# Script

## An introduction

## (For Beginners)

ZAHNER
SCIENTIFIC INSTRUMENTS

01/2021

# 1 Introduction

In research/industries, simple electrochemical experiments are carried out multiple times to ensure the quality and reproducibility of a system/product under investigation (i.e., testing of a battery). Using our script program, a simple script can be written, which runs the desired experiments (EIS, potential/current experiments etc) single or multiple times in a predefined order and simultaneously save the measured data in a designated folder. Scripts can also be used to run the experiments when predefined conditions (i.e., current stability, trigger) are met.

The sample scripts from *Section 5* of this manual are also provided in the **c:\thales\script\myscript** folder. The user is welcome to use these scripts as they are or modify them according to his or her requirements. After reading this script manual and working on the sample scripts, the users will be able to write his/her scripts.

This manual will serve as a starting point for users to familiarize themselves with the Zahner script software. For writing the more complex scripts, please read our **Script** and **andiBASIC** manuals. For further assistance contact us at ***contact@zahner.de***.

# 2 Starting script program

For starting a script program, turn on your Thales software. In the classic view of Thales software, a **"Script"** icon is provided, click on this icon to open the script sequencer window.

A "Script Sequencer" window contains different buttons, details of which are given below

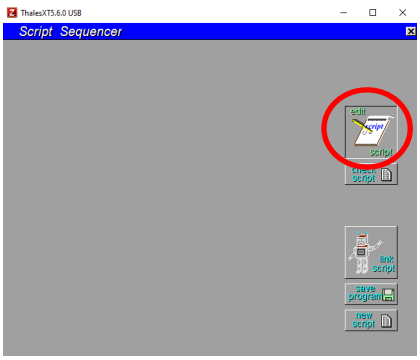
***edit script:*** to open a script window (ZEDIT USB)

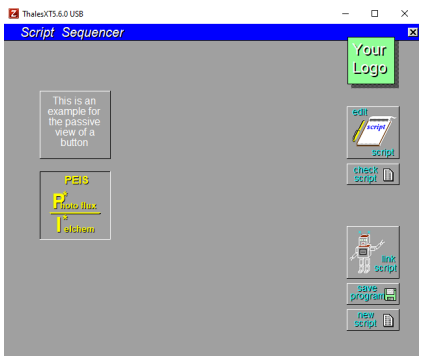
***check script:*** to check for the changes in the script

***link script:*** to link the script to the script software

***save program:*** to save the script

***new script:*** to start a new script program

By clicking on the *edit script*, a new window (***ZEDIT USB***) will open. The user can write his or her script in this window.

Once a script is written, then click on "*link script"*. Now new buttons will appear on the "Script Sequencer" window which will be used to execute the script.

When any change is made in the script, the *"link script"* has to be clicked so that the software can recognize these changes and implement them in the next measurement.

For two scripts in a ZEDIT USB window (explained below), the second button (with yellow text) will only appear when the cursor hovers over the button.

When a script is run, then the progress of the script can be viewed in the Z-Trace window. Z Trace window can be open via clicking on the small Z on the top left part of the Thales software.

# 3 Writing a script

## 3.1 Color scheme

In the ***ZEDIT USB*** window, a script can be written with different kinds of commands, variables, constant and remarks, etc. To easily differentiate between different parts of the script and to make reading the script easy, a color scheme is used. Details of this color scheme are given below

| | |
|---|---|
| `Macro commands` | : red + bold (i.e. start/end script, open/save measurement file etc.) |
| `SCRIPT system variables` | : black + bold (setting potentiostatic or galvanostatic mode etc.) |
| `andiBASIC commands` | : pink (loop functions, condition function – `if, goto`-, etc.) |
| `Software remarks` | : green (comments on script-not processed by the processor) |
| `String constants` | : blue (file or variable names, folder paths etc.) |
| `Data` | : cyan |

If a script is opened via the *edit script* icon then the **Z EDIT USB** window will open. A script can be written, modified, and saved from this format. Only the script opened in the **Z EDIT USB** window will be able to link with the script software.

