External Lock Implementation

Implementation Guide: Frequency locking

Note: this guide is for implementation of the PPCL590 firmware in combination with an external frequency reference. This guide may give some helpful information on how PPCL590 works, but is not a manual for that product.

Pure Photonics provides a PPCL590 product with an external gas-cell integrated within the module. The absorption lines of the gascell serve as reference for a locking mechanism. The PPCL590 is realizing high sensitivity to frequency variations through electronics means (down to 1kHz/ADCcount), however the underlying firmware can be used to implement a frequency locking loop using a user's preferred feedback reference with either higher or lower sensitivity.

Pre-requisites

To be able to operate this custom firmware the device should be configured with FM modulation (this input should remain open, as it is controlled internally, but the associated circuit modification is required) and have the Clean Measurement input for a feedback signal. Optionally an analog output can be used (see 'complex operation').

The firmware required for this operation is specific for this application and needs to be loaded to the unit. Only specific timestamps are suitable. The specific timestamps are listed in the appendix.

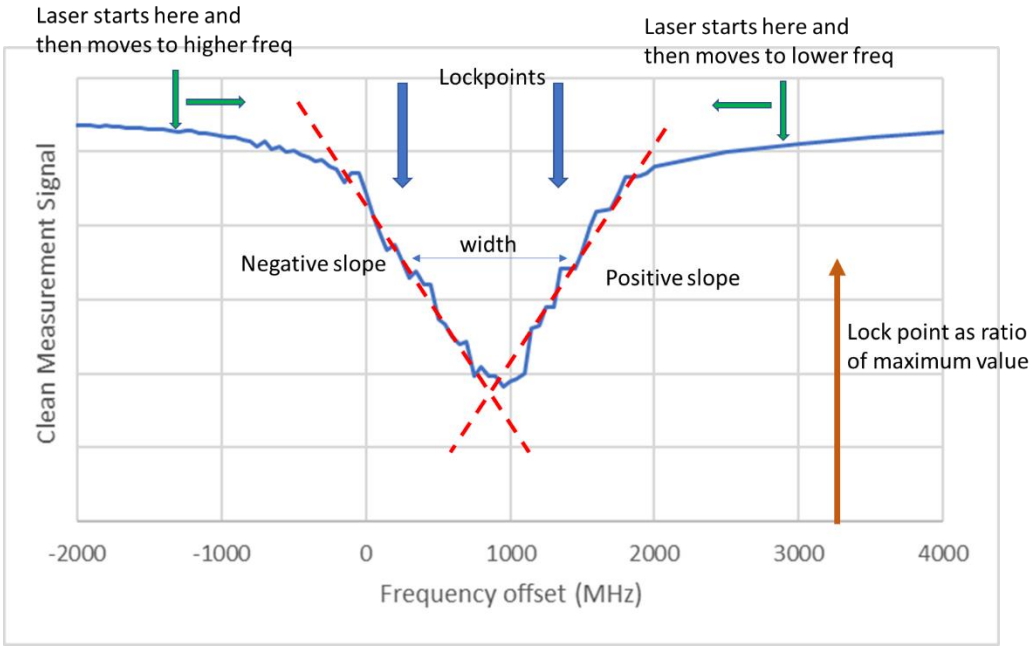
1. Configuration guide

Configuration

The firmware is operated as per the PPCL590 manual. With channel set to 1 any frequency can be accessed. With a channel > 1 the locking mechanism is enabled. For this purpose, once the channel is set, four numbers are pulled from memory:

- Lock frequency
- Slope-ratio: this is the slope of the Clean Measurement input versus Frequency at the lock point as a fraction of the full power. E.g. when the full power is 40,000 counts and the ratio is 0.0001 then 4 counts is 1MHz. In our PPCL590 we can achieve slopes of 1000 counts/MHz. The sign of this slope-ratio is important.
- Lock-ratio: this is the target depth of the lock-point versus maximum response. E.g. if locking to half of max, this value would be 0.5. In case the resonance is a peak, the value needs to be larger than 1.0.
- Width of feature: this is the FWHM of the absorption peak in units of 10 MHz. Currently this is not being used.

In the graph below a typical absorption curve is shown. The blue arrows indicate the two available lock frequencies. The slope is indicated by the red dotted line. The lock-ratio is illustrated by the red arrow.



