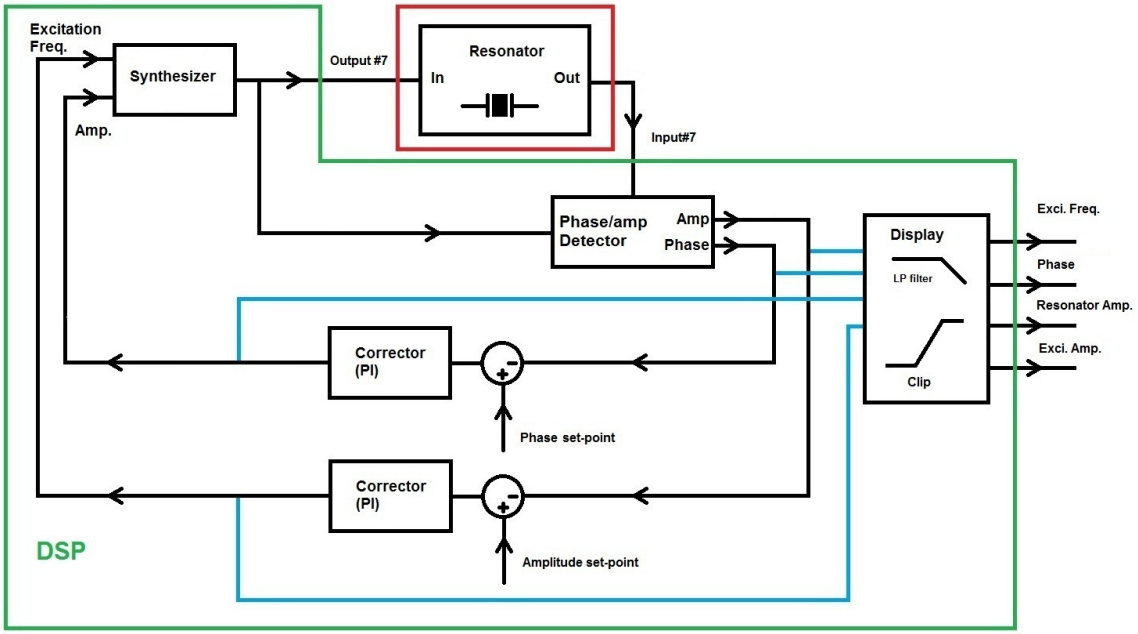
SPM PLL API

User's Manual

Version 1.5



by

Soft dB

In association with

Convergence Instruments

October 2011

1040, avenue Belvédère, suite 215
Québec (Québec) G1S 3G3 Canada
Tél.: (418) 686-0993 Fax: (418) 686-2043

1 Introduction

The PLL API provides a library of functions that allows a developer to integrate PLL functionality to Scanning-Probe Microscopy applications running on the SignalRanger_mk3 platform.

The API requires the presence of a special version of the SR2_A810 IO board that includes a very stable Temperature-Compensated Oscillator (TCXO) as its main clock source. As a consequence, the API also requires a special FPGA logic to support the added TCXO on SR2_A810.

This FPGA logic can be loaded by the PC at high level or placed in the flash memory of the DSP in the boot section. The FPGA file name is *SR2_A810_TCXO_V200.rbt*.

The PLL API is designed to run at 150 KHz input and output sampling rate. This is the rated maximum for the SR2_A810 board, and is high enough to properly control up to the second harmonic of a 32 kHz resonator.

