PDH_Lock_5001 Instruction Manual Rev0.1

Benjamin M. Sparkes

20 May 2011

This code is released under the Creative Commons Attribution-NonCommerical-ShareAlike 3.0 Licence (`http://creativecommons.org/licenses/by-nc-sa/3.0/`). If you do use the code and feel that it has contributed significantly to any work accepted for publication we ask that you acknowledge the following work:

B. M. Sparkes, H. M. Chrzanowski, D. P. Parrain, B. C. Buchler, P. K. Lam, and T. Symul, *A scalable, self-analyzing digital locking system for use on quantum optics experiments*, arXiv:1105.3795v1 (2011).

We are interested in cataloguing modifications made to the code. If you do modify the code in a substantial way we would appreciate it if you would notify us and send us a copy of the code so that we can place this on our website for others to download.

If you have any comments, suggestions or queries please send an e-mail to ben.sparkes@anu.edu.au.

Contents

Appendix A

PDH Lock 5001 Instruction Manual - Rev. 0.1

A.1 Introduction

The goal of this work is to provide an easily modifiable code to lock multiple cavities and interferometers for quantum optics experiments. This code uses the Pound-Drever-Hall locking technique [4, 7] to allow for automatic re-locking. The code also includes a White Noise Generator that can be used to investigate the frequency response of the locks, as well as sequential locking logic. For more detail on the background and motivation for this work, please see Ref. [8] attached.

A.2 Set-up

This code requires National Instruments LabVIEW 2010® (32 bit) to run. Other requirements and basic set-up are detailed below.

A.2.1 Hardware

Below is a list of equipment that has been used while developing the *PDH_Lock_5001* code. The bracketed number indicate the numbers required for 8 locks.

- PXI chassis - NI PXI-1042Q with PXI-8106 embedded controller[1];
- FPGA cards - NI PXI-7852R (x2);
- Frequency generator - NI PXI-5404 (x1);
- AIP/AOP break out box (BoB) - made in house, 8 analog inputs and 8 analog outputs using Connector 0 line from FPGA (x2), also FPGA 1 needs to have 8 digital output lines (DIO0-7) connected for clock programming protocols (see below);
- High speed ADC - AD9460BSVZ-80 (x8);
- Clock generator - AD9959/BZ (x2).

---

[1] Only 1 chassis and controller required for 16 locks.

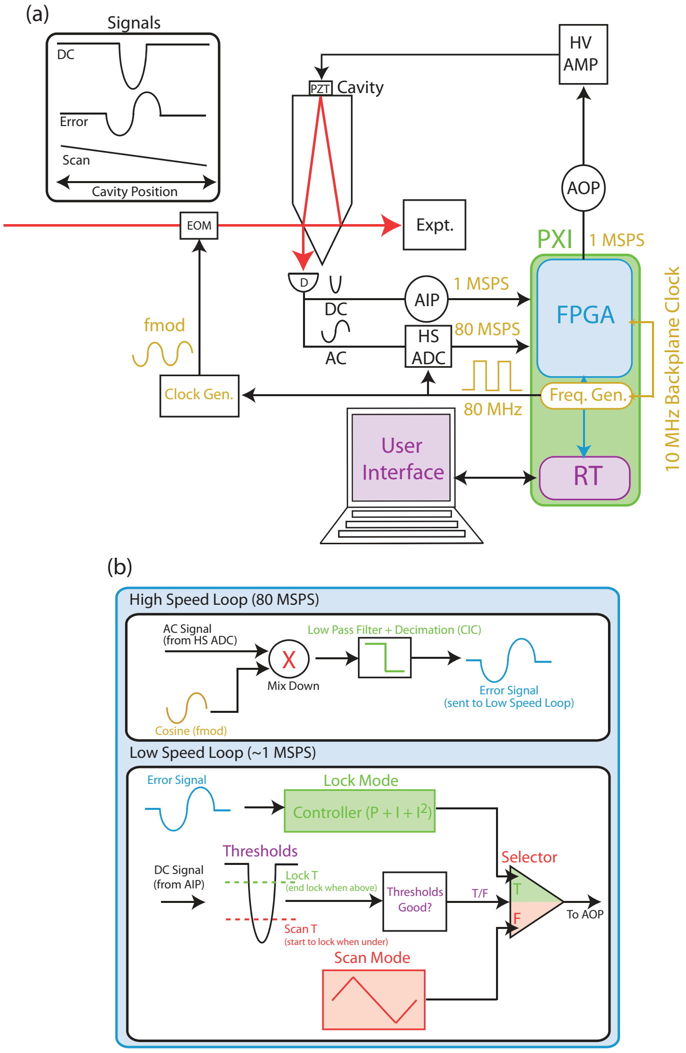


Figure A.1: Set-Up.(a) Depiction of the physical system and hardware required for digital locking using this code, in this case for a mode-cleaner cavity. PXI - PXI chassis, FPGA - field programmable gate array card; Freq. Gen. - frequency generator; RT - Real Time controller; AIP - analog input (sampling frequency < 1 M sample s^{-1}); AOP - analog output (sampling frequency < 1 M sample s^{-1}); HS ADC - High Speed analog to digital converter (80 M samples s^{-1}); D - detector; Clock Gen. - digital clock generation board; fmod - modulation frequency for lock; EOM - electro-optic modulator; PZT - piezo-electric transducer; HV Amp - high voltage amplifier; and Expt. - rest of experiment. Inset shows the different signals which are used for locking: DC - dc transmission/reflection signal,; Error - error signal; Scan - scan function (provided by FPGA code). (b) Depiction of LabVIEW® 2010 code used to program FPGA cards. In the high speed loop (running at 80 M samples s^{-1}) the input signal from the HS ADC is mixed down with a cosine signal at fmod (generated by a look-up table in LabVIEW®). The signal then passes through a low pass filter (LPF) and is decimated to produce the error signal. This error signal is, in turn, passed to the low speed loop (running at $\simeq 750$ k samples s^{-1}) where, depending on the value of the DC signal and the threshold limits imposed by the user, the system will move from Scan Mode, where the cavity is scanned using a sawtooth function, to Lock Mode, where a Proportional-Integral-Double Integral (PII) controller is used to keep the system on resonance with the laser. The code will move from Scan to Lock mode when the DC signal moves below the user defined Scan Threshold and will then remain in Lock mode until the DC signal rises above the Lock Threshold. If this occurs, the cavity will Scan again until the system returns to resonance, etc.

Figure A.1(a) shows the basic set-up for a single lock. In all cases the Freq. Gen. signal (at 80 MHz) must be split (in our case using a MiniCircuits splitter, model number: ZSC-2-1), with one output being sent to the clock generators and one to the HS ADCs, both of which are split again as required[2]. The modulated signal from the detectors (normally on an ac output) are sent one to each HS ADC. As each FPGA card has only two digital connectors (1 and 2), to send four digital signals from the ADCs to one FPGA requires combining the signals from two ADCs into one connector. In the code it is assumed that the first ADC of each pair is wired to lines DIO0-15 of the relevant connector and the second is wired to lines DIO16-31.

The low speed AIP/AOPs do not require clocking and will run between 1 M sample s^{-1} and 750 k samples s^{-1}. Normally 4 AIPs and 5 AOPs are used per BoB (AIP0-3, AOP0-4): the four inputs for the DC signal; and the first four outputs (AOP0-3) for the Scan/Lock signal which are, in turn, sent to the High Voltage amplifier. AOP4 carries a scan trigger for external hardware (see Section A.7.2 for more details). Also, at least the first (master) FPGA BoB needs to have access to 8 digital lines. These are used to program two AD9959 DDS (i.e. clock generation) boards. Six digital lines (DIO0-2,4,6-7) are sent to all boards, with DIO5,3 sent to DDS boards one and two respectively (for more information on AD9959 programming please download the relevant documentation from the Analog Devices website [1]). The outputs from the Clock Gen. box are sent to the related modulators.

A.2.2 Software

As mentioned above, this code was developed using LabVIEW 2010® (32 bit) and therefore this version, or better, is required to run it. An overview of the FPGA code is shown in Figure A.1(b), and consists of two loops: a high speed loop running at 80 M samples s^{-1}; and a low speed loop running at approximately 750 k samples s^{-1} (limited by the AIPs/AOPs). In the high speed loop the modulated signals from the HS ADCs are demodulated with a cosine function generated using a look-up table at the same frequency as the modulations (i.e. fmod). These pass through a Cascaded-Integrator-Comb (CIC) filter [5] to produce error signals which are then sent to the low speed loop.

There are two main components to the low speed loop: a Lock Mode (consisting of a P, I and I^2 controller); and a Scan Mode (consisting of a sawtooth scan function). If the code is set to **AutoLock** the code will move from the Scan Mode to Lock Mode depending on the thresholds set for the DC signal: the code will start to lock when the DC signal dips below the Scan Threshold and will stay in Lock Mode until the signal rises above the Lock Threshold. If the lock has no DC component (i.e. a Michelson-style lock) the absolute value of the error signal can be used to set the thresholds instead (see Section A.4 for more information). The low speed loop includes a white noise generator (with the output also being sent to the same AOP as the Scan and Lock signals).

