Getting started with PalmSens SDK for LabVIEW

Based on PalmSens SDK v5.10

Last revision: December 22, 2022

www.palmsens.com

PalmSens

Contents

1 Contents of the PalmSens SDK

The PalmSens SDK contains the following .NET libraries and LabVIEW classes, controls and Vis in the PalmSens folder:

Libraries Folder
Contains all the necessary .NET libraries

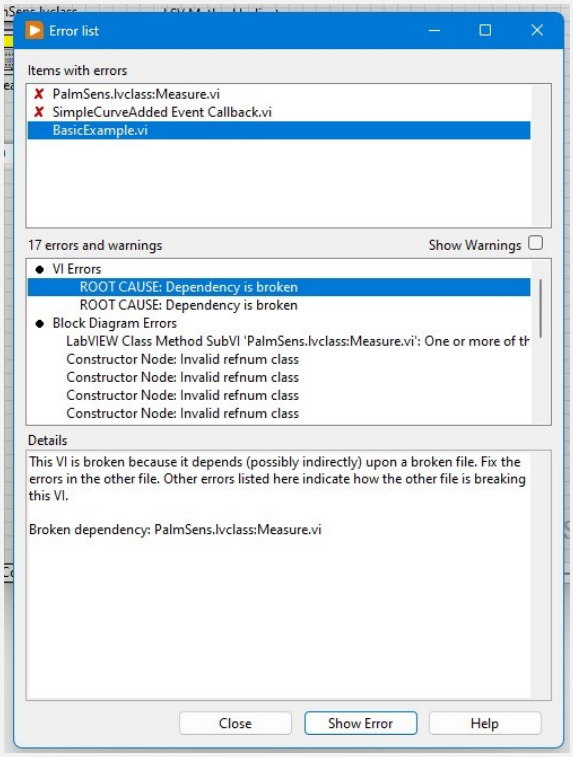
PalmSens LabVIEW class
Class that implements the basic features required to use our instruments in LabVIEW.

- **Initialise**: Initialises the class and .NET libraries.
- **Dispose**: Frees up memory used by .NET libraries.
- **ListInstruments**: Returns an array of the available instruments.
- **Connect**: Connects to the specified instrument.
- **Disconnect**: Disconnects from the connected instrument.
- **Measure**: Performs a measurement.
- **AbortMeasurement**: Aborts the current measurement

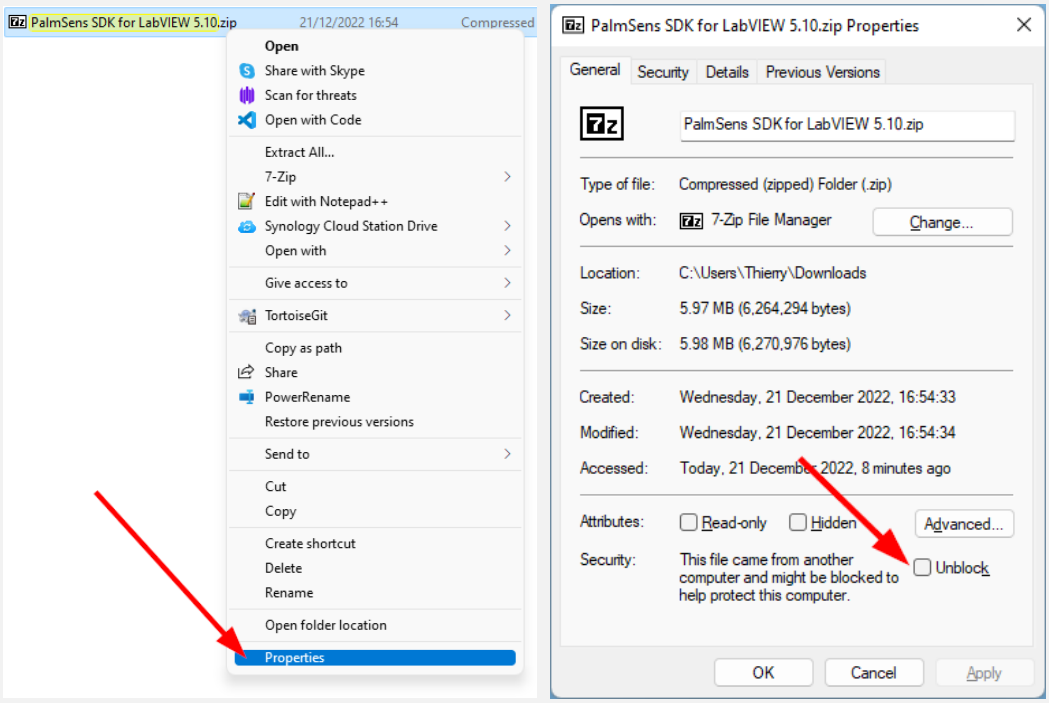
MeasurementResults and LiveCurveResults controls
Type definitions for the in and outputs of the Measure function.

Event Callback Vis
VIs used by the Measure function to receive data from .NET events.

Broken dependency error when using the SDK
When you have downloaded the SDK and extract its contents Windows will have blocked the required .NET library .dll files. When you try to execute anything in the SDK that uses these libraries you will get an error like this.



To prevent this error you will need to manually unblock the PalmSens SDK for LabVIEW.zip file before unpacking its contents. To do this go to right click on the zip file and select Properties. Then click on the Unblock checkbox and click on OK.



If this does not resolve the issue you may need to do this for each file in the PalmSens\Libraries subfolder.

1.1 Example programs

The following examples are included.

Example – BasicExample
Demonstrates how to use the PalmSens class to run a measurement.

Example – MethodSCRIPTExample
Demonstrates how to use the PalmSens class to run a MethodSCRIPT™ measurement.

Example – BasicUIExample
Demonstrates how to use the PalmSens class to run and plot a measurement in real-time and the recommended method to abort, disconnect, or terminate the app during an active measurement.

1.2 Compatible devices and firmware

	Minimum required firmware version
EmStat	3.7
EmStat2	7.7
EmStat3	7.7
EmStat3+	7.7

EmStat4	1.2
EmStat Go	7.7
EmStat Pico	1.3
Sensit Smart	1.3
Sensit BT	1.3
MultiEmStat	7.7
PalmSens3	2.8
PalmSens4	1.7
MultiPalmSens4	1.7

2 Setting up the SDK for your LabVIEW project

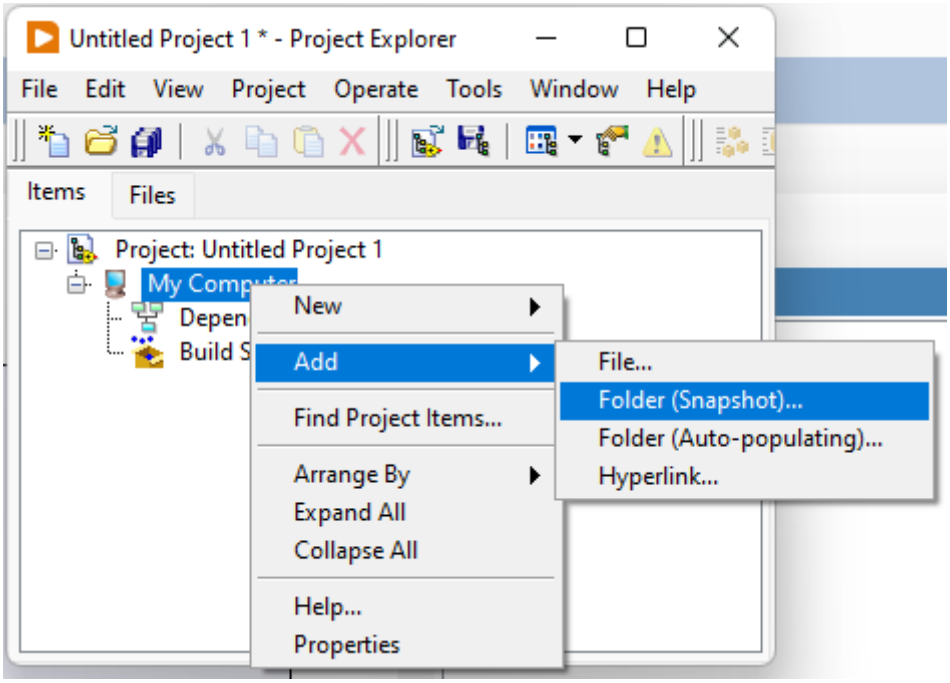
2.1 Requirements

To discover our instruments and use our SDK libraries our drivers and the .NET/c++ runtimes need to be installed. We highly recommend installing PSTrace or MultiTrace as this installs all the required dependencies.

The following diagram shows the inheritance structure of the Method classes:

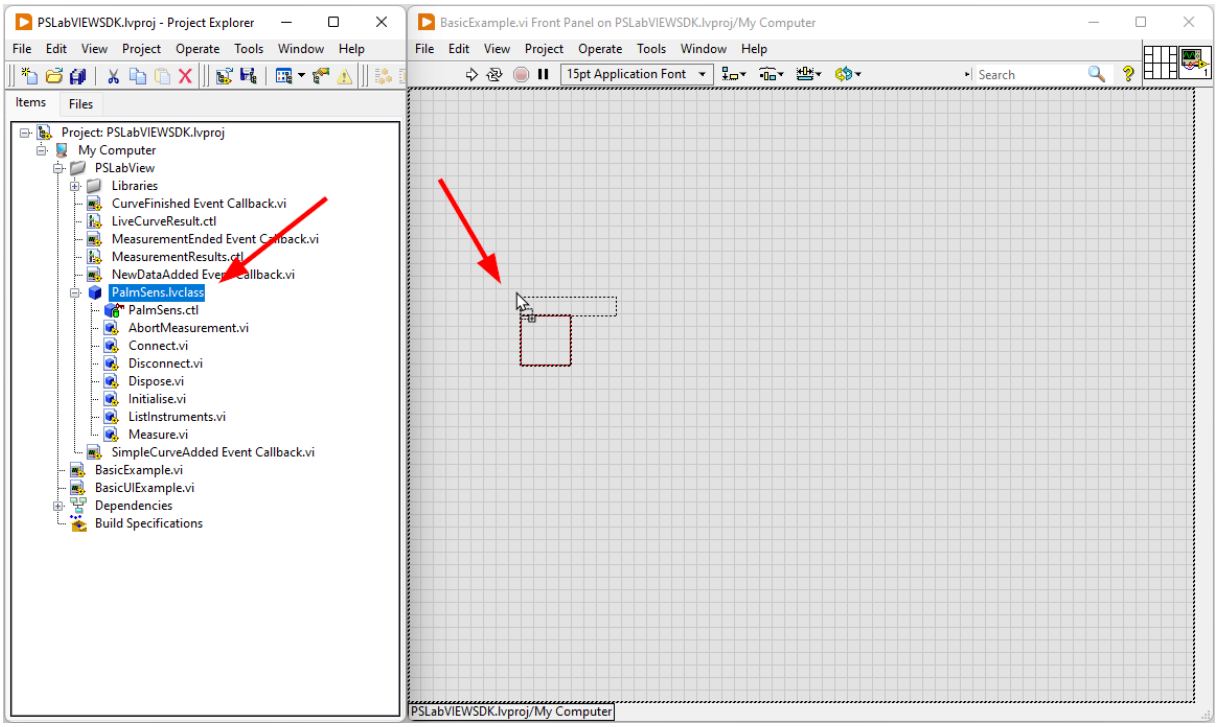
2.2 Adding the PalmSens folder

To add the SDK to a new or existing LabVIEW project add the PalmSens folder. Right click on the My Computer entry in the Project Explorer, go to Add and select Folder. Both the Snapshot and Auto-populating options will work.