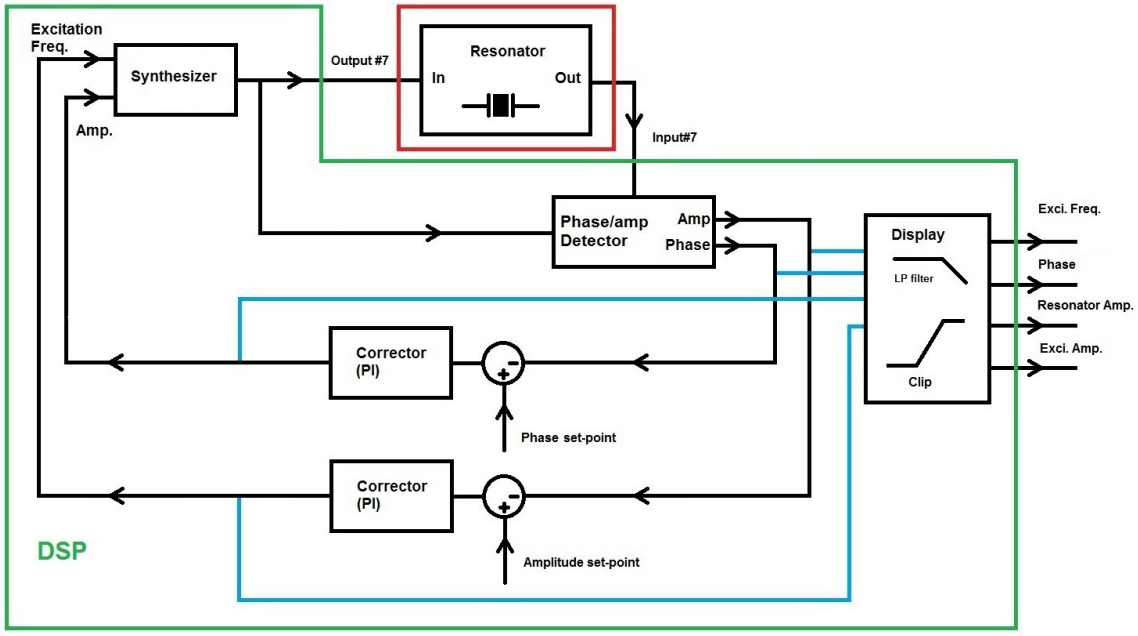
2 Technical Data

TBD

3 General Description of the API

3.1 Theory of Operation

The following figure presents a schematic of the PLL process running on the DSP:



PLL schematic

The synthesizer (sinusoidal generator), the *Phase/Amplitude Convergent Detector* (*PAC* detector), the phase controller, the amplitude controller and the display filter and clip operations are all done in firmware by the DSP. All these functions are provided through an API (Application Programming Interface) that takes the form of a function library (PAC_pll.lib) and associated header file (PAC_pll.h) files.

3.2 Code Size and Performance

The API DSP code size is 6976 Bytes, the data section size is 604 Bytes and the buffer section is 8000000 Bytes.

The number of CPU cycles required for operation is presented below:

- Phase and amplitude controllers OFF (the synthesizer, the phase detector and the display module are activated): 1255 cycles
- Phase controller ON and amplitude controller OFF: 1620 cycles
- All modules ON: 1718 cycles

At 150 kHz, the DSP has about 3300 cycles available for the data processing between two consecutive input and output samples. When both controllers are activated the PLL operation consumes about 52% of the computational power of the DSP.

3.3 API Functions Summary

The API includes a set of functions and global variables, which allow the configuration and control of the PLL. The list of the API functions is presented below:

- **StartPLL(int ioIN, int ioOUT):** This function is called from the main() of the user DSP code before all other functions and variable initializations. The oiIN and ioOUT arguments are the ADC and DAC numbers (between 0 and 7).
- **DataprocessPLL():** This function is called at every execution of (from the) *dataprocess()* of the *SR3PRO_A810* API library, at 150 kHz.
- **ChangeFreqHL():** This function changes the sine generator set-up.
- **OutputSignalSetUp_HL():** This function changes the output display signal set-up.
- **TestPhasePIResp_HL():** This function measures the step response of the phase controller.
- **TestAmpPIResp_HL():** This function measures the step response of the amplitude controller.

The next sections describe in details these functions and the associated global variables. Also, procedures are presented to start both phase and amplitude controllers, to acquire the PLL signals, to measure the resonator's frequency response and to measure the step responses of both controllers. The conversion equations (from physical units to DSP values) are presented when required.

Note: Whenever calculations are necessary to obtain proper values for the API variables we strongly suggest calculating all values using double precision float numbers (64-bit). This ensures the proper precision for all the API variables.

4 API Function and Variables

The next sections describe the functions and variables of all the API modules.

4.1 SR2_A810 API Requirements

The user DSP code that uses the PLL API (*PAC_pll.lib* et PAC_pll.h) must be linked with the *SR3PRO_A810Driver.lib* file and use the *SR3PRO_A810Driver.h* header file.

