What is Lars Arduino based GPSDO Controller with 1ns resolution TIC?

By Lars Walenius, Aug 11, 2017

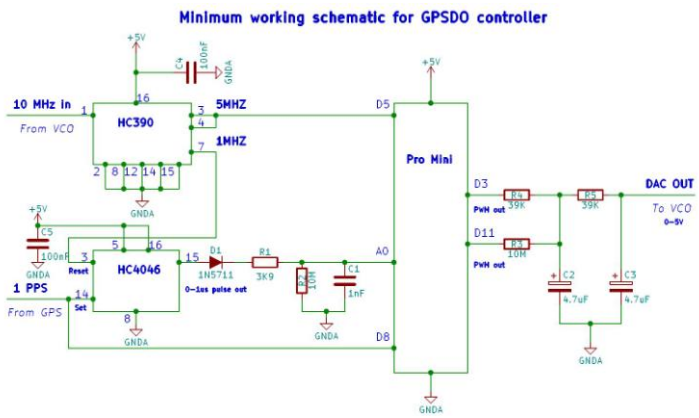
This is a flexible GPSDO (GPS Disciplined Oscillator) controller mainly for the experimenter and will work as a complete GPSDO with a GPS receiver that can output 1PPS (pulse per second) and a 10MHz VCO (voltage controlled oscillator).

I have built several complete GPSDO prototypes to test both hardware and software but probably it still has bugs.

As it has a TIC (Time Interval Counter) with a resolution of 1ns (nanosecond) it is also useful as a cheap and simple frequency counter for 10MHz oscillators.

I started the design many years ago, around 2010 I think, just because I was lazy and didn´t want to tweak my 10MHz house standard. The design is very inspired from other designs and mostly from Brooke Shera's excellent design in QST 1998. The only thing I am proud of is the very simple 1ns TIC.

The software is tailored to my needs so many will say it is far too complex but it contains the functions I want. Of course, I have a long list with other functions I also want but that are another story. I have many times thought of using another processor than the ATmega 328P but think it has it charm to use this simple processor and the Arduino platform. I also should say I have worked too long as an analog and RF engineer so software is not my best discipline and another reason to stay with the Arduino.



The hardware is simple. It is an Arduino controller with just a few HCMOS IC´s and discrete components. The Arduino and HCMOS circuit measures the time difference between the 1PPS and 10MHZ and outputs a voltage to the 10MHZ oscillator, so basically a PLL (Phase Locked Loop). As the 1PPS isn´t so reliable the software takes care of some conditions such as missing or wrong pulses to some extent. I guess this might be a reason it is called a disciplined oscillator instead of PLL.

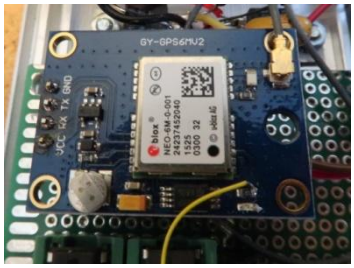
The resolution of the time difference measured is about 1ns. The range is 10ms (milliseconds) as both an analog hardware measurement with a range of 1000 ns and a timer measurement are combined.

The main loop is a normal PI-loop (Proportional + Integral) in software. As the 1PPS has a lot of jitter, normally in the range 10-50ns p-p, the time measured between the 1PPS and 10MHz is filtered with a low-pass filter in the software.

The hardware may seem simple but the performance is still mostly dependent of the GPS receiver and 10MHZ oscillator. So, first I recommend selecting a GPS receiver and oscillator depending on your requirement on stability.

GPS receiver

Almost all receivers with 1PPS out will work. The PPS pulse shall be positive and enough to drive the inputs. So far 3.3-5V pulses from the receivers have worked for me. I have tested both normal position only receivers and timing receivers. Timing receivers gives the chance to have better performance but if you have no receiver I would recommend to start with a uBlox NEO-6M or 7M that can be had, mounted on a board, for less than 10USD at eBay.



I recommend the blue boards that I have tested several without problems. With the red boards, I have just had problems. On the blue boards the 1PPS is taken from the resistor before the flashing LED on the uBlox pin side.