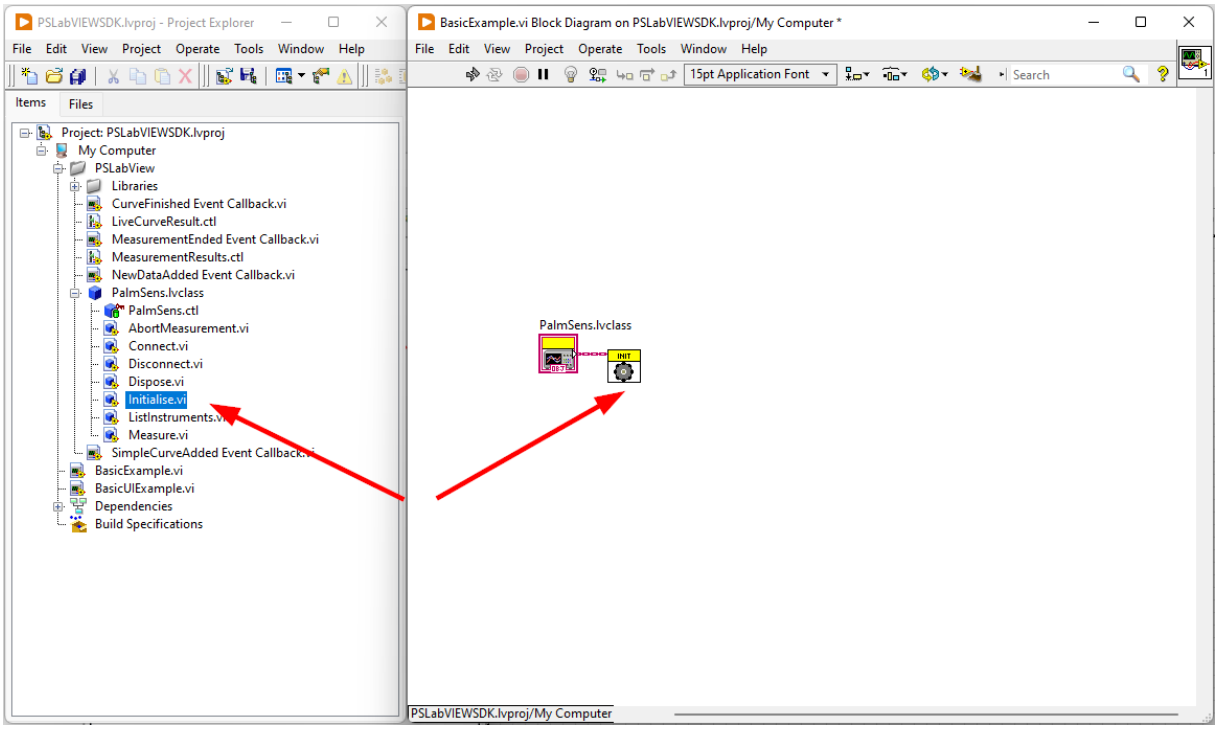


2.3 Creating an instance of the PalmSens Class

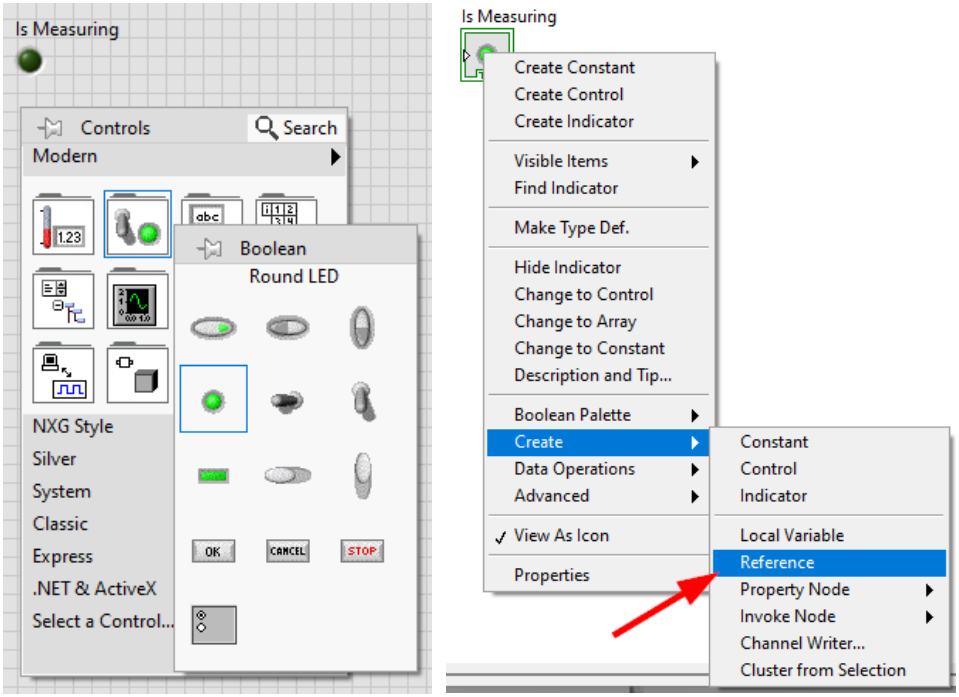
To create an instance of the PalmSens class select it in the Project Explorer and drag it to your VI.



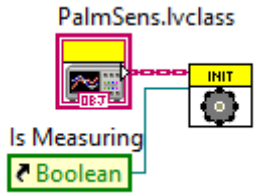
Switch to the block diagram and drag and drop the Initialise function and connect it to the PalmSens.lvclass block.



Finally, a reference to a Boolean indicator needs to provided to the Initialise block. This indicator will signal when the instrument is running a measurement and when the measurement is finished. Switch back to the Front Panel and add a Boolean indicator. Then switch back to the block diagram right click on the indicator, go to Create and select Reference.



Finally, connect the Boolean reference to the Init block terminal to complete the setup of the PalmSens LabVIEW class.



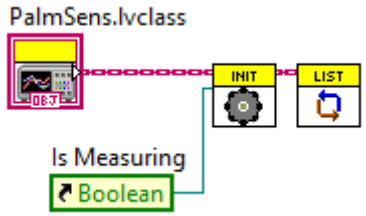
The following chapter will demonstrate how to use the functions of this class.

3 Connecting and Measuring

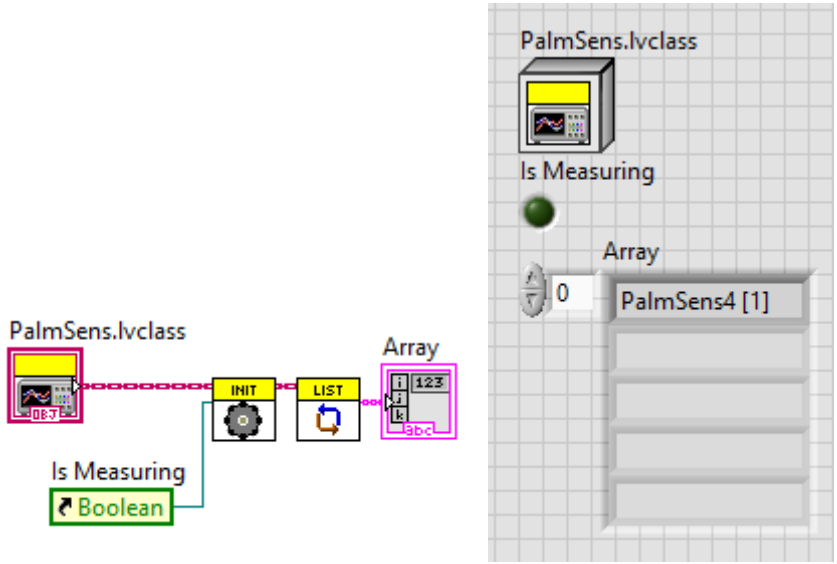
The following chapter details how to discover all available instruments, connect to an instrument, run measurements on the device and finally how to properly close a connection to a device. If you have not done so already follow the steps in chapter 2 to set up the LabVIEW class in your VI. For reference the result of the following steps are available in the BasicExample VI.

3.1 Discovering available instruments

To get a list of all available/connected instruments drag and drop the ListInstruments function VI from the PalmSens lvclass into your block diagram and connect it to the initialized instance of the PalmSens class.

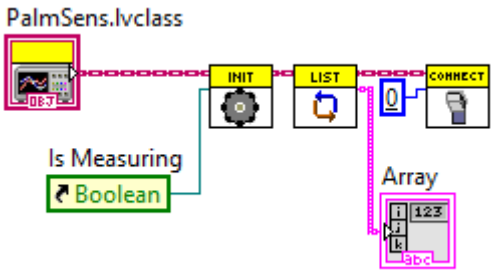


The result of the ListInstruments function is an array of strings. For this example, we have added an indicator to show the results.

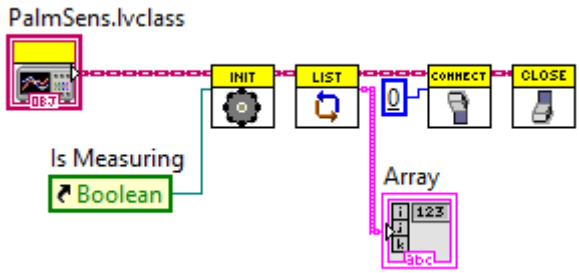


3.2 Connecting/disconnecting to/from instruments

Each instance of the PalmSens class can be connected to a single instrument. To connect to an instrument drag and drop the Connect function VI from the PalmSens lvclass into your block diagram and connect it to the initialized instance of the PalmSens class. To specify which instrument to connect to an integer value must be provided, this value is the index of the instrument in the array obtained from the ListInstruments block.



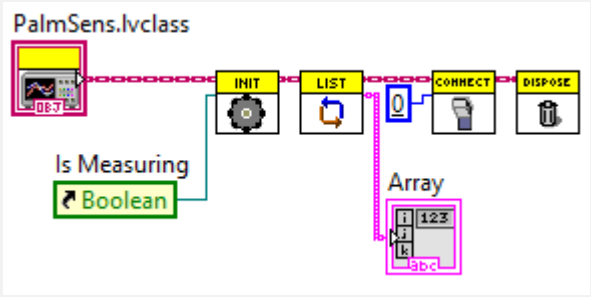
This diagram assumes there is an instrument connected. To disconnect from an instrument add the Disconnect function VI from the PalmSens lvclass into your block diagram and connect it after the Connect function.



Memory usage

Stopping a running VI with the abort button can result in memory leaks in the .NET libraries which can eventually cause the LabVIEW VI to crash.

Therefore it is highly recommended to always dispose the PalmSens class using the Dispose function. This function will abort any running measurements, disconnect the instruments and properly clean up the objects in the .NET libraries.