The code of the PLL (.text section of the *PAC_pll.lib*) must be linked in L1PRAM to obtain the best performance. The *PLLBuffers* section must be linked in the DDR2 memory (add this line in the cmd file: *PLLBuffers > DDR2*).

The *SR2_A810* hardware must be set to sample at 150 kHz. The input range can be set to either +-5V or +-10V. Below are the values for the *SR2_A810* set-up:

Variable name	Description
_ADCRange	16-bit value 0 :+-10V 1 :+-5V
_FreqDiv	16-bit value. The sampling rate must be set at 150 kHz all the time **_FreqDiv** = 5.

4.2 Phase/Amplitude Convergent Detector

The *PAC* detector (Phase/Amplitude Convergent detector) is based on a patent pending technique. This technique provides fast and precise phase and amplitude measurements on a wide frequency tracking range. The only parameter of the *PAC* detector is the variable *_Tau_PAC* (its time constant).

Variable name	Description
_Tau_PAC	32-bit value which represents the time constant of the phase/amplitude detector: $$\_Tau\_PAC = \frac{(2^{22}-1)*1.5E-5}{Tau(s)}$$ *Where Tau is the detector's time constant, defined in s.* The minimum value of *_TauPAC* is 1 and the maximum value is 2^22-1. The maximum *_Tau_PAC* corresponds to a time constant of 15 us. This gives a fast but noisy measurement of the phase and amplitude. At the other end of the scale, a value of *_Tau_PAC*=1 gives a time constant of 63s. That value gives a precise but extremely slow measurement of the phase and amplitude. A value of 0 freezes the phase and amplitude outputs. Typical time constant values are between 15 us and 1.5 ms.

	The set-up of the *PAC* time-constant limits the overall bandwidth of the PLL. The bandwidth of the *PAC* detector in Hz is: 1/(2*Pi*Tau(s)). The actual bandwidth of the whole PLL may be lower depending on the set-up of the phase and amplitude controllers and the display low pass filter set-up. Notes: - If the excitation amplitude is 0, we suggest keeping the time constant at +inf (it means ***_Tau_PAC***=0) to avoid wrong measurements of the phase/amplitude. - ***_Tau_PAC*** must be written using an atomic write.

4.3 Sine Generator

The sine generator is based on a special technique developed by Soft-dB. For more details about the sine generator, see the document *Fast and High-Precision Sine Generator for a TMS320C54x fixed-Point DSP* - Alex Boudreau and Bruno Paillard, article published on the site of globaldsp.com. The sine generation is done by iterative rotation of a complex number by a fixed angular step. This provides a very high-precision and low-distortion output sinusoid. In this implementation the Signal to Noise Ratio of the output sinusoid is –185 dB). The output frequency is expressed through the Real and Imaginary parts of the rotation step. The function **ChangeFreqHL()** must be called after writing these variables to implement them.

Variable name	Description
_deltaReT and _deltaImT	These are the 32-bit Real and Imaginary parts of the angular rotation step. The angular rotation step is a complex number that lies on the unit circle (its norm is 1). Its phase represents the angular step that is applied to the rotating vector at each sampling period. The following equations calculate the values of the real and imaginary parts, as a function of the target frequency. $$\_deltaReT = RE(r = 2^{31} - 1, ph = \frac{2 * \pi * FsineHz}{150000})$$ $$\_deltaImT = IM(r = 2^{31} - 1, ph = \frac{2 * \pi * FsineHz}{150000})$$ *where FsineHz is the desired frequency of the sine wave, r is the norm of the rotation step, and ph is the phase of the rotation step in radians.* Note: - The frequency generator set-up cannot be changed while the phase controller is ON. - The function **ChangeFreqHL()** must be called after modifying the variables to implement the change.
_PI_Phase_Out	This is the output of the phase corrector (32-bit), It has the dimension of a frequency command. This variable must

	be initialized with the following value when the sine generator is set: $$\_PI\_Phase\_Out=\frac{(2^{29}-1)*2*\pi*F\sin eHz}{150000}$$ *where FsineHz is the desired frequency of the sine wave in Hz.* Note: - This variable cannot be modified by the user while the phase controller is ON. The phase corrector will overwrite the variable at every sampling time.
_volumeSine	32-bit value representing the output amplitude of the sine generator: $$\_volumeSine=\frac{(2^{22}-1)*VolumeVolt}{10}$$ *where VolumeVolt is the output volume in volt.* Note: - To change only the excitation volume it is not necessary to call the function ChangeFreqHL() after changing the value of **_volumeSine**. - The value of **_volumeSine** can be changed asynchronously anytime while the amplitude controller is OFF. If the amplitude controller is ON, it will overwrite the variable **_volumeSine** at every sampling period. - **_volumeSine** must be written using an atomic write.