These datapoints are being stored in memory in groups of 16 bytes. E.g. channel 2 corresponds to the first 16 bytes, channel 3 to the next 16. Each group of 16 bytes is configured as:

- 4 bytes for frequency as a floating point number. Unit THz. **The frequency value has an offset of 190THz, so 193.43 is coded at 3.43THz and 188.4 is coded as -1.6THz**
- 4 bytes for lockslope as a floating point number. Unit %counts/MHz
- 4 bytes for lockdepth as a floating point number. Unit is a ratio.
- 1 byte for feature width as an integer in units of 10MHz
- Last 3 bytes are not used.

Refer to https://www.h-schmidt.net/FloatConverter/IEEE754.html on how to calculate the byte representation of a floating point number.

Programming

NOTE: never do any programming to an actual PPCL590 unit. This will overwrite the calibration constants.

The above numbers need to be programmed to permanent memory. This is done by sending the data byte-by-byte to register 0xFD, with datafield (0xF000 + byte value). Every byte needs to be written, so 16 bytes need to be send for each setpoint. It is recommended to add at least 2 sets of 16 bytes with value 0xFF as a finish sequence. Once all data is provided send datafield 0xFFFF to register 0xFD. Note that the total number of bytes can not exceed 512 (30 channels + 2 dummy channels).

A good floating point converter can be found at: https://www.h-schmidt.net/FloatConverter/IEEE754.html.

For example:

- Lock frequency: 195.8934 -> converts to 5.8934. Hexadecimal representation: 0x 40bc96bc
- Lock slope: 0.00345. Hexadecimal representation: 0x3b621965
- Lockdepth: 0.65. Hexadecimal representation: 0x3f266666
- Width: 600 MHz. We would write the value 60. Hexadecimal representation: 0x3c

So then we would write to the unit:

- **0xf040** to register 0xFD # lockpoint
- **0xf0BC** to register 0xFD
- **0xf096** to register 0xFD
- **0xf0BC** to register 0xFD
- 0xf03B to register 0xFD # slope
- 0xf062 to register 0xFD
- 0xf019 to register 0xFD
- 0xf065 to register 0xFD
- 0xf03F to register 0xFD # lockratio
- 0xf026 to register 0xFD
- 0xf026 to register 0xFD

- 0xf026 to register 0xFD
- 0xf03C to register 0xFD # width
- 0xf0FF to register 0xFD # this is unused
- 0xf0FF to register 0xFD # this is unused
- 0xf0FF to register 0xFD # this is unused
- …..
- Repeat the above for other lock points…..
- ….
- 0xffff to register 0xFD # this initiates the save to memory

Operating sequence

Once the laser is turned on (SENA=1), the laser will automatically go through the following sequences when channel# is larger than 1:

- Turn the laser on in standard dithermode at 2GHz away from the target frequency. When the slope is negative and lockratio < 1 or slope is positive and lockratio > 1 it will be 2GHz lower than the lockpoint. When the slope is positive and lockratio < 1 or slope if negative and lockratio > 1 it will be 2GHz higher than the lockpoint
- Lockstate 1: See complex operation for more information. In simple operation, the voltage on the CM1 input needs to be within the capture range of 0.62 – 1.88V. If not the analog output value will go up, until the clean measurement input drops to a value less than 1.88V.
- Lockstate 2: Adjust the analog output value to lock-depth*current value. If the analog output > 0 then the clean measurement lockpoint is 0x8000. If the analog output equals 0 then the clean measurement lockpoint is lock-depth*current CM value.
- Lockstate 3: Move the frequency (through FTF) towards the lockpoint, until the target clean measurement value is achieved.
- Lockstate 4: Switch on the whispermode
- Lockstate 5: Wait until whispermode has stabilized
- Lockstate 6: Not used
- Lockstate 7: Change frequency offset until the Clean Measurement signal is in locking range
- Lockstate 8: Adjust the internal PZT position to keep the Clean Measurement signal within lock. If the Clean Measurement signal moves out of locking range, the firmware automatically goes back to state 7.