The *Master* and *Slave* FPGA codes differ in that the *Master* code contains a trigger for the Freq. Gen. (to try to minimise the issue of the ADC and FPGA running out of sync - though if this occurs then it can be rectified by varying the clock phase, see Section A.7.2) and also code to run two Clock Gen. boards. The *Slave* code does not contain this extra logic. For more than 8 locks the *Master* code should be compiled on the third FPGA (minus the Freq. Gen. triggering code) and the *Slave* code on the fourth.

---

[2] NB: To avoid clock phase issues between HS ADCs ensure that the same cable length and splitters are used for this secondary - as well as tertiary and quaternary - splitting. See Section A.8.

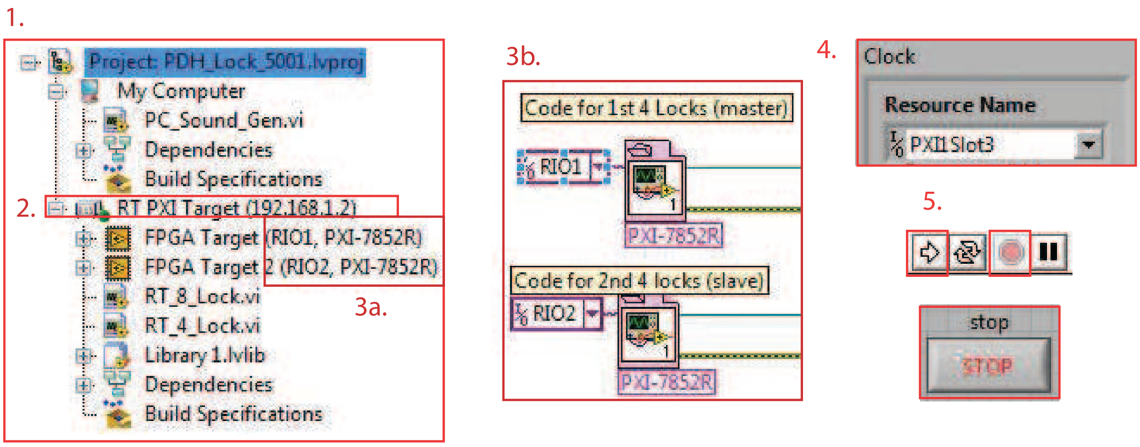


Figure A.2: First Time. The menus and controls which need to be altered before using the code for the first time. The numbers and boxes refer to the instructions in the text.

The Real Time code acts as the interface between the FPGAs and the user, as well as to allow communication between FPGAs for sequential locking purposes (see Section A.7.3). It also controls the Freq. Gen. cards. Only one RT code is required for all FPGAs on one chassis. Included in the *PDH_Lock_5001* project are RT codes for 4 and 8 locks. These can be extended to 12 or 16 without much additional coding.

A.3 First Time

The following instructions are for when opening the project for the first time. Before running for the first time make sure all components and drivers for LabVIEW® 2010 are installed.

1. Open the *PDH_Lock_5001* folder, select the *PDH_Lock_5001.lvproj* project file.

2. In the project folder right click on the *RT PXI Target* and select *Properties*. In this menu enter the correct name and IP address of your PXI chassis (if unsure check using Measurement and Automation Explorer - MAX).

3. Make sure that the RIO# of the FPGA cards are correct (check using MAX). To change, right click on the relevant *FPGA Target* and select *Properties*. If you do change the RIO# here you will also have to alter this in the RT codes by opening the RT code, using Ctrl E to bring up the Block Diagram, scroll to the far left hand side of the code and change the Dialog box before the relevant FPGA.

4. Check the **Resource Name** of the clock (located in the **Clock and Timing and Misc** tab of the **System Inputs/Outputs** folder) against that in MAX. Change if different.

5. To run the code select the white right arrow icon in the top left hand corner of the Front Panel. To stop the code use either the red octagon next to the start button (this is for force quit) or, better, use the **Stop** button included in the code located above the **System Inputs/Outputs** folder.

A.4 Locking the System

Below are listed the suggested steps to achieve lock using the *PDH_Lock_5001* code (assuming the hardware has been set-up correctly). An overview of the various menus and more in depth description of the various controls is included in Section A.7.

1. Make sure that the Scope (Located in the **Graphs** menu) is set to the correct lock using **Scope Trigger Channel** and the Scope is turned on (located in the **Scope** tab of the **system Inputs/Outputs** menu), this will allow you to see your progress on the computer screen (see Figure A.5). If the scope is not displaying or triggering correctly try increasing/decreasing the **Acquisition Rate (kHz)** and **Scope Time Out (ms)** controls (not shown).

2. On the **System Inputs/Outputs** menu select the **Lock** # tab of interest and then choose the modulation frequency of the lock via the **Freq (MHz)** control in the **Signal** cluster. This automatically changes the frequency sent to the relevant EOM by the DDS board as well as the demodulation frequency in the code[3].

3. Set the **Scan Amplitude (mVpp)**, **Scan Offset (mV)** and **Scan Freq Step** and **SFS Bit Shift** to achieve a reasonable scan range and rate (see **Scan Freq (MHz)** indicator), all controls are located in the **Controller** cluster. The range of the AOPs is -10 V → 10 V.

4. An error signal should now be visible of the scope. If not, check that the correct modulation is being sent to the EOM and that the right input signal is being received by LabVIEW®. To check if the input is working properly, try sending the modulation signal straight to the HS ADC input, which should produce a straight line on the Scope.

5. Once an error signal has been produced, use the **BS** control in the **Signal** cluster to increase/decrease the size of the error signal by factors of 2 (to avoid saturation).

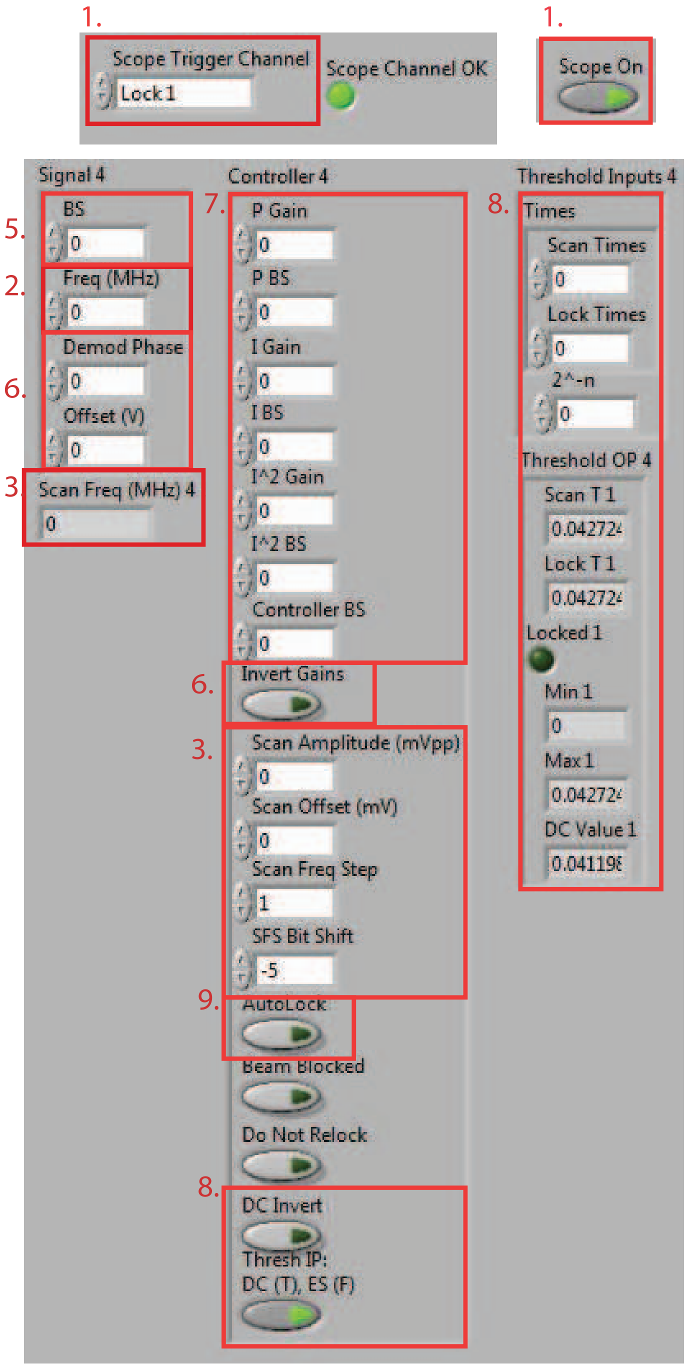
6. Maximise the size of the error signal using the **Demod Phase** control. If the error signal is 180^0 out of phase (see Figure A.1(a) inset), use the **Invert Gains** control to avoid having to use negative gains. If there is a constant DC offset when the phase is maximised, use the **Offset (V)** control to remove it.