4.4 Phase Controller

The phase controller keeps the resonator phase at the desired set point: normally the phase at the resonance. The output of the phase controller is the excitation frequency. The controller output is limited within an adjustable minimum and maximum frequencies. See section 7.0 for the procedure to start the controllers.

Variable name	Description
_ctrlmode_Phase	32-bit value to set the controller ON(1) or OFF(0). Set the controller OFF while setting-up the controller parameters.
_setpoint_Phase	32-bit value. The set point of the phase controller: $$\_setpoint\_Phase=\frac{(2^{22}-1)*\pi*PhSetPoint}{180}$$ *where PhSetPoint is the phase set point between –180 and +180 degrees.*

	Note: - Normally, the phase set point is adjusted to the resonator phase at the resonance. - **_setpoint_Phase** must be written using an atomic write.
_corrmax_Phase	64-bit value. Maximum excitation frequency at the output of the phase controller: $$\_corr\max\_Phase=\frac{(2^{58}-1)*2*\pi*F\max}{150000}$$ *Fmax is the maximum excitation frequency in Hz.* Note: - **_corrmax_Phase** must be written using an atomic write.
_corrmin_Phase	64-bit value. Minimum excitation frequency at the output of the phase controller: $$\_corr\min\_Phase=\frac{(2^{58}-1)*2*\pi*F\min}{150000}$$ *Fmin is the minimum excitation frequency in Hz.* Note: - **_corrmin_Phase** must be written using an atomic write.
_mem_I_Phase	64-bit value. Memory of the phase controller integrator. This variable must be initialized to the value of the initial frequency: $$\_mem\_I\_Phase=\frac{(2^{58}-1)*2*\pi*Fstart}{150000}$$ *where Fstart is the initial excitation frequency for the phase controller in Hz.* Note: - Normally, the initial frequency is the resonator resonance frequency. - To avoid discontinuities in the excitation frequency when the phase controller transitions from OFF to ON, we suggest to initialize **_mem_I_Phase** to the current excitation frequency just prior to setting the controller ON - **_mem_I_Phase** must be written using an atomic write.
_icoef_Phase	32-bit value. I gain of the phase controller:

	$$\_icoef\_Phase = 10^{\left(\frac{Igain}{20}\right)} * (2^{29} - 1) * Isign$$ *Where Igain is defined in dB (max +12 dB) and Isign is the sign of the gain (-1 or +1).* Note: - **_icoef_Phase** must be written using an atomic write.
_pcoef_Phase	32-bit value. P gain of the phase controller: $$\_pcoef\_Phase = 10^{\left(\frac{Pgain}{20}\right)} * (2^{29} - 1) * Psign$$ *Where Pgain is defined in dB (max +12 dB) and Psign is the sign of the gain (-1 or +1).* Note: - **_pcoef_Phase** must be written using an atomic write.
_errorPI_Phase	32-bit value. Must be set to 0 before switching the controller ON. Note: - **_errorPI_Phase** must be written using an atomic write.

The following expressions can be used to determine the optimal gains of the phase controller (these formulas must be used with a PAC time constant set to 15us):

Kp = 20*log10(1.6575e-5*F_c)

Ki = 20*log10(1.7357e-10*F_c^2)

Where Fc is the desired bandwidth of the controller in Hz (the suggested range is between 1.5 Hz to 4.5kHz).

These gains can be used directly in the P and I formulas (values P_{gain} and I_{gain}). Note that the maximum value for both gains is +12 dB. The gain must be clipped at +12dB if the maximum value is reached. Most of the time the P gain reaches the maximum before the I gain. In this case, the reduction applied on the P gain must be applied on the I gain to assure a stable behavior of the controller.