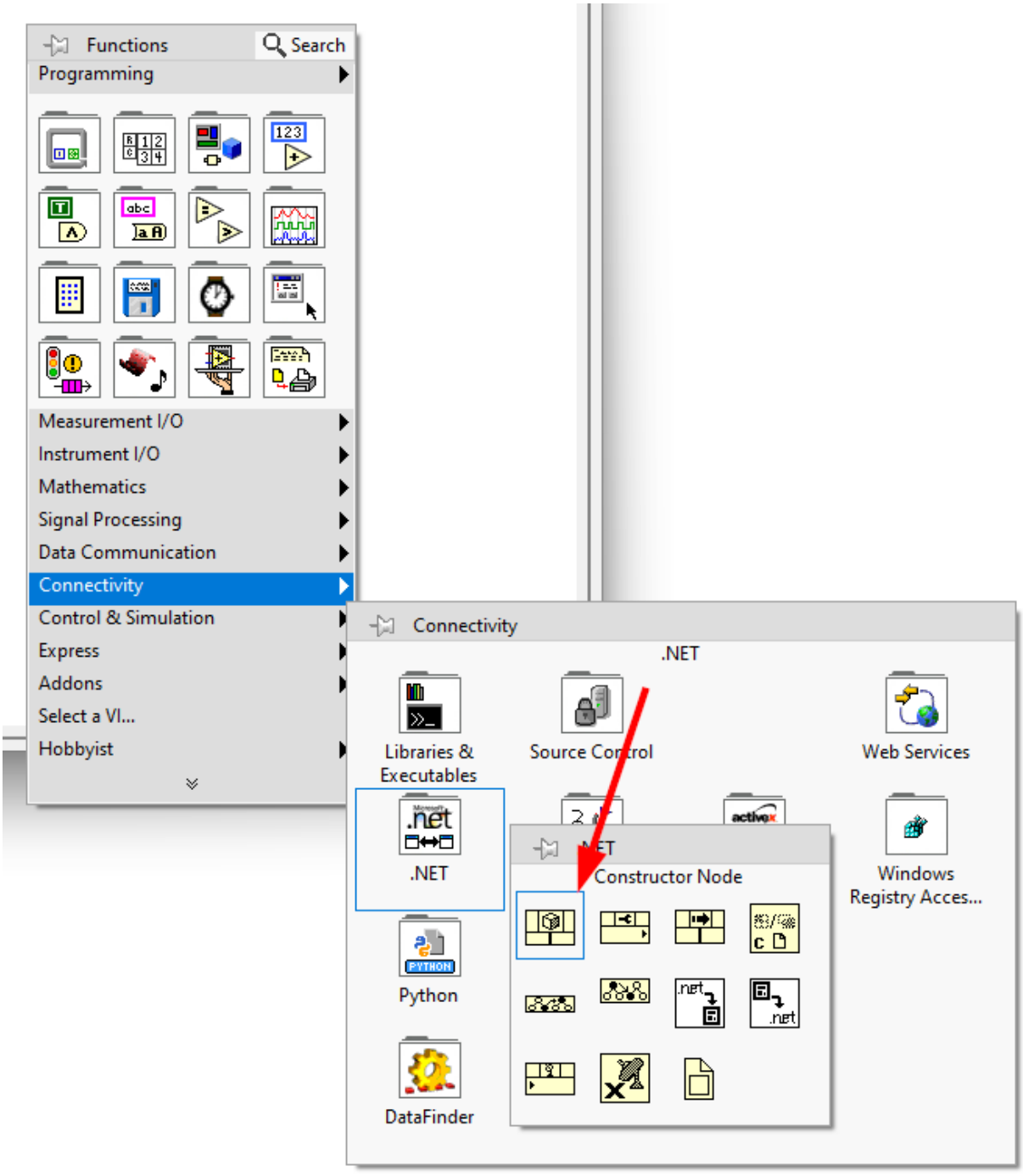
3.3 Measuring

Starting a measurement is done by sending method parameters to a PalmSens/EmStat device.

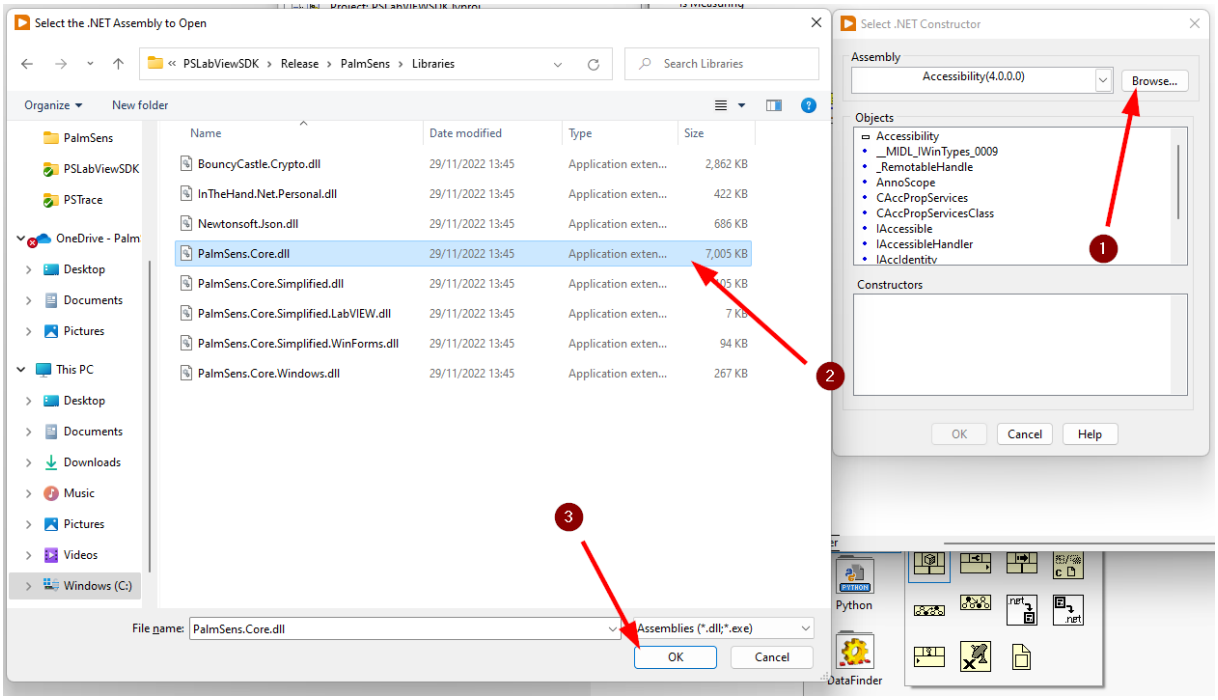
3.3.1 Creating a method

To run a measurement on your instrument, you will first have to create an instance of a method that defines the parameters for the technique to run. Appendix A provides an overview of all techniques and their respective parameters.

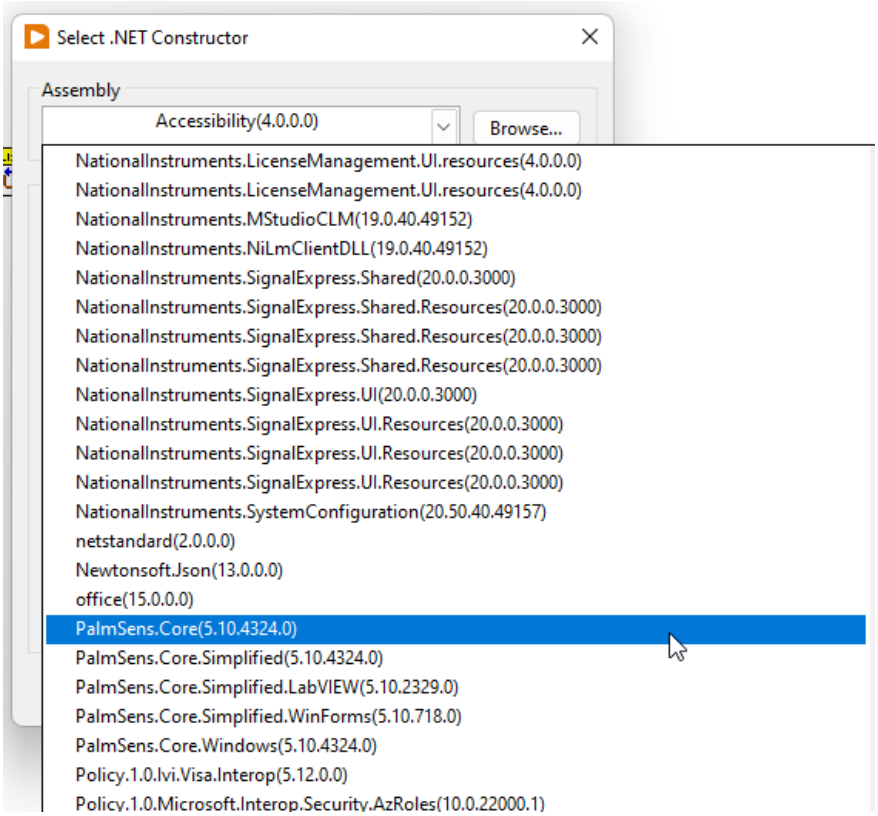
To create an instance of a method in LabVIEW add a .NET constructor node to your block diagram.



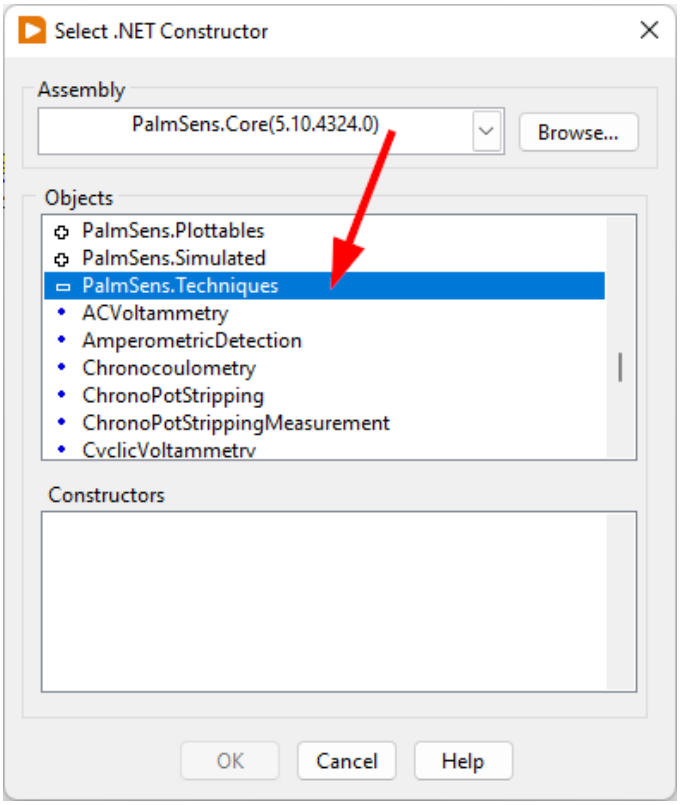
This will open Select .NET Constructor window. (1) Click on browse, (2) navigate to the PalmSens/Libraries folder included with the SDK and select the PalmSens.Core.dll, and (3) then click on OK.



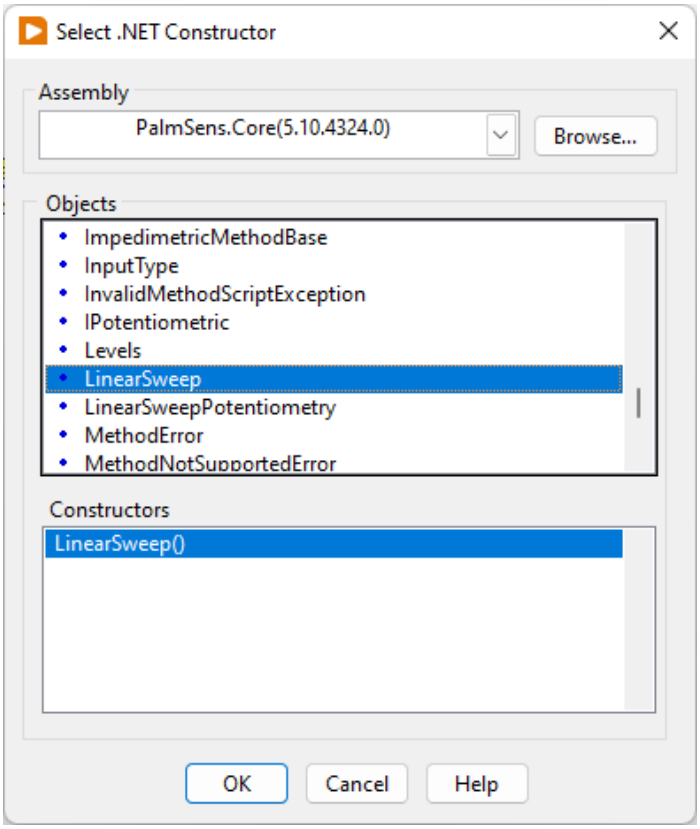
Alternatively you can select the PalmSens.Core Assembly from the dropdown list in the .NET Constructor Window.



