from the RS232 port usually stabilised at a small positive number rather than zero. The reason for this is not clear, but may be due to leakage around the charge pump circuitry, or rectification of the RF in the varicap causing a constant offset. Its effect does not appear to be important to operation. Without programming-in the 20s initialisation period, there were times, depending on the initial random starting count, when the loop would fail to lock up at all, and other times when lock would occur after a few minutes.

By using the output as a clock for my AD9850 DDS module, the output frequency is divided down and the phase blip is reduced by the same proportion. At 137kHz, for example, a 360° blip at 4MHz appears as a blip of 11° every second, always returning to the same value. When averaged out over a 30-second signalling element, the net phase shift is very close to zero. An LED flashes with the 1PPS signal, and the duration of flash is related to the width of the charge pump pulse. Therefore, when locked, the LED gives a short flash but, during the lock-up phase, the flash is of varying duration and, if the GPS pulse is not present, the LED does not flash at all.

CONSTRUCTION AND SETTING UP

THE CIRCUIT LAYOUT is not critical, apart from that around the VCXO. Here, wires need to be as short as possible and the loop filter components need to be mounted close to the varicap to avoid picking up interference which is then coupled onto the output signal. A double-sided PCB has been produced which makes use of normal 2.54mm pitch DIL ICs, but surface mount resistors