7. Set some gain for the **P Gain** and **I Gain** components of the controller, as well as the **Controller BS** (normally starting around -15)

8. Set the **Scan Times** and **Lock Times** values (Scan Times $\leq$ Lock Times), as well as the **2∧-n** control to set the scan and lock thresholds (see Section A.7.1). Make sure that, if there is a DC signal for the lock, it is displayed on the scope. If not, then toggle the **Thresh IP: DC (T), ES (F)** switch on. If it is a Michelson-style lock, with no DC signal, the absolute value of the error signal can be used instead by turning the **Thresh IP: DC(T), ES(F)** switch to false. Also check that the DC signal is the correct way up (see Figure A.1(a) inset), if not then use the **DC Invert** control. The maximum, minimum and current values of the DC input, as well as

---

[3] To change the modulation depth of the signal, use the relevant **Amp IP Cluster** for the DDS board located in the **Clocks and Timing** tab, see Section A.7.2.



Scope Trigger Channel: Lock 1

BS: 0
Freq (MHz): 0
Demod Phase: 0
Offset (V): 0
Scan Freq (MHz) 4: 0

P Gain: 0
P BS: 0
I Gain: 0
I BS: 0
I^2 Gain: 0
I^2 BS: 0
Controller BS: 0
Scan Amplitude (mVpp): 0
Scan Offset (mV): 0
Scan Freq Step: 1
SFS Bit Shift: -5

Scan Times: 0
Lock Times: 0
2^-n: 0
Scan T 1: 0.04272
Lock T 1: 0.04272
Min 1: 0
Max 1: 0.04272
DC Value 1: 0.04119

Figure A.3: Locking the System. The menus and controls used to lock the system. The numbers and boxes refer to the instructions in text.

the threshold values, are shown in the **Threshold OP** cluster. Set the **I1 Max Value** and **I2 Max Value** thresholds in the **Clocks and Timing and Misc** tab (not pictured). This will cause the locks to reset if the I or I^2 controller values rise above these levels. The I and I^2 values can be viewed in the **Re-Ordered IP/OPs** folder in the **I Values** tab.

9. Set the thresholds using the **Threshold Inputs** cluster. The Scan and Lock thresholds are fractions of the size of the DC signal determined by **Scan Times** $\times 2^{-n}$ and **Lock Times** $\times 2^{-n}$. The higher the fraction, the closer to the bottom of the DC signal the threshold will be. The system will lock once the DC value goes below the Scan Threshold and remain locked until it rises above the Lock Threshold, therefore ensure that **Scan Times** $\geq$ **Lock Times**

10. Turn on the **Autolock** control. The system will do one full scan to determine the maximum and minimum DC values and calculate the Scan and Lock thresholds (see **Threshold OP** cluster) before trying to lock (see previous step). If the system does not lock, make sure that the phase is correct (see Figure A.1(a) inset for correct orientation for positive gains), if this is not the case try the **Invert Gains** control. If this does not work, then try decreasing (or increasing) the gains, decreasing the scan frequency, the scan amplitude, or decrease the scan and lock times. You may also want to check the dependencies of the lock to make sure that it is not **Beam Blocked** (see Section A.6). Once locked, to optimise the gains see Section A.5.2.

11. As long as **Autolock** is on then the system will either be locked or trying to find lock. If lock is dropped and the system cannot satisfy the set Scan and Lock thresholds (due to power drifting of the laser etc.) the maximum and minimum values will be retaken after a time **Lock TimeOut (s)** located in the **Clock and Timing and Misc** tab. If it is not desirable for the system to relock then select the **Do Not Relock** control. In this case, once the lock comes unlocked the code will move back to Scan Mode until it is turned off. Also, to draw attention to this, if the *PC_Sound_Gen.vi* code is run in parallel with the RT locking code, a sound will be made when a lock has come unlocked and **Do Not Relock** is active for it.

12. Finally, if you desire to stop the lock exactly where it is and freeze the maximum and minimum DC values (for instance if you will be blocking the beam before the lock), select **Beam Blocked**.

A.5 Lock Optimisation

In this section we will go through a brief explanation of how the code is designed to allow for lock optimisation.

A.5.1 Control Theory

Figure A.4(a) shows the basics of a closed loop feedback system, as well as the definitions we will use in this section[4]. The controller transfer function H, used for calculating GH

---

[4] Note that we define the system bandwidth to be the unity gain frequency, and the phase margin follows from the Nyquist stability criterion [6].

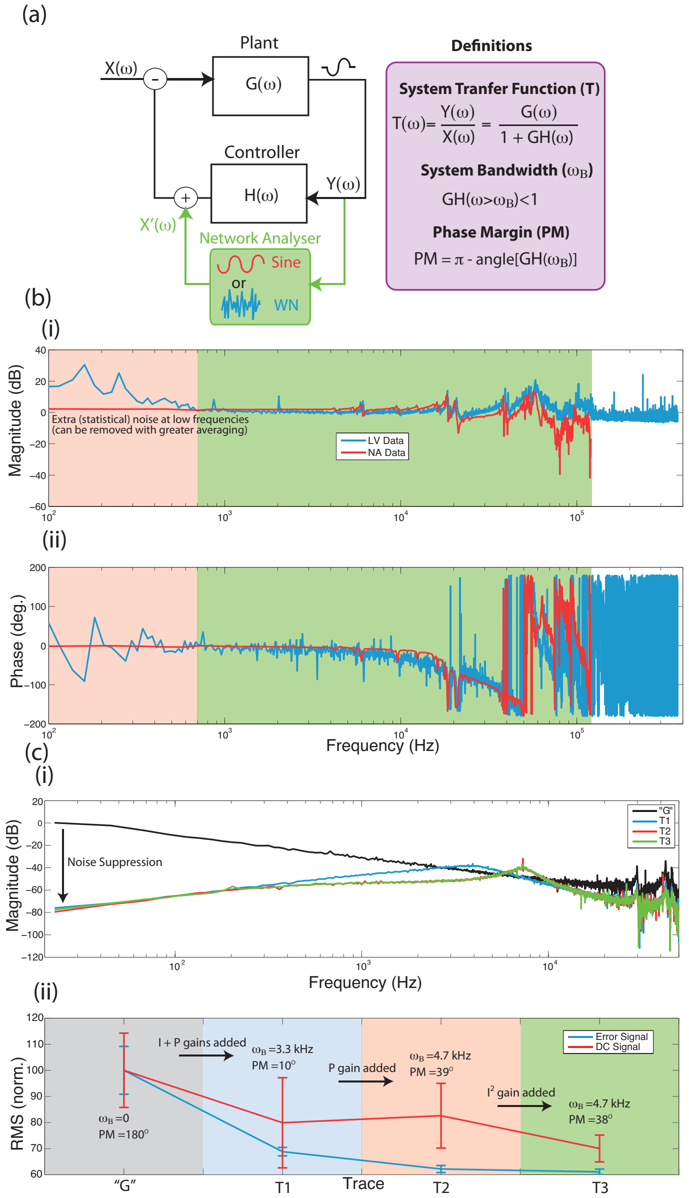


Figure A.4: Lock Optimisation. (a) Block diagram of a general closed loop feedback system with a Plant (the system to be controlled) with frequency response $G(\omega)$, and the Controller (to control the Plant using the output signal $Y(\omega)$) with frequency response $H(\omega)$. Also included is the noise added to the system $X(\omega)$. A network analyser (shown in green) can be used to measure the System Transfer Function $T(\omega)$ using either a swept sine wave (Sine) or white noise (WN), in both cases the size of the signal $X'(\omega) \gg X(\omega)$. The inset shows some of the definitions used to parametrise the locks. (b) Comparison of magnitude and phase plots for an unlocked mode-cleaner cavity using both a store bought network analyser (NA Data) and a white noise generator developed in LabVIEW® (LV Data). The traces agree very well, except for some excess noise at low frequencies for the white noise generator. (c)(i)The magnitude of T measured with different controller gains, with G being only very lightly locked to approximate the Plant and increasing gains used for $T1$ to $T3$, to optimise the lock. The suppression of noise at low frequencies with the PII controller can be seen here. (ii) Root Mean Squared (RMS) measurements for both the error and DC signals for each of the traces in (i). These were taken with extra white noise present in the system and again $X'(\omega) \gg X(\omega)$. The error bars were determined from the standard deviation of the 10 values used for each set of gains. Also shown are the bandwidths and phase margins measured by the LabVIEW® code as part of its locking analysis.

which determines the system bandwidth and phase margin, is a simulated H using the following equation [3]

$$H_{sim}(z) = k_P + k_I \cdot \frac{z}{1+z^2} + k_{II} \cdot \left(\frac{z}{1+z^2}\right)^2, \tag{A.1}$$

(where k are the gains, $z = exp(if/f_s)$, and f_s is the sampling frequency of the controller system). Another way of determining H is by calculating it through G and T, $H_{calc} = (G - T)/(GH)$, however this does not work once the controller gain drops below a certain level (close to unity gain).