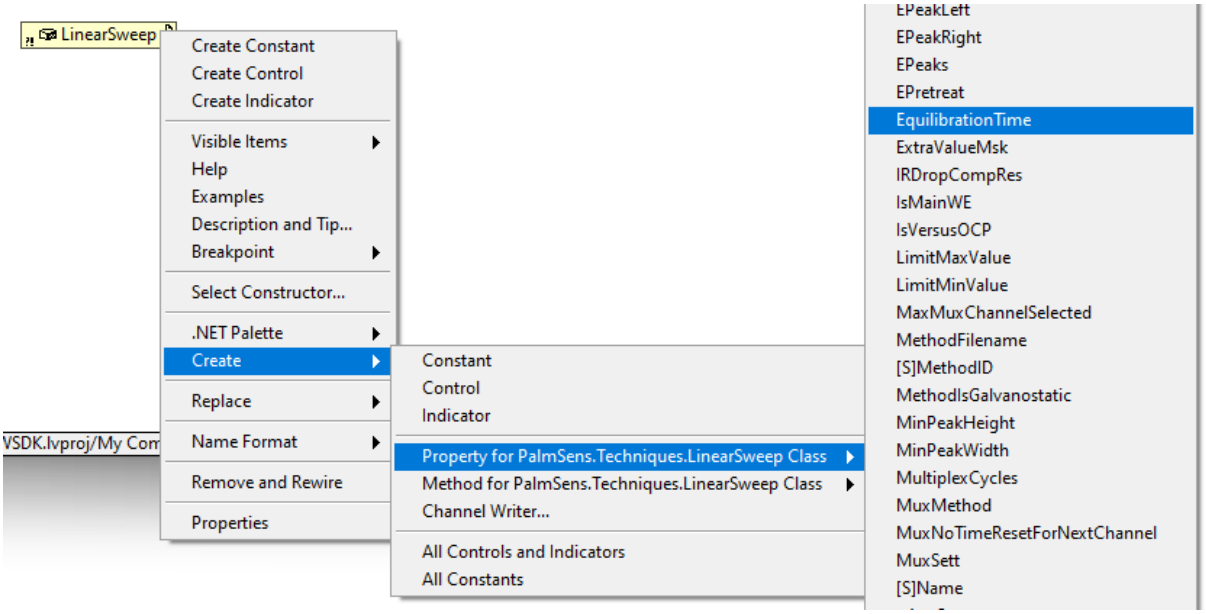
Then double click on the PalmSens.Techniques item near the bottom of the list to expand it.



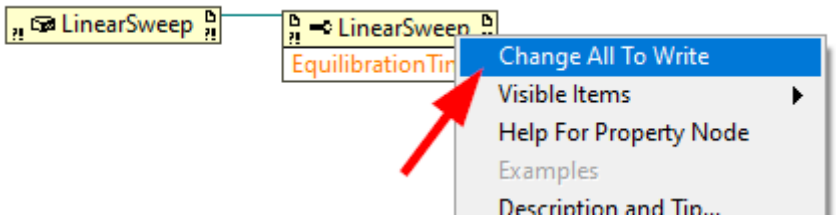
Select the technique you want to run and click on OK, for reference see appendix A and the PSTrace help documentation. For this example we will use the Linear Sweep Voltammetry technique.



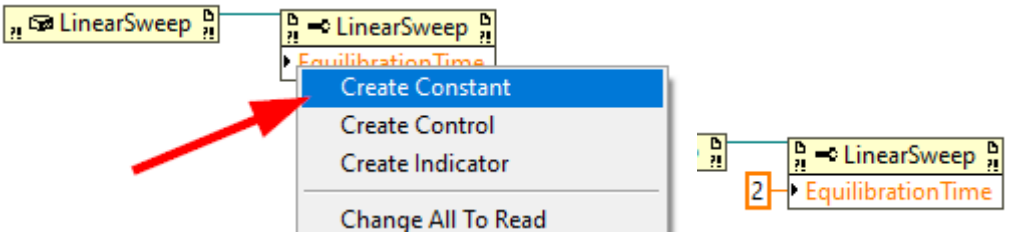
To define the parameters for a technique right click on the constructor node, go to Create, then to Property for PalmSens.Techniques… Class and select the parameter you want to set, Appendix A lists the relevant parameters for each technique.



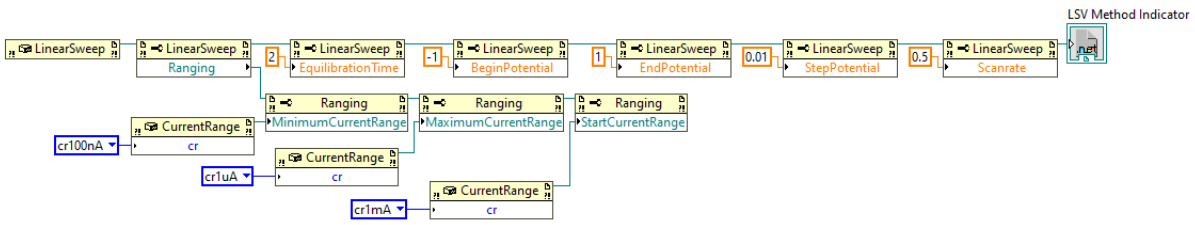
Connect the property to the constructor node and change it to write by right clicking on it.



Then create a constant by right clicking on the node and set that to the desired value.

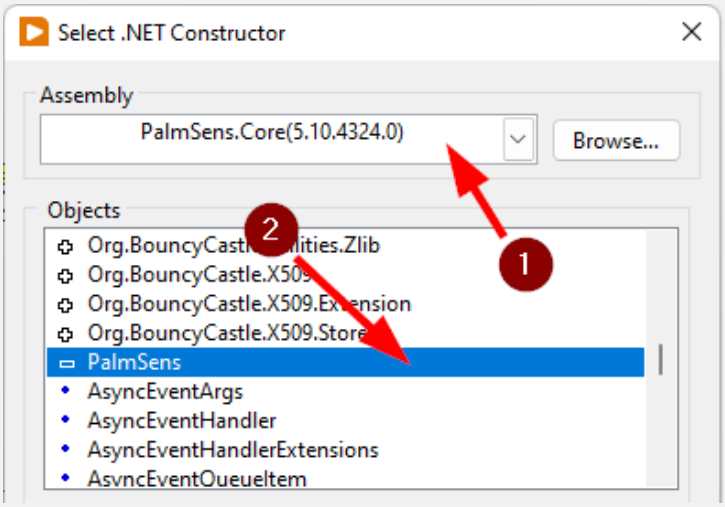


An example of what a configured method looks like

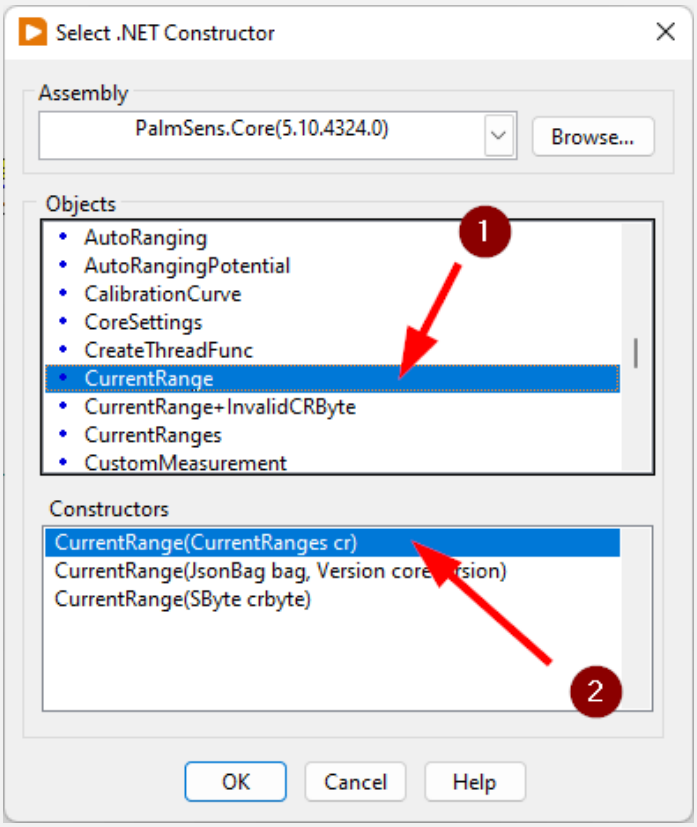


Current/Potential Ranges

Current and potential ranges are respectively stored in the Ranging and RangingPotential parameters. To set the minimum, maximum, and starting range you will need to create an instances of the current/potential ranges by adding .NET constructor nodes.

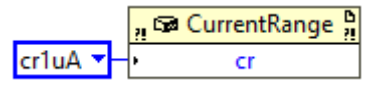


(1) Make sure that you selected the PalmSens.Core Assembly in the Select .NET Constructor window and (2) expand the PalmSens item by double clicking on it.

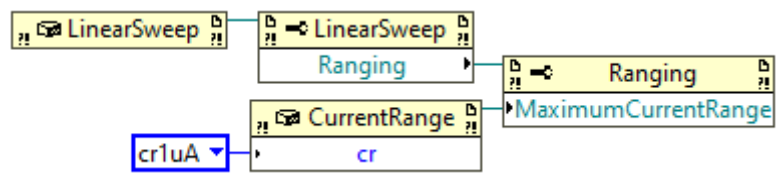


(1) Select CurrentRange, or PotentialRange when defining the Potential Ranges, and (2) select the constructor with the CurrentRanges cr or PotentialRanges pr argument respectively.

Current/Potential Ranges, continued

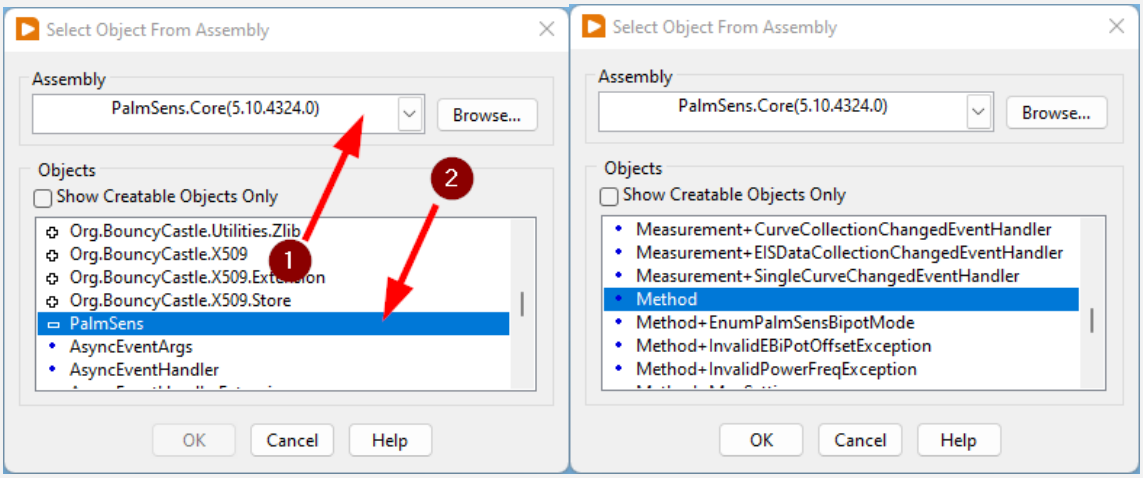


Add a constant value to the cr/pr node and select the range from the list. These current ranges can then be set to the Ranging/RangingPotential Maximum, Minimum, and Start parameters.



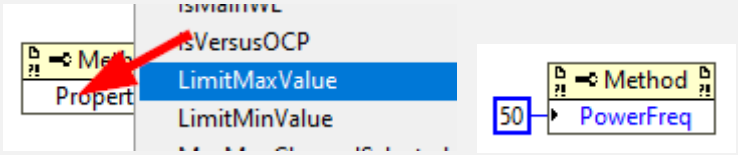
Mains Frequency

To eliminate noise induced by other electrical appliances it is highly recommended to set your regional mains frequency (50/60 Hz) in the static property PalmSens.Method.PowerFreq. Add a .NET property node to your Block Diagram.



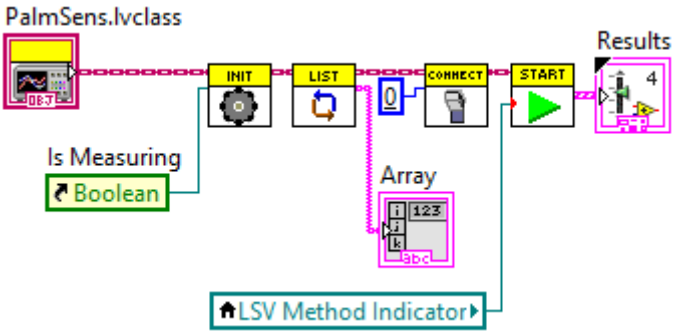
(1) Make sure that you selected the PalmSens.Core Assembly in the Select Object From Assembly window and (2) expand the PalmSens item by double clicking on it.

