nanoSurf®

Programmers Manual

easyPLL

Software version 2.5

Table of Contents

About this manual

This manual describes how to control their easyPLL Digital FM Detector from another Windows program. This manual applies to the easyPLL Control software version 2.5. The easyPLL control software is described in the *easyPLL reference manual*.

The easyPLL COM Automation Server

The easyPLL COM Automation Server makes it possible to control the easyPLL Digital FM-Detector from another Windows program. This chapter explanains what a COM automation server is, and gives examples of how to control a the COM server from LabView and Visual Basic Script. The chapter *easyPLL Server Properties and Methods* is a reference of the COM servers object properties and method, listed in alphabetical order.

For those who use non-windows operating systems, or those who do not wish to use the COM automation server, the easyPLL C software library option is available. The functions of this library are not described in this manual, however the functions in this library are similar to the ones described here.

What is a COM Automation Server ?

The abbreviation COM stands for 'Component Object Model', which is a Microsoft standard for building interoperable software components. The COM standard describes how a program (called server) can publish its functionality to other programs (called client). The clients can then use the functions of the server using this published information. The functionality can even be used if the client and server are on different computers, connected by a network, independent of the programming language in which the programs were written.

The COM automation standard is defined using the COM standard. The COM automation standard was necessary because the basic COM standard only defines the internal principle how to access the functions of a server by a client. But the client needs prior to its own compilation the information about the servers function details in order to be able to access them. This is a problem for scripting languages like VisualBasic or other programs like LabView which should be able to access unknown servers during run time. This problem was solved by the COM Automation standard. A COM Automation Server publishes its functionality in such a way that COM Automation Clients can ask the server during run

time about its functions and access them afterwards. The information about the servers function are stored in the servers exe-file, and in a binary file with the extension '.tlb' which can be loaded by a client.

The easyPLL control software can act as a server according to the COM Automations standard. Many programming environments and software packages are able to access the easyPLL as a client:

Programming environments:

Visual C++, Visual Basic, Delphi, Windows Scripting Host, LabView, ...

Other software packages:

MathLab, MathCAD, Excel, Word, Internet Explorer, ...

The next two sections contain two examples of how to access the easyPLL Control software. The first example demonstrates the integration of the easyPLL in LabView. The second example shows how to access of the easyPLL in the Windows Scripting Host environment (also called WSH) using Visual Basic Script.

LabView as an easyPLL client

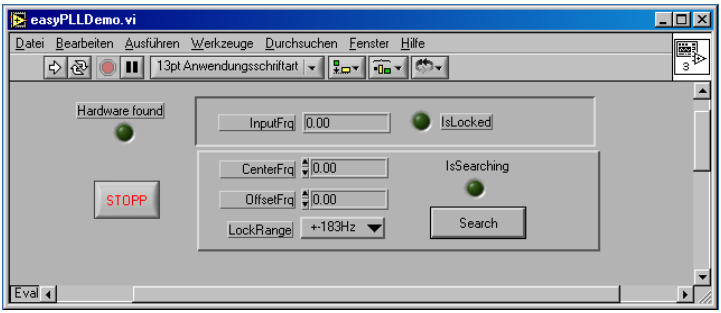
The LabView demo is a simple virtual instrument that monitors the status of the easyPLL, and allows the user to start a search for the resonance frequency. You can use this example as a starting point for your own control software. Use the next screen shots of the front panel and block diagram as a guide through the following detailed explanations.

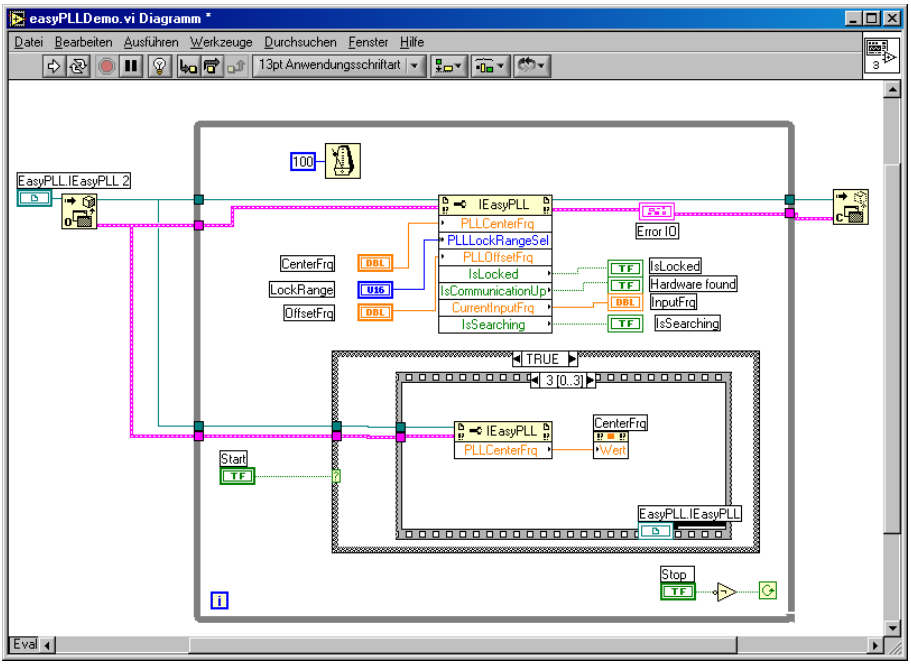
The example on how you can control some functions of the easyPLL from a LabView Virtual Instrument is provided with the instrument, or can be downloaded from the Nanosurf Website by registered users. The example is its called 'easyPLLDemo.vi', and is normally installed in the LabView subdirectory of your easyPLL installation directory, typically 'C:\program files\Nanosurf\easyPLL\LabView\'.