A Network Analyser (shown in green) can be used to measure the magnitude $(20 \cdot log_{10}|T|)$ and phase plots of the system. In our system we use a white noise generator function contained in the LabVIEW® code and average over many cycles, whilst a store bought network analyser[5] usually uses a swept sine wave. Figure A.4(b) shows a comparison of these methods with good agreement at higher frequencies ($\geq$ 700 Hz). There is some statistical noise at low frequencies for the LabVIEW® white noise generator, but this can be improved by increasing the number of averages taken for measurements. Measuring the Plant transfer function $G(\omega)$ can be difficult as it requires the system to be unlocked, therefore allowing it to drift off resonance. Figure A.4(b) was taken using a Michelson interferometer set up at the rear of a cavity, not requiring the system to be locked. However, a very small amount of gain can also be used to give an approximation of G and this is what is used for the lock optimisation protocol discussed below along with H_{sim}.

A.5.2 Lock Optimisation Protocol

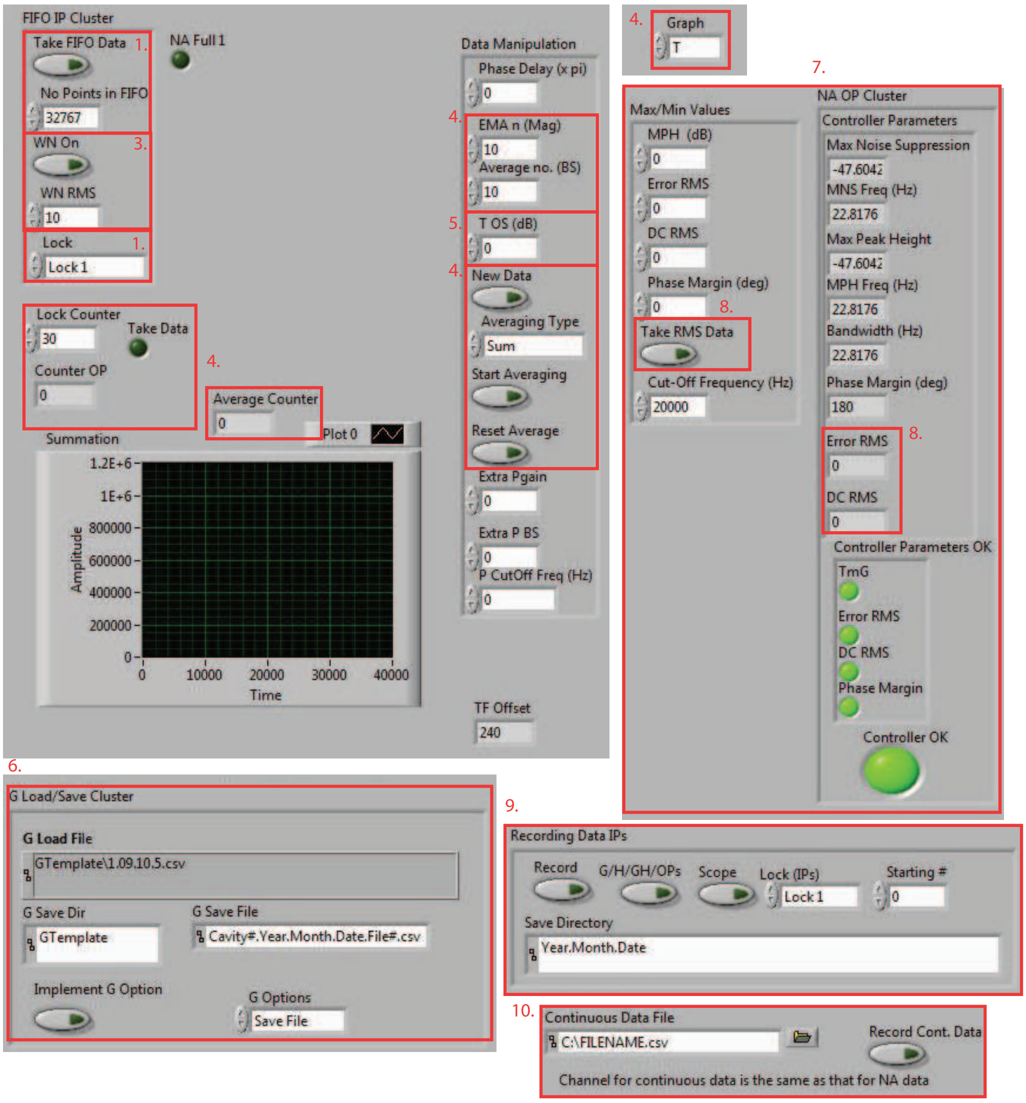
The idea behind the lock optimisation protocol is to find the locking parameters which allows for: large bandwidths ω_B; large noise suppression at frequencies below ω_B; stable locking (as measured by the phase margin); no (or minimal) excitation of resonances; and low RMS on the DC and error signals. This is achieved here mainly by investigating the transfer functions of the plant and the system (see Figure A.4). Below is a suggested method for using the code to help optimise the locking parameters if no previous G trace has been taken.

1. Turn the network analyser on using the **Take FIFO Data** switch in the **FIFO IP** cluster. Select the correct **Lock #** and make sure that **No Points in FIFO** is non-zero[6].
2. Lock the system of interest using as little gain as possible (ideally hardly any I, and definitely no I^2).
3. Use **WN On** to start sending WN to the channel of interest. For best results make sure that **WN RMS** is much larger than the system noise.
4. To display data on the **Bode Plots** tab of the **Graphs** menu make sure **Graph** is set to T and turn on the **New Data** switch in the **Data Manipulation** cluster and also click the **Start Averaging** control. There are two different averaging methods available in this code - a summation (*Sum*) which will only display new data after all traces have been collected (chosen using the **Average no. (BS)** control for

---

[5] Here, a Model MS4630B from Anritsu.

[6] For best results use the maximum number of FIFO points: 32767.



FIFO IP Cluster
1. Take FIFO Data
No Points in FIFO: 32767
NA Full 1
3. WN On
WN RMS: 10
1. Lock: Lock 1
Lock Counter: 30
Take Data
Counter OP: 0
4. Average Counter: 0
Summation
Plot 0

Data Manipulation
Phase Delay (x pi): 0
4. EMA n (Mag): 10
Average no. (BS): 10
5. T OS (dB): 0
4. New Data
Averaging Type: Sum
Start Averaging
Reset Average
Extra Pgain: 0
Extra P BS: 0
P CutOff Freq (Hz): 0
TF Offset: 240

4. Graph: T

7.
Max/Min Values
MPH (dB): 0
Error RMS: 0
DC RMS: 0
Phase Margin (deg): 0
8. Take RMS Data
Cut-Off Frequency (Hz): 20000

NA OP Cluster
Controller Parameters
Max Noise Suppression: -47.6042
MNS Freq (Hz): 22.8176
Max Peak Height: -47.6042
MPH Freq (Hz): 22.8176
Bandwidth (Hz): 22.8176
Phase Margin (deg): 180
8. Error RMS: 0
DC RMS: 0
Controller Parameters OK
TmG
Error RMS
DC RMS
Phase Margin
Controller OK

6. G Load/Save Cluster
G Load File: GTemplate\1.09.10.5.csv
G Save Dir: GTemplate
G Save File: Cavity#.Year.Month.Date.File#.csv
Implement G Option
G Options: Save File

9. Recording Data IPs
Record
G/H/GH/OPs
Scope
Lock (IPs): Lock 1
Starting #: 0
Save Directory: Year.Month.Date

10. Continuous Data File: C:\FILENAME.csv
Record Cont. Data
Channel for continuous data is the same as that for NA data

Figure A.5: Lock Optimisation Protocol. The menus and controls used to investigate and optimise a lock. The numbers and boxes refer to the instructions in text.

2^{BS} traces), or an exponential moving average (*EMA*) filter (with α chosen using **EMA n (Mag)**) which will constantly update but will be biased towards more recent traces. These (or no averaging) can be chosen using the **Averaging Type** control. Data will only be taken after the system have been locked for the number of counts determined by the **Lock Counter** control and displayed on the **Counter OP** indicator, this can be seen from the **Take Data** indicator. If the data on the **Bode Plots** tab vanishes, or if the **Average Counter** won't increase when **Take Data** is true, then use the **Reset Average** to start again (this problem normally occurs when the WN is switched off and then switched on again).

5. Once a trace has been created, turning **New Data** off will preserve it (if so wished). Sometimes extra gain will be present in the system, to normalise the G trace use the **T OS (dB)** control to centre the trace.

6. Once happy with the G trace, use the **G Load/Save** cluster, located in the **Saving Data** tab, to select a directory and file name to save to on the PXI hard drive. Select *Save File* in the **G Option** box and then click the **Implement G Option** switch to implement the save. This will also load the trace to the G trace bode plot (which can be viewed using the **Graph** control on the **Bode Plots** tab of **Graphs** menu. Previously recorded G traces can also be loaded here.