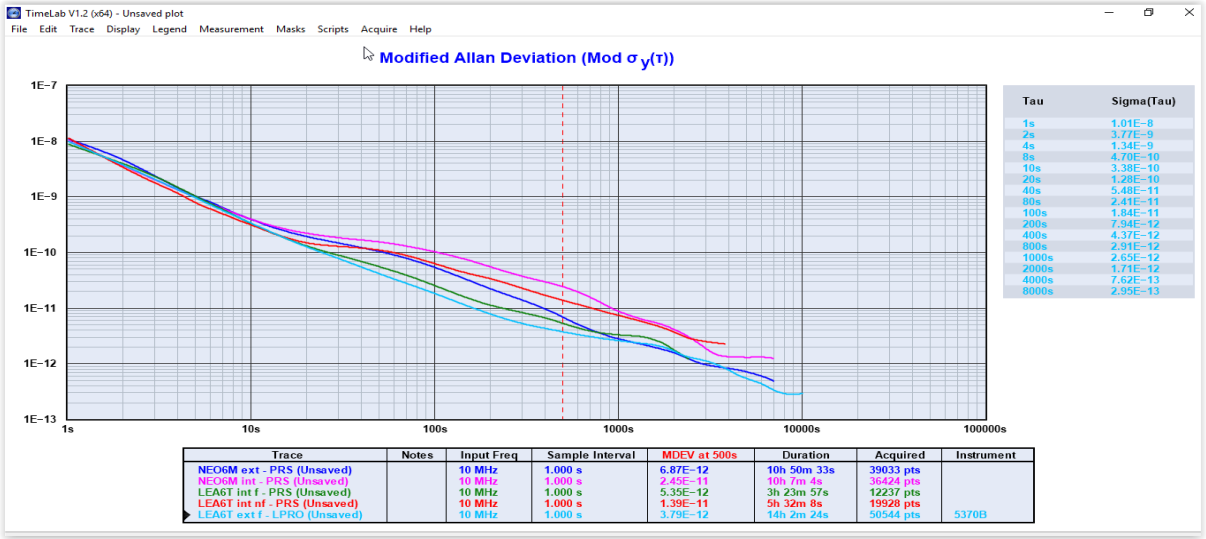
My recommendation before you build the controller is to test a GPS receiver.

Modern GPS receivers like the uBlox often work inside a building but having an antenna outside is better even if just near the window. I have used both cheap patch antennas (around 3-5USD on eBay) to more expensive antennas with good results.

Using a program like the manufacturers own (eg uBlox) or other programs like Lady Heather that can show you how many satellites you have and the signal strength is very useful. Seeing at least 4 satellites is preferred even if timing receivers might work down to one satellite. With outdoor antennas, I often see 50dB Signal to Noise Ratio but in-house on the upper floor I have only 30dB but the signal is still ok to lock the GPSDO. With S/N ratios of 20dB it is a large risk that the signal is too bad. I also have a lock on the Altitude charts available in the u-center, if you have deviations above 50 meters you may have problems with the timing also. With a poor in-door antenna position I loosed lock of the GPSDO even with seven satellites used but the S/N ratio averaged only 20db and with an altitude peak of about 80 meters wrong the GPSDO loosed lock as the time deviation was more than 100ns for more than 16 seconds (PPS lock filter in software is 16 seconds).

The GPSDO controller works best if the PPS signal only is available when it is valid. Check if the PPS stops if you have a really bad signal or the antenna is disconnected. Some modules have settings to disable the PPS output. Check for your particular receiver.

Some examples of what you can get with NEO6M and LEA6T GPS modules:



In this EEVblog thread you can see some more of my results with different receivers and internal or external antennas: https://www.eevblog.com/forum/projects/ocxo-stable-reference-and-control-voltages/50/

10MHz oscillator

The Arduino GPSDO controller described can handle voltage controlled 10MHz oscillators with a range of 1ppb (part per billion or 1E-9)) to 6.5ppm (part per million or 1E-6) with a DAC output voltage of 0-5 Volt and DAC output impedance of 78kohm.

If your oscillator doesn't have a range of 0-5V it should be quite simple to add a resistor network and/or an op-amp to change the range. If your VCXO (voltage controlled crystal oscillator) for example have a range of 30ppm for 0-3.3V a simple resistive divider with a couple of resistors or a trim pot will be useful to restrict the range to 6ppm.

Note: Oscillators with inverse control characteristics, that is: lower frequency with higher control voltage, needs hardware inverting op-amp or similar or changed software.