Feedback information on operation

- Read operation on register 0x93: read clean measurement signal
- Read operation on register 0x94: read analog output signal (complex operation)
- Read operation on register 0x96: read RMS frequency error in kHz (data field = 0, integration/update time is 1sec; data field = 1, integration/update time is 20sec)
- Read operation on register 0xFD with datafield 0x9000: lock state

- Read operation on register 0xFD with datafield 0x9001: control signal applied on the internal PZT. This value would ideally be centered around the midpoint 0x8000

Controls of operation

- Read register 0xFD with datafield 0xe000 + multiplier: with multiplier being a scaling factor for the response to the error signal. In our PPCL590 we use the value 50, as we have a very sensitive response. Higher values are likely needed for other implementations. The multiplier is between 1 and 4095.

Enable for manual control

NOTE: never do any programming to an actual PPCL590 unit. This will overwrite the calibration constants.

On devices shipped after January 2024, the below setting is correct. However, on earlier units this may not be the case.

You can check the setting by reading register 0xFD with data 0xB4E7. The response should be 0x3F80. If that is not the case, the following fix needs to be made one time.

Read from register 0xFD with data:

- 0xC4E7
- 0xC03F
- 0xC4E8
- 0xC080
- 0xC4E9
- 0xC000
- 0xC4EA
- 0xC000
- 0xCFFF

After this sequence, power cycle the laser after 5 seconds (or more) and repeat the read operation with 0xB4E7 to see if the issue is fixed.

Complex operation

The laser stability that can be achieved is partially determined by the inherent stability of the laser and by the sensitivity of the frequency detection. In our experience, most frequency differentiators are not very steep, and just using the clean measurement input limits the sensitivity that can be obtained. E.g. if you find a differentiator that goes from max to 0 in 100MHz, then you could achieve (assuming using 75% of CM range at max power) 65535*0.75/100 = 491 counts/MHz. Typically this ideal situation is not even realized.

The analog output provides an auxiliary signal that can be used to improve the range. It assumes that with an increasing value of the analog output, the value on the Clean

Measurement input goes down. So with a proper external circuit configuration, it can be used to zoom to the operating range.

For example a value of 10 would mean that for each increment of 1 of the analog output the input of the clean measurement goes down by 10. In such a configuration the sensitivity and responsivity improves by a factor 10.

The multiplier value can be set through the command line interface. Convert the multiplier to a floating point number, resulting in 4 hexa-decimal 8 byte values. E.g. for a value of 10 it would be 41 20 00 00.

This can be loaded to the dictionary with:

```
it.connect(x,9600)
it.oop()
test=it.toModulePacket()
test.register(0xFD)
test.data(0xC4E7)
it.packet(test)
test.data(0xC000+0x41)
it.packet(test)
test.data(0xC4E8)
it.packet(test)
test.data(0xC000+0x20)
it.packet(test)
test.data(0xC4E9)
it.packet(test)
test.data(0xC000+0x00)
it.packet(test)
test.data(0xC4EA)
it.packet(test)
test.data(0xC000+0x00)
it.packet(test)
test.data(0xCFFF)
it.packet(test)
```

This sequency can also be saved in a text file (test.txt) and be executed as it.script('test.txt') within the CLI.

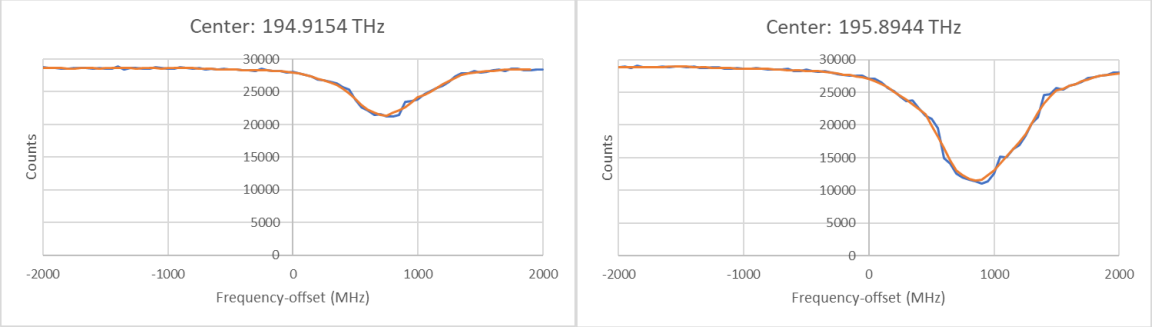
2. Case Study

We modified our PPCL590 gascell based configuration to serve as a case-study. The special measures to increase the sensitivity of the signal have been removed and the photo-diode current on the gascell is linearly converted into a voltage signal. The circuit is tuned to ensure that 16dBm of output power provides about 50,000 counts on the ADC input (1.9V).