4.5 Amplitude Controller

The amplitude controller keeps the resonator output amplitude to the desired set point. The output of the amplitude controller is the excitation amplitude. Like the phase controller, the minimum and the maximum outputs are specified by the user to limit the amplitude controller. See section 7.0 for the procedure to start the controllers.

Variable name	Description

_ctrlmode_Amp	32-bit value to set the controller ON(1) or OFF(0). Set the controller OFF during the set-up of the controller parameters.
_setpoint_Amp	32-bit value. Set point for the resonator output amplitude: $$\_setpoint\_Amp = \frac{(2^{29}-1)*AmpSetPoint}{10}$$ *where AmpSetPoint is the set point amplitude in Volt. If the input range of the A810 analog board is +-5V, replace the 10 by 5 in the above equation.* Note: - **_setpoint_Amp** must be written using an atomic write.
_corrmax_Amp	64-bit value. Maximum amplitude controller output: $$\_corrmax\_Amp = \frac{(2^{58}-1)*AmpMax}{10}$$ *AmpMax is the maximum of the amplitude controller output in Volt.* Note: - **_corrmax_Amp** must be written using an atomic write.
_corrmin_Amp	64-bit value. Minimum amplitude controller output: $$\_corrmin\_Amp = \frac{(2^{58}-1)*AmpMin}{10}$$ *AmpMin is the minimum amplitude controller output in Volt.* Note: - **_corrmin_Amp** must be written using an atomic write.
_mem_I_Amp	64-bit value. Memory of the amplitude controller integrator. This variable must be initialized to the value of the initial output amplitude: $$\_mem\_I\_Amp = \frac{(2^{58}-1)*AmpStart}{10}$$ *AmpStart is the initial output of the amplitude controller in Volt.* Note: - To avoid discontinuities in the output amplitude when the amplitude controller transitions from OFF to ON, we suggest to initialize **_mem_I_Amp** to the current output volume just prior setting the controller to ON. - **_mem_I_Amp** must be written using an atomic write.

_icoef_Amp	32-bit value. I gain of the amplitude controller: $$\_icoef\_Amp = 10^{\left(\frac{IgainAmp}{20}\right)} * (2^{29} - 1) * IsignAmp$$ *Where IgainAmp is defined in dB (max +12 dB) and IsignAmp is the sign of the gain (-1 or +1).* Note: - **_icoef_Amp** must be written using an atomic write.
_pcoef_Amp	32-bit value. P gain of the amplitude controller: $$\_pcoef\_Amp = 10^{\left(\frac{PgainAmp}{20}\right)} * (2^{29} - 1) * PsignAmp$$ *Where PgainAmp is defined in dB (max +12 dB) and PsignAmp is the sign of the gain (-1 or +1).* Note: - **_pcoef_Amp** must be written using an atomic write.
_errorPI_Amp	32-bit value. Must be set to 0 before switching the controller ON. Note: - **_errorPI_Amp** must be written using an atomic write.

The following expressions can be used to determine the optimal gains of the amplitude controller (these formulas must be used with a PAC time constant set to 15us):

Kp = $20*\log10(0.08045*QF_c/Gain_{res}F_0)$

Ki = $20*\log10(8.4243e\text{-}7*QF_c^2/Gain_{res}F_0)$

Where :

$Gain_{res}$ is the gain of the resonator at the resonance
Q is the Q factor of the resonator
F_0 is the frequency at the resonance in Hz
Fc is the desired bandwidth of the controller in Hz (the suggested range is between 1.5 Hz to 10Hz).

These gains can be used directly in the P and I formulas (values $P_{gainAmp}$ and $I_{gainAamp}$). Note that the maximum value for both gains is +12 dB. The gain must be clipped at +12dB if the maximum value is reached. Most of the time the P gain reaches the maximum before the I gain. In this case, the reduction applied on the P gain must be applied on the I gain to assure a stable behavior of the controller.

4.6 Output Signal Display Set-up

The output signals of the PLL are:

- Excitation frequency
- Excitation amplitude

- Resonator phase
- Resonator amplitude.