Use LabView's ActiveX function blocks in the diagram of your virtual instrument to control the functionality of the easyPLL. Four function block types are needed:

- The 'ActiveX Open'-block to start the easyPLL Server program

- The 'ActiveX Close'-block to stop the easyPLL Server after executing the VI.
- The 'ActiveX Method'-block to call the easyPLL methods to send it commands.
- The 'ActiveX Property'-block to read or write the easyPLL properties to change and/or read its configuration and status information .





Front and Block diagram of easyPLLDemo.vi

First, a connection between LabView and the easyPLL control software is established using the 'ActiveX Open' function block.

- Place this block from the palette 'Functions->Communication->ActiveX'.

Now connect the block to the easyPLL Control Software:

- Clicking the ActiveX Open block with the right mouse button and selecting the menu item 'Select ActiveX...->Search'.
- Click the 'Browse' button in the dialog to search for the easyPLL's type library with the filename 'easyPLL.tlb'. This file is located in you easyPLL installation directory, which typically is 'C:\program files\Nanosurf\easyPLL\'.

A list of creatable objects is opened after selecting this file. This list contains the name 'easyPLL.Application' as creatable object.

- Select 'easyPLL.Application' and click 'OK'.

The object is now connected to the 'ActiveX Open' block. The outputs of this block should be connected to the corresponding inputs of the other ActiveX function blocks.

The example program uses the easyPLL automation server properties to read or write the status and settings of the easyPLL. In order to do this, create an 'ActiveX Property' function block and connect it to the 'ActiveX Open' block:

- Create the block analogous to the 'ActiveX Open' function block.
- Select the specific property by clicking the lower part of the 'ActiveX Property' block with the right mouse button, and select a property from the list in the 'Property>' submenu.
- Select whether to read or write the property using the menu item 'Change to read' or 'Change to write' in the same submenu.

The current read/write status of the property is indicated by a small arrow. In the example, the settings 'PLLCenterFrq', 'PLLLockRangeSel' and 'PLLOffsetFrq' are written to by the LabView program in order to

change the easyPLL settings, and the properties 'IsLocked', 'IsCommunicationUp', 'CurrentInputFrq' and IsSearching' are used to monitor the status of the PLL.

The procedure is the same for method calls: Insert the block 'ActiveX Method', wire it and select the desired method in the pop up menu. Take care to only call a method at a timed interval, or a specific event, do not call it continuously. Refer to the easyPLL example on how you can integrate the method call in a sequence.

To close the easyPLL Server you place the function block 'ActiveX Close' in the diagram and wire its two inputs to the corresponding outputs of the 'ActiveX Open' block.

Refer to the LabView documentation and examples on ActiveX for more detailed description on how to use the ActiveX function blocks.

Visual Basic as an easyPLL client

You can control all of the functionality of the easyPLL from a windows shell script. In newer version of the Windows operating systems (starting from Windows ME/2000) Microsoft distributes the so called Windows Scripting Host (WSH). With the WSH you are able to write shell scripts in a language like Visual Basic Script (.vbs) or JavaScript (.js). VisualBasicScript is also used in applications like Internet Explorer, Word or Excel to give to user the possibility to enhance the functionality of this software package. We will demonstrate how you to use Visual Basic Script to control the easyPLL's COM Automation Server using an example. You find the example in the sub directory 'VisualBasicScript' of your installation directory (typical c:\program files\Nanosurf\easyPLL\) its called 'easyPLLDemo.vbs'. Use this example as a starting point for your own control software.

```
dim objEasyPLL
set objEasyPLL = CreateObject("easyPLL.Application")
objEasyPLL.LoadConfig("")

if objEasyPLL.IsCommunicationUp = TRUE then
```

```
  objEasyPLL.SearchCenterFrqStart()
  do while objEasyPLL.IsSearching = TRUE
  loop
  wscript.echo "CenterFrq: " &
                objEasyPLL.PLLCenterFrq & "Hz"
else
  wscript.echo "Hardware not found"
end if

objEasyPLL = NUL
```