I have tested the controller with both VCXO, VCTCXO (voltage controlled temperature compensated XO), OCXO (Oven Controlled XO) and rubidium oscillators. Some of my experiences: The cheap VCXO for a few USD from e.g. Digikey and Mouser works but has just slightly better performance than the 10MHz you can have from the NEO-7M, that is about 7-8 digits for short times (seconds). The cheap VCTCXO I bought from Digikey are not better as they sometimes make jumps in the region of 50ppb (5E-8). For experiments, they work. The better VCTCXO´s as DOT050V from Digikey works well and I have got ADEV´s below 1E-10 all the way from 1s. See figure 8 in appendix. Most of my prototypes have had OCXO´s from eBay. As they can be had for about 10-50USD they are a bargain if they work. Some have reported just problems but that is not my experience. I have had one that just works a couple of minutes. Some have had quite frequent small jumps in the 1E-10 range but not really a problem. Some are excellent with ADEVs in the low E-12 or even high E-13 for some Tau´s (time scale for ADEV´s).

I recommend buffering the 10MHz oscillator. Not so much for the Arduino controller but for your outputs that you will use with different loads. Otherwise you will have jumps every time you connect or disconnect a load. I have used 74AC04 and often even 74HC04 even if they are slightly worse but if you just need ADEV's (see description below) in the upper E-12 range the HC is ok. If you want 50ohm outputs, I have paralleled three inverters each with 120 or 150ohm series resistors. I have also added an external LCL tee-filter to get a sine output if needed. I have got about 11-13dBm sine output with an L of 10uH and C about 50pF (trimmed for maximum output).

Remember that the loop that controls the oscillator mainly changes with very low frequency so the noise above 1Hz is mainly due to the oscillator (and 1Hz from the PPS with overtones if bad grounding). If you have requirements on the noise for example for instruments or RF work the oscillator is most important.

What is ADEV?

One way to learn more is to check Wikipedia for Allan Variance https://en.wikipedia.org/wiki/Allan_variance . This gives a very good description.

Otherwise a very simplified way of describing ADEV (Allan Deviation) is to say it is the frequency stability (not absolute accuracy) described as the standard deviation of difference of two nearby frequency measurements with a gate time of Tau in seconds. So, if you have an ADEV of 1E-10 at Tau 10 seconds and you have a perfect frequency counter, the readings will fluctuate several parts in E-10 (as the ADEV is just standard deviation not peak to peak) that is in the 10^{th} digit but this is not the accuracy.

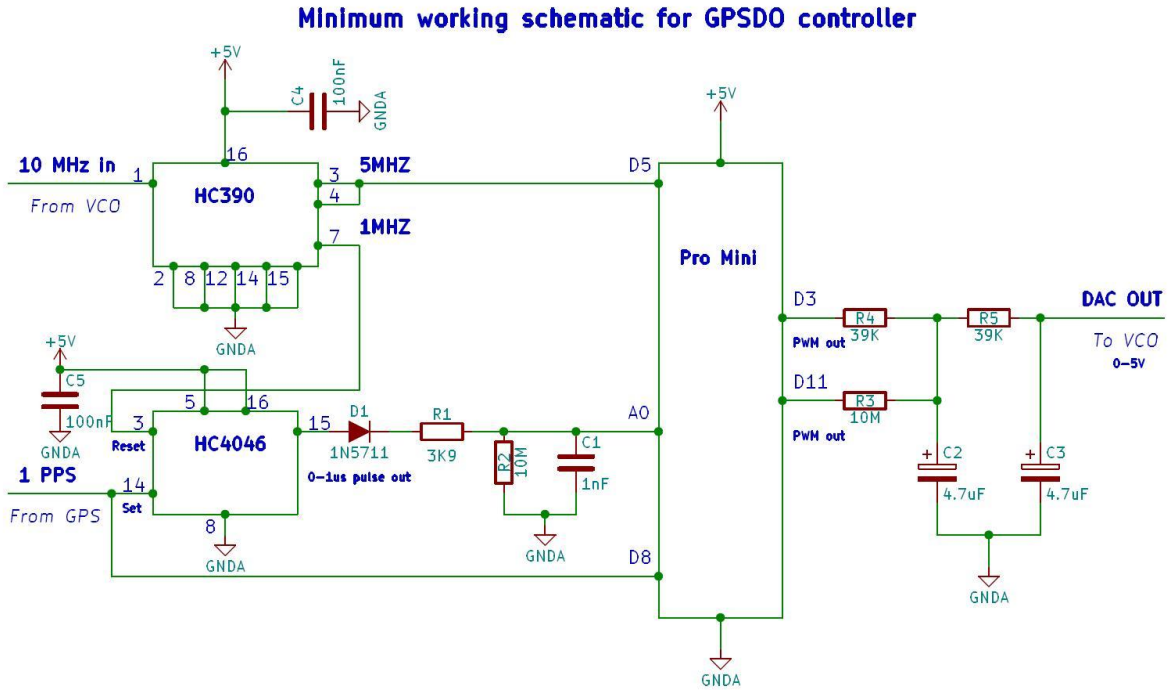
For a GPSDO that is locked to a "perfect" reference the absolute accuracy may be as good if the loop is tight so the oscillator doesn´t deviate for longer times (that is for larger Taus). A free running oscillator drifts away with time and so will have worse ADEV at larger Taus.