These signals are filtered with a first order low-pass filter and the output of the filter is clipped to the user-specified range. The set-up of the display module is done through the PLL function: **OutputSignalSetUp()**. Before running the function, the following parameters must be set:

Variable name	Description
_LPSelect_Phase	32-bit value that selects of the frequency cut-off for the first order LP filter applied to the signals (Phase and Excitation Frequency). The selections for **_LPSelect_Phase** are: **0:** Filter-off **1:** 16548 Hz **2:** 6868 Hz **3:** 3188 Hz **4:** 1541 Hz **5:** 757 Hz **6:** 376 Hz **7:** 187 Hz **8:** 93 Hz **9:** 47 Hz **10:** 23 Hz **11:** 12 Hz **12:** 5.8 Hz **13:** 2.9 Hz **14:** 1.5 Hz
_LPSelect_Amp	32-bit value that selects of the frequency cut-off for the first order LP filter applied to the signals (Resonator and Excitation Amplitudes). The selections for **_LPSelect_Amp** are: **0:** Filter-off **1:** 16548 Hz **2:** 6868 Hz **3:** 3188 Hz **4:** 1541 Hz **5:** 757 Hz **6:** 376 Hz **7:** 187 Hz **8:** 93 Hz **9:** 47 Hz **10:** 23 Hz **11:** 12 Hz **12:** 5.8 Hz **13:** 2.9 Hz **14:** 1.5 Hz
_ClipMin(4) and _ClipMax(4)	Two arrays of four 32-bit values. Minimum and maximum values for the output signals. Below are the equation to calculate each element of the arrays: **0:Excitation Frequency**

	$$\_ClipMin(0)=\frac{(2^{29}-1)*2*\pi*FMin}{150000}+1$$ $$\_ClipMax(0)=\frac{(2^{29}-1)*2*\pi*FMax}{150000}-1$$ *Where Fmin and Fmax are in Hz.* **1:Resonator Phase** $$\_ClipMin(1)=\frac{(2^{29}-1)*\pi*Phase\min}{180}+1$$ $$\_ClipMax(1)=\frac{(2^{29}-1)*\pi*Phase\max}{180}-1$$ *Where Both Phasemin and Phasemax values (in degrees) are referenced to zero. So, the minimal phase is –x degrees and the maximum phase is +x degrees.* **2:Excitation Amplitude** $$\_ClipMin(2)=\frac{(2^{29}-1)*AmpExciMin}{10}+1$$ $$\_ClipMax(2)=\frac{(2^{29}-1)*AmpExciMax}{10}-1$$ *Where AmpExciMin (in Volt) cannot be negative.* **3:Resonator Amplitude** $$\_ClipMin(3)=\frac{(2^{29}-1)*Amp\mathrm{Reso}Min}{10}+1$$ $$\_ClipMax(3)=\frac{(2^{29}-1)*Amp\mathrm{Reso}Max}{10}-1$$ *Where AmpResoMin (in Volt) cannot be negative. Also, if an input range of +-5V is used, both AmpResoMin and AmpResoMax must be divided by 5 instead of 10.* Note: - **_ClipMin** and **_ClipMax** vectors must be written using an atomic write.

5 Signal Acquisition

The PLL API can be used to store two selected signals for display purposes. The signal storage is done by block through two dedicated 32-bit vectors (maximum length of 1000000 samples).

Variable name	Description
_pSignal1 and _pSignal2	The signal selection is done through these two 32-bit pointers. These pointers must be initialized with the address of the desired variables before starting the acquisition process. Here is the list of possible variables: **Resonator Output Signal:** variable name *_InputFiltered*. Input 7 signal, high pass filtered at 3.2 kHz. Use the following equation to convert the signal in Volts: $ResonatorOutput(V)=10*\frac{Value}{2^{22}-1}$ *If the input range is +-5V, use 5 instead of 10 is the last equation.* **Excitation Signal:** variable name *_SineOut0*. Output 7 signal. Use the following equation to convert the signal in Volts: $ExcitationSignal(V)=10*\frac{Value}{2^{22}-1}$ **Resonator Phase (no LP filter):** variable name *_phase*. Raw phase at the output of the phase detector centered on the *_setpoint_Phase* value. Use the following equation to convert the signal in Degrees: $ResonatorPhase(\deg)=180*\frac{Value}{(2^{22}-1)*\pi}$ **Excitation Freq. (no LP filter):** variable name *_PI_Phase_Out*. Raw excitation frequency. Use the following equation to convert the signal in Hz: $ExcitationFreq(Hz)=150000*\frac{Value}{(2^{29}-1)*2*\pi}$ **Resonator Amp. (no LP filter):** variable name *_amp_estimation*. Raw resonator amplitude at the output of the phase/amp detector. Use the following equation to convert the signal in Volts: $ResonatorAmp(V)=10*\frac{Value}{2^{22}-1}$