Select Method and in the Block Diagram click on the property and select PowerFreq.



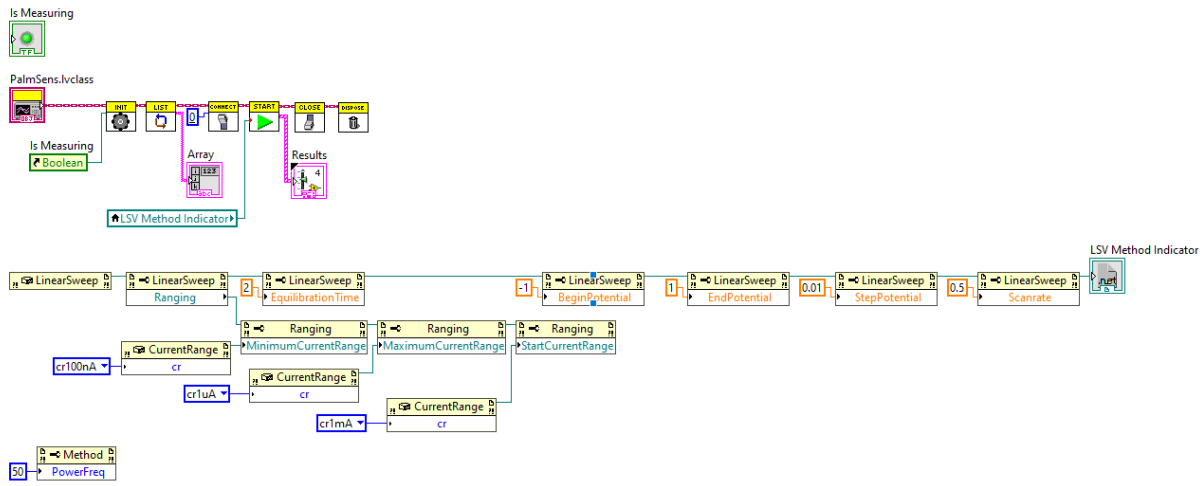
3.3.2 Running a measurement

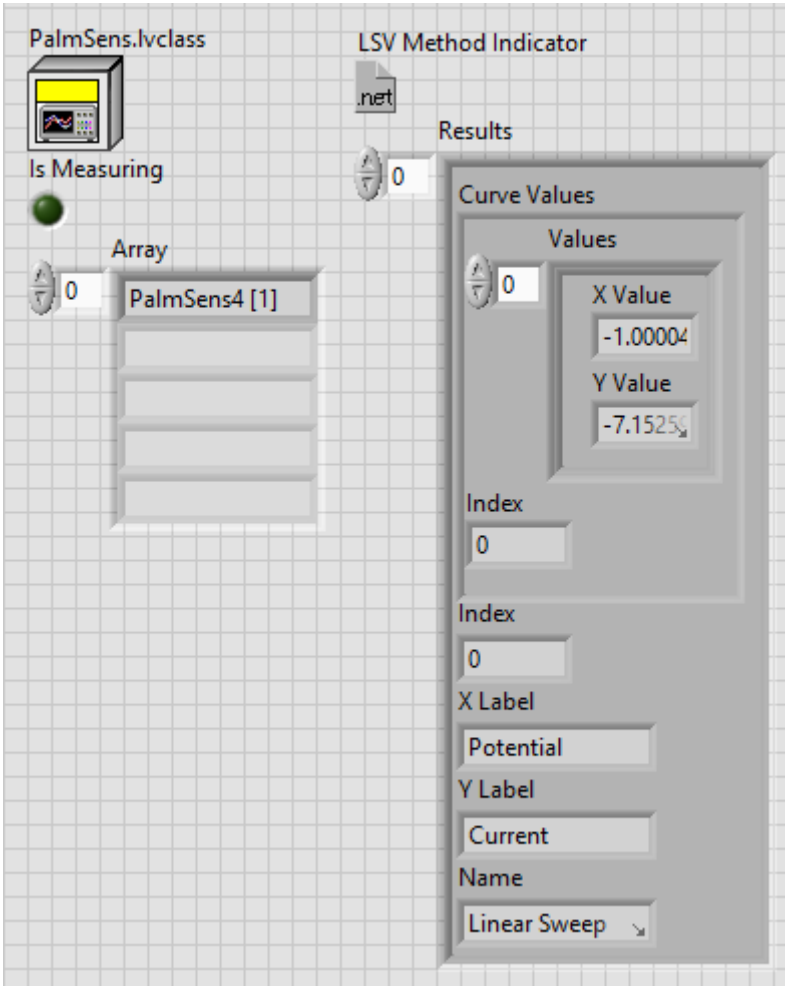
To run a measurement you must be connected to an instrument, 3.2, and need an instance of a method, 3.3.1. To run a measurent drag and drop the Measure function VI from the PalmSens lvclass into your block diagram and connect it to the PalmSens class.



Make sure to connect the method to the input. The output can be stored in an indicator, the easiest way to view the results is to right click on the ouput node and select create indicator. The type of the output is defined in MeasurementResults.ctl, it is a set of x and y values with strings for the name and units. Similar to PSTrace a Linear Sweep Voltammetry measurement will give you one set of current and potential values, a Cyclic Voltammetry measurement will give you multiple sets of current and potential values corresponding to the amount of scans, and an Chronopotentiometry / Amperometric Detection measurement will give you a set of current and time values. When extra values are also recorded these will return as additional sets of x and y values and the same applies to multiplexer scan results.

The final diagram of the BasicExample VI.





Blocking behavior of Measure function
The Measure function will block the VI until the measurement is complete, for more information on this refer to chapter 4 and the BasicUIExample.

3.4 MethodSCRIPT™

The MethodSCRIPT™ scripting language is designed to integrate our OEM potentiostat (modules) effortlessly in your hardware setup or product.

MethodSCRIPT™ allows developers to program a human-readable script directly into the potentiostat module by means of a serial (TTL) connection. The simple script language allows for running all supported electrochemical techniques and makes it easy to combine different measurements and other tasks.

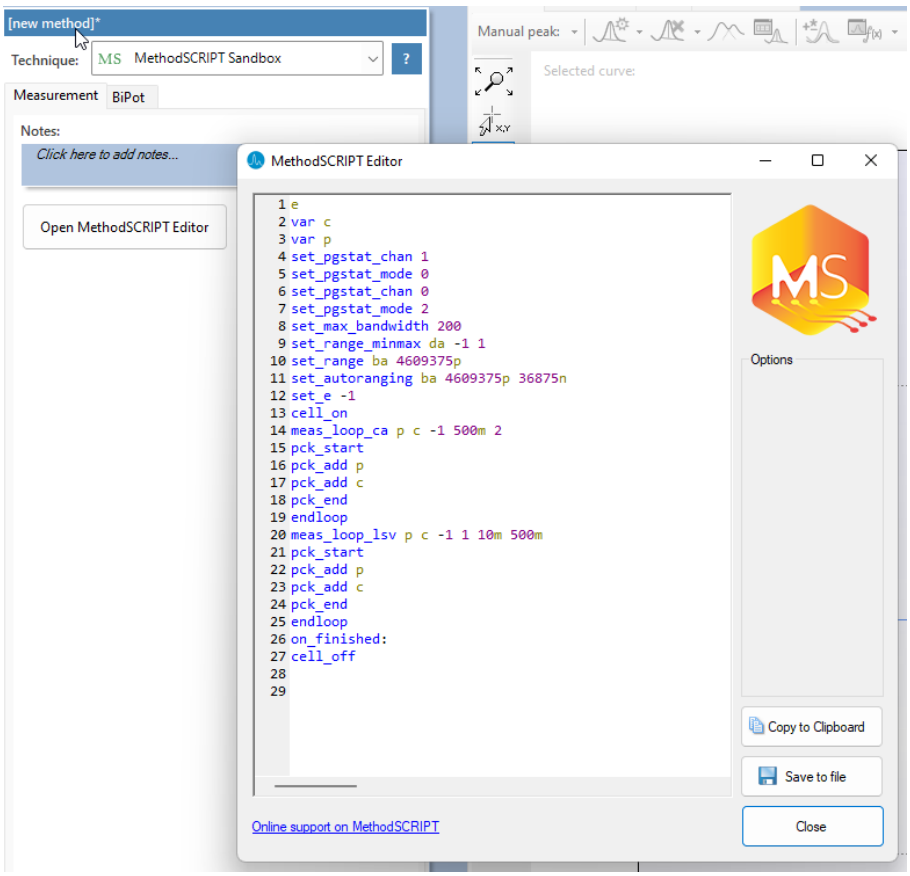
More script features include:

- Use of variables
- (Nested) loops
- Logging results to an SD card
- Digital I/O for example for waiting for an external trigger
- Reading auxiliary values like pH or temperature
- Going to sleep or hibernate mode

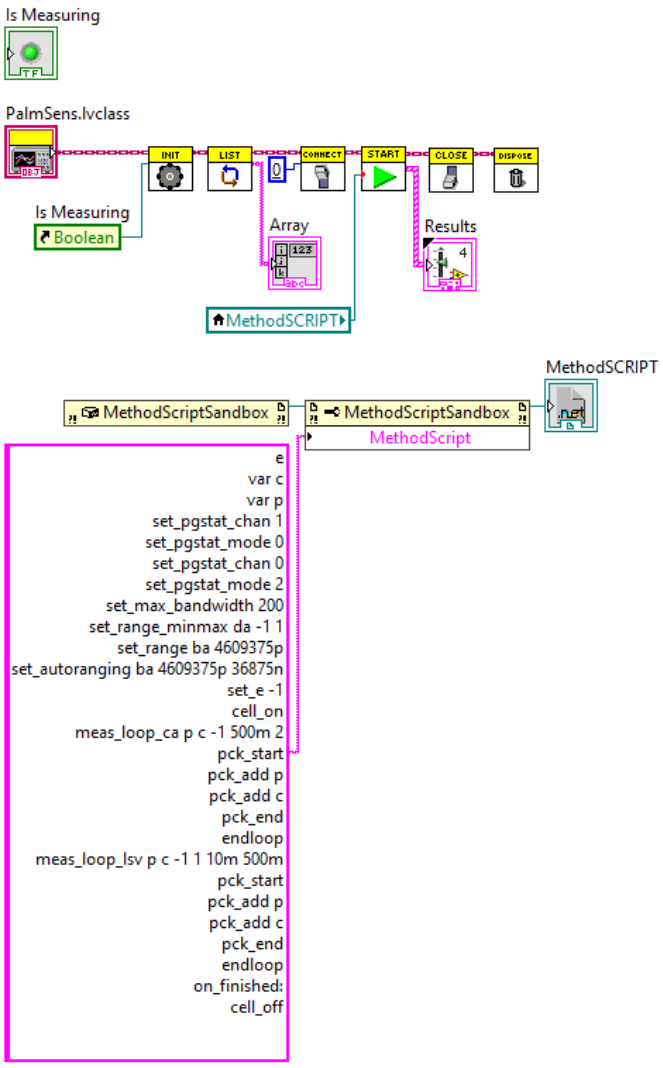
See for more information: www.palmsens.com/methodscript

3.4.1 Sandbox Measurements

PSTrace includes an option to make use MethodSCRIPT™ Sandbox to write and run scripts. This is a great place to test MethodSCRIPT™ measurements to see what the result would be. That script can then be used in the MethodScriptSandbox technique in the SDK as demonstrated below.



The MethodSCRIPTExample VI demonstrates how to run this measurement on a compatible instrument, i.e. the Sensit, EmStat Pico and EmStat4 series instruments.



SandboxMeasurements parse and store the variables sent in pcks. Sets of x and y values are generated automatically for each meas_loop that defines a pck with two or more variables, scripts with multiple meas_loops will generate sets. The first variable in the pck will be set as the x-axis and a set is created for each subsequent variable in the pck. Please note that to plot data versus time you will need to add a variable with the time to the pck.

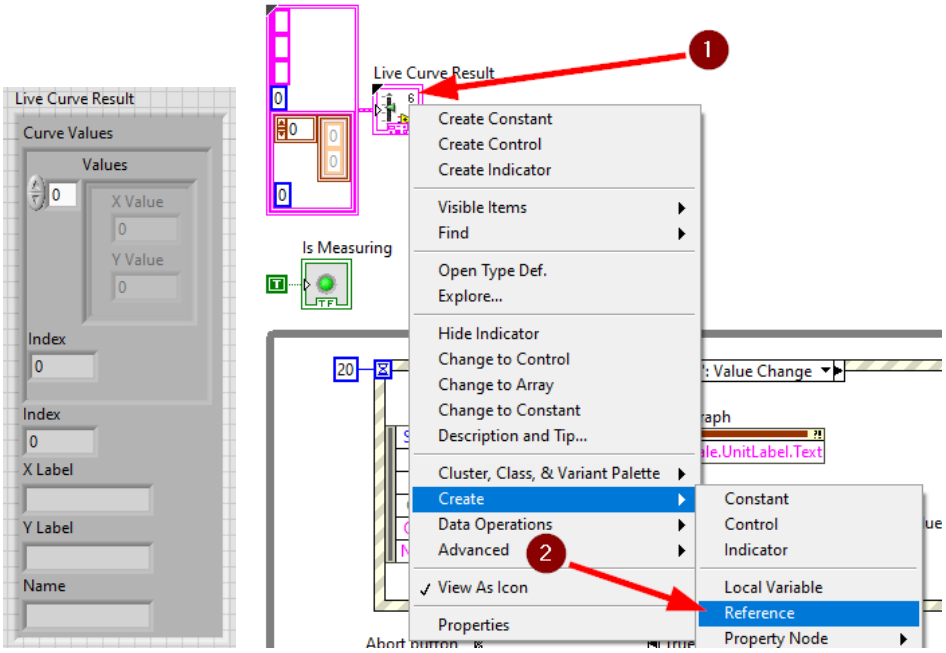
In the example above two sets of x and y values will be generated.

4 Control and visualization of running measurements

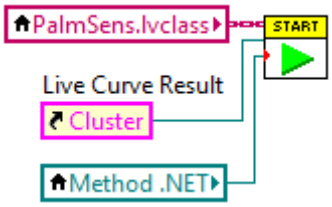
When a measurement is running the VI or loop the measure function VI is in will be blocked until the measurement is done. This chapter and the BasicUIExample detail how you can work around this to plot/process results in real-time and abort a running measurement.

4.1.1 Real-time visualization/processing of measurement data

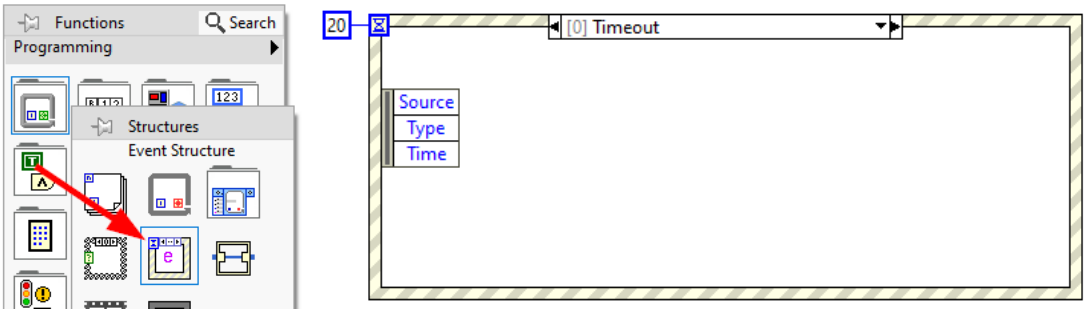
The Measure function VI has an input terminal to which you can connect a reference to an indicator of the cluster defined in the LiveCurveResult.ctl type definition. You can add this by dragging and dropping the LiveCurveResult.ctl on to your front panel.



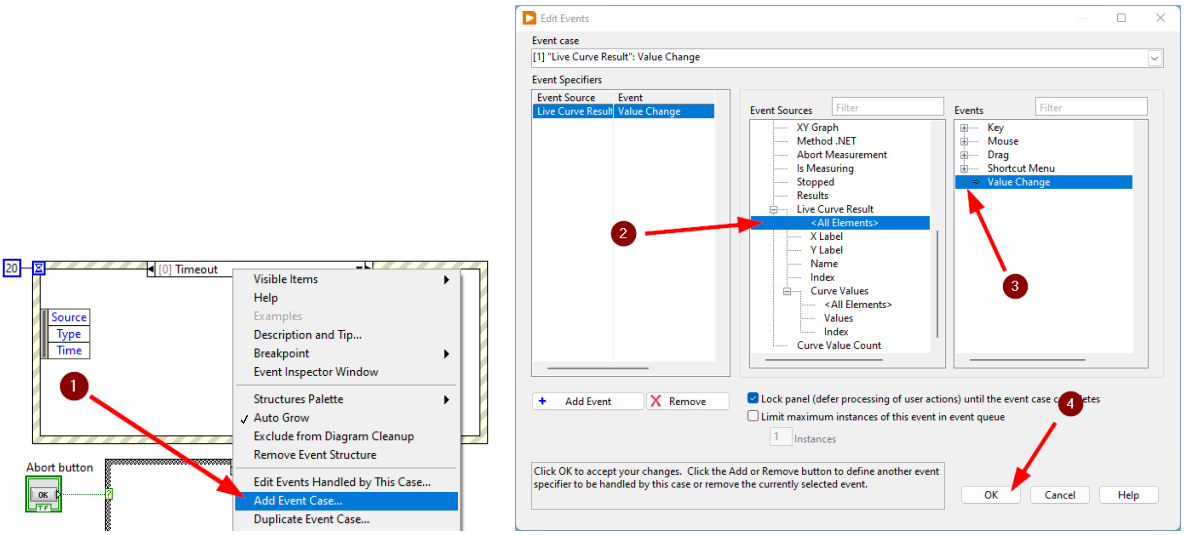
(1) Go to the indicator for the LiveCurveResult in the block diagram and (2) right click on it, go to Create and select Reference. The resulting reference can then be connected to the Measure function VI.



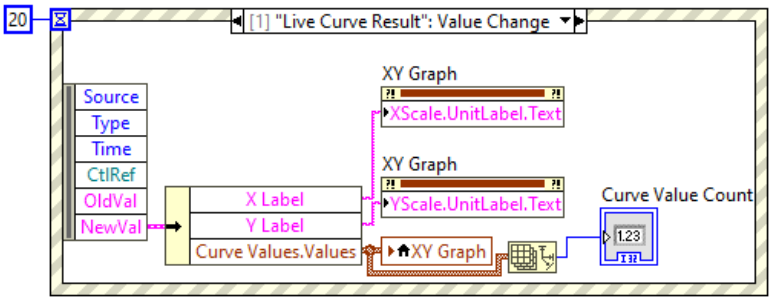
The values of the LiveCurveResult will be updated during while the measurement is running and LabVIEW receives a signal for each of these updates. The event block allows you to execute something each time a signal is received. To receive measurement data in real-time the Event Structure should be placed inside a loop.



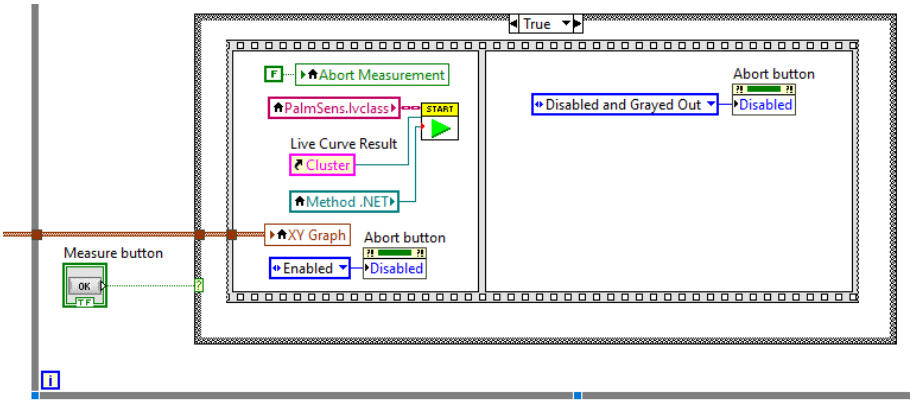
The Event Structure has a timeout event setup by default, if you want to be able to use the loop the Event Structure is placed in for other things it is highly recommended to define the timeout of the Event Structure in the top left corner.

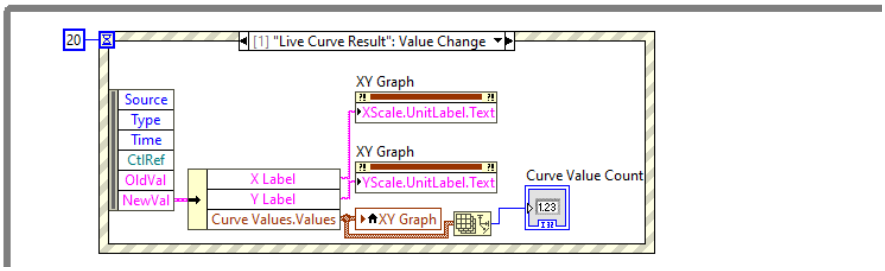


(1) Next you will need to add an Event case to the Event Structure. (2) In the Event window expand the LiveCurveResult in the Event Sources frame and select <All Elements>, (3) then select Value Change in the Events frame, and (4) click on OK.



The BasicUIExample uses this Event Structure to update the plot.

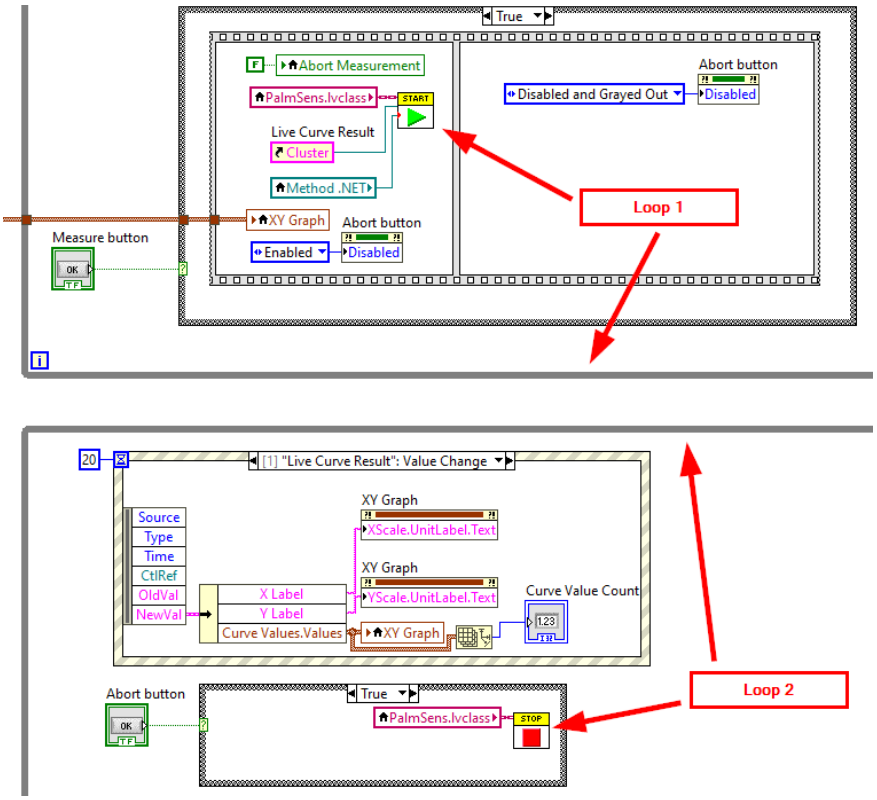




To be able to visualize/process these results the Measure function VI and event structure cannot be in the same loop.