7. Now increase the P, I, and I^2 gains to improve the lock. The tools located in the **Controller Measures** tab can be used to monitor properties of the lock such as maximum noise suppression, system bandwidth and phase margin, as well as looking at lock RMS (see next step). The **Max/Min Values** in the **Controller Measures** tab (see Section A.7.6) allow you to set limits on various parameters and the indicators let you know whether these have been achieved. The different traces can also be viewed in the **Bode Plots** tab. As with G, there may be extra gain on the controller. To take this into account, use the **Comparison Plots** tab of the **Graphs** menu to see the difference between the H simulated through Equation A.1 and H calculated through T and G (see Section A.5.1). The **Extra Pgain**, **Extra P BS**, and **P CutOff Freq (Hz)** can then be used to match the two. An example of lock optimisation is shown in Figure A.4(c). Normally this involves adding I, then P gains in an iterative process to increase noise suppression and bandwidth without exciting resonances, increasing RMS (see Section A.7.6), or lowering the PM too much, followed by I^2 gain to again increase noise suppression at low frequencies. No derivative component is included to avoid exciting resonances at higher frequencies.

8. The RMS of both the error signal and DC signal of the system can be investigated by clicking the **Take RMS Data** switch in the **Max/Min Vales** cluster of the **Controller Measures** tab. This information is then shown in the **NA OP Cluster** cluster. The RMS data is taken from the Scope, therefore to use this you must ensure that **Scope Trigger Channel** is set to the desired lock. Also, ensure that the **WN RMS** is the same when comparing RMS for different gains.

9. T, H, GH and the **Controller Parameters**, as well as scope traces and inputs for a lock can also be saved from the **Recording Data IPs** cluster in the **Saving Data** tab. Once a save directory is selected files will save in numerical order starting from 0. The saving path and next file number will be displayed in the **Recording Data OPs**. You can alter the next file number using the **Starting #** control in

the **Recording Data IPs** cluster. If the file already exists the **Next File already exists** indicator will light up.

10. To investigate the long term stability of the system, the code can also take continuous data for one lock at a time in approximately 1 s time intervals. The channel that will be saved is the same channel as selected in the **FIFO IP CLuster** of the **Transfer Functions** tab.

11. To recover the saved files from the PXI hard drive you can either use the LabVIEW® Meausrement and Automation Explorer (MAX) or a file transfer protocol application such as Filezilla®.

A.6 Sequential Locking

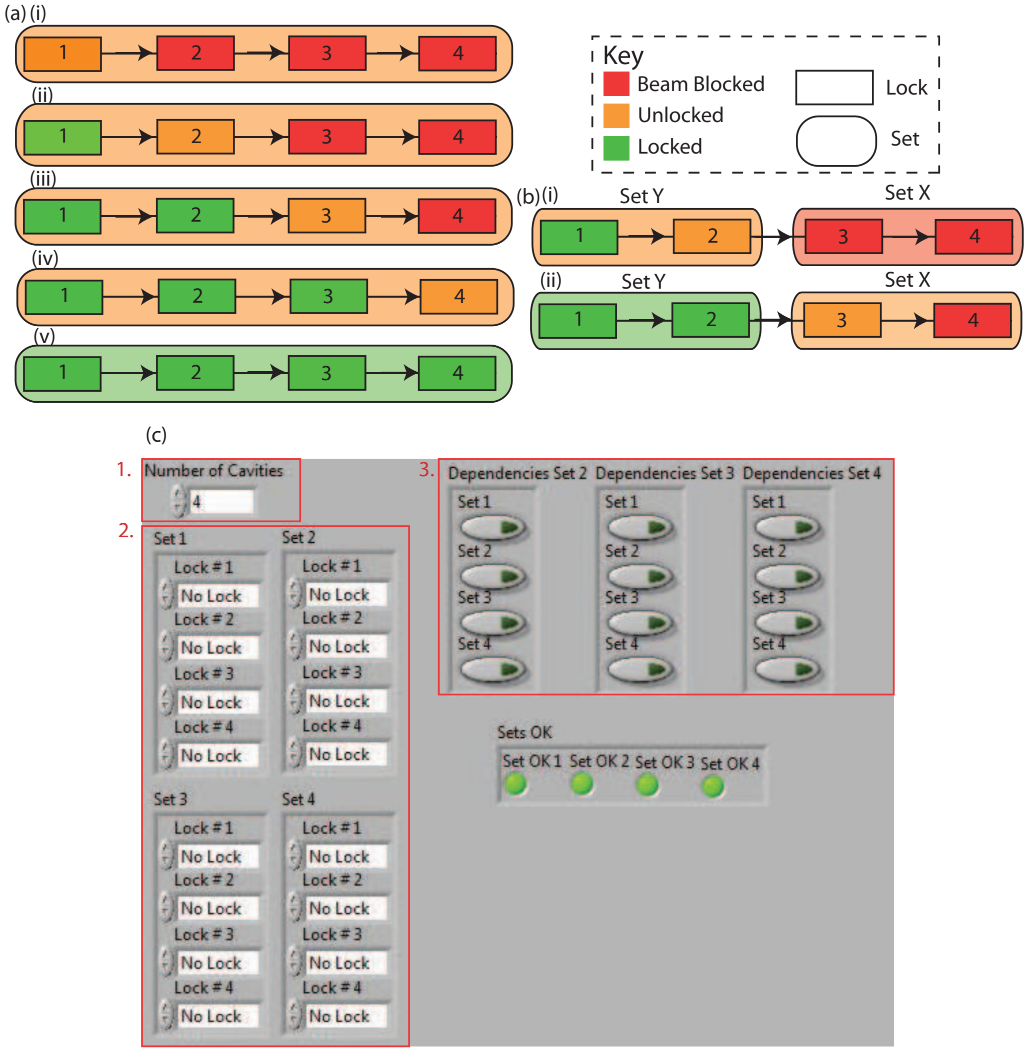
The sequential locking aspect of the code is handy if the experiment to be controlled contains many locks that depend on previous locks. The main idea behind this logic is that if any of the locks that a subsequent lock, Lock X, depends on drop lock, then Lock X will become automatically **Beam Blocked** until all the locks it depends on re-lock. This is illustrated in Figure A.6(a). For maximum flexibility we group locks in to sets of locks. The dependence between locks in a set is defined to be linear: with Lock #4 depending on Lock #3, which depends on Lock #2, which depends on Lock #1. Sets, on the other hand, can depend on multiple other sets and all locks within a set will be **Beam Blocked** unless all locks within the sets that it depends on are locked. This is illustrated in Figure A.6 (b).

Below are the suggested steps for setting up the sequential locking for your system. All controls are situated in the **Sequential Locking** tab of the **System Inputs/Outputs** folder.

1. Set the number of locks present in the system with **Number of Cavities**

2. Set 1 is defined within the code to be the first set and therefore cannot depend on any other set. Keeping this in mind, place all locks within the provided sets. The best way of doing this is to place all those lock that follow sequentially into one set using the **Set #** clusters or, if more than four locks follow sequentially, into multiple sequential sets in the order that the locks follow on the experiment. Once you come to a lock that depends on two subsequent locks then put this lock as the first lock of a new set and continue in this fashion until all locks have been entered into a set. The sets don't have to be full, if this is the case then put *No Lock* in the remaining **Lock #X** boxes. Check the **Sets OK** indicators. If a set is not OK, make sure that one lock does not appear twice in one set, or in two different sets. Also check that you have not put more locks in sets than you selected in **Number of Cavities**. Also, *No Lock* cannot be selected as anything but the last locks in a set. Sets can span FPGA cards.

3. Now set the dependencies of the sets using the **Dependencies Set #**. If a set follows sequentially from one other set then simply select the one set, if the first lock in a set depends on two other sets then select two, etc. Check the **Sets OK** indicators are all still lit. If not make sure that there is no interdependencies (i.e. Set X depending on Set Y, which depends on Set X), and that no set depends on itself.



Number of Cavities 4

Set 1
Lock # 1 No Lock
Lock # 2 No Lock
Lock # 3 No Lock
Lock # 4 No Lock

Set 2
Lock # 1 No Lock
Lock # 2 No Lock
Lock # 3 No Lock
Lock # 4 No Lock

Set 3
Lock # 1 No Lock
Lock # 2 No Lock
Lock # 3 No Lock
Lock # 4 No Lock

Set 4
Lock # 1 No Lock
Lock # 2 No Lock
Lock # 3 No Lock
Lock # 4 No Lock

Figure A.6: Sequential Locking. A schematic diagram of the principles behind the sequential locking logic developed for the code for (a) a single set of four locks with Lock 4 depending on Lock 3 depending on Lock 2 depending on Lock 1 (as indicated by the direction of the arrows), and (b) for two sets of two locks with Set X depending on Set Y. (c) The controls used to set up the sequential locking. The numbers and boxes refer to the instructions in text.

A.7 Menu Overview

This section provides an overview of the various menus located in the *RT_#_Lock* codes, with more in depth descriptions of the controls they contain. Most of these are located in the **System Inputs/Outputs** folder, but some are also located in the **Graphs** and **Re-ordered IP/OPs** folders.

A.7.1 Lock

Each lock menu (one for each lock present in the system) contains three input clusters and one indicator cluster. These are used to determine all parameters with respect to the individual, lock including controller gains, fmod and locking thresholds.