The nice thing is that the ADEV for a GPS is only bad at low Taus but gets better and better for longer Taus so by combining these two in the PI-loop gives the best of two worlds. At some Tau, the ADEV for the GPS and oscillator are about the same (line crosses). At this point the loop should change from the oscillator to the GPS, of course it isn´t a switch-over but a gradual change between the GPS and oscillator. This is selected by setting the time constant for the PI-loop. As the error signal is filtered it is sometimes better to use the cross over point of MDEV.

In the appendix you can see some examples of ADEV etc.

The software can be set through the serial port for time constants from 4 to 32000 seconds with command "t"
Typical values for an XO might be 5-30 seconds, for a good TCXO 30-100s, OCXO 100-1000s and a rubidium 2000-10000s.
With the XO you can get ADEVs in the E-8 -9 range, for a good TCXO E-10 to -11 and an OCXO E-11 to -12 and a rubidium oscillator (at Taus greater than 1000secs) in the E-13 range. At Taus of a day all GPSDO's correctly built have ADEVs in the E-13 and some even high E-14 ranges.

Hardware description



The “fine” TIC with a range of 1us consists of the 10MHz divided down to 1MHz fed into an HC4046 reset pin and the 1PPS fed into the HC4046 set pin. The 1PPS sets the HC4046 output high and starts to charge the 1nF capacitor through the diode and 3.9kohm resistor. Within 1 us the stop pulse from the 10MHz divided down to 1 MHz set the HC4046 output low and charging stops. As the diode is in series the low level on the HC4046 will not discharge the capacitor. The leakage current on the ATmega 328P is quite low so the main discharge will be through the 10Mohm. As the 1PPS gives an interrupt to the processor the ADC will measure the voltage almost immediately after the charge is ended. The ADC start time is not synchronous with the 1PPS but the internal 16MHz so a timing error of several us gives a ripple on the reading if the discharge is too fast. I used 1Mohm before and that gave a ripple of several ns. The stability of the TIC at around the midpoint of 500ns is good. I have measured around 0.3ns/°C. Most of it comes from the about 2mV/°C change on the diode I guess. The linearity of the TIC is of course not perfect as it is more or less an RC circuit so it is exponential. The good thing with the ATmega328p is that is has a selectable ADC range so I have chosen the 1.1V range. That makes the nonlinearity quite small. Some other DIY GPSDO that has adapted my TIC like Nick Sayers and the Elector GPSDO has tried with a FET as current source as they have the full ADC range. So far I have seen no evidence that this is better, it also adds a component with much more variation than a simple resistor. Screenshots from the Elektor GPSDO ramp still has much worse linearity than using the 1.1v range of the 328P. My software also has the possibility to linearize the slope. I have got the INL down to around 1ns by using that. If you use the controller only as a TIC I have managed to get ADEVs below 1E-9 at 1s with the serial line directly feeding the excellent Timelab software. See examples in appendix.

10MHz divided by 2 is fed to timer1 that make the TIC range much wider in the software. The timer1 need to be fed with less than half of the processor clock frequency so that is the reason for the division. The resolution of 200ns still is enough to “overlap” the 1000ns resolution of the fine TIC so the software can combine them.

The 16 bit DAC is implemented as a PWM-DAC. The ATmega328p has two 8-bit PWM´s that are used to get the 16 bits. The two 8-bit outputs are combined with two resistors with a ratio of about 256. With 1% resistors, it is possible to get at least 13-14bits of monotonicity that is the most important parameter. The stability is mainly dependent on the 5V regulator. Often the 7805 types have around 100ppm/C or more that will limit the performance. So far I have used different types of 7805 with good results despite that. The noise is not the problem as the output is heavily filtered. One way to make this controller better is to change to one or two monotonic DACs for example MAX5217 with a more stable reference like the MAX6070. Changing the software for that should probably not be very difficult but I have not tried. For me the PWM-DAC has worked fine so far. For those skeptical that a 16 bit DAC is enough I recommend to use the GPSDO simulator found on leapsecond.com to simulate different resolutions. This simulator also shows that 1ns is more than enough for most GPSDO's. I have one Isotemp 131-100 OCXO based GPSDO with a "poor" resolution of 1.2E-11 (about 1ppm VCO range) that have ADEVs in the E-12 range. See figure 8 in appendix.