When a saved script is opened directly from a folder then it will open in the **THALES EDITOR** window. Here one can modify the script and save it. However, linking with the software is not possible. For linking the script, it must be opened in the **Z EDIT USB** window via clicking on the *edit script* button in Zahner script program

## 3.2 Basic structure

For every script, a basic structure is needed. In the ***ZEDIT USB*** window, the script is written within this basic structure. This basic structure is shown below

```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1
.
.
.
SCRIPT_END
```

First, a path is provided where the script file is saved. We suggest making a "myscript" folder in your script folder in *c:\thales\script*. After providing the path, the `SCRIPT1` command is provided to start the script and `SCRIPT_END` to end the script. All the script code will be written between these two commands.

In a single script file, up to 9 scripts can be written and applied to the system under investigation. The basic structure with 9 scripts is shown below

```
SCRIPT_PATH$="c:\thales\script\myscript" 'folder path for script
SCRIPT1    'start script 1
.
.          'script body
.
SCRIPT_END 'end script 1
SCRIPT2
.
.
.
SCRIPT_END
SCRIPT3
.
.
.
SCRIPT_END
.
.
.
.
.
.
.
.
SCRIPT8
.
.
.
SCRIPT_END
SCRIPT9
.
.
.
SCRIPT_END
```

Comments in the scripts can be written after an apostrophe (’), this part is not read by the script software. An example of such comments is given above

To properly run a script, few additional buttons are required. To access these buttons, copy the *scr_logo.icd*, and the *scr_b1.icd* logo from ***C:\Thales\Script*** to the folder where the script is saved.

The **myscript** folder provided with this manual contains the sample scripts, required logo-files and rule files. Please save this **myscript** folder in **C:\Thales\Script** for the sample scripts to properly work.

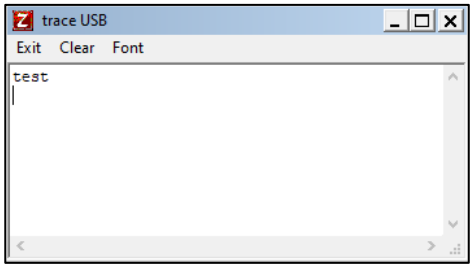
The script is saved in the **AMOS textfiles (*.is_)**.

## 3.3 Z-Trace USB window

Z Trace USB window shows us the results of a simple script and the progress of the script during operation. In the example given below, a very simple script is written and the result of this script is shown in the Z Trace USB window.

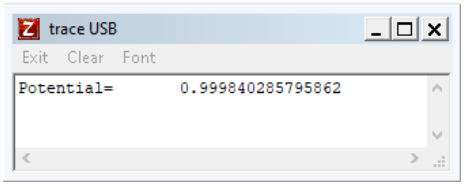
```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1     'start script
lprint"test"     'print test
SCRIPT_END 'end script
```

SCRIPT_PATH$ sets the path where the script files (buttons, graphics, etc.) are present.




## 3.4 Turn potentiostat on/off

Here, the potentiostat is turned on and a potential of 1 V is applied. The potential is measured and shown in the ***Z trace USB*** window. At the end of the script, the potentiostat is turned off.




```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1          'start script
GETPARAM   'get current configuration of potentiostat
Gal=0
GAL=0                 'turn on potentiostatic mode (Gal=0 and GAL=0)
Pset=1                     'set potential to 1 V
Pot=-1                     'turn potentiostat ON
SETUPECW              'apply the settings above
POTENTIAL             'measure potential
lprint"Potential=",Pact     'print potential value
Pot=0                 'turn potentiostat OFF
SETUPECW              'apply the settings above
SCRIPT_END                 'end script
```

If the potentiostat has to be turned on before loading a rule-file (explained later) then it is crucial to use the GETPARAM command in combination with the SETUPECW command. Otherwise, ECW hardware may load invalid parameters and exhibit an erroneous behavior.

Once the GETPARAM command is used in the script with the SETUPECW command then for the rest of the script, a new GETPARAM command is not required and only the SETUPECW command can be used.

