Method SCRIPT SDK Example





Last revision: March ? 2019

© 2019 PalmSens BV

www.palmsens.com

## Contents:

* PSEsPicoLib.py
* MSConsoleExample.py
* MSPlotCV.py
* MSPlotEIS.py
* \MethodSCRIPT files\ MSExampleCV.mscr
* \MethodSCRIPT files\ MSExampleEIS.mscr

The examples are based on Python under Windows using the Spyder IDE as part of the Anaconda distribution.

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.

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

This example opens a communication port, sends a MethodSCRIPT file, read back the data , parses the data and prints the parsed data (variable type, value, unit) to the console. The meta data (status,currentrange) is not parsed in this example.

A connected Emstat Pico will create a comport with a generated (enumerated) port number. To find this number one can look in the Windows device manager

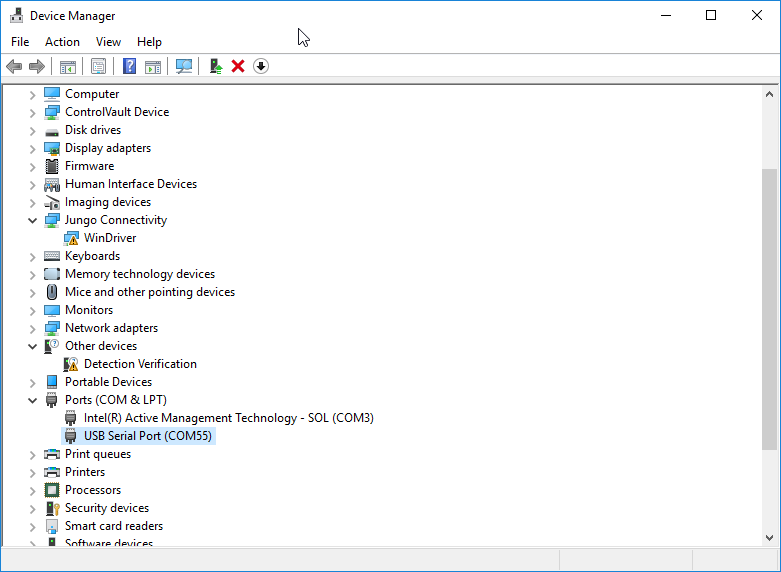


Figure 1; Available com ports in device manager

We use COM55 as our Emstat Pico comport and set the myport variable accordingly.

import serial

import os.path

import PSEsPicoLib

#script specific settings

MSfilepath = ".\\MethodSCRIPT files"

MScriptFile = "MSExampleCV.mscr"

#combine the path and filename

MScriptFile = os.path.join(MSfilepath, MScriptFile)

#initialization and open the port

ser = serial.Serial() #Create an instance of the serial object

myport = "COM55" #set the comport

if PSEsPicoLib.OpenComport(ser,myport,1): #open myport with 1 sec timeout

print("Succesfuly opened: " + ser.port )

try:

if PSEsPicoLib.IsConnected(ser): #Check if EmstatPico is connected

print("Connected!")

print(PSEsPicoLib.GetVersion(ser)) #Print the version

with open(MScriptFile) as f: #open the script file

content = f.readlines() #read all the contents

for scriptline in content: #read the content line by line

print(scriptline.strip()) #print the line to send without the linefeed (stripped)

ser.write(bytes(scriptline, 'ascii')) #send the scriptline to the device

#Fetch the data comming back from the device

while True:

response = ser.readline() #read until linefeed is read or timeout is expired

res\_line = response.decode("ascii") #decode the returned bytes to an ascii string

print("read data: " + res\_line.strip()) #print the data read without the linefeed

if res\_line.startswith('P'): #data point start with P

pck = res\_line[1:len(res\_line)] #ignore last and first character

for v in pck.split(';'): #value fields are seperated by a semicolon

str\_vt = v[0:2] #get the value-type

str\_var = v[2:2+8] #strip out value type, we ignore it for now

value = PSEsPicoLib.ParseVarString(str\_var) #Parse the value

var\_type = PSEsPicoLib.GetVarType(str\_vt) #Get the variable type

var\_unit = PSEsPicoLib.GetValueUnit(str\_vt) #Get the unit

print(var\_type + "=" +str(value) + " " + str(var\_unit) )

if (res\_line == '\n'): #Check on termination of data from the device

break #exit while loop

else:

print("Unable to connected!")

ser.close() #close the comport

except Exception as e1: #catch exception

print("error communicating...: " + str(e1)) #print the exception

else:

print("cannot open serial port ")

### Parsing the response

Each line of response returned by the function ser.readline(), can be further parsed if it is identified to be a data package. Here’s a sample response (raw data) from a Linear Sweep Voltammetric measurement.