Signals

This cluster is for controlling the modulation/demodulation process, with:

- **BS** - the bit shift of the error signal (signal $\times 2^{BS}$) to ensure that it remains within the ±10 V range on the scope (i.e. it does not saturate the I16 initial number range for the error signal in the slow loop);
- **Freq (MHz)** - the modulation/demodulation frequency for the signal. This value has a set step size of 0.3125 MHz to ensure that the FPGA sine look-up table and Clock. Gen. card are always at exactly the same frequency. The maximum frequency that should be used is 40 MHz, though staying below 35 MHz is advised to ensure a clean frequency output from the AD9959 card;
- **Demod Phase** - Changes the phase of the demodulating cosine (in degrees) to allow for optimisation of the error signal. The optimal phase may change between runs and so it is advised to optimise it each time the code is run;
- **Offset (V)** - due to mis-alignment etc. the error signal may not be centred around 0 V on the scope and so an artificial offset may need to be added to rectify this problem, and allow for locking to the resonance point.

Controller

The controller cluster controls the relative gains for the various parts of the controller, as well as the scan function and whether or not to lock the system, with:

- **P Gain** - gain for proportional controller;
- **P BS** - Bit Shift of P signal (k_p =PGain$\times 2^{PBS}$);
- **I Gain** - Gain of single integrator controller;
- **I BS** - Bit Shift of I signal (K_i = IGain $\times 2^{IBS}$);
- **I∧2 Gain** - Gain of double integrator controller;
- **I∧2 BS** - Bit shift of signal out of single integrator before entering double integrator (k_{ii} =I∧2 Gain $\times 2^{I\wedge 2BS}$);
- **Controller BS** - Common BS for P, I, and I^2 controllers;

- **Invert Gains** - If the signal is 180^o out of phase (i.e. the high side of the error signal is on the high side of the scan, the opposite of Figure A.1(a) inset) then, instead of adding/subtracting 180^o from the phase in the **Signals** cluster or using negative gains, this button can be set to true;

- **Scan Amplitude (mVpp)** - Amplitude of the sawtooth scan function in mVpp. The maximum amplitude is 20,000 mVpp, and this is only possible is **Scan Offset (mVpp)** = 0 (i.e. maximum AOP range= ±10 V);

- **Scan Offset (mVpp)** - Offset of scan from a 0 V centre point;

- **Scan Freq Step** - Determines the scan frequency in concert with **SFS Bit Shift** (see below);

- **SFS Bit Shift** - Bit shift for scan speed, with **Scan Freq (MHz)** being given by $\frac{80}{108\times1000} \times SFS \times 2^{SFSBS}$. See **Scan Freq (MHz)** indicator for actual scan frequency;

- **AutoLock** - If true the system will scan its full range once to set the Min and Max DC values for the Scan and Lock thresholds, before starting to lock. The system will move to Lock Mode when the DC value drops below the Scan Threshold, and will remain locked until it rises above the Scan Threshold (see below), when it will revert to Scan Mode. If the system cannot find lock after the amount of time set by **Lock TimeOut** (see Section A.7.2) then the system will undergo another complete scan to reset the maximum and minimum DC limits and try again;

- **Beam Blocked** - If the beam is going to be blocked before the cavity, set Beam Blocked to true to freeze the output signal at its current position (i.e. no change to locking signal or scan), as well as preserve the current threshold limits until it is set to false again;

- **Do Not Relock** - If you do not want to code to re-lock a lock once it has come unlocked, then select this control. When the lock comes unlocked the system will move the Scan Mode until **Do Not Relock** is deselected. Also, when this occurs, a signal is sent to the *PC_Sound_Gen.vi* code to make a sound and alert the user to this event;

- **DC Invert** - The current way the thresholds are set up is for a DC signal such as that shown in Figure A.1(a) inset, with the trough below the background signal. If the DC signal, due to detector gain/using a transmission signal etc., is inverted (i.e. a peak rather than a trough) then, by turning DC Invert to true it will invert the signal (i.e. $\times - 1$), allowing the thresholds to remain the same and **AutoLock** to function without the need to recompile the code;

- **Thresh DC(T), ES(F)** - Allows the threshold for **AutoLock** to be set from either the DC signal (T) or the absolute value of the error signals (F).

Threshold Inputs

Controls the scan and lock thresholds.

- **Scan/Lock Times** - Sets the Scan/Lock thresholds as a fraction of peak height (see next). Ideally the Scan Threshold is set to be as high as possible to avoid locking to any peaks aside from the main peak, while the lower the Lock Threshold is, the less likely the system is to move back in to Scan Mode while the system is still locked (i.e. Error Signal $\neq 0$ outside of resonance). If no DC signal is available for a lock (for instance a Michelson style lock) and/or you just want the system to lock without unlocking, then set both to 0;
- **2^{-n}** - Sets n. The Scan and Lock Thresholds are set by **Scan/LockTimes** $\times 2^{-n} \times$(Max-Min) (i.e. as a fraction of the peak height). The higher they are, the closer the thresholds are to the bottom of the peak.

Threshold OP

Shows the threshold data.

- **Scan T** - The Scan Threshold limit, determined by **Scan Times** $\times 2^{-n} \times$(Max-Min). The system will enter Lock Mode when the DC value drops below the Scan Threshold;
- **Lock T** - The Lock Threshold limit, determined by **Lock Times** $\times 2^{-n} \times$(Max-Min). The system will leave Lock Mode when the DC value rises above the Lock Threshold;
- **Locked** - Tells whether the system is currently in Scan Mode (F) or Lock Mode (T), and therefore whether the system is on resonance;
- **Min** - The minimum DC value, measured in V, used to calculate thresholds (see above);
- **Max** - The maximum DC value, measured in V, used to calculate thresholds (see above);
- **DC Value** - Current DC value of the system, measured in V.

Scan Frequency (MHz)

- Displays the frequency of the scan, in MHz, calculated from **Scan Freq Step** and **SFS BS** as described above.

A.7.2 Clocks and Timing and Misc

This tab controls the frequency generator, as well the clock generator signals. Also included are the lock timeout function, output scan trigger control, and indicators relating to FPGA code timing.

Clock

Controls for Freq. Gen.

- **Resource Name** - Normally this is just the slot on the PXI chassis in which the Freq. Gen. card is located (i.e. *PXI1Slot#*). To check, use the Measurement and Automation Explorer (MAX);

- **Frequency** - This should be left at 80 MHz for clocking the ADCs. All Clock Gen. programming assumes an 80 MHz input clock;
- **Amplitude (V)** - The amplitude of the Freq. Gen. card has a small range (see LabVIEW® documentation [2]);
- **Phase** - Phase of Freq. Gen. If the input signals from the HS ADCs appear overly noisy or small, altering the clock phase may be able to fix this (see Section A.8).

Clock Board IP Cluster

Controls for all Clock Gen. boards.

- **Clk Loop Speed** - Speed at which the clock programming loop on the Master FPGA runs. This should always be greater than 107, but as the clock boards do not require fast programming, a value of 100,000 seems to work fine;
- **TTL Multiplier** - for technical reasons, the AD9959 boards used produce a much nicer output if the input 80 MHz signal is multiplied by a factor of 4. It is recommended to keep this value though it can be altered (see AD9959 documentation [1]);
- **Clk Reset** - If, for any reason, the clock generator stops working then resetting it may fix the problem. If this does not work, then the code should be stopped and rebooted. If this does not work, stop the code, turn the clock power supply off and then back on, and then start the code again (see Section A.8).

Amp/Phase Clusters

These are controls for individual clocks (as indicated). These clusters set the amplitude of the modulation signal (with a maximum of 1023), as well as the phase of the signals. It is recommended to use the **Phase** control in the **Lock** tabs to alter the phase of the error signals (see previous section). However, if the relative phase between two clock signals is important, then this is the place to alter them.

Lock TimeOut (s)

Sets the length of time (in seconds) the system will try to **AutoLock** before resetting the max and min limits and trying again.

FPGA Timing Outputs

These indicators give information about FPGA running speeds.

- **PI @ 1 MHz** - Indicates whether the FPGA code is running at the specified speed (normally 80 MHz/108 due to timing restriction of AIPs/AOPs). One indicator per FPGA;
- **PI Tick Delay** - Indicates the tick delay. Especially important if **PI @ 1 MHz** = False. One indicator per FPGA;
- **Clk @ CLS** - Indicates whether clock programming loop is able to run at specified speed (see **Clock Board IP Cluster** section above);

- **PI Clk Delay** - Indicates the clock loop tick delay. Especially important if **Clk @ CLS** = False.

Integrator Thresholds

- **I1 Max Value** - Set the maximum value for all integrator controllers in the code;
- **I2 Max Value** - Set the maximum value for all double integrator controllers in the code, if **I∧2 Gain** is set to zero then this is ignored in the code. The actual integral values from all locks can be seen in the **Re-Ordered IP/OPs** folder in the **I Values** tab.