A two-line code is necessary to choose a potentiostatic, galvanostatic, or pseudo-galvanostatic mode.

| | GAL=1 | GAL=0 | GAL=-1 |
|---|---|---|---|
| Gal=0 | - | Potentiostatic mode | Pseudo-galvanostatic |
| Gal=-1 | Galvanostatic mode | | |

In the above-provided script, **Gal=0** and **GAL=0** are used to turn the potentiostat on.

By clicking on commands (i.e. GETPARAM) with the right button of the mouse, the **Script** and **andiBASIC** manuals can be open to the page where that respective command is explained. This information is very handy for beginners as they have to check the meaning of different commands frequently.

## 3.5 Choosing different channels

Basic Zennium series potentiostats can be extended with the power potentiostats, electronic loads, and multiplexers. These additional devices are connected via EPC42 cards. To access these devices following script code is used

```
DEV% = 1 'Choose channel 1 of EPC42
SETUPECW 'apply the settings above
```

A Zennium series potentiostat can hold 4 EPC42 cards, and each card has 4 channels so to choose different channels, the DEV% =1,2,3 ….. 16 are used. In the case of RMux or PMux, the first 16 channels are assigned to RMux or PMux and the EPC42 channels start from DEV% =17 onwards.

To access the main Zennium series potentiostat, use the following script code.

```
DEV% = 0 'Channel 0 is always assigned to the main Zennium series potentiostat
SETUPECW 'apply the settings above
```

## 3.6 Loop function

### 3.6.1 Return-until

The scripts provided in section 3.4 can be easily modified with a simple `repeat - until` loop command. This will allow the electrochemical workstation (ECW) to keep measuring the potential until a condition is met. A simple trigger is also recommended to stop the loop prematurely or if the code results in a never-ending loop. In the script below, the potentiostat will keep on measuring the applied potential of 0.1 V every 500 ms until the key "q" is pressed on the keyboard.

```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1                          'start script
      GETPARAM                   'get current configuration of potentiostat
      Gal=0
      GAL=0                           'turn on potentiostatic mode
      Pset=0.1                   'set potential to 0.1 V
      Pot=-1                          'turn potentiostat ON
      SETUPECW                   'apply the settings above
            repeat                    'start loop
            POTENTIAL                 'measure potential
            lprint"Potential=",Pact   'print potential value
            pause 500                 'wait 500 milli-seconds (ms)
            get KEY$                  'access keybord
            until KEY$="q"            'press "q" button on keyboard to stop loop
      Pot=0                           'turn potentiostat OFF
      SETUPECW                   'apply the settings above
SCRIPT_END                       'end script
```

### 3.6.2 For-to [step], next

Besides `repeat - until` loop, `for to (step) – next` command is also used for the loop function. In the previous example (`repeat - until`), the potentiostat will keep on measuring potential until the loop is stopped by pressing "q". With `for to (step) – next` loop, the potentiostat will measure the potential for predefined number and then will exit the loop.

In this script on the following page, the potential will be measured 100 times, every 500 ms. To stop the loop prematurely a new function `if - goto` is introduced. Here pressing, the "q" will force the script software to go to the `LOOPEND::` code.

During script, `POTENTIAL` or `CURRENT` measurements take around 30 ms hence with the script, it is not possible to measure the potential/current faster than the time interval of 30 ms. Therefore, it is advised that for accurate measurements don't run the loops with a time interval of less than 100ms.

```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1                         'start script
      GETPARAM                  'get current configuration of potentiostat
      Gal=0
      GAL=0                           'turn on potentiostatic mode
      Pset=0.1                  'set potential to 0.1 V
      Pot=-1                          'turn potentiostat ON
      SETUPECW                  'apply the settings above
            for t=1 to 100 step 1   'start loop
            POTENTIAL               'measure potential
            lprint"Potential=",Pact 'print potential value
            pause 500               'wait 500 milli-seconds (ms)
            get KEY$                'access keyboard
            if KEY$="q" goto LOOPEND:    'press "q" button on keyboard to stop loop
            next                    'next loop
          LOOPEND::
      Pot=0                           'turn potentiostat OFF
      SETUPECW                  'apply the settings above
SCRIPT_END
```

