MethodSCRIPT Example - Python





Last revision: March 22 2019

© 2019 PalmSens BV

www.palmsens.com

## Contents:

The example “MSConsoleExample.py” found in the “/MethodSCRIPTExample\_Python” folder demonstrates basic communication with the EmStat Pico using Python.

The “MSPlotCV.py” example demonstrates the common electrochemical technique: Cyclic Voltammetry and plots the resulting voltammogram.

The “MSPlotEIS.py” example demonstrates the Electrochemical Impedance Spectroscopy technique and plot the resulting Nyquist and Bode plots.

## Examples:

### Example 1: Console Example (MSConsoleExample.py)

This example opens a communication port, sends a MethodSCRIPT file, reads and parses the data and prints the parsed data (variable type, value, unit) on the console. The metadata (status, currentrange) are not parsed in this example.

The name of the com port connected to the EmStat Pico has to be set by looking up the in the Device manager in Control Panel in Windows, as shown below.

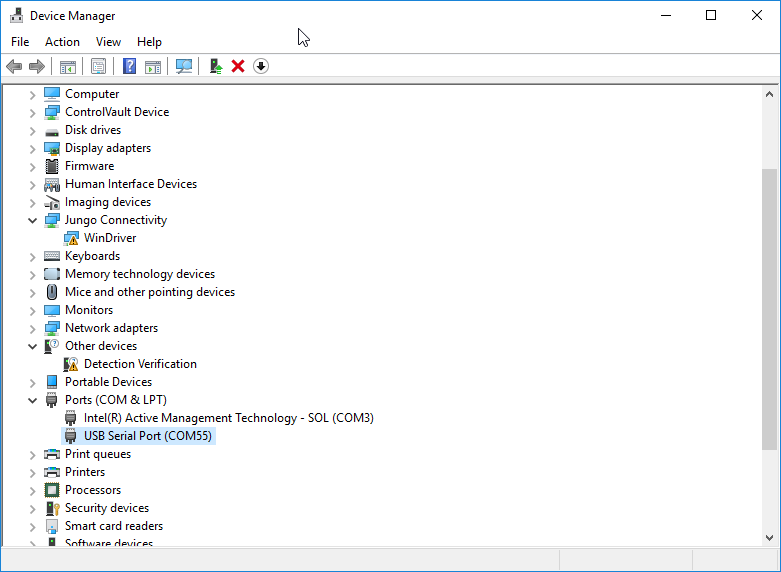


Figure 1: Available com ports in device manager

COM55 is being used as the Emstat Pico com port in this example.

myport = "COM55" #set the comport

### Example 2: Cyclic Voltammetry Plot Example (MSPlotCV.py)

This example performs a CV (Cyclic Voltammetry) measurement on a Palmsens Dummy Cell WE A (RedOx circuit) and plots the I vs E curve. The raw data from the measurement is also saved in a result file with *.dat* extension in the *data* sub folder.

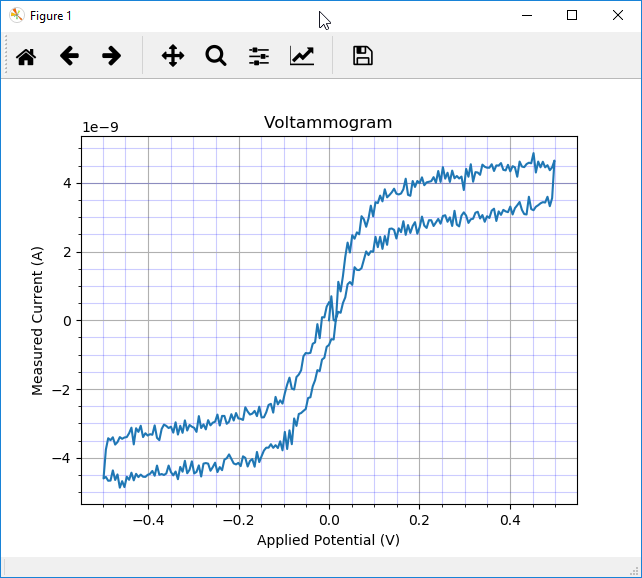


Figure 2: CV on Palmsens Dummy Cell WE A (RedOx circuit)

### Example 3: EIS Plot Example (MSPlotEIS.py)

This example performs an EIS (Electrochemical Impedance Spectroscopy) scan on a Palmsens Dummy Cell WE C (Randles circuit) and generates a Nyquist plot and a Bode plot as shown below.