	If the input range is +-5V, use 5 instead of 10 in the equation. **Excitation Amp. (no LP filter):** variable name *_volumeSine*. Generator output amplitude. Use the following equation to convert the signal in Volts: $ExcitationAmp(V)=10*\frac{Value}{2^{22}-1}$ **Excitation Freq. (with LP filter):** variable name *_Filter64Out*. Filtered excitation frequency. Use the following equation to convert the signal in Hz: $ExcitationFreq(Hz)=150000*\frac{Value}{(2^{29}-1)*2*\pi}$ **Resonator Phase (with LP filter):** variable name (*_Filter64Out*+4). Filtered phase at the output of the phase detector centered on the *_setpoint_Phase* value. Use the following equation to convert the signal in Degrees: $ResonatorPhase(\deg)=180*\frac{Value}{(2^{29}-1)*\pi}$ **Resonator Amp. (with LP filter):** variable name (*_Filter64Out*+8). Filtered resonator amplitude at the output of the phase/amp detector. Use the following equation to convert the signal in Volts: $ResonatorAmp(V)=10*\frac{Value}{2^{29}-1}$ *Use 5 instead of 10 in the last equation if the input range is +-5V.* **Excitation Amp. (with LP filter):** variable name (*_Filter64Out*+12). Filtered excitation amplitude. Use the following equation to convert the signal in Volts: $ExcitationAmp(V)=10*\frac{Value}{2^{29}-1}$
_blcklen	32-bit value. Desired block size. Must be set with to desired length-1. Maximum value 999999. The signal storage automatically starts on the DSP when the ***_blcklen*** is not equal to –1. Note: - **_blcklen** must be written using an atomic write.
_Signal1 and _Signal2	Two arrays of 32-bit values. These are the two vectors used to store the selected signals during the acquisition process. Always read these vectors using a non-atomic read to avoid blocking the DSP for a long time, since these vectors are usually quite large.

To initiate the storage of one block of each of the two selected signals, follow the procedure below:

1) Initialize **_pSignal1** and **_pSignal2** to the desired variable addresses (see the list above).
2) Set **_blcklen** to the desired block length–1 (maximum 999999).
3) Wait for **_blcklen=-1**.
4) Read both vectors **_Signal1** and **_Signal2** (with non-atomic reads). Note that the first element of the Signal1 or Signal2 vector is the last sample of the block. So, the arrays are stored in reverse order.
5) Convert both signals from I32 values to the physical units using the above equations.

6 Resonator Response Measurement

Before starting the PLL controllers, the resonator frequency response must usually be measured to determine the resonance frequency, the phase at the resonance and the Q factor. The following procedure explains how to measure the resonator frequency response:

1) Switch both controllers (amplitude/phase) OFF
   _ctrlmode_Amp=0 and **_ctrlmode_Phase=0**
2) Set the phase set point to zero
   _setpoint_Phase=0
   The phase controller set point also acts as the phase reference for relative phase measurements. To be able to measure the absolute phase of the resonator, the phase reference must be set to 0.
3) Set the ***_Tau_PAC*** (time constant of the phase detector). To obtain a low noise result, we suggest setting the time constant to 1ms or higher (see section 4.2).
4) Set the sine generator to the starting frequency and the desired output volume (see section 4.3).
5) Wait at least 1.5s to allow the resonator output to initially stabilize.
6) Read the phase (***_phase***) and the amplitude (***_amp_estimation***). To obtain the phase in Degree use the following equation:

   $$Phase(\deg)=180*\frac{\_Phase}{(2^{22}-1)*\pi}$$

   To obtain the amplitude in Volt use the following equation:

   $$Amp(V)=10*\frac{amp\_estimation}{2^{22}-1} \text{ or } Amp(V)=5*\frac{amp\_estimation}{2^{22}-1}$$

   if a +-5V input range is used. The excitation output volume must be taken into account to obtain the gain of the resonator.
7) Change the frequency of the sine generator and make a pause to allow the resonator to stabilize. The duration of the pause depends on the Q factor and the frequency step. For instance, for a Q factor of 25k and a step size of 0.05 Hz, a measurement pause of 300ms is necessary.
8) Repeat steps 6 and 7 to complete the frequency scan and to obtain the resonator frequency response.
9) Determine the resonance frequency, the phase at the resonance and the Q factor from the measured frequency responses (amplitude and phase). The Q factor can be estimated using two methods:
   - **Half maximum:** Freq. at the resonance/delta Freq. at –3 dB
   - Derivative: $Q=\frac{Freq_{res}*\frac{d_{phase}}{d_f}^{atFreq_{res}}}{106}$
10) Write the ***_setpoint_Phase*** variable to its original value or use the newly measured phase at the resonance.

Normally, the phase of the frequency response graph is unwrapped to facilitate the reading. However, when the phase at the resonance is read on the graph and written back to the DSP memory (the ***_setpoint_Phase*** variable), be sure to use a phase value between +180 and –180 degrees:

If UnwrapPhase <0: $Phase=UnwrapPhase-(180*floor(\frac{UnwrapPhase}{-180}))$

If UnwrapPhase >0: $Phase=UnwrapPhase-(180*floor(\frac{UnwrapPhase}{180}))$

7 Controller Startup

The following procedure explains how to start the controllers:

1) Set the controller OFF if this is not already the case.
2) Set the ***_Tau_PAC*** (phase/amplitude detector time constant) to the desired value. If a fast tracking of the phase is necessary, a short time constant is required. But, if a very clean but slow tracking is preferred, the time constant can be increased.
3) Be sure that the output volume for the sine generator is not zero.
4) Set the output signal parameters (see section 4.6).
5) Set the controller parameter (see section 4.4 or 4.5)
6) Switch the controller ON.

8 Controller Closed-Loop Step Response Measurement

The PLL API includes special functions to measure the closed-loop step response of both controllers. These functions add a small adjustable step to the controller set point and measure the input and output of the controller in response to the step. This measurement provides the step response and allows the estimation of the time constant/bandwidth of the controllers. The following procedure explains the technique that can be used to measure the controller response:

1) Switch the controller ON (phase and/or amplitude) with the desired P and I gains (See section 7.0).
2) Initialize ***_deltaReT*** (32-bit value) with a small phase step (1 or 2 degrees for instance) using this equation: $\frac{PhStep*(2^{29}-1)*\pi}{180}$ where PhStep is the phase step in degree. For the amplitude controller, the variable ***_deltaReT*** must be set with a small amplitude step (+0.05V for instance) using this equation: $\frac{AmpStep*(2^{29}-1)}{10}$ replace the 10 by 5 if the input range is +-5V.
3) Initialize ***_pSignal1*** and ***_pSignal2*** to measure the resonator phase (variable ***_Filter64Out*****+4**) and the excitation frequency (variable ***_Filter64Out***) for the phase controller case. For the amplitude controller case, init ***_pSignal1*** and ***_pSignal2*** to measure the resonator amplitude (variable ***_Filter64Out*****+8**) and the excitation amplitude (variable ***_Filter64Out*****+12**).
4) Start the function
   _TestPIResp_HL (phase controller case) or
   _TestAmpPIResp_HL (amplitude controller case)
   to start the measurement of the step response. These functions block until the end of the acquisition block before returning.
5) Read the ***_Signal1*** and ***_Signal2*** vectors (using non-atomic reads) and convert the signals in units. See section 5.0 for conversion equations.
6) The ***_Signal1*** represent the controller closed-loop step response and can be used to estimate the time constant and the bandwidth.

9 Long-Term History Analysis

In order to estimate the long-term stability and the noise on the excitation frequency or other signals, we suggest constructing a history graph based on periodic reads (at 1/10s for instance) of single samples of the desired signal at high level on the PC. See section 5.0 for variable name.