Scan Trig OP

Each FPGA has a scan trigger output, through AOP4, which outputs a scan signal at the same frequency as the lock selected using the **Scan Trig OP** controls, but with a set amplitude and offset. This is designed to help trigger external scopes/other pieces of hardware.

A.7.3 Sequential Locking

This tab determines the sequential locking set-up for the experiment (i.e. which locks depend on which other locks). Apart from the indicators here there are also indicators located above the **System Inputs/Outputs** folder.

- **Number of Cavities** - Input the number of cavities (or locks) present in the experiment;
- **Set #** - A set is defined here to be up to four locks which follow one after another (i.e. Lock # 4 is dependent on Lock # 3, which is dependent on Lock # 2, which is dependent on Lock # 1). This means that if Lock # 1 is not locked then locks # 2-4 will be **Beam Blocked** automatically, while if Lock # 2 is not locked then locks # 3-4 will be **Beam Blocked** etc. (see Figure A.6(a)). If less than four locks are used in a set, set the remaining spaces to *No Lock*. Sets cannot contain the same lock twice, and different sets may not contain the same lock. Sets can span across FPGA cards;
- **Dependencies Set #** - This determines which other sets Set # depends on. Note than Set 1 has no dependencies by definition. If a set is dependent on another set (i.e. Set X depends on Set Y), Set X needs all locks that make up Set Y to be locked otherwise all locks in Set X will be **Beam Blocked** (see Figure A.6(b)). Sets cannot be interrelated (i.e. Set X cannot depend on Set Y if Set Y already depends on Set X) and sets cannot depend on themselves;
- **Sets OK** - Indicates whether the sets satisfy the requirements mentioned previously in this section;
- **Set Locked** - Located above the folder, this cluster indicates whether all the locks that make up an individual set are locked;
- **BB Total** - Indicates which locks are **Beam Blocked** (either manually or automatically). These values are sent into the FPGA code;

- **All is Locked** - If all sets are locked then this will be true, otherwise it will be false.

A.7.4 Scope

To investigate the state of each lock, a scope has been included in the code. This scope (output located in the **Graphs** folder) shows the error signal, scan signal and DC signal of a lock, as well as one parameter of choice for another of the three locks on the same FPGA.

FPGA IPs

These are the scope inputs that are sent to the FPGA:

- **Trigger Direction** - Determines whether the scope triggers off the rising edge or falling edge of the scan function. The scan trigger has a set amplitude and no offset but is at the correct frequency;
- **Trigger Delay (ms)** - Determines the amount of time in milliseconds the code waits before taking data after triggering;
- **Acquisition Rate (kHz)** - Determines the rate of data collection in kHz between data points. If the scope is not triggering properly try increasing/decreasing the acquisition rate;
- **Trigger Offset (V)** - Used to move the triggering point up or down the scan function;
- **Trigger Channel** - This control is disabled. Instead, the channel the scope triggers off is determined by the control in the **Graphs** folder (see below);
- **Channel Selector** - Used to select which channels to display. Currently the code is set up so that the first three channels to be displayed are the Error Signal, Scan Trigger, and DC Signal for the lock of interest (determined by **Scope Trigger Channel**, see below). The fourth channel can be set to any other channel on the FPGA.

Other Scope Parameters

These are the scope parameters located in the RT code:

- **Scope On** - Determines whether to turn on the scope (and therefore display information on the screen) or not;
- **Scope Time Out (ms)** - Determines the time out time for the scope (if the scope is not triggered) before displaying the remaining points on the screen. If the scope screen starts to flicker, or the **Scope Full** indicator (see below) flashes, increase **Scope Time Out (ms**;
- **RT/Offset** - Used to offset the four signals displayed on the scope;
- **RT/Scaling** - Used to scale the four signals displayed on the scope;

- **Scope Trigger Channel** - Determines which lock to trigger scope off and display channels (located in **Scope** tab in **Graphs** folder). If Lock 1 → 4 is selected then it will show data from FPGA 1, and if Lock 5 → 8 is selected then it will display data from FPGA 2;
- **Scope Channel OK** - Determines if desired trigger channel is **Beam Blocked** (if true then it will turn off the scope to avoid timing out);
- **Scope Full** - Indicates whether scope FIFO[7] is full;
- **Scope Size** - Indicates the number of points in the scope FIFO (max of 4095).

A.7.5 Transfer Functions

This menu is to control the white noise generator and data analysis (see Section A.5.2). The data is displayed on the **Bode Plots** tab in the **Graphs** folder.

FIFO IP Cluster

These parameters control the WN generator:

- **Take FIFO Data** - Turns on Network Analyser FIFO to display information;
- **No Points in FIFO** - Number of points in Network Analyser FIFO (max of 32,767);
- **WN On** - Turns on (or off) white noise on the desired channel;
- **WN RMS** - Determines the amplitude on the WN signal;
- **Lock** - Which channel to send the white noise to, and therefore which channel to display information for. Also determines which lock information is sent to any **Continuous Data File** being recorded.

Data Manipulation

These parameters control the manipulation of data taken with the WN generator including averaging:

- **Phase Delay (x pi)** - Depending on the sign of the controller and phase of error signal, it may be necessary to add π to the current displayed T transfer function to correct the phase. This can be done here;
- **EMA n** - α for the Exponential Moving Average Filter (if used) for transfer function data ($Y_t = 2^{-\alpha}\left[X_t + (2^{\alpha} - 1)\,Y_{t-1}\right]$);
- **Average no. (BS)** - Number of traces taken for Summation average ($2^{Average\_no.\_(BS)}$, if used) for transfer function data;
- **T OS (dB)** - To correct for extra gain added to the current displayed T transfer function, so that 0 dB corresponds to no gain for the G trace (see sections A.5.1 and A.7.7);

---

[7] first-in-first-out memory used to transfer scope data from the FPGA to the RT.

- **New Data** - Lets the system take new data if set to true, otherwise will keep current data and not update when next average is complete;
- **Averaging Type** - Determines whether to use summation average (will only display new data once the number of traces indicated have been taken), or exponential moving average filter (constantly updates trace, but gives greater importance to more recent traces) to smooth transfer function data and allow for better resolution at low frequencies;
- **Start Averaging** - Must be pressed once to start system displaying data on **Bode Plots** screen;
- **Reset Average** - If system stops displaying data (normally when white noise is turned off and then back on again), hit reset to start displaying data again;
- **Extra Pgain** - As was the case with extra T OS, sometimes there seems to be a constant offset between the calculated H value (from T and G) and the simulated H value (using Equation A.1). The different between the two can be seen from the **Comparison Plots** tab of the **Graphs** folder. The **Extra Pgain** can be used to compensate for this;
- **Extra P BS** - As was the case with the controller parameters (see Section A.7.1), this BS is used in tandem with the **Extra Pgain**;
- **P CutOff Freq** - Determines the frequency range (starting from 0 Hz to **P CutOff Freq**) for the compensation.

Other Transfer Function Parameters

- **Graph** - determines which parameter to plot on **Bode Plots** tab of **Graphs** folder, where it is located. Can display T, G (if taken and saved/loaded, see Section A.5.2), H, GH, as well as current error signal (top graph) and white noise (bottom graph) traces;
- **Lock Counter** - To ensure that data is only taken when the system is locked and stable, the **Lock Counter** enables users to set a number of counts the system must have been locked for before data will be taken. If this is not an issue, simply set to 0;
- **Counter OP** - Indicates current # of **Lock Counter** (up to Max);
- **Take Data** - Indicates whether **Lock Counter** currently enables data to be taken;
- **Average Counter** - Displays the number of averages that have been taken so far in *Sum* mode;
- **Summation Graph** - Displays current Summation average data. If this goes blank in *Sum* mode then it is necessary to use **Reset Average**;
- **NA Full** - Indicates whether network analyser FIFO is full.

A.7.6 Controller Measures

This menu is for displaying analysis for the lock of interest (see Section A.5.2).

Max/Min Values

This is where the user can set their desired max/min values for certain locking parameters:

- **MPH (dB)** - Input the maximum peak height which can be tolerated for the lock up to **Cut-Off Frequency (Hz)** (see below);
- **Error RMS** - Input the maximum error signal RMS that can be tolerated;
- **DC RMS** - Input the maximum DC RMS that can be tolerated;
- **Phase Margin (deg)** - Input the minimum phase margin that can be tolerated up to **Cut-Off Frequency (Hz)** (see below);
- **Take RMS Data** - Click to take an RMS data set for both DC and error signals of the lock currently displayed on the scope;
- **Cut-Off Frequency (Hz)** - Input a frequency that will be a maximum for the locking analysis values (i.e. system will ignore all frequencies above this).