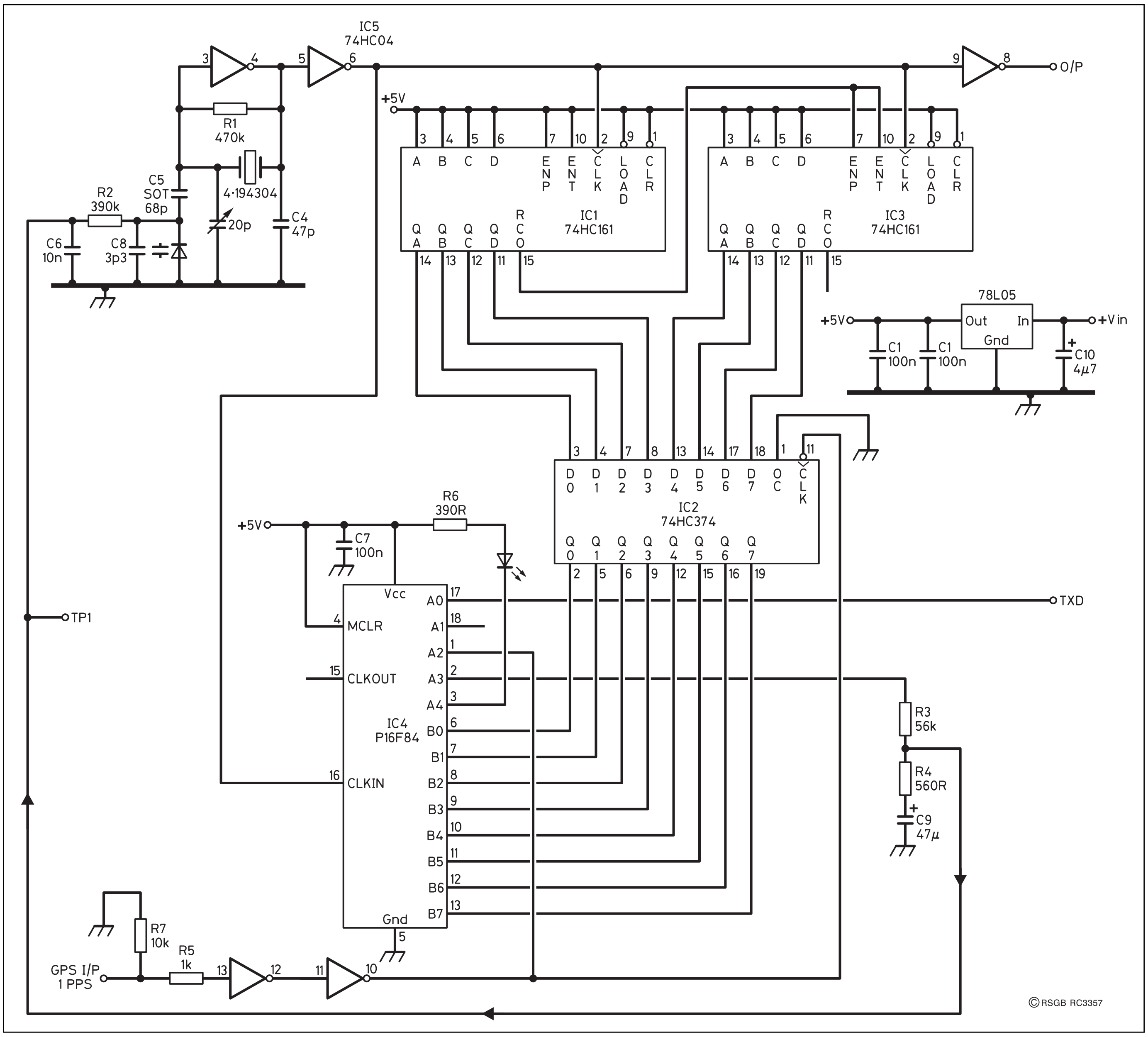

Fig 1: Complete circuit diagram of the GPS-locked frequency source for LF.

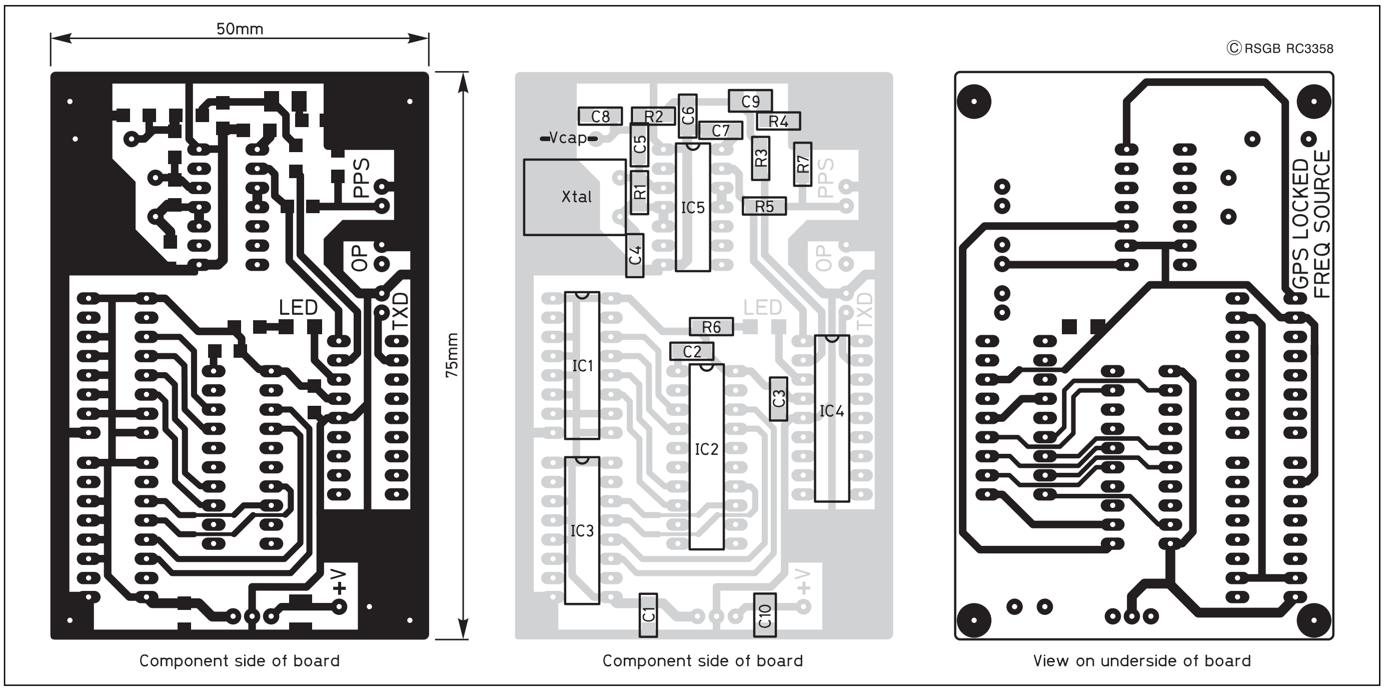

Fig 2: Tracking details for the PCB.

and capacitors. (No apologies are offered for the SMT construction, I do not have many stocks of wire-ended components these days, just SMT ones.) The prototype layout is given in **Fig 2**.

There is a certain amount of initialisation of the circuit necessary before satisfactory operation can be guaranteed. Firstly, the VCXO has to be set to the correct frequency. Before soldering in the RC loop filter components, connect a potentiometer to the varicap decoupling resistor, shown as TP1 on the diagram. Set to mid-rail at 2.5V, and adjust the preset capacitor for a frequency as close as possible to the correct value. Swing the tuning voltage over the range 0 - 5V, and ensure that the output frequency shifts by no more than about 100Hz in either direction. Much more than this, and a false lock 256Hz away is theoretically possible if the crystal drifts over time; much less that this, lock-up time and drift may be a problem. To change the tuning range, alter the value of the 'select on test' capacitor experimentally. Once the VCXO tuning has been set up, insert the remaining loop filter components. Connect the 1PPS output from a GPS receiver to the board and switch on. It may be convenient to monitor the loop tuning voltage with a high-impedance meter at this stage. If one is available, connect PIC port A0 to the RXD line of an ASCII terminal set to 1200 baud, 8-bit, no parity, to monitor the loop-locking progress.

For the first 20 seconds after switch-on, the LED will stay off and the voltage at TP1 should rise to 2.5V, at which time the output frequency will be close to the correct value if the setting-up procedure was followed carefully. After this time has elapsed, the LED will start to flash in synchronism with the GPS pulses, the flash duration appearing to vary in an apparently random manner. If the counter error value is being monitored, it will also show a rapidly varying number each second but, after about 30 seconds to a minute, a pattern will start to emerge and the value will gradually converge to a constant value, not far removed from zero; the duration of the LED flashes will also shorten. The voltage at TP1 should now be stable and, after a few minutes, the loop should have stabilised, at which point the output frequency is locked. If you listen to this signal on an SSB receiver, it should give a slight blip every second which should be just about audible, but how much depends on how musical your ears are !

There is plenty of scope for further experimentation, particularly around the loop to decrease chirp / phase blips and improve lock-up times. One idea that would be worth investigating is to have a non-linear relationship between error value and pulse width, something that is straightforward to implement in software, but complex in a PLL built completely in hardware.

PIC SOFTWARE

THE PIC SOFTWARE STABIL01 is supplied in the *RadCom Plus* area of the RSGB Members-Only website. Examination of the source code in the .ASM file should show the operation; I hope it is well documented enough!

An updated version of the software, STABIL02, is available. This now sends debug information at 19200 baud rather than 1200, and has improved balance between positive and negative pulse durations, leading to a more stable lock up condition. Programmed PICs can be made available if you cannot blow your own.

KNOWN PROBLEMS

- The loop error value is rarely stabilised at exactly zero during operation. This is probably caused by rectification of the RF in the VCXO by the varicap, which the loop has to fight. It could be cured by adding an op-amp buffer to isolate the varicap drive from the loop filter itself, but the effect does not appear to cause a problem, so no buffer has been included. Increasing the value of the 390k resistor feeding the varicap reduces this effect, but this shouldn't be made too high, as spurious pick-up could then become more of a problem. Charge-pump capacitor leakage will show the same effect if any is present, and is particularly noticeable when a voltmeter is connected to TP1.
- The latching of the counter by the GPS pulse is asynchronous to the counter increment, so there is a probability (a certainty in fact) that at some point the reading will try to be latched at the very instant it is changing; minimising this, in fact, was the reason for using a synchronous counter here. Fortunately, there is a simple work-around. During testing, is was noticed that a read like this always appeared to result in a latched value of 0xFF, or all ones. The software detects this condition and ignores that particular value, treating it as if a value of 128 had been read and issuing no pulse. The loop will never be so stable that this glitch situation will occur repeatedly, and the tracking continues uninterrupted at the next pulse. The resultant 2-second delay may cause a slightly larger blip than usual, but this should not be unacceptable in practice. ♦