The testing is performed with the timestamp 30726 firmware.

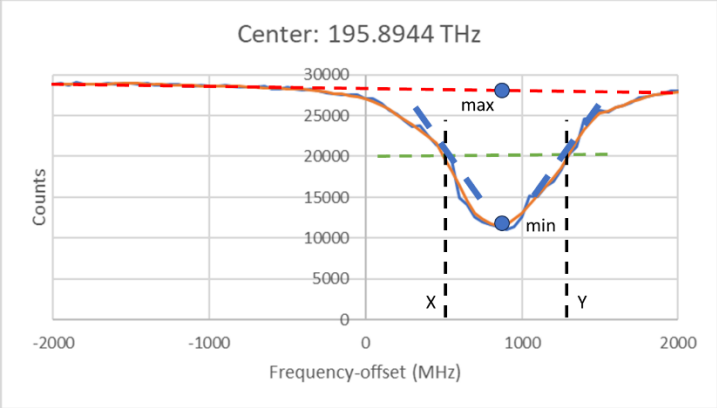
First we characterize the gas-cell. To do this we set the laser to each known gascell absorption peak and scan the frequency with the use of the FTF function (register 0x62). We scan the FTF from -2000 to + 2000 (-2GHz to +2GHz) around this frequency. We do this at 3 power levels (7dBm, 13.5dBm, 16dBm), though in our configuration each will give the same result (we use the 3 tests at each frequency to average our readings).

The below figure shows the measured response at 2 different frequencies (one with a deep absorption and one with a less deep absorption). Blue is actual data, orange is an averaged value.



We process such data by averaging and finding the frequency offset points (X, Y) where the absorption is half of the max absorption. We also determine the slope (in counts/MHz) at those points. That data allows us to determine at each peak:

- Frequency = center + X or center + Y
- Slope of the dip in 1/MHZ (negative at X and positive at Y), calculated as counts per MHz, divided by the counts at the X,Y (max – 0.5*(max-min))
- Depth of the dip = 1- 0.5*(max - min) / max
- Width of the dip = (Y-X)



For the peaks in the graph the following values are use:

Frequency	Slope	Depth	Width (* 10 MHz)
194.91585	- 0.000525	0.88	65
194.91650	+0.000495	0.88	65
195.89488	- 0.001038	0.71	78
195.89567	+0.001020	0.71	78

We then convert these values into integer values. See below for the conversion on the two graphs before. **Note the frequency expressed as an offset from 190 THz.**

Frequency	Slope	Depth	Width (* 10 MHz)
409D4EA5	BA09C009	3F612777	41
409D53F8	3A01D97F	3F612777	41
40BCA2E2	BA881B00	3F357D68	4E
40BCA94D	3A85B31A	0.71	78

For the exact conversion, please refer to:

- https://www.h-schmidt.net/FloatConverter/IEEE754.html.

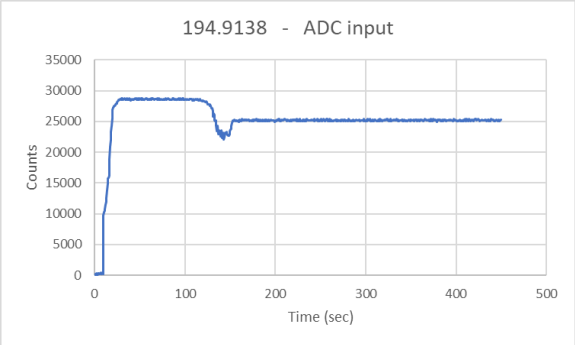
We then upload these values to the memory. We do this by addressing (read) register 0xFD. We start with the data for channel 2 and proceed channel by channel (16 integers per channel).