4.1.2 Controlling the instrument when a measurement is running

To add the functionality of aborting a running measurement drag and drop the AbortMeasurement function VI from the PalmSens lvclass into your block diagram.



Make sure that the AbortMeasurement function VI and the Measure function VI are placed in separate loops. Otherwise, the most likely scenario will be that LabVIEW will postpone executing the abort command until after the measurement is finished. This also applies applies to the Disconnect and Dispose function VI commands and any other UI or blocks that you want to be able to execute in parallel to a measurement.

5 Appendix A: Parameters for each technique

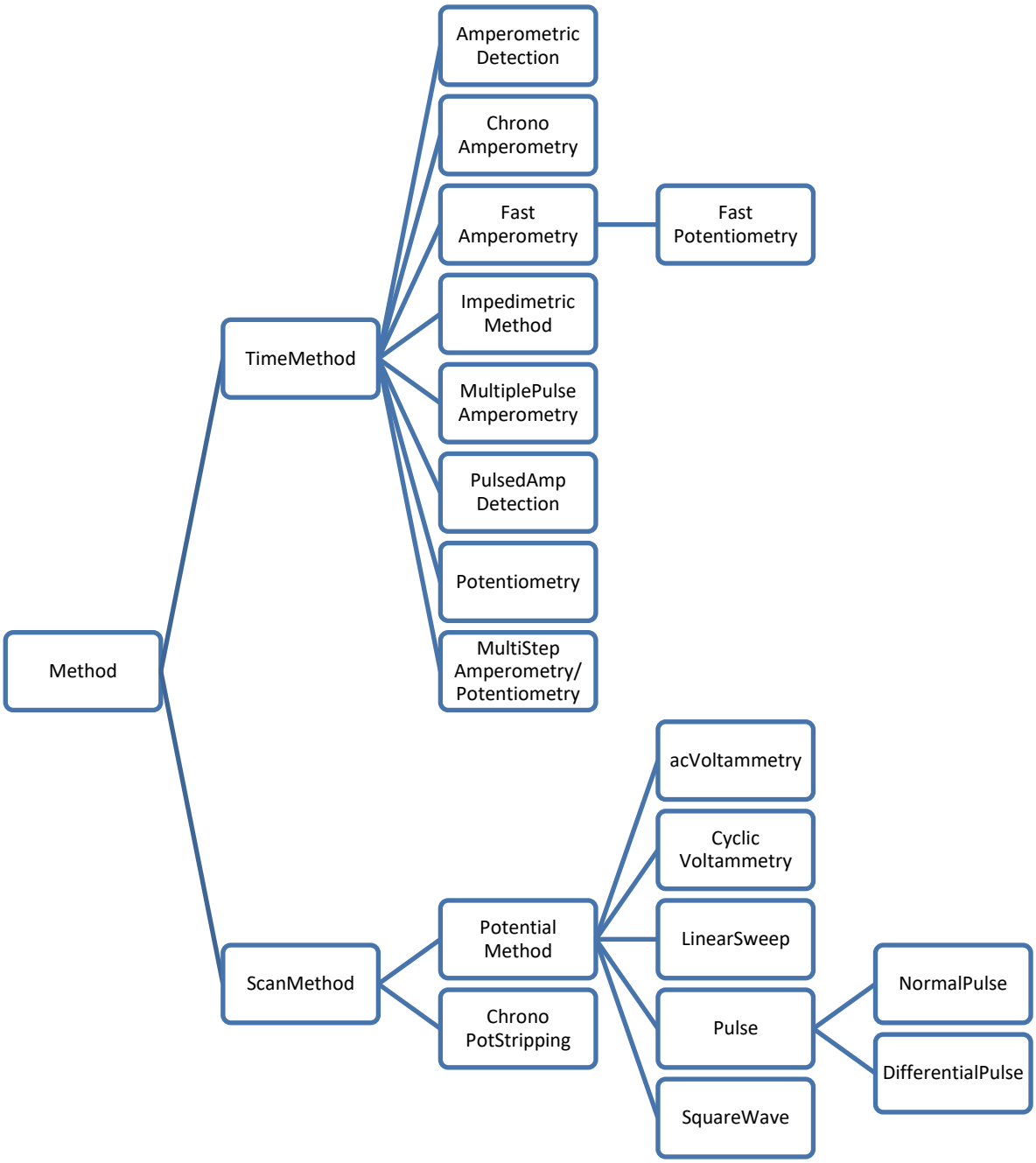
All applicable parameters for each technique can be found here. For the inheritance hierarchy of the the techniques, see section 3 in this document. See section ‘Available techniques’ in the PSTrace manual for more information about the techniques.

Each technique is identified by a specific integer value. This integer value can be used to create a class derived from the corresponding technique, as follows:

```
PalmSens.Method.FromTechniqueNumber(integervalue)
```

The integer values are indicated in this appendix inside the brackets [ ] following each technique name.

The techniques are also directly available from the **PalmSens.Techniques** namespace. Please refer to the PSTrace manual for explanations and expected values for each parameter.


Method
TimeMethod
Amperometric Detection
Chrono Amperometry
Fast Amperometry
Fast Potentiometry
Impedimetric Method
MultiplePulse Amperometry
PulsedAmp Detection
Potentiometry
MultiStep Amperometry/ Potentiometry
ScanMethod
Potential Method
acVoltammetry
Cyclic Voltammetry
LinearSweep
Pulse
NormalPulse
DifferentialPulse
SquareWave
Chrono PotStripping

5.1 Common properties

Property	Description	Type
Technique	The technique number used in the firmware	System.Int
Notes	Some user notes for use with this method	System.String
StandbyPotential	Standby Potential (for use with cell on after measurement)	System.Float
StandbyTime	Standby time (for use with cell on after measurement)	System.Float
CellOnAfterMeasurement	Enable/disable cell after measurement	System.Boolean
MinPeakHeight	Determines the minimum peak height in µA. Peaks lower than this value are neglected.	System.Float
MinPeakWidth	The minimum peak width, in the unit of the curves X axis. Peaks narrower than this value are neglected.	System.Float
SmoothLevel	The smoothlevel to be used. -1 = none 0 = no smooth (spike rejection only) 1 = 5 points 2 = 9 points 3 = 15 points 4 = 25 points	System.Int
Ranging	Ranging information, settings defining the minimum/maximum/starting current range	PalmSens.Method.Ranging
PowerFreq	Adjusts sampling on instrument to account for mains frequency. It accepts two values: 50 for 50Hz 60 for 60Hz	System.Int

5.2 Pretreatment settings

The following properties specify the measurements pretreatment settings:

Property	Description	Type
ConditioningPotential	Conditioning potential in volt	System.Float
ConditioningTime	Conditioning duration in seconds	System.Float
DepositionPotential	Deposition potential in volt	System.Float
DepositionTime	Deposition duration in seconds	System.Float
EquilibrationTime	Equilibration duration in seconds. BeginPotential is applied during equilibration and the device switches to the appropriate current range	System.Float

5.3 Linear Sweep Voltammetry (LSV) [0]

Class: Palmsens.Techniques.LinearSweep

Property	Description	Type
BeginPotential	Potential where scan starts.	System.Float
EndPotential	Potential where measurement stops.	System.Float
StepPotential	Step potential	System.Float
Scanrate	The applied scan rate. The applicable range depends on the value of E step since the data acquisition rate is limited by the connected instrument.	System.Float

5.4 Differential Pulse Voltammetry (DPV) [1]

Class: Palmsens.Techniques.DifferentialPulse

Property	Description	Type
BeginPotential	Potential where scan starts.	System.Float
EndPotential	Potential where measurement stops.	System.Float
StepPotential	Step potential	System.Float
Scanrate	The applied scan rate. The applicable range depends on the value of E step since the data acquisition rate is limited by the connected instrument.	System.Float
PulsePotential	Pulse potential	System.Float
PulseTime	The pulse time	System.Float

5.5 Square Wave Voltammetry (SWV) [2]

Class: Palmsens.Techniques.SquareWave

Property	Description	Type
BeginPotential	Potential where scan starts.	System.Float
EndPotential	Potential where measurement stops.	System.Float
StepPotential	Step potential	System.Float
PulseAmplitude	Amplitude of square wave pulse. Values are half peak-to-peak.	System.Float
Frequency	The frequency of the square wave	System.Float

5.6 Normal Pulse Voltammetry (NPV) [3]

Class: Palmsens.Techniques.NormalPulse

Property	Description	Type
BeginPotential	Potential where scan starts.	System.Float
EndPotential	Potential where measurement stops.	System.Float
StepPotential	Step potential	System.Float
Scanrate	The applied scan rate. The applicable range depends on the value of E step since the data acquisition rate is limited by the connected instrument.	System.Float
PulseTime	The pulse time	System.Float

5.7 AC Voltammetry (ACV) [4]

Class: Palmsens.Techniques.ACVoltammetry

Property	Description	Type
BeginPotential	Potential where scan starts.	System.Float
EndPotential	Potential where measurement stops.	System.Float
StepPotential	Step potential	System.Float
SineWaveAmplitude	Amplitude of sine wave. Values are RMS	System.Float
Frequency	The frequency of the AC signal	System.Float

5.8 Cyclic Voltammetry (CV) [5]

Class: Palmsens.Techniques.CyclicVoltammetry

Property	Description	Type
BeginPotential	Potential where scan starts and stops.	System.Float
Vtx1Potential	First potential where direction reverses.	System.Float
Vtx2Potential	Second potential where direction reverses.	System.Float
StepPotential	Step potential	System.Float
Scanrate	The applied scan rate. The applicable range depends on the value of E step since the data acquisition rate is limited by the connected instrument.	System.Float
nScans	The number of repetitions for this scan	System.Float

5.8.1 Fast Cyclic Voltammetry Scans

Class: Palmsens.Techniques.FastCyclicVoltammetry

Outdated class. PalmSens 3 and 4 only. CV's with sampling over 5000 data points per second, use the regular **Palmsens.Techniques.CyclicVoltammetry()** constructor instead.

5.9 Chronopotentiometric Stripping (SCP) [6]

Class: PalmSens.Techniques.ChronoPotStripping

Property	Description	Type
EndPotential	Potential where measurement stops.	System.Float
MeasurementTime	The maximum measurement time. This value should always exceed the required measurement time. It only limits the time of the measurement. When the potential response is erroneously and E end is not found within this time, the measurement is aborted.	System.Float
AppliedCurrentRange	The applied current range	PalmSens. CurrentRange
Istrip	If specified as 0, the method is called chemical stripping otherwise it is constant current stripping. The current is expressed in the applied current range.	System.Float

5.10 Chronoamperometry (CA) [7]

Class: PalmSens.Techniques.AmperometricDetection

Property	Description	Type
Potential	Potential during measurement.	System.Float
IntervalTime	Time between two current samples.	System.Float
RunTime	Total run time of scan.	System.Float

5.11 Pulsed Amperometric Detection (PAD) [8]

Class: PalmSens.Techniques.PulsedAmpDetection

Property	Description	Type
Potential	The dc or base potential.	System.Float
PulsePotentialAD	Potential in pulse. Note that this value is not relative to dc/base potential, given above.	System.Float
PulseTime	The pulse time.	System.Float
tMode	DC: I(dc) measurement is performed at potential E pulse: I(pulse) measurement is performed at potential E pulse differential: I(dif) measurement is I(pulse) - I(dc)	PalmSens.Techniques. PulsedAmpDetection. enumMode
IntervalTime	Time between two current samples.	System.Float
RunTime	Total run time of scan.	System.Float

5.12 Fast Amperometry (FAM) [9]

Class: PalmSens.Techniques.FastAmperometry

Property	Description	Type
EqPotentialFA	Equilibration potential at which the measurement starts.	System.Float
Potential	Potential during measurement.	System.Float
IntervalTimeF	Time between two current samples.	System.Float
RunTime	Total run time of scan.	System.Float

5.13 Chronopotentiometry (CP) [10]

Class: PalmSens.Techniques.Potentiometry

Property	Description	Type
Current	The current to apply. The unit of the value is the applied current range. So if 10 uA is the applied current range and 1.5 is given as value, the applied current will be 15 uA.	System.Float
AppliedCurrentRange	The applied current range.	PalmSens. CurrentRange
RunTime	Total run time of scan.	System.Float
IntervalTime	Time between two potential samples.	System.Float

5.13.1 Open Circuit Potentiometry (OCP)

Class: PalmSens.Techniques.OpenCircuitPotentiometry

The same as setting the Current to 0.