The hardware and software can use ADC2 to measure temperature and also for temperature compensation. The software has a setting to read °C for an LM35 or 10k NTC with beta 3950 + pull-up 39k, 47k or 68k. The value is output every second on the serial line. As the NTC is cheapest I have used that most of the time. Adding a capacitor (1nf or larger) in parallel with the ADC inputs are recommended. But easiest is to have an LM35 connected to ADC2.

Using ADC1 to read temperature or something else is optional. The input range is about 0-1.1v. I have often used an NTC + pull-up connected near the processor and the ADC2 sensor near the OCXO. The software outputs the value from ADC1 every second to the serial port.

ADC3 also has a range of 0-1.1V and can be read by a command ("f3") on the serial line. I have only used it to read the lamp voltage on my rubidium based GPSDO.

A LED + resistor connected to D13 will give an indication that the loop is locked. After five time constants with a slightly filtered (tc=16s) TIC value within 100ns it will light. If the DAC value, in locked state, is near the ends (<3000 or >62535) it will flash. If you have "no PPS" it will give a brief double flash. I added this to give a visual indication if the antenna is disconnected. It will of course also indicate at all other conditions that give no PPS.

Power supply:

In the beginning I powered my prototypes from lab power supplies without extra regulators and that worked quite well. But when I mounted them in boxes I added regulators.

I have used MC7805 type regulators for the Arduino pro mini and HCMOS power. I have also used the same regulator for the GPS receiver. If you use one regulator for both be careful to take the +5V directly from the regulator in two different paths as the GPS receivers often have very fluctuating current that might modulate the Arduino PWM-DAC.

For OCXO's with 5V I have used a 7805 both for the OCXO and CMOS buffer stage (not the same as for the Pro Mini and GPS). For 12V OCXO's I have used 7812 style regulators and added an extra 7805 for the CMOS buffer stage powered from the 7812.

For the power input I have used a 5.5/2.1mm jack and added a 1N5819 in series for reverse power protection. Most of the time I have used cheap 15V stabilized switched mains adapters even if I don´t know if they have given a lot of noise. If you are using the GPSDO to drive radio equipment or instruments that needs low phase noise it might be better to use a linear supply.

Building the hardware - selecting components

For the processor I from the beginning used an Arduino UNO board, I have also used ATmega328P IC with separate 16MHz xtal and UNO boot loader. But what I prefer are the Arduino ProMini 5V 16MHz boards for under 2USD found on eBay. So far I haven´t had any problems with them and have used both types with crystals and ceramic resonators. They require that you have a USB-TTL converter with DTR (to be able to program) but I still find them best.

The HC4046 and HC390 might also be HCT without problems.

I have selected the diode due to the low capacitance but am not sure if that is a real problem. Using a schottky is also preferable due to the low voltage drop even if a 1N4148 probably will work (I have tested one). 1N5711 and 1N6263 are types I have tested several of. BAS70 could probably be a cheaper alternative. If the capacitance in the diode is too high, it will discharge the 1nF a little when the HC4046 output goes low.

The 1nF capacitor shall be NPO ceramic or some other temperature stable type. I have mostly used SMD0805 types. For the 4.7uF I have used tantalum drops with 25 or 35V ratings as I have hoped they have lower leakage current. The leakage current by itself is not a problem if it is stable but if it varies with temperature it will change the output voltage due to the voltage drop over the 39+39k resistors.

Having decoupling capacitors is necessary. Use 10-100nF at the IC's +5v.

The resistors I use have been 1%. No special types. I have found that SMD0805 types fit well on the bottom side of prototype boards with 0.1 inch spacing.

Just for experimenting it isn´t necessary to have a metal box but for a lab standard I think it is mandatory to have a well shielded box and put some effort to have good grounding e.g. ground planes or good connection to the box at relevant points. My first GPSDO was an Arduino UNO with a prototype shield and together with the 10MHz it gave a lot of disturbance to other parts in my lab.

Good grounding is not only necessary because of the 10MHz but also on the very sensitive voltage control signal. I have the last 4.7uF (sometimes I have used 10uF) close to the OCXO for example. The GPS modules often have very fluctuating supply current so having good control of the ground currents is necessary if you don´t want a lot of 1Hz modulation on the 10MHz output. Just a blinking LED might give too much modulation if not properly handled. As the 5V to the controller is also used for the PWM DAC it is important that fluctuating supply current don´t give ripple on the 5V (might be another reason to change to a DAC with separate Vref).