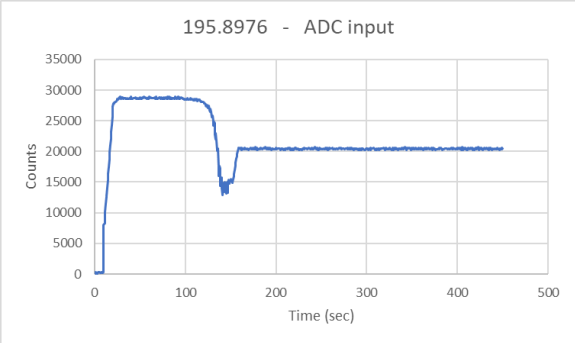
After loading the desired data we end with a read of register 0xFD with value (0xFFFF). That will perform the load of the values to the non volatile memory.

After this the device should be set for operating in locked mode.

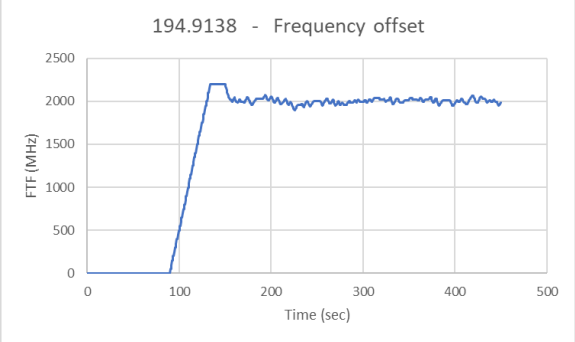
We ran the test with this data on our demo device and enabled the different channels and monitored the output for 400 seconds. Below are the outputs for two channels that are the left hand slope of the first absorption peak and the right hand slope of the second absorption peak.

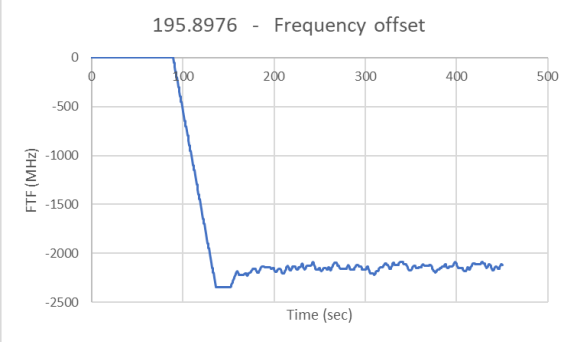
We read the reading from the external system through register 0x94. As per the below graphs, the device activates away from the main peak (transparent state, around 30,000 counts for 13.5dBm). Then the device starts to move the frequency towards the absorption peak until it has dropped to the desired spot on the slope. After that it locks into that power level and stabilizes.



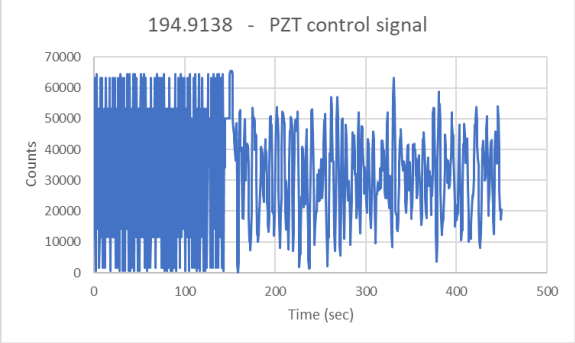


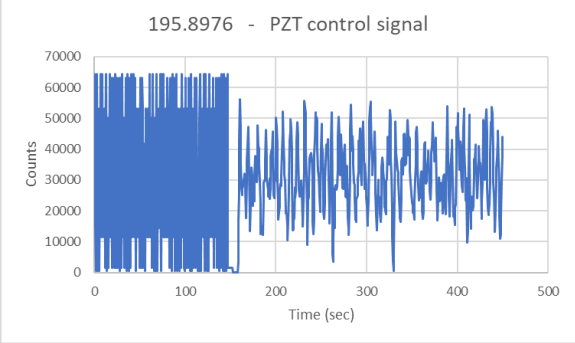
We monitor the FTF value through register 0x62. On the left hand slope, The device starts 2000MHz below the target value and moves up in frequency until it locks. On the right hand slope the device starts 2000MHz above the target value and moves down. After stabilization you can see the FTF value moving around, to ensure that the feedback signal to the PZT remains centered.