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 the response, various identifiers are used to identify the type of response packages. For example, In the above sample response package,

1. ‘e’ marks the beginning of a response.
2. ‘M’ marks the beginning of a measurement loop.
3. ‘P’ marks the beginning of a row of data package.
4. “\*\n” marks the end of measurement.
5. “\n” marks the end of response.

The following information can be found in the data packages received from the device.

* Potential (set cell potential in V)
* Current (measured current in A)

In case of Impedance spectroscopy measurements, the following data values can be obtained from the response.

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

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

* CurrentStatus (OK, underload, overload, overload warning)
* CurrentRange (the current range in use at the moment)
* Noise (Noise)

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

This example performs a CV and plots the I vs E.

The shown plot is the result when the Palmsens Dummy Cell WE A (RedOx circuit) is used.

In the

import serial

import os.path

import PSEsPicoLib

import matplotlib.pyplot as plt

#script specific settings

MSfilepath = ".\\MethodSCRIPT files"

MScriptFile = "MSExampleCV.mscr"

#combine the path and filename

MScriptPathandFile = os.path.join(MSfilepath, MScriptFile)

#initialization and open the port

ser = serial.Serial() #Create an instance of the serial object

myport = "COM55" #set the comport

if PSEsPicoLib.OpenComport(ser,myport,1): #open myport with 1 sec timeout

print("Succesfuly opened: " + ser.port )

try:

if PSEsPicoLib.IsConnected(ser): #Check if EmstatPico is connected

print("Connected!")

# Send the MethodSCRIPT file

PSEsPicoLib.SendScriptFile(ser,MScriptPathandFile)

#Get the results and store it in datafile

datafile=PSEsPicoLib.GetResults(ser) # fetch the results

#Create "data" subfolder

(prefix, sep, suffix) = MScriptFile.rpartition('.') #split the file-extension and the filename

ResultFile = prefix + '.dat' #change the extension to .dat

ResultPath = MSfilepath+"\\data" #use subfolder for the data

try:

os.mkdir(ResultPath)

except OSError:

print ("Creation of the directory %s failed" % ResultPath)

else:

print ("Successfully created the directory %s " % ResultPath)

ResultFile = os.path.join(ResultPath, ResultFile) #combine the path and the filename

ResultFile = PSEsPicoLib.CheckFileExistAndRename(ResultFile) #Rename the file if it exists to a unique name by add the date+time

#print(ResultFile)

f = open(ResultFile,"w+") #Open file for writing

f.write(datafile) #write data to file

f.close() #close file

else:

print("Unable to connected!")

ser.close() #close the comport

except Exception as e1: #catch exception

print("error communicating...: " + str(e1)) #print the exception

else:

print("cannot open serial port ")

value\_matrix = PSEsPicoLib.ParseResultFile(ResultFile) #Parse result file to Value matrix

applied\_potential=PSEsPicoLib.GetColumnFromMatrix(value\_matrix,0) #Get the applied potentials

measured\_current=PSEsPicoLib.GetColumnFromMatrix(value\_matrix,1) #Get the measured current

plt.figure(1)

plt.plot(applied\_potential,measured\_current)

plt.title("Voltammogram")

plt.xlabel("Applied Potential (V)")

plt.ylabel("Measured Current (A)")

plt.show()

plt.grid(b=True, which='major')

plt.grid(b=True, which='minor', color='b', linestyle='-', alpha=0.2)

plt.minorticks\_on()