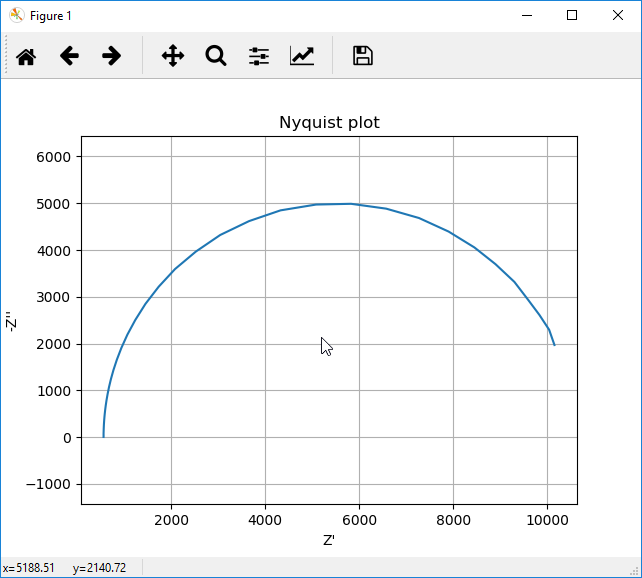
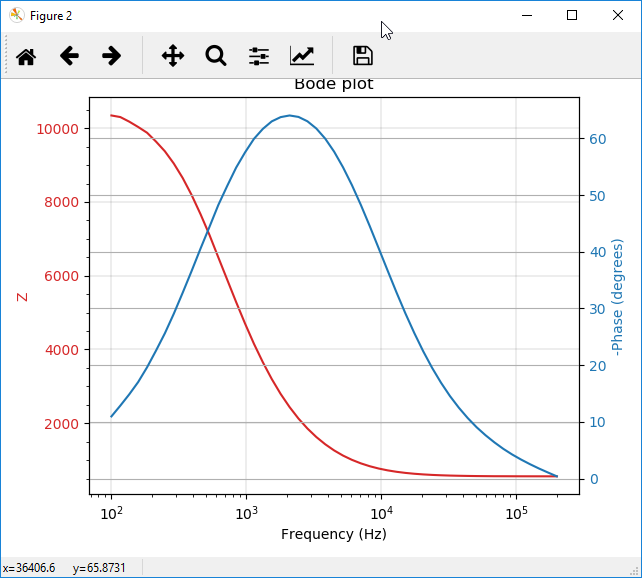
 

Figure 3: Nyquist plot Figure 4: Bode plot

## Communications

The examples have been developed using the Spyder IDE as part of the Anaconda distribution.

### Connecting to the device

The examples use the serial library for serial communication with the device.

### Sending a MethodSCRIPT

The MethodSCRIPT can be read from a txt file stored in the *MScriptFile* and then sent to the device. In this example the MethodSCRIPT files are stored in the *MethodSCRIPT files* directory.

### Receiving the measurement packages

Once the script file is sent to the device, the measurement packages can be read continuously from the device as shown below.

response = ser.readline()

### Parsing the response

Each measurement data package returned by the method *ser.readline()*, can be parsed further to obitain the actual data valuesHere’s a set of data packages received from a Linear Sweep Voltammetry (LSV) measurement on a dummy cell with 10 kOhm resistance.

eM0000\n

Pda7F85F3Fu;ba48D503Dp,10,288\n

Pda7F9234Bu;ba4E2C324p,10,288\n

Pda806EC24u;baAE16C6Dp,10,288\n

Pda807B031u;baB360495p,10,288\n

\*\n

\n

While parsing a measurement package, various identifiers are used to identify the type of package. For example, In the above sample,

1. ‘e’ is the confirmation of the “execute MethodSCRIPT” command.
2. ‘M’ marks the beginning of a measurement loop.
3. ‘P’ marks the beginning of a measurement data package.
4. “\*\n” marks the end of a measurement loop.
5. “\n” marks the end of the MethodSCRIPT.

The data values to be received from a measurement can be sent through ‘pck*’* commands in the MethodSCRIPT. Most techniques return the data values Potential (set cell potential in V) and Current (measured current in A). These can be sent with the MethodSCRIPT.

In case of Electrochemical Impedance Spectroscopy (EIS) measurements, the following *variable types*  can be sent with the MethodSCRIPT and received as measurement data values.

* Frequency (set frequency in Hz)
* Real part of complex Impedance (measured impedance Ohm)
* Imaginary part of complex Impedance (measured impedance in Ohm)

The following metadata values can also be obtained from the data packages, if present.

* CurrentStatus (OK, Underload, Overload, Overload warning)
* CurrentRange (the current range in use)
* Noise