If due to erroneous script, the potentiostat enters a never-ending loop, then the user can press the USB knob (on the front panel of the Zennium series potentiostats) downwards to *nmi*. This will force the potentiostat to stop the ongoing measurement. If you interrupt a measurement loop then pay attention to the potentiostat because depending upon the measurement, the potentiostat will stay on and will not be turned off with nmi. If the potentiostat is ON then the **POT** LED at the front panel of the Zennium series potentiostat will be **orange** and if the potentiostat is off then the LED will be **green**.

## 3.7 Subroutines

The subroutines are a handy tool in script coding which allows the use of a single function multiple times in a script. For example, if in a script the user wants to turn on and/or turn off the potentiostat multiple times between different measurements. Then he or she can create subroutines for turning on or turning off the potentiostat and use those despite witing the whole script. Subroutines are usually written at the end of the script. A `gosub` command is used for the subroutines. The script provided in the Section 3.6.2 is modified using the subroutines

```
SCRIPT_PATH$="c:\thales\script\myscript"
SCRIPT1 'start script
gosubPOT_ON:    'go to subroutine 1- potentiostat ON
       for t=1 to 100 step 1    'start loop
       POTENTIAL           'measure potential
       lprint"Potential=",Pact   'print potential value
       pause 500           'wait 500 milli-seconds (ms)
       get KEY$            'access keybord
       if KEY$="q" goto LOOPEND:     'press "q" button on keyboard to stop loop
       next                      'next loop
     LOOPEND::
gosubPOT_OFF:              'go to subroutine 2- potentiostat OFF
gotoMENU                   'go to the main menu of Script Sequencer

       POT_ON::            'Subroutine 1 - potentiostat ON
       GETPARAM            'get current configuration of potentiostat
       Gal=0
       GAL=0                     'turn on potentiostatic mode
       Pset=0.1            'set potential to 0.1 V
       Pot=-1                    'turn potentiostat ON
       SETUPECW            'apply the settings above
       return                    'return to script


     POT_OFF::             'Subroutine 1 - potentiostat OFF
       Pot=0                     'turn potentiostat OFF
       SETUPECW            'apply the settings above
       return              'return to scriptine 1 - potentiostat ON
SCRIPT_END
```

In script make sure not to use **a space**, or a **hyphen (-)** in a string or a variable. Hyphen (-) is used as a mathematical operator **(minus)**.

# 4 Electrochemical measurements

After learning the basics of script coding, the next step is the coding for the standard measurement methods using script software.

1. Electrochemical impedance spectroscopy (EIS)
2. Cyclic voltammetry (CV)
3. Current-voltage curve (IE)
4. Polarization curve (POL)

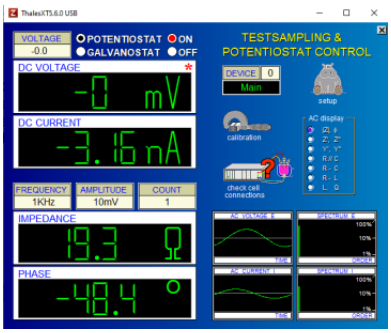
## 4.1 Rule file

For complex measurements (i.e. EIS), a script that includes all the required parameters have to be written. These parameters include a potentiostatic/galvanostatic mode, voltage amplitude, lower frequency limit, upper-frequency limit, applied bias voltage and step-per-decade for different frequency ranges, etc. Similar is true for other measurement processes, that for each measurement, one has to provide information about each parameter/variable. To simplify this process, a rule-file is used which contains all the parameters, needed for the measurement. This way the user does not have to code for these parameters, making the script code for these measurements very simple. For generating a rule file, a process provided below is followed.