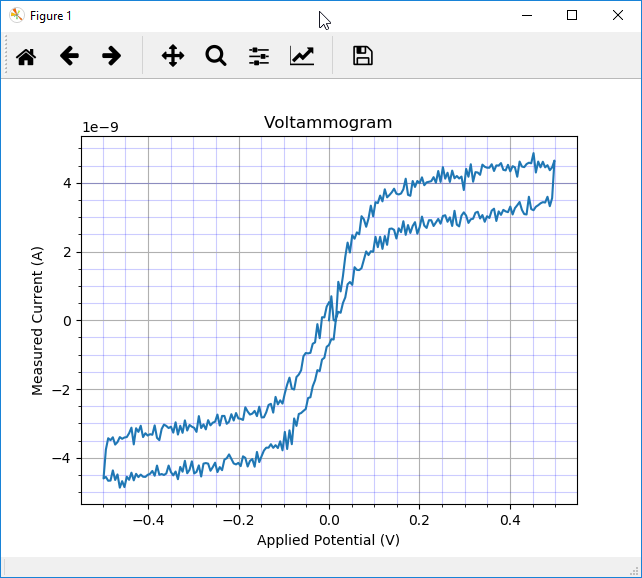


Figure 2; CV on Palmsens Dummy Cell WE A (RedOx circuit)

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

This example performs a EIS scan and generated a Nyquist plot and a Bode plot .

The shown plots are the result when the Palmsens Dummy Cell WE C (Randles circuit) is used.

Part-1 of the code connects to Emstat Pico and sends the MethodSCRIPT file and saves the results to the filename given by ResultFile using the functions from PSEsPicoLib.py.

Part-2 parses the data stored in ResultFile to a matrix given by value\_matrix where the first column (0) holds the applied frequencies, the second (1) the real part of the complex impedance and the third (2) the imaginary part of the complex impedance. The complex impedance is composed from the real and imaginary parts and the absolute impedance (Z) and phase are calculated from the complex impedance: Zcomplex . Nyquist plot and the Bode plot are generated accordingly. At last the results are saved to a Comma Separated File enabling the results to be imported in other applications like Excel.

#### Part-1

import serial

import os.path

import PSEsPicoLib

import matplotlib.pyplot as plt

import numpy as np

#script specific settings

MSfilepath = ".\\MethodSCRIPT files"

MScriptFile = "MSExampleEIS.mscr"

#combine the path and filename

MScriptPathandFile = os.path.join(MSfilepath, MScriptFile)

#initialization and open the port

ser = serial.Serial() #Create an instance of the serial object

myport = "COM55" #set the comport

if PSEsPicoLib.OpenComport(ser,myport,1): #open myport with 1 sec timeout

print("Succesfuly opened: " + ser.port )

try:

if PSEsPicoLib.IsConnected(ser): #Check if EmstatPico is connected

print("Connected!")

# Send the MethodSCRIPT file

PSEsPicoLib.SendScriptFile(ser,MScriptPathandFile)

#Get the results and store it in datafile

datafile=PSEsPicoLib.GetResults(ser) # fetch the results

#Create "data" subfolder

(prefix, sep, suffix) = MScriptFile.rpartition('.') #split the file-extension and the filename

ResultFile = prefix + '.dat' #change the extension to .dat

ResultPath = MSfilepath+"\\data" #use subfolder for the data

try:

os.mkdir(ResultPath)

except OSError:

print ("Creation of the directory %s failed" % ResultPath)

else:

print ("Successfully created the directory %s " % ResultPath)

ResultFile = os.path.join(ResultPath, ResultFile) #combine the path and the filename

ResultFile = PSEsPicoLib.CheckFileExistAndRename(ResultFile) #Rename the file if it exists to a unique name by add the date+time

#print(ResultFile)

f = open(ResultFile,"w+") #Open file for writing

f.write(datafile) #write data to file

f.close() #close file

else:

print("Unable to connected!")

ser.close() #close the comport

except Exception as e1: #catch exception

print("error communicating...: " + str(e1)) #print the exception

else:

print("cannot open serial port ")

#### Part-2

value\_matrix = PSEsPicoLib.ParseResultFile(ResultFile) #Parse result file to Value matrix

applied\_frequency=PSEsPicoLib.GetColumnFromMatrix(value\_matrix,0) #Get the applied frequencies

measured\_zreal=PSEsPicoLib.GetColumnFromMatrix(value\_matrix,1) #Get the measured real part of the complex impedance

measured\_zimag=PSEsPicoLib.GetColumnFromMatrix(value\_matrix,2) #Get the measured imaginary part of the complex impedance

#Calculate Z and Phase

measured\_zimag = -measured\_zimag #invert the imaginary part for the electrochemist convention

Zcomplex= measured\_zreal + 1j\*measured\_zimag #compose the complex impedance

Zphase=np.angle(Zcomplex, deg=True) #Get the phase from the complex impedance in degrees