Property	Description	Type
RunTime	Total run time of scan.	System.Float
IntervalTime	Time between two potential samples.	System.Float

5.14 Multiple Pulse Amperometry (MPAD) [11]

Class: PalmSens.Techniques.MultiplePulseAmperometry

Property	Description	Type
E1	First potential level in which the current is recorded	System.Float
E2	Second applied potential level	System.Float
E3	Third applied potential level	System.Float
t1	The duration of the first applied potential	System.Float
t2	The duration of the second applied potential	System.Float
t3	The duration of the third applied potential	System.Float
RunTime	Total run time of scan.	System.Float

5.15 Electrochemical Impedance Spectroscopy (EIS)

Class: PalmSens.Techniques.ImpedimetricMethod

The most common properties are described first. These are used for a typical EIS measurement, a scan over a specified range of frequencies (i.e. using the default properties **ScanType** = **ImpedimetricMethod**.
enumScanType.FixedPotential and **FreqType** = **ImpedimetricMethod.enumFrequencyType.Scan**). The additional properties used for a **TimeScan** and a **PotentialScan** are detailed separately in next sections.

Property	Description	Type
ScanType	Scan type specifies whether a single or multiple frequency scans are performed. When set to FixedPotential a single scan will be performed, this is the recommended setting. **The TimeScan and PotentialScan are not fully supported in the SDK**, we highly recommend you to implement yourself. A TimeScan performs repeated scans at a given time interval within a specified time range. A PotentialScan performs scans where the DC Potential of the applied sine is incremented within a specified range. A PotentialScan should not be performed versus the OCP.	ImpedimetricMethod. enumScanType

Potential	The DC potential of the applied sine	System.Float
Eac	The amplitude of the applied sine in RMS (Root Mean Square)	System.Float
FreqType	Frequency type specifies whether to perform a scan on a range of frequencies or to measure a single frequency. The latter option can be used in combination with a TimeScan or a PotentialScan.	ImpedimetricMethod.enumFrequencyType
MaxFrequency	The highest frequency in the scan, also the frequency at which the measurement is started	System.Float
MinFrequency	The lowest frequency in the scan	System.Float
nFrequencies	The number of frequencies included in the scan	System.Int
SamplingTime	Each measurement point of the impedance spectrum is performed during the period specified by SamplingTime. This means that the number of measured sine waves is equal to SamplingTime * frequency. If this value is less than 1 sine wave, the sampling is extended to 1 / frequency. So for a measurement at a frequency, at least one complete sine wave is measured. Reasonable values for the sampling are in the range of 0.1 to 1 s.	System.Float
MaxEqTime	The impedance measurement requires a stationary state. This means that before the actual measurement starts, the sine wave is applied during MaxEqTime only to reach the stationary state. The maximum number of equilibration sine waves is however 5. The minimum number of equilibration sines is set to 1, but for very low frequencies, this time is limited by MaxEqTime. The maximum time to wait for stationary state is determined by the value of this parameter. A reasonable value might be 5 seconds. In this case this parameter is only relevant when the lowest frequency is less than 1/ 5 s so 0.2 Hz.	System.Float

5.15.1 Time Scan

In a Time Scan impedance spectroscopy measurements are repeated for a specific amount of time at a specific interval. The SDK does not support this feature fully, we recommend you to design your own implementation for this that suits your demands.

Property	Description	Type
RunTime	RunTime is not the total time of the measurement, but the time in which a measurement iteration can be started. If a frequency scan takes 18 seconds and is measured at an interval of 19 seconds for a RunTime of 40 seconds three iterations will be performed.	System.Float
IntervalTime	IntervalTime specifies the interval at which a measurement iteration should be performed, however if a measurement iteration takes longer than the interval time the next measurement will not be triggered until after it has been completed.	System.Float

5.15.2 Potential Scan

In a Potential Scan impedance spectroscopy measurements are repeated over a range of DC potential values. The SDK does not support this feature fully, we recommend you to design your own implementation for this that suits your demands.

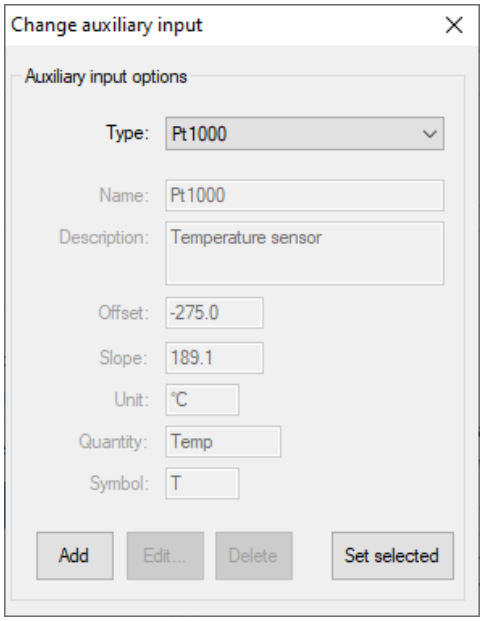
Property	Description	Type
BeginPotential	The DC potential of the applied sine wave to start the series of iterative measurements at.	System.Float
EndPotential	The DC potential of the applied sine wave at which the series of iterative measurements ends.	System.Float
StepPotential	The size of DC potential step to iterate with.	System.Float

5.16 Recording extra values (BiPot, Aux, CE Potential…)

The **PalmSens.Method.ExtraValueMsk** property allows you to record an additional value during your measurement. Not all techniques support recording extra values, the **SupportsAuxInput** and **SupportsBipot** properties are used to indicate whether a technique supports the recording of these values. The default value for **PalmSens.Method.ExtraValueMsk** is **PalmSens.ExtraValueMask.None**.

- None, no extra value recorded (default)
- Current
- Potential
- WE2, record BiPot readings (The behavior of the second working electrode is defined with the method's **BipotModePS** property. **EnumPalmSensBipotMode.Constant** sets it to a fixed potential and **EnumPalmSensBipotMode.Offset** sets it to an offset of the primary working electrode. The value in Volt of the fixed or offset potential is defined with the method's **BiPotPotential** property.)

- AuxInput, similar to PSTrace it is possible to configure the readings of the auxilliary input. Using the **PalmSens.AuxInput.AuxiliaryInput** class you can assign a name, offset, gain and unit to the auxilliary input. The following example demonstrates how to set up the Pt1000 temperature sensor from PSTrace.




```
psCommSimpleWinForms.comm.AuxInputSelected = new PalmSens.AuxInput.Auxiliar
yInputType(true, "Pt1000", "Temperature sensor", -275f, 189.1f,
             new PalmSens.Units.Temperature());
```

  The can be ignored and set to true, the second argument is the name, third is the description, fourth the offset, fifth the slope and the final argument is an instance of one of the unit classes in the **PalmSens.Units** namespace.
- Reverse, record reverse current as used by Square Wave Voltammetry
- PolyStatWE, not supported in the PalmSens SDK
- DCcurrent, record the DC current as used with AC Voltammetry
- CEPotential, PalmSens 4 only

The PSSDKBiPotAuxExample example project demonstrates how to record extra values.

5.17 Multiplexer

The **PalmSens.Method** class is also used to specify the multiplexer settings for sequential and alternating measurements. Alternating multiplexer measurements restricted to the chronoamperometry and chronopotentiometry techniques.

The enumerator property **PalmSens.Method.MuxMethod** defines the type multiplexer measurement.

```
methodCA.MuxMethod = MuxMethod.None; //Default setting, no multiplexer
methodCA.MuxMethod = MuxMethod.Alternatingly;
methodCA.MuxMethod = MuxMethod.Sequentially;

//The channels on which to measure are specified in a boolean array
PalmSens.Method.UseMuxChannel: methodCA.UseMuxChannel = new bool[] { true, true,
false, false, false, false, false, true };
```

The code above will perform a measurement on the first two and last channels of an 8-channel multiplexer. For a 16-channel multiplexer you would also need to assign true or false to the last 8 channels.

Alternating multiplexer measurement can only measure on successive channels and must start with the first channel (i.e. it is possible to alternatingly measure on channels 1 through 4 but it is not possible to alternatingly measure on channel 1, 3 and 5). The multiplexer functionality is demonstrated in the PSSDKMultiplexerExample project.

5.17.1 Multiplexer settings

When using a MUX8-R2 multiplexer the multiplexer settings must be set digitally instead of via the physical switches on the earlier multiplexer models. The type of multiplexer should be specified in the connected device's capabilities, when the multiplexer is connected before connecting to the software the capabilities are updated automatically. Otherwise, when using the MUX8-R2 the **PalmSens.Devices.DeviceCapabilities.MuxType** should be set to **PalmSens.Comm.MuxType.Protocol** manually or by calling **PalmSens.Comm.CommManager.ClientConnection.ReadMuxInfo**, **PalmSens.Comm.CommManager.ClientConnection.ReadMuxInfoAsync** when connected asynchronously.

For the MUX8-R2 the settings for a measurement are set in **PalmSens.Method.MuxSett** property with an instance of the **PalmSens.Method.MuxSettings** class. For manual control these settings can be set using the **PalmSens.Comm.ClientConnection.SetMuxSettings** function, **PalmSens.Comm.ClientConnection.SetMuxSettingsAsync** when connected asynchronously.

```
method.MuxSett = new Method.MuxSettings(false)
{
        CommonCERE = false,
        ConnSEWE = false,
        ConnectCERE = true,
        OCPMode = false,
        SwitchBoxOn = false,
        UnselWE = Method.MuxSettings.UnselWESetting.FLOAT
};
```

5.18 Versus OCP

The versus open circuit potential settings (OCP) are defined in the **PalmSens.Method.OCPmode**, **PalmSens.Method.OCPMaxOCPTime**, and **PalmSens.Method.OCPStabilityCriterion** properties. The OCPmode is a bitmask specifies which of the following technique dependent properties or combination thereof will be measured versus the OCP potential:

- Linear Sweep Voltammetry:
  - BeginPotential = 1
  - EndPotential = 2
- (Fast) Cyclic Voltammetry
  - Vtx1Potential = 1
  - Vtx2Potential = 2
  - BeginPotential = 4
- Chronoamperometry
  - Potential = 1
- Impedance Spectroscopy (Fixed potential and Time Scan)
  - Potential = 1
- Impedance Spectroscopy (Potential Scan)
  - BeginPotential = 1
  - EndPotential = 2

The progress and result of the versus OCP measurement step are reported in the **PalmSens.Comm.MeasureVersusOCP** class, which can be obtained by subscribing to the

PalmSens.Comm.CommManager.DeterminingVersusOCP event which is raised when the versus OCP measurement step is started.

```
//Defining versus OCP measurement step for a Cyclic Voltammetry measurement
_methodCV.OCPmode = 7; //Measure the (Vtx1Potential) 1 + (Vtx2Potential) 2 +
(BeginPotential) 4 = 7 versus the OCP potential
_methodCV.OCPMaxOCPTime = 10; //Sets the maximum time the versus OCP step can take to
10 seconds
_methodCV.OCPStabilityCriterion = 0.02f; //The OCP measurement will stop when the
change in potential over time is less than 0.02mV/s, when set to 0 the OCP measurement
step will always run for the OCPMaxOCPTime
```

5.19 Properties for EmStat Pico

There are two method parameters specific to the EmStat Pico. The **PalmSens.Method.PGStatMode** property sets the mode in which the measurement should be run, low power, high speed or max range. This mode can be set for all techniques but Electrochemical Impedance Spectroscopy. The second property is **PalmSens.Method.SelectedPotentiostatChannel** which let you choose on which channel the EmStat Pico should run the measurement.