NA OP Cluster

This cluster displays the locking parameters measured by the system:

- **Max Noise Suppression** - Displays the value of the maximum noise suppression of the system (below **Cut-Off Frequency (Hz)**);
- **MNS Freq (Hz)** - The frequency at which the maximum noise suppression occurs;
- **Max Peak Height** - Displays the value of the maximum peak height of the system (below **Cut-Off Frequency (Hz)**);
- **MPH Freq** - The frequency at which the maximum peak occurs;
- **Bandwidth (Hz)** - Displays the bandwidth of the system (i.e. the unity gain frequency ω_B of the GH graph);
- **Phase Margin (deg)** - Displays the phase margin of the system (i.e. the phase of GH above $-\pi$ at ω_B);
- **Error RMS** - Displays the Error signal RMS value from the previous RMS run;
- **DC RMS** - Displays the DC signal RMS value from the previous RMS run;
- **Controller Parameters OK** - Indicators to show whether the system parameters agree with the **Max/Min Values** entered (**TmG** = max peak height of T);
- **Controller OK** - Indicates whether all four of the above indicators agree or not with the **Max/Min Values** entered.

A.7.7 Saving Data

This menu allows for saving of data from different sources - either one off or continuous.

Recording Data IPs/OPs

This cluster is for recording transfer function data:

- **Record** - Press to save current T trace (as .csv), along with f - the frequency axis scale;
- **G/H/GH/OPs** - Turn on to also save these traces along with T. Outputs are those located in the **Controller Parameters** cluster;
- **Scope** - Will also save scope data;
- **Lock IPs** - Will also save a subset of the lock inputs if selected (In order: **Signal/BS**; **P Gain**; **I Gain**; **I∧2 Gain**; **I∧2 BS**; **Controller BS**; **P BS**; and **I BS**);
- **Save Directory** - Determines which directory to save data to (in RT home directory on the PXI chassis hard drive). You should always create a new directory for each set of data (i.e. # 0-X) to avoid saving over previous data, see below;
- **Starting #** - Determine starting number for traces (will increase by 1 after each data is saved);
- **appended path** - Displays the path of the previous saved file, will be of form *SaveDirectory/File#/T.csv* etc.);
- **Next File #** - Displays the number of the next file # to save (can be changed by changing **Starting #**);
- **Next File already exists** - This indicator tell you if the next file # already exists on the PXI hard drive. If you then proceed to take data the file will be saved over.

Continuous Data

This cluster is for saving continuous data, in $\simeq 1$ s intervals. It will record the DC value of the lock selected for **FIFO IP Cluster** in the **Transfer Functions** tab, as well as the time interval between points in ms. It will only save data if **Take Data** is true (see Section A.7.5). If this is not desirable then set **Lock Counter** to 0.

- **Continuous Data File** - To name and select type (by changing extension, i.e. .csv) of file to save;
- **Record Continuous Data** - Turn on to take data and turn off to stop taking data;
- **Next Cont. File Exists** - This indicator tells you if your selected file for the next set of continuous data already exists on the PXI hard drive. If you proceed to take data then it will be saved over the top of the previous data.

G Load/Save Cluster

This cluster is for saving G data to allow for locking parameters to be calculated (see Section A.5.2).

- **G Save Dir** - Determines directory (inside RT home directory) to save G data to;

- **G Save File** - Determines the file name (and extension, i.e. .csv) for *G* save file;
- **G Load File** - Determines the directory and file name (as well as extension) for *G* file to load;
- **G Options** - Determine desired *G* operation (save/load file);
- **Implement G Option** - Push to implement currently selected **G option**.

A.7.8 Re-ordered IP/OPs and Errors

The folder on the right hand side of the **Graphs** folder is where all variables are placed before they are re-bundled into a user friendly format, and they are only included here as they must be included somewhere for coding reasons and for initial debugging. The only tab which provides useful information is the **Errors** tab. Above the **System Inputs/Outputs** folder is an **Error** indicator. If this flashes true then it means that an error has been detected for either the Freq. Gen., one of the FPGAs (most probably to do with the FIFO read out and time-out), or with the continuous data saving code. To determine which, look at the **Errors** tab and one should display an error code and message to help fix the problem.

A.7.9 PC_Sound_Gen.vi

This code, located in the PC component of the project, can be used in conjunction with the RT locking codes to produce sounds due to various events. The RT locking codes communicate to the *PC_Sound_Gen.vi* code using Shared Variables. Currently the only event that will register is if a lock becomes unlocked when **Do Not Relock** is selected, but this can be altered in the RT code. To make use of the sound generation option simply run this cod in conjunction with the RT locking code.

Gen Sound IP Cluster

These values have been preset and it is inadvisable to change any of these controls, except the **Frequency** input, to alter the pitch of the signal.

Sound Format

Again, these values have been preset and it is inadvisable to change any of these unless there is no sound being produced or errors occur.

Other Controls and Indicators

- **Volume** - Changes the volume of the sound, ranging from 0-100;
- **Test** - Toggle to test out speakers (sound should be generated when **Test** = true);
- **Mute** - To stop sound when code is running put **Mute** = true;
- **Length** - Sets length of sound signal: the greater the value the longer each sound will be and the slower the refresh rate;
- **Came Unlocked** - This indicates when a sound event (see above for current sound event) occurs in the RT.

A.8 Known Problems

Though we have done our best to remove any bugs from the code, there are still some issues that we have not fixed. These are listed below.

- **Extra Noise on Error Signals** - This was mentioned in Section A.7.2. Though the Master FPGA code is set up to trigger the Freq. Gen., sometimes the signal from the Freq. Gen. does not trigger the HS ADCs in sync with the running of the FPGA code, leading to a signal with excess noise. This can be removed by scanning the phase on the Freq. Gen., though if multiple HS ADCs are being run with the same source and the cable lengths are not the same it is possible that there will not exist a region where all HS ADCs are in sync. This is something that should be considered when setting up the HS ADCs.

- **Real Time Time-Out** - Sometimes there is an unexplained loss of connection between the PC and the Real-Time controller (normally with a message such as: *Waiting for Real-Time target (rt PXI target) to respond*). The code continues to run on the FPGAs and the RT when this happens, it just does not allow for updates to/from the PC. Connection can be re-established by re-deploying the RT code. We have found this a major issue with LabVIEW® 2009, but seems to be fixed in LabVIEW® 2010.

- **Clock Gen. Issues** - More of a hardware problem, however, we have found that sometimes the AD9959 DDS boards will stop sending out their modulation signals. This normally occurs when there is activity around their power supply and we hope that moving to a shielded rack mountable system will fix this problem. In the mean time, the best way to fix the problem is to stop the code, turn off and then back on the clock power supply, and then re-deploy the code.

- **AutoLock Not Working** - This can occur for a few reasons: The first step would be to check that the phase of the error signal is correct (and not 180^o out), after this you may want to consider lowering both the Scan Threshold and Lock Threshold (this is especially the case for small DC signals on a large background) to allow for a greater locking region. Sometimes also the scan is too fast to allow for the controller to take effect before the Lock Threshold is reached and you may want to slow this down. If the system starts locking to the top or bottom of the error signal rather than the zero point this indicates that you are using too much gain and that you should play with the ratios of the three different gains to try to rectify this.

- **Scope Not Triggering Correclty** - The first thing to check is that the selected scope channel is not beam blocked. If the scope is displaying a signal but not triggering, try increasing/decreasing the acquisition rate. If the scope is flashing, try increasing the scope time out or increasing the scan frequency.

- **Code Not Compiling** - The FPGA code for *PHD_Lock_5001* is very close to filling the FPGA memory, therefore if you are trying to add extra logic to the code then there is every likelihood that it will not compile the first or second times. Try at least three times to make sure. If that doesn't work then a LabVIEW® 2009 version of the FPGA code is available upon request, as we have found that the code compiles much more easily on the earlier version (but runs much better on 2010).

A.9 Acknowledgements

I would like to thank all my fellow authors on Ref. [8]: Helen Chrzanowski; David Parrain; Dr. Ben Buchler; Prof. Ping Koy Lam; and Dr. Thomas Symul, for all their help and support with this project. I would also like to thank Dr. Dan Shaddock for helpful hints given, and Neil Hinchey and Oliver Thearle for their help developing the hardware to accompany the code.

Bibliography

[1] Analog devices homepage; http://www.analog.com/en/index.html.

[2] National instruments homepage; http://www.ni.com/.

[3] R.C. Dorf and R.H. Bishop. *Modern Control Systems.* Addison-Wesley, eighth edition, 1998.

[4] R.W.P. Drever, J.L. Hall, F.V. Kowalski, J. Hough, G.M. Ford, A.J. Munley, and H. Ward. Laser phase and frequency stabilization using an optical resonator. *Appl. Phys. B*, 31:97, 1983.

[5] E. B. Hogenauer. An economical class of digital filters for decimation and interpolation. *IEEE Trans. Acoustic, Speech, and Signal Processing*, 29:155, 1981.

[6] H. Nyquist. Regeneration theory. *Bell System Technical Journal*, 11:126, 1932.

[7] R.V. Pound. Electronic frequency stabilization of microwave oscillators. *Rev. Sci. Instrum.*, 17:490, 1946.

[8] B. M. Sparkes, H. M. Chrzanowski, D. P. Parrain, B. C. Buchler, P. K. Lam, and T. Symul. A scalable, self-analyzing digital locking system for use on quantum optics experiments. *e-print*, arXiv:1105.3795v1, 2011.