Z=np.abs(Zcomplex) #Get the impedance value

#show the Nyquist plot as figure 1

plt.figure(1)

plt.plot(measured\_zreal,measured\_zimag)

plt.title('Nyquist plot')

plt.axis('equal')

plt.grid()

plt.xlabel("Z\'")

plt.ylabel("-Z\'\'")

#show the Bode plot as dual y-axis

fig, ax1 = plt.subplots()

color = 'tab:red' #plot the impedance in Red

ax1.set\_xlabel('Frequency (Hz)') #X-axes is Frequency

ax1.set\_ylabel('Z', color=color) #axes-1 is Z (impedance)

ax1.semilogx(applied\_frequency, Z, color=color) #X-axis is logarithmic

ax1.tick\_params(axis='y', labelcolor=color) #show ticks

## Turn on the minor TICKS, which are required for the minor GRID

ax1.minorticks\_on()

# Customize the major grid

ax1.grid(which='major', linestyle='-', linewidth='0.1', color='black')

ax2 = ax1.twinx() # instantiate a second axes that shares the same x-axis

color = 'tab:blue'

ax2.set\_ylabel("-Phase (degrees)", color=color) # we already handled the x-label with ax1

ax2.semilogx(applied\_frequency, Zphase, color=color)

ax2.tick\_params(axis='y', labelcolor=color)

fig.tight\_layout() # otherwise the right y-label is slightly clipped

plt.grid(True,which="both")

plt.title('Bode plot')

plt.show()

#Save results as comma seperated values (.csv) file

(prefix, sep, suffix) = ResultFile.rpartition('.')

CSVFile = prefix + '.csv'

print(CSVFile)

np.savetxt(CSVFile,np.transpose([applied\_frequency,measured\_zreal,measured\_zimag,Z,Zphase]) , delimiter=',')

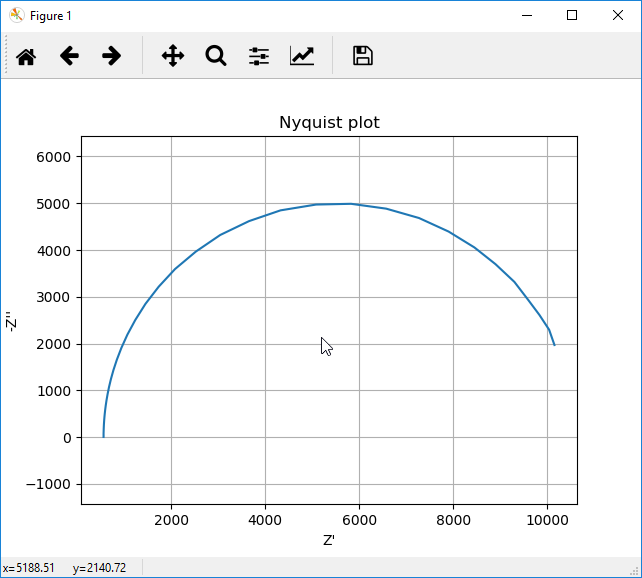


Figure 3; Nyquist plot of Palmsens Dummy Cel WE C (Randles circuit)

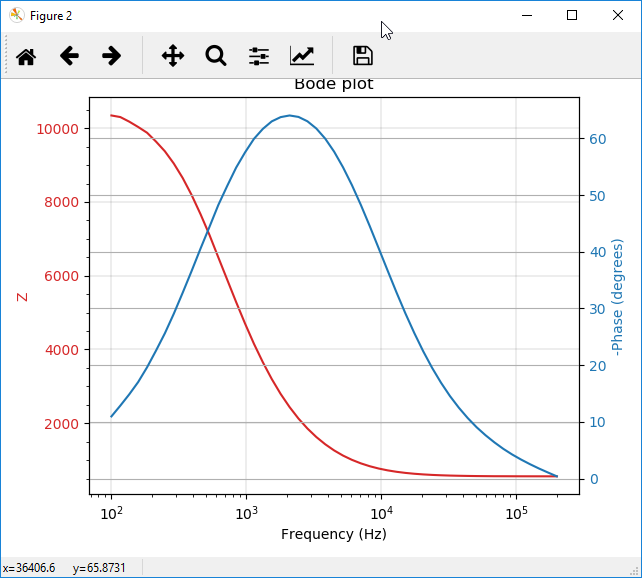


Figure 4;Bode plot of Palmsens Dummy Cel WE C (Randles circuit)