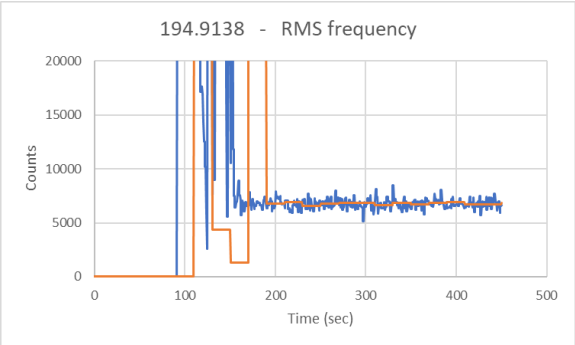


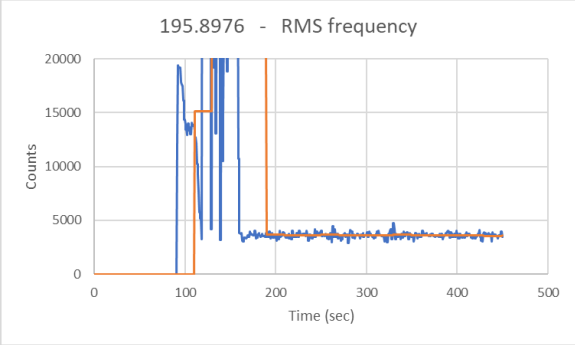
We monitor the output to the PZT through the register 0xFD with datafield 0x9001 (read operation). We can see that the PZT is working hard to keep the signal close to the target. Most importantly the correction signal stays within the range of the control (0-65535) and does not rail.



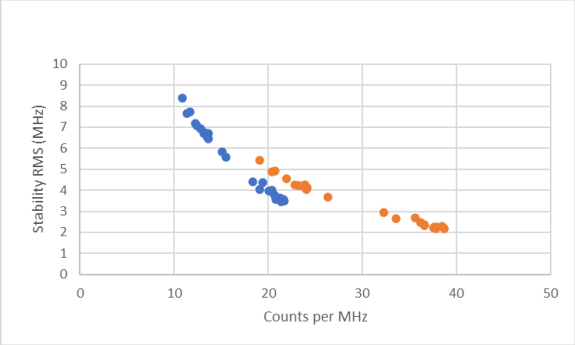


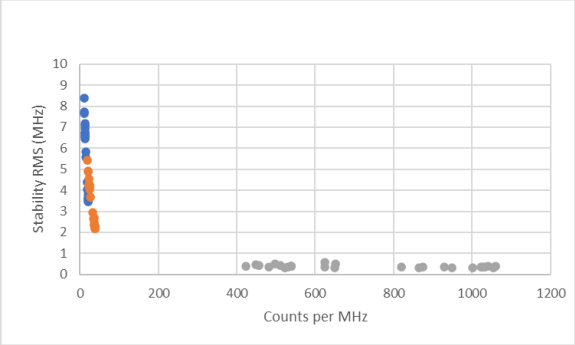
The module keeps measuring the frequency error (based on the ADC measurement and the loaded slope) and keeps an RMS error value. When reading register 0x96 with value 0 we get the average over 1 second (blue) and with value 1 we get the average over 20 seconds (orange).





Based on the data the module successfully locks at the different channels and stabilizes the frequency. The main key to stabilization is the counts per MHz in the slope. More resolution (higher value) gives a better stability. For these two peaks at 30,000 count, the slope is respectively (30,000*0.0005 =) 15 and (30,000*0.001=) 30. Plotting the observed stability versus the slope (blue, 13.5dBm), we get the below relationship. We also included tests at 16dBm (orange, with a higher count, the slope is higher).





For further reference, the data for our PPCL590 is included in the second graph so show capabilities with higher resolution. Also, it likely shows that beyond 150 counts per MHz other effects are limiting the frequency stability that can be realized.

3. Appendix

The following firmware timestamps support this functionality. A higher timestamp indicates that the firmware is compiled later, which makes it likely that the firmware is more mature or has more functionality.

The timestamp can be obtained from the release register. This returns a string with at the end TX xxx.yyy.000. The timestamp is xxx*256+yyyy.

- 28316 (a previous version; do not use anymore)
- 30765 (version used for case-study. Does not yet allow for the use of peaks)
- 33581 released version for PPCL590