For a rule file (i.e., for EIS measurement), follow the procedure provided below

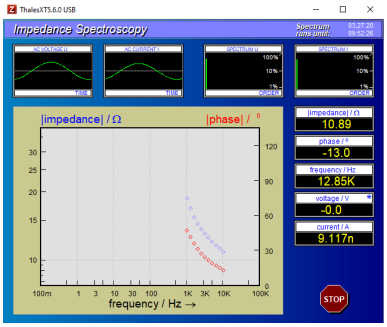

1. Select EIS from Thales software


2. Input desired EIS parameters & click on “control potentiostat”

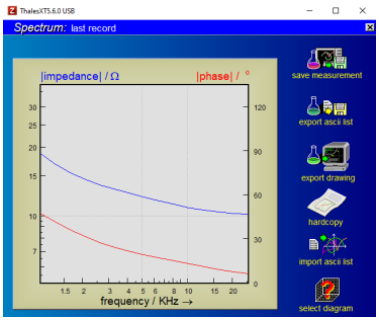



3. Input desired amplitude and bias voltage etc. Click “Esc”

Clicking “Esc” will bring you back to the 2nd window and then click “start recording”.




4. Measure at least 5 data points and then click “stop”




5. Save measurement as “EIS_Rule_file” in folder c:\thales\script\myscript

Now, this saved file has all the required parameters saved in it and can be used as a base file (**rule-file**) for the EIS measurement(s) in the script. In script code, a "path-command" to the rule-file is necessary, from where the script program will take the rule file for measurement. For EIS measurement, the following code can be added to the script.

```
MEAS_OPEN_EIS(65,"c:\thales\script\myscript","EIS_Rule_file")'open EIS rule file
MEAS_EIS                                                    'perform EIS
MEAS_SAVE_EIS(65," c:\thales\script\myscript","EIS1")       'save EIS with name EIS1
```

**NOTE:** "65" indicates that the data location is in the computer drive.

## 4.2 Script – EIS measurement at OCP

The script in this section illustrates the measurement of an EIS spectrum for the system under investigation at the open circuit potential (OCP).

```
SCRIPT_PATH$=,"c:\thales\script\myscript" 'script path
SCRIPT1                                  'start script
MEAS_OPEN_EIS(65,"c:\thales\script\myscript","EIS_Rule_file")'open EIS rule file
MEAS_EIS                                 'perform EIS at OCP
MEAS_SAVE_EIS(65," c:\thales\script\myscript","EIS1")'save EIS1 (part1)
SCRIPT_END                                   'end script
```

Here,

```
GETPARAM
Gal=0
GAL=0                                    'turn on potentiostatic mode
Pot=-1                                       'turn potentiostat ON
SETUPECW                                 'apply the settings above
```

is not necessary because the potentiostatic/galvanostatic mode and potentiostat turn on/off information is also provided in the saved rule file. Since in the rule file, the EIS is measured with potentiostat turned off (at OCP) hence

```
Pot=0                                    'turn potentiostat OFF
SETUPECW                                 'apply the settings above
```

is also not necessary. However, if EIS is measured at some bias potential then after the EIS coding, code for turning off the potentiostat is necessary.

In the sample scripts, the `GETPARAM` commands are always provided even if the rule file is opened at the start of the script. This is done as a precaution that upon modifying the scripts, the user does not apply any potential or current at the start of the script without `GETPARAM`.

Similarly, for CV, POL, and IE, a rule file can be saved, and then a script can be written for the before-mentioned measurements. Script code for these measurements is provided below

**CV:**

```
MEAS_OPEN_CV(65,"c:\thales\script\myscript","CV_Rule_file")'open CV rule file
MEAS_CV                                          'perform CV
MEAS_SAVE_CV(65," c:\thales\script\myscript","CV1")      'save CV with name CV1
```

**POL:**

```
MEAS_OPEN_POL(65,"c:\thales\script\myscript","POL_Rule_file")'open POL rule file
MEAS_POL                                          'perform POL
MEAS_SAVE_POL(65," c:\thales\script\myscript","POL1")      'save POL with name POL1
```

**IE:**

```
MEAS_OPEN_IE(65,"c:\thales\script\myscript","IE_Rule_file") 'open IE rule file
MEAS_IE                                          'perform IE
MEAS_SAVE_IE(65," c:\thales\script\myscript","IE1")         'save IE with name IE1
```

## 4.3 ASCII export

Script program saves the measurements in the special formats (i.e., ***.ism** format for EIS measurement). If an ASCII export is required then the measured spectrum is opened in SIM and then exported into ASCII format.

```
CLRSIM                                          'initialize SIM
ANA_OPEN_EIS(65,"c:\thales\script\myscript","EIS1")  'open saved EIS1 file
ANA_XASCII(65,"c:\thales\script\myscript","EIS1")    'save EIS ASCII with name EIS1
```

The script provided in *Section 4.2* is extended for the ASCII export

```
SCRIPT_PATH$=,"c:\thales\script\myscript" 'script path
SCRIPT1                                 'start script
MEAS_OPEN_EIS(65,"c:\thales\script\myscript","EIS_Rule_file")'open EIS rule file
MEAS_EIS                                'perform EIS at OCP
MEAS_SAVE_EIS(65," c:\thales\script\myscript","EIS1")'save EIS1 (part1)
CLRSIM                                          'initialize SIM
ANA_OPEN_EIS(65,"c:\thales\script\myscript","EIS1")  'open EIS1 (saved in part1)
ANA_XASCII(65,"c:\thales\script\myscript","EIS1")    'save EIS ASCII with name EIS1
SCRIPT_END                                    'end script
```

Similarly for CV and IE, an ASCII export can be made using the code below.

**For CV:**

```
CLRSIM                                          'initialize SIM
ANA_OPEN_CV(65,"c:\thales\script\myscript","CV1")    'open saved CV1 file
ANA_PLOT_CV(1)                                'plot CV graph
ANA_XASCII(65,"c:\thales\script\myscript","CV1")    'save CV ASCII with name CV1
```

An ASCII export is not possible with “POL measurement”.

**For IE:**

```
CLRSIM                                          'initialize SIM
ANA_OPEN_IE(65,"c:\thales\script\myscript","IE1")    'open saved IE1 file
ANA_PLOT_IE(1)                                  'plot IE graph
ANA_XASCII(65,"c:\thales\script\myscript","IE1")     'save IE ASCII with name IE1
```

`ANA_PLOT_XX(1)` is required for the ASCII export of the CV and IE measurements.

For an ASCII export, first saving the file (`MEAS_SAVE_XXX`) is required. A direct ASCII export after measurement is not possible.

# 5 Sample scripts

In this part of the manual, different sample scripts are provided. These scripts deal with the standard tests and can be used as provided or modified to the user's desires. These scripts should provide a starting point for the users to start using the script software. The sample scripts provided below in the rest of this manual are also provided with required rule files and logo files in the folder **c:\thales\scripts\myscripts**.

## 5.1 Loop function

When a script is required to run multiple measurements until a condition is met then loop commands **for—to step**, **next,** and **repeat — until** come handy. The script provided below shows a script code with **for—to step**, **next**. In the script provided below

1. the potentiostat is turned on and a potential of -0.5 V is applied
2. Potential and current is measured and printed
3. In the loop, the potential is changed from -0.5 V to 0.5 V with a 0.1 V step
4. Potential and current is measured in each cycle

```
SCRIPT1
GETPARAM              'get current configuration of potentiostat
Gal=0
GAL=0                 'turn potentiostat on
Pset=-0.5             'set potential to -0.5 V
Pot=-1                     'turn potentiostat ON
SETUPECW              'apply setting provided above
POTENTIAL             'measure potential
CURRENT               'measure current
lprint("POT"),Pact         'print potential (in Z trace USB window)
lprint("CUR"),Cact         'print current (in Z trace USB window)
for Psetpoint= -0.5 to 0.5 step 0.1  'run loop for set potential from -0.5 to 0.5 V
Pset=Psetpoint        'link potentiostat potential to Psetpoint
SETUPECW              'apply setting provided above
POTENTIAL             'measure potential
CURRENT               'measure current
lprint("POT"),Pact,("CUR"),Cact 'print potential and current
get KEY$              'read keyboard
if KEY$="q" goto LOOPEND: 'press q key to end the loop prematurely
next                  'start the loop again
LOOPEND::             'loop end command
Pot=0                 'turn potentiostat OFF
SETUPECW              'apply setting provided above
gotoMENU              'go to the main menu of the Script software
SCRIPT_END
```

## 5.2 Series measurements

In the folder **c:\thales\scripts\myscripts\EIS-CV-POL-measurements**, a script is provided with the rule files. This script deals with the different measurement methods looped in a cycle and the whole cycle is repeated 3 times producing a total of 9 files in the "measured-spectra" folder.

Initially, 3 rule files are produced for POL, CV, and EIS which contain the desired parameters for the intended measurements. The parameters for each measurement are listed below

**POL:** Potential: 10mV, time: 60 seconds, sampling interval: 5 seconds

**CV:** Potential range: 0 to 0.5 V, Scan rate: 100 mV/s, Samples: 1000 samples/cycle, Measure 10 cycles

**EIS:** Potentiostatic mode, DC bias: @ OCP, Frequency range: 10 Hz to 10 kHz, Amplitude: 5mV

```
SCRIPT_PATH$="path" 'path to script
SCRIPT1
     GETPARAM ' get current configuration of potentiostat
     Gal=0 ' potentiostatic mode
     GAL=0 ' potentiostatic mode
     Pot=-1  'switch potentiostat i.e. cell state = on
     SETUPECW ' apply the settings above
     forNumber=1 to 3  'Run 3 loops
     MEAS_OPEN_POL(65,"path_rule_file","pol")'open pol rule file
     MEAS_POL        'measure polarization curve
      pack pol_NAME$, using"pol-data@@",Number 'name code, for giving different name
      to the files which  will be created due to loop
     MEAS_SAVE_POL(65,"@path_save_folder",pol_NAME$)'save polarization curve

      MEAS_OPEN_CV(65," path_rule_file ","cv")'open CV rule file
     MEAS_CV              'measure CV
     pack cv_NAME$, using"cv-data@@",Number 'name code, for giving different name  to
the files which will be created due to loop
     MEAS_SAVE_CV(65,"@path_save_folder",cv_NAME$)        'save CV

      MEAS_OPEN_EIS(65,"path_rule_file","eis")'open EIS rule file
     MEAS_EIS                              ' measure EIS
     pack eis_NAME$, using"eis-data@@",Number 'name code, for giving different name to
the files which will be created due to loop
     MEAS_SAVE_EIS(65,"@path_save_folder",eis_NAME$) 'save EIS
     Pot=0      'turn potentiostat off
     SETUPECW   'apply settings provided above
     next       'start next loop
     gotoMENU   'go to script menu
SCRIPT_END
```

In a loop script, multiple files are produced. Hence it must be made sure that all the files have different names. If the name of the files will be the same then with every loop, the

previously saved files will be overwritten by the newly saved file. For this a string (i.e., *pol_NAME$*) is used in the command given below

```
pack pol_NAME$, using"pol-data@@",Number
```

Here, the **pack** command will assign the Number value of the loop to the **pol-data@@**. This will result in the file numbers as given below

For loop 01: Number =01, measured file name is **pol-data01**

For loop 02: Number =02, measured file name is **pol-data02**

For loop 03: Number =03, measured file name is **pol-data03**

Besides the rule file, some additional files are also needed for the logo and the buttons in the script program. These files must always be copied in every new folder for a new script to properly run the script program. These files are named as *scr_logo.icd*, *scr_b1.icd*, *scr_b2.icd* and *scr_b3.icd* and so on. *b1*, *b2,* and *b3…* files will produce buttons for the script 1, script 2 and script 3… respectively.

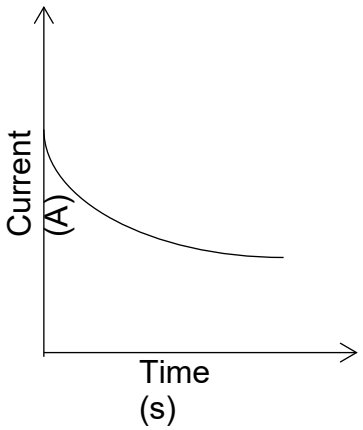
## 5.3 ASCII export

In the folder **c:\thales\scripts\myscripts\ASCII-export,** a script is provided with the rule. The script not only carryout EIS measurement and save data but also save data in ASCII format. For ASCII export, the saved spectrum (EIS) is loaded into the SIM software and then is saved as an ASCII export. An ASCII export without initializing SIM is not possible with the script program.

```
SCRIPT_PATH$="c:\thales\script\myscript\ASCII-export"
SCRIPT1
GETPARAM ' get current configuration of potentiostat
Gal=0 ' potentiostatic mode
GAL=0 ' potentiostatic mode
Pot=-1  'switch potentiostat i.e. cell state = on
SETUPECW ' apply the settings above
forNumber=1 to 3 step 1
MEAS_OPEN_EIS(65,"c:\thales\script\myscript\ASCII-export","eis")'open EIS rule file
MEAS_EIS   'measure EIS
pack eis_NAME$, using"eis-data@@",Number 'Name files with different number during loop
MEAS_SAVE_EIS(65,"@c:\thales\script\myscript\ASCII-export",eis_NAME$)'Save   EIS   file
(File 1)
CLRSIM 'Clear SIM, whatever is in SIM, clean it (initializing SIM)
ANA_OPEN_EIS(65,"c:\thales\script\myscript\ASCII-export",eis_NAME$)  'open measure EIS
file (File 1-measured recently)
ANA_XASCII(65,"@c:\thales\script\myscript\ASCII-export",eis_NAME$)'export as txt file
Pot=0      'turn potentiostat off
SETUPECW   'apply the setting provided above
next       'next loop
gotoMENU   'go to menu
SCRIPT_END
```

## 5.4 EIS after current stabilization

In many applications, the user has to wait before the system stabilizes and then carry out the EIS measurement. With the script program, the changing variable can be used as a trigger and when the changing variable meets the user-defined limit (i.e. current or voltage stabilization) then the script program carries out the measurement. A sample script for such a process is provided in the folder **C:\THALES\script\myscript\EIS-after-current-stabilization**. In the script below, current is measured every 500 ms and then the current stabilization is defined as the change in current between two consecutive current values is less than 100 pA. As this condition is met, the script software starts measuring an EIS.




```
SCRIPT_PATH$="c:\thales\script\myscript\EIS-after-current-stabilization"
SCRIPT1
      GETPARAM          'get current configuration of potentiostat
      Gal=0
      GAL=0             'potentiostatic mode
      Pot=-1            'switch potentiostat i.e. cell state = on
      SETUPECW          'apply the settings above
      CURRENT           'measure current
      lprint("current0"),Cact 'print current
      current0=Cact               'assign measured current name of "current0"
      pause 500                   'wait for 500ms
      repeat                      'start loop with repeat command
      CURRENT                     'measure current
      lprint("current"),Cact      'print new current value (step A)
      trigger=abs(Cact-current0) 'get an absolute difference between both current value,
      lprint("trigger"),trigger  'print current difference, named as "trigger"
      current0=Cact    'name current measured in step A as "current0" for next loop
      pause 500  'wait 500 ms
      get KEY$   'access keyboard
      until KEY$="q" or trigger<1e-10 'keep looping until the current difference is
            less than 100pA or key "q" is pressed
      MEAS_OPEN_EIS(65,"c:\path","eis") 'open EIS rule file
      MEAS_EIS   'measure EIS
      MEAS_SAVE_EIS(65,"@c:\path","eis_data") 'save EIS
            Pot=0 'turn potentiostat off
            SETUPECW 'apply the settings above
      gotoMENU 'go to script menu
SCRIPT_END
```

## 5.5 EIS at different bias potential

One pre-requisite of EIS is linearity. In EIS, a very small potential amplitude is used to ensure that only the linear response of the SