Written by Hans Summers
Thursday, 04 June 2015 03:03 - Last Updated Monday, 10 October 2016 09:48

Here's a simple and easy single-chip 10.000000000MHz GPS-disciplined frequency reference, based

10MHz OCXO

I am re-using the 10MHz OCXO, type HCD71, from my earlier GPS Reference Project . For various reasons I decided to retire that reference, one of which was that these days I am more used to using C programming than assembler, and that project is coded in Assembler. The power supply situation is a bit difficult in that project too, it overheats with basic linear regulators (7812, 7805 etc); I had tried switching regulators but they are too noisy for the microcontroller to work nicely. Therefore I'm re-using the HCD71 in this project. It has an analogue voltage input which can steer the output frequency over a range of +/- 100Hz.

GPS Receiver Module

For the GPS disciplined timing I am using an SKM61 GPS receiver module from QRP Labs . This module has high sensitivity and a very precise 1 pulse per second (pps) time pulse, with 10ns rms accuracy. The single GPS receiver module regulates both the output frequency of my Ultimate3S QRSS/WSPR transmitter and also this 10MHz frequency reference. They are both connected in parallel to the 1pps output of the SKM61 .

Circuit

The circuit diagram (schematic, if you prefer) is shown below. Click it for the full-size version. As you can see, it is very simple!

{gallery}gpsref2/circuit{/gallery}

The actual stabilisation work is done in the software in the 8-bit Atmel AVR ATtiny24 microcontroller. This 14-pin chip has a 2KByte Flash program memory and 128 bytes of SRAM (for data variables in the program). Relevant items of note in the circuit, are as follows.

I am using a BC109 transistor as a buffer amplifier between 10MHz OCXO's sinewave output, and the ATtiny24 microcontroller's clock input. The fuses on the microcontroller are set to use the externally provided clock, 10MHz in this case. This works well and without any fuss.

The 10MHz sinewave output from the HCD71 OCXO passes through a 7-element Low Pass Filter to attenuate any remaining harmonics which may be present. For convenience, I used the 30m Low Pass Filter Kit from QRP Labs
. The output goes to a BNC connector on the front panel.

The voltage control input of the HCD71 OCXO can pull the frequency over a range of 10MHz +/- 100Hz. I produce this voltage using the 16-bit Pulse Width Modulation (PWM) output of the ATtiny24 microcontroller, and a simple R-C integrator from a 3.9K resistor and 470uF capacitor (time constant: 1.8 seconds). The calculated frequency change at the output of the OCXO, for a 1-bit (least significant bit) change in the PWM value, is 0.0025Hz. This figure governs the basic accuracy attainable by this 10MHz frequency reference. A more elaborate circuit could be used here to obtain higher accuracy, this is a matter for future improvement I think - for now this 0.5 ppb precision is adequate for my initial purposes.

Three LEDs are provided for visual indication of what is going on in the device. These are set up to show respectively the 1pps signal; direction of frequency correction; and frequency lock status. The LEDs I happened to find in the junkbox are yellow LEDs in a nice panel-mounting housing. They contain an internal current limiting resistor and there is a label on the LED that says 6V 15mmA (sic. I assume that they mean 15mA). I could have connected these directly to the microcontroller output pins. However I wanted to avoid any additional current loading of the microcontroller, that may cause the supply voltage to sag. No voltage regulator is completely ideally perfect. It only takes a 76uV (0.000076V) change in the supply voltage, to equal a 1-bit change in the PWM value. I don't want to risk that kind of tiny voltage change, when the LEDs are switched on and off. So I decided to switch the LEDs to the 13.8V supply using 2N7000 transistors. The load change on the microcontroller voltage supply is negligible.

Finally, to the voltage regulator arrangement. You can see from the circuit diagram, it is a little elaborate. Possibly overkill. What I wanted to do was ensure a very stable 5V supply to the

ATtiny24 microcontroller which cannot be affected by anything else which is going on, e.g. changing load due to the LEDs going on/off, or the oven heater taking different current consumption. The LM2940-12 is a low dropout regulator, to produce +12V for the 10MHz OCXO device, from the 13.8V supply voltage. A simple 7812 regulator may not be suitable here because the voltage dropout may exceed 1.8V (the difference between 12V and 13.8V). Following the +12V supply, a 7808 takes the voltage down to 8V, and from there a TO92-style 100mA 78L05 produces 5V for the microcontroller. To avoid any other interference with the 5V, I also provided a separate 5V regulator using a 7805 which powers the SKM61 GPS receiver module, separately from the microcontroller. I used lots of capacitors all over the place, maybe too many, but the junkbox anyway contains too many, so what.

Finally note that the 9-pin D-type connector on the back panel is used to connect both the GPS receiver module (Gnd, +5V supply, and 1pps) and also a 5-way connector for programming the AVR microcontroller. The MOSI, MISO, SCK and RES signal connections are not shown on the circuit diagram, but obviously they just connect to the relevant pins on the microcontroller.

Software

The code to control this is short and simple. It uses just the 16-bit Timer1 which is clocked from the 10MHz system clock without dividers. The OC1B PWM output drives the 10MHz OCXO steering voltage as described previously. The 1pps input from the GPS module is used to gate a frequency counter, that also uses the 16-bit Timer1. Very simply, if the frequency is too high, then the PWM value is adjusted downwards, resulting in a decrease in the OCXO control voltage, which reduces the frequency. Correspondingly if the frequency measurement shows it is too low, it is steered slightly upwards.

The trick is all in deciding a suitable algorithm for choosing the step size at each interval. In the end, you want the step size to be very tiny, 1 Least Significant Bit (LSB) of the PWM, so that each step nudges the OCXO output frequency only by a very tiny amount in each direction. However, on power up, you want a much larger step so that the initial lock does not take a very long time. For example: there is one correction per second, if the step was 1 LSB, in theory it may have to steer from the starting value of 32768 all the way to one extreme or the other, at a rate of 1 LSB per second; that could take up to 32768 / 3600 hours (9 hours). We don't want that. A locking time of a few minutes is acceptable, 9 hours is not.

So, it is lots of fun trying to work out the best methods to start with a large step that finds the approximate correct value for the steering voltage, and then get as quickly as possible down to 1 LSB step size for finest accuracy.

The software also controls the LEDs. The "PPS" LED just flashes in sync with the incoming 1pps pulse (kind of a health-check of the GPS receiving system, like a heartbeat). The "DIR" (direction) LED is lit each time the correction is in an upwards direction, and switched off otherwise. The "LOCK" LED is lit when the step size is less than 10, which corresponds to a delivered output freuqency of something like 10MHz +/- 0.05Hz.

Construction photos

The circuit is built onto two PCBs that are stacked one above the other, to fit in the small aluminium box for the project. Click the photos in the albums below for the larger size images.

{gallery}gpsref2/1{/gallery}

1. Taken during construction, this photo shows the processor PCB. This 80 x 37mm PCB is a spare, from another project. It has a socket for the ATtiny24 microcontroller and a small prototyping area on the right hand side, which is large enough to build the other parts of the circuit.

2. Here's the lower PCB (processor) installed in the box, with all connections wired. The black cable from near the blue capacitor (470uF) is a screened cable carrying the OCXO steering voltage to the OCXO.

3. A close-up showing the wiring to the 9-pin D connector.

4. A close-up showing the voltage regulator installations. The three TO220 voltage regulators and their smoothing/decoupling capacitors are bolted directly to the bottom of the aluminium

case, acting as a heatsink. In practice the whole case, including the quite hot oven (OCXO) only gets very slightly warm. Most of the circuit consumes very low power except for the 10MHz OCXO, and this is powered only from the LM2940-12 regulator, which drops only 1.8V from the 13.8V supply. So the heat dissipation is really quite small.

{gallery}gpsref2/2{/gallery}

1. This photo shows the second PCB installed, bolted on top of the first one using 12mm nylon hex spacers. The second PCB is the same kind as the first one. Another leftover spare from a different project. I'm not using any of the traces on the PCB. It's just providing a suitable frame to fix the HCD71 OCXO on to. On the right you can see the QRP Labs 30m LPF PCB. I am feeding the 10MHz signal the wrong way through the filter, but that doesn't matter, it's symmetrical anyway and there aren't any PCB traces that it's connecting to.

2. Another photo showing the nice neat stacked construction of the two PCBs.

3. Here's the back of the box, showing 9-pin D connector, and a 2.1mm Power Connector for the 13.8V supply voltage input.

4. Finally, here's the operating unit, working nicely. I put it sitting on top of my Ultimate3S QRSS/WSPR Transmitter
since anyway they are sharing the same GPS receiver module 1pps signal for their GPS disciplining action. On top of that, my
QRP power meter
.

Spectrum Analyser results

More info to follow shortly...

{gallery}gpsref2/3{/gallery}

{gallery}gpsref2/4{/gallery}