In order to be able to send commands to the easyPLL you have to establish a connection to it. This is done by creating a object variable with the command `CreateObject()`. The result of this command should be stored in an object variable. This object is used to refer to the easyPLL when calling its methods and accessing its properties. Commands and Properties are separated with a dot from the object name: `object.method` and `object.property`.

- First create an easyPLL object and store it in the variable `objEasyPLL`.
- Load the current easyPLL configuration from the easypll.ini file by calling the method `LoadConfig("")`.
- Find out whether the hardware is connected an powered up by reading the property `IsCommunicationUp`.
- If this is OK, start a frequency search by calling the method `SearchCenterFrqStart()`.
- Waits until the search is finished by checking the property `.IsSearching`.
- Read the new centre frequency from the property `PLLCenterFrq` and display it with the `echo` method of the standard `wscript` class.

easyPLL Server Properties and Methods

This chapter describes the object properties and methods of the easyPLL COM Automation Server. First the properties and methods are listed in groups with similar functions, then each individual object and propertiy is described, listed in alphabetical order.

Functional Overview

The following properties and methods are available:

Status

Type	Property
boolean	ShowAppWindow
boolean	IsConnected
boolean	IsCommunicationUp
boolean	IsLocked
double	CurrentInputFrq
double	CurrentInputPhase
double	CurrentPLLFrq
short	InputGainCheck

General settings and controls

Type	Property
double	PLLCenterFrq
boolean	IsSearching
double	PLLOffsetFrq
short	PLLLockRangeSel
Type	**Method**
boolean	SearchCenterFrqStart()
void	SearchCenterFrqAbort()

Advanced Input/Output Settings

Type	Property
short	InputGainSel

boolean	InputAutoGain
short	OutputGainSel
boolean	OutputLowPass
boolean	OutputPosPolarity
boolean	TipGuardActive
boolean	TipGuardPosPolarity
short	OperatingMode

Phase/Frequency plot

Type	**Property**
double	PhaseSweepEndFrq
double	PhaseSweepStartFrq
short	PhaseSweepSamples
short	PhaseSweepTimePerSample
boolean	IsPhaseSweepMeasuring
Type	**Method**
boolean	PhaseSweepStart()
void	PhaseSweepAbort()
double	GetPhaseSweepResultAt(short pos)
boolean	PhaseSweepSave(BSTR Filename)

Configuration

Type	**Name**
double	SearchStartFrq
double	SearchEndFrq
double	SearchStepFrq
double	SearchStepTime
short	StatusCheckRate
double	CalibQuartzFrq
double	CalibPhaseCoeff
Type	**Method**
boolean	LoadConfig()
boolean	SaveConfig()

CalibPhaseCoeff Property

The CalibPhaseCoeff property sets or returns the calibration coefficient of the phase measurement for the currently used hardware.

Syntax

```
double object.CalibPhaseCoef   (read/write)
```

Description

The easyPLL software needs a correction coefficient for a correct phase measurement in the 'Const. Frequency' mode because this can vary from hardware to hardware. It is normally stored in the easypll.ini file which is correctly setup to a specific hardware at installation of the easyPLL software.

To recalibrate the easyPLL you have to proceed the following:

- Connect the Ref. out and the Input BNC with a cable.
- Set the CenterFrq to 100kHz and the OffsetFrq to 0Hz
- Activate the 'Const. Frequency' operating mode
- Measure with a voltage meter the voltage at the Output-BNC. The absolute value of this voltage is the new calibration coefficient.

The phase calibration coefficient should be in the range of 4V to 4.5V.

See Also

CalibQuartzFrq Property

CalibQuartzFrq Property

The CalibQuartzFrq property sets or returns the calibration coefficient of the frequency measurement for the currently used hardware.

Syntax

```
double object.CalibQuartzFrq (read/write)
```

Description

The easyPLL software needs a correction coefficient for a correct frequency measurement because the build in quartz reference frequency can vary from hardware to hardware. It is normally stored in the easypll.ini file which is correctly setup to a specific hardware at installation of the easyPLL software.

To recalibrate the easyPLL you have to proceed the following:

- Connect the Input-BNC with a calibrated reference frequency of 1MHz.
- Set the CenterFrq to 1MHz and the OffsetFrq to 0Hz.
- Set the LockRange to +-183Hz.
- Set the current quartz calibration coeff. to 24000000.0Hz.
- Activate the 'PLL FM Detector' operating mode.
- Push about 10 times the 'Search' button and calculate the mean value of the displayed CenterFrq's. Name it *CurCenterFrq* .
- Calculate: new QuartzFrqCoeff = 24e6 * 1e6 / *CurCenterFrq.*

The new quartz coefficient should be in the range of 24MHz +-500Hz.

See Also

CalibPhaseCoeff Property

CurrentInputFrq Property

The CurrentInputFrq property returns the currently measured input frequency in 'PLL FM Detector' operating mode in [Hz].

Syntax

```
double object.CurrentInputFrq   (read only)
```

Description

The easyPLL is constantly measuring the input frequency. The value for this property is updated at an approximately sampling rate defined by the property StatusCheckRate.

This property is only returning valid results in 'PLL FM Detector' operating mode and if property IsLocked returns true.

See Also

StatusCheckRate Property

IsLocked Property

CurrentInputPhase Property

CurrentInputPhase Property

The CurrentInputPhase property returns the currently measured input phase in 'Const. Frequency' operating mode in [°].

Syntax

```
double object.CurrentInputPhase  (read only)
```

Description

The easyPLL is constantly measuring the phase shift between the internal reference oscillator and the input signal. The value for this property is updated at an approximately sampling rate defined by the property StatusCheckRate.

This property is only returning valid results in 'Const. Frequency' operating mode.

See Also

StatusCheckRate Property

CurrentInputFrq Property

CurrentPLLFrq Property

The CurrentPLLFrq property returns the currently measured frequency deviation of the PLL in 'PLL FM. Detector' operating mode in [Hz].

Syntax

```
double object.CurrentPLLFrq     (read only)
```

Description

The easyPLL is constantly measuring the frequency of the input signal.

The result of this measure is a difference between the input frequency and a reference frequency defined by the user with the properties PLLCenterFrq and PLLOffsetFrq. Its value corresponds to the 'dF'-output signal at the easyPLL FM Detector hardware front panel. This property is updated at an approximately sampling rate defined by the property StatusCheckRate.

This property is only returning valid results in 'PLL FM. Detector' operating mode and if property IsLocked returns true.

See Also

StatusCheckRate Property

PLLCenterFrq Property

PLLOffsetFrq Property

PLLLockRange Property

IsLocked Property

GetPhaseSweepResultAt Method

The GetPhaseSweepResultAt returns the phase shift value in [°] of one measurement point from the last Phase/Frequency plot measurement.

Syntax

```
double object.GetPhaseSweepResultAt(short pos)
```

`pos` position of the data point to be returned

Description

This method returns the phase shift value in [°] of the measurement point given by pos. The corresponding excitation frequency can be calculated using

```
Freq = PhaseSweepStartFrq + (PhaseSweepStartFrq -
PhaseSweepStartFrq) * (pos / (PhaseSweepSamples -
1)) ....
```

The position pos should be smaller than PhaseSweepSamples, the first measurement point has position 0.

See Also

PhaseSweepStartFrq Property

PhaseSweepStartFrq Property

PhaseSweepSamples Property

InputAutoGain Property

The InputAutoGain property determines whether the easyPLL control software is automatically selecting the input voltage range or not.

Syntax

```
boolean object.InputAutoGain   (read/write)
```

Description

The easyPLL FM Detector hardware is using an variable gain amplifier to adapt its internal electronics to the input signals voltage range. If the input voltage is not corresponding to the current selected amplifier gain the easyPLL hardware detect this and light the over or under range LED.

The easyPLL Control Software can automatically adjust the gain of the input amplifier to follow the input signal voltage. The InputAutoGain property switch this function on or off.

In some cases it could be wise not to enable this function because at the time the gain switch is done the pll can shortly unlock which could influence a ongoing measurement.

See Also

InputGainSel Property

InputGainCheck Property

InputGainCheck Property

The InputGainCheck property reads the input gain status to check for correct gain setting.

Syntax

```
short object.InputGainCheck   (read only)
```

Value	Gain status
0	input signal is OK and in range
1	input signal overloads the amplifier
-1	input signal is too low for amplifier

Description

The InputGainCheck property returns a status information about the variable gain amplifier's signal condition. This input amplifier should be optimal adjusted to the applied input signal voltage.

If the status is not OK then the gain should be adjusted with the property InputGainSel. This can also be automatically be done with the InputAutoGain property.

See Also

InputGainSel Property

InputAutoGain Property

InputGainSel Property

The InputGainSel property determines the gain of the input amplifier.

Syntax

```
short object.InputGainSel (read/write)
```

Value	Input range
0	<10V
1	<1.7V
2	<310mV
3	<54mV
4	<9mV

Description

The InputGainSel property determines the gain of the input amplifier of the easyPLL hardware to adapt the electronics to the input signal voltage.

To check if the correct gain is selected the property InputGainCheck can be used or an automatic adjustment by software can be activated with the property InputAutoGain.

See Also

InputGainCheck Property

InputAutoGain Property

IsCommunicationUp Property

The IsCommunicationUp returns true when the communication protocol to the easyPLL hardware detects no errors.

Syntax

```
boolean object.IsCommunicationUp (read only)
```

Description

The easyPLL Control software is checking periodically if a communication with the hardware is possible or not. A reason for not to be able to communicate could be a wrong setting of the LPT port, a not connected parallel port cable, not switch on hardware, ...

There is also the property IsConnected returns true when the setting of the software about the LPT port is correct.

See Also

IsConnected Property

IsConnected Property

The IsConnected property returns true when the LPT port is correctly configured.

Syntax

```
boolean object.IsConnected  (read only)
```

Description

The easyPLL Control software needs to know about the LPT port the easyPLL FM Detector hardware is connected to. This property returns true when the easyPLL control software has access to this LPT port.

There is also the property IsCommunicationUp which checks the possibility of sending commands to the hardware.

See Also

IsCommunicationUp Property

IsLocked Property

The IsLocked property returns true when the PLL is locked onto the measurement frequency.

Syntax

```
boolean object.IsLocked (read only)
```

Description

In order to be sure that all settings of the hardware are correct and the system is able to measure the actual input signal's frequency this property is useful. This property returns true if all settings are OK and the hardware's pll circuit is able to lock its internal reference oscillator to the input signal.

The CurrentInputFrq and the CurrentPLLFrq properties are returning only valid values if this IsLocked is true. If it is returning false the settings has to be adjusted. This is done by the PLLCenterFrq and PLLLockRangeSel properties. Or automatically by calling the method SearchCenterFrqStart().

See Also

PLLCenterFrq Property

PLLLockRangeSel Property

SearchCenterFrqStart() Method

IsPhaseSweepMeasuring Property

The IsPhaseSweepMeasuring property returns true when a Phase/Frequency measurement is in progress.

Syntax

```
boolean object.IsPhaseSweepMeasuring    (read only)
```

Description

This property is monitoring the state of the Phase/Frequency measurement function.

When a search is started by the PhaseSweepStart() method, this property is set to true until the search is finished or stopped by a call to PhaseSweepAbort().

See Also

PhaseSweepStart() Method

PhaseSweepAbort() Method

IsSearching Property

The IsSearching property returns true when a frequency search is running.

Syntax

```
boolean object.IsSearching (read only)
```

Description

This property is monitoring the state of the automatic search function.

If a search is started by the method SearchCenterFrqStart() this property is set to true until the search is finished or stopped by a method call to SearchCenterFrqAbort().

See Also

SearchCenterFrqStart() Method

SearchCenterFrqAbort() Method

LoadConfig Method

The LoadConfig method loads a configuration from the configuration file.

Syntax

```
boolean object.LoadConfig(BSTR groupname)
```

`groupname`	name of the configuration where the configuration data is loaded from

Description

This method loads a configuration group from the configuration file.

A configuration group consists of all the properties and can be stored in a named group in the configuration file easypll.ini.

Typically a configuration group is loaded at the beginning of a client program.

The easyPLL control software stores its own configuration in the group 'EasyPLL-LastConfig' which is loaded at the program start up and stored at users exit.

See Also

SaveConfig() Method

OperatingMode Property

The OperatingMode property determines the active measuring mode.

Syntax

```
short object.OperatingMode     (read/write)
```

Value	Operating mode
0	Frequency measure mode (PLL FM Detector)
1	Phase measure mode (Const. Frequency)

Description

The easyPLL FM Detector is able to measure in different modes.

In each mode it is measuring a different property of the input signal.

Either the frequency or the phase can be measured.

This property is selecting on of the modes. In the phase measuring mode some property and methods are not available:

- PLLLockRange
- TipGuardActive
- CurrentInputFrq
- CurrentPLLFrq
- IsSearching
- SearchCenterFrqStart/Abort()

See Also

CurrentInputFrq Property

CurrentPLLFrq Property

CurrentInputPhase Property

OutputGainSel Property

The OutputGainSel property determines the output amplifier gain.

Syntax

```
short object.OutputGainSel     (read/write)
```

Value	Gain	Output voltage
0	x0.1	-1.0V .. +1.0V
1	x1.0	-10.0V .. +10.0V
2	x10.0	-10.0V .. +10.0V
3	x100.0	-10.0V .. +10.0V

Description

With this property the gain of the amplifier for the 'Output'-BNC signal can be defined. This is useful to adapt the easyPLL hardware to different scan electronics.

Pay attention that a gain of x10 and x100 is decreasing the lock range also by the factor of 10 or 100!

See Also

OutputPosPolarity Property

OutputLowPass Property

PLLLockRange Property

OutputLowPass Property

The OutputLowPass property enables the output low pass filter when true.

Syntax

```
boolean object.OutputLowPass   (read/write)
```

Description

With this property the low pass filter for the signal at the 'Output'-BNC signal can be switched on and off. If the full speed of the easyPLL demodulation bandwidth is not used because of a low scan speed the low pass filter can decrease the noise heavily.

The filter has a 3dB cut of frequency of about 380Hz.

See Also

OutputPosPolarity Property

OutputGainSel Property

OutputPosPolarity Property

The OutputPosPolarity sets sign of the output signal to positive when true.

Syntax

```
boolean object.OutputPosPolarity    (read/write)
```

Description

With this property the sign of the signal at the 'Output'-BNC signal can be inverted or not. This is useful to adapt the easyPLL hardware to different scan electronics.

A positive sign mean a voltage increase at the output if the frequency rises and a decrease of the output voltage if the input frequency drops.

A negative sign mean a voltage increase at the output if the frequency drops and a decrease of the output voltage if the input frequency increases.

See Also

OutputLowPass Property

OutputGainSel Property

PhaseSweepAbort Method

The PhaseSweepAbort method aborts an active Phase/frequency plot measurement.

Syntax

```
void     PhaseSweepAbort(void)
```

This method stops the Phase/frequency plot measurement. A the measurement is started with the method PhaseSweepStart() method. The IsPhaseSweepMeasuring property has the value true when a measurement is in progress.

See Also

PhaseSweepStart() Method

IsPhaseSweepMeasuring Property

PhaseSweepEndFrq Property

The PhaseSweepEndFrq property sets the end of the frequency range for the Phase/Frequency measurement in [Hz].

Syntax

```
double object.PhaseSweepEndFrq (read/write)
```

Description

With this property the end or highest frequency of the search range for the automatic search of a input frequency is defined.

This is used by the method PhaseSweepStart().

See Also

PhaseSweepStart() Method

PhaseSweepStartFrq Property

PhaseSweepSamples Property

PhaseSweepTimePerSample Property

PhaseSweepSamples Property

The PhaseSweepSamples property sets the number of frequency samples that is taken in the Phase/Frequency measurement.

Syntax

```
short object.PhaseSweepSamples (read/write)
```

Description

Determines the number of points that are taken in the Phase/Frequency measurement. The first point is the PhaseSweepStartFrq, the last point is the PhaseSweepEndFrq, all other points are evenly distributed between these points

This is used by the method PhaseSweepStart().

See Also

PhaseSweepStart() Method

PhaseSweepStartFrq Property

PhaseSweepStepFrq Property

PhaseSweepTimePerSample Property

PhaseSweepSave Method

The PhaseSweepSave method stores the last Phase/Frequency plot measurement to a file.

Syntax

`boolean object.PhaseSweepSave(BSTR Filename)`

`Filename` name of the text file where the Phase/Frequency plot is saved.

Description

This method stores the last Phase/Frequency measurement to a ASCII-text file with the CSV extension.

See Also

PhaseSweepStart() Method

PhaseSweepStart Method

The PhaseSweepStart method starts a Phase/frequency plot measurement.

Syntax

```
boolean object.PhaseSweepStart()
```

Description

This function starts a Phase/Frequency plot measurement with PhaseSweepSamples samples, in the frequency range from PhaseSweepStartFrq to PhaseSweepEndFrq.

Monitor the property IsPhaseSweepMeasuring to find whether a sweep is still in progress or not.

Call the method PhaseSweepAbort to interrupt the measurement.

See Also

PhaseSweepEndFrq Property

PhaseSweepStartFrq Property

PhaseSweepSamples Property

PhaseSweepTimePerSample Property

IsPhaseSweepMeasuring Property

PhaseSweepAbort() Method

PhaseSweepStartFrq Property

The PhaseSweepStartFrq property sets the start of the frequency range for the Phase/Frequency measurement in [Hz].

Syntax

```
double object.PhaseSweepStartFrq    (read/write)
```

Description

With this property the end or highest frequency of the search range for the automatic search of a input frequency is defined.

This is used by the method PhaseSweepStart().

See Also

PhaseSweepStart() Method

PhaseSweepEndFrq Property

PhaseSweepSamples Property

PhaseSweepTimePerSample Property

PhaseSweepTimePerSample Property

The PhaseSweepTimePerSample property sets the time interval between two phase shift measurements in the Phase/Frequency measurement in [ms]. ....

Syntax

```
short object.PhaseSweepTimePerSample    (read/write)
```

Description

With this property the time interval between changing the excitation two search cycles during the automatic search scan of the input frequency is defined.

Its value should be of the order of the Q factor divided by the resonance frequency of the sensor. Although the phase shift changes instantaneously, it may take some time before the phase shift reacts when the vibration amplitude was very low.

This is used by the PhaseSweepStart() method.

See Also

PhaseSweepStart() Method

PhaseSweepStartFrq Property

PhaseSweepEndFrq Property

PhaseSweepSamples Property

PLLCenterFrq Property

The PLLCenterFrq property sets the reference frequency of the PLL in [Hz].

Syntax

```
double object.PLLCenterFrq     (read/write)
```

Description

With this property the easyPLL's reference frequency is set. The input frequency is compared to this reference frequency and the 'dF' output signal of the PLL is the difference between this two frequencies.

The PLL is able to measure an input frequency in the range of this centre frequency plus or minus the frequency range defined by the property PLLLockRangeSel.

An additional offset can be added with the property PLLOffsetFrq to the difference to be able to define a set point for the Z-feedback controller of your scan electronics.

See Also

PLLOffsetFrq Property

PLLLockRangeSel Property

PLLLockRangeSel Property

The PLLLockRangeSel property selects the measurable frequency range and output resolution.

Syntax

```
short object.PLLLockRangeSel   (read/write)
```

Value	Frq. range	Resolution
0	+-183Hz	18.31Hz/V (out. gain x1)
1	+-366Hz	36.62Hz/V (out. gain x1)
2	+-732Hz	73.24Hz/V (out. gain x1)

Description

With this property and the PLLCenterFrq property the range of the measurable input frequency is defined. The centre frequency determines the frequency for which the dF-Output is at 0V and the lock range determines the frequency range around this centre which can be measured. A input frequency of exactly the frequency PLLCenterFrq + PLLLockRange the dF-Output is at 10V (out. gain x1).

See Also

OutputGainSel Property

PLLCenterFrq Property

PLLOffsetFrq Property

The PLLOffsetFrq property sets the offset added to the dF-Output in [Hz].

Syntax

```
double object.PLLOffsetFrq     (read/write)
```

Description

With this property an offset can be added to the PLLs centre frequency. This offset shifts the signal at the ‘dF’-Output.

Prior to the approach the PLLCenterFrq is adjusted so that the dF-Output is 0Hz with a call to SearchCenterFrqStart(). Then the set point can be defined with this offset property in order to get a shifted output signal so that a z-feedback controller which is regulating it error input signal to zero keep the cantilever at an frequency shift defined by this property.

See Also

PLLCenterFrq Property

PLLLockRangeSel Property

SaveConfig Method

The SaveConfig method stores the current configuration to the easyPLL configuration file.

Syntax

```
boolean object.SaveConfig(BSTR groupname)
```

`groupname`	Name of the configuration group where the configuration is stored.

Description

This method stores the current configuration to the configuration group with the name groupname in the configuration file.

A configuration group consists of all the properties and is stored as a named group in the configuration file easypll.ini.

Typically a configuration set is stored immediately prior the end of a client program.

The easyPLL control software stores its own configuration in the group ‘EasyPLL-LastConfig’ which is loaded at the program start up and stored at users exit.

See Also

LoadConfig() Method

SearchCenterFrqAbort Method

The SearchCenterFrqAbort method aborts an active frequency search sweep process.

Syntax

```
void object.SearchCenterFrqAbort(void)
```

Description

This method stops an active frequency search sweep. A sweep is started with the method SearchCenterFrqStart() method. The property IsSearching has the value true when a sweep is in progress.

See Also

SearchCenterFrqStart() Method

IsSearching Property

SearchCenterFrqStart Method

The SearchCenterFrqStart method starts the search for the input frequency process.

Syntax

```
boolean object.SearchCenterFrqStart(void)
```

Description

This method starts a process of automatic adjustment of the centre frequency. If the PLL is locked the current input frequency a fine adjustment process is executed and the current input frequency is transferred to the centre frequency to set the dF-Output zero. If the PLL is unlocked a coarse search for the input frequency is started until the PLL locks or an upper frequency limit is reached. Is a PLL lock found during the sweep of the reference frequency the same fine adjustment process as above is executed.

The frequency range of the sweep is defined by the properties SearchStart/ SearchEndFrq, SearchStepFrq and SearchStepTime.

Monitor the property IsSearching to find whether a sweep is still in progress or not.

Call the method SearchCenterFrqAbort to interrupt a search sweep in progress.

See Also

SearchCenterFrqAbort() Method

IsSearching Property

SearchStartFrq Property

SearchEndFrq Property

SearchStepFrq Property

SearchStepTime Property

SearchEndFrq Property

The SearchEndFrq property sets the highest frequency for the automatic search in [Hz].

Syntax

```
double object.SearchEndFrq     (read/write)
```

Description

With this property the end or highest frequency of the search range for the automatic search of a input frequency is defined.

This is used by the method SearchCenterFrqStart().

See Also

SearchCenterFrqStart() Method

SearchStartFrq Property

SearchStepFrq Property

SearchStepTime Property

SearchStartFrq Property

The SearchStartFrq property sets the lowest frequency for the automatic search in [Hz].

Syntax

```
double object.SearchStartFrq   (read/write)
```

Description

With this property the start or lowest frequency of the search range for the automatic search of a input frequency is defined.

This is used by the method SearchCenterFrqStart().

See Also

SearchCenterFrqStart() Method

SearchStartFrq Property

SearchStepFrq Property

SearchStepTime Property

SearchStepFrq Property

The SearchStepFrq property sets the frequency increment for the automatic search in [Hz].

Syntax

```
double object.SearchStepFrq    (read/write)
```

Description

With this property the frequency increment between two search cycles during the automatic search scan of the input frequency is defined.

The used LockRange during the scan is the +-732Hz range. Therefore this value should be about 700Hz in order to be able to find a input signal during the scan. A lower value increases the precision but lower the search speed.

This is used by the method SearchCenterFrqStart().

See Also

SearchCenterFrqStart() Method

SearchStartFrq Property

SearchStepFrq Property

SearchStepTime Property

SearchStepTime Property

The SearchStepTime property sets the time interval between two search cycles during the automatic search in [s].

Syntax

```
double object.SearchStepTime   (read/write)
```

Description

With this property the time interval between two search cycles during the automatic search scan of the input frequency is defined.

The PLL needs some time do be able to lock in to a frequency from the lock out state. Therefore this value should be about 60ms in order to be able to find a input signal during the scan. A higher value increases the precession but lower the search speed.

This is used by the method SearchCenterFrqStart().

See Also

SearchCenterFrqStart() Method

SearchStartFrq Property

SearchStepFrq Property

SearchStepFrq Property

ShowAppWindow Property

The ShowAppWindow property makes the easyPLL control software application window visible when true.

Syntax

```
boolean object.ShowAppWindow   (read/write)
```

Description

With this property the user interface of the easyPLL control software can be displayed or hidden. Therefore the client can control if the user can adjust controls on the user interface or not.

See Also

none

StatusCheckRate Property

The StatusCheckRate property determines the time interval status information are read from the easyPLL hardware in [ms].

Syntax

```
short object.StatusCheckRate        (read/write)
```

Description

With this property the time interval between two status check is defined.

At each status check different hardware properties are read from the easyPLL FM Detector hardware. The following properties are updated:

- IsCommunicationUp
- IsLocked
- CurrentInputFrq
- CurrentPLLFrq
- CurrentInputPhase
- InputGainCheck

A standard check rate of 300ms is recommended to avoid high PC work load.

See Also

none.

TipGuardActive Property

The TipGuardActive property activates the TipGuard feature when true.

Syntax

```
boolean object.TipGuardActive  (read/write)
```

Description

With this property the TipGuard feature is activated or disabled.

The TipGuard is a functionality which protects the tip from crashing into the sample if the z-feedback controller of the scan electronics is not setup properly and the tip is getting so close to the surface that the oscillation cannot be hold. This is detected by the TipGuard circuit and the output of the easyPLL is forced to a maximum level in order to force the z-feedback controller to retract the tip at maximum speed.

After the oscillation is back the signal of the easyPLL output is normal again and the z-feedback controller can re-approach to the surface again.

See Also

TipGuardPosPolarity Property

TipGuardPosPolarity Property

The TipGuardPosPolarity property sets the direction of the output enforcement to a positive value when true.

Syntax

```
boolean object.TipGuardPosPolarity (read/write)
```

Description

With this property the direction of the enforcement is set.

In correspondence to the z-feedback controller the voltage of the enforcement has to be set right. If the z-controller withdraw the tip with a positive input voltage at its feedback error input, then the polarity has to be set to positive. If the z-feedback controller needs an negative voltage to withdraw the tip the polarity has to be negative.

See Also

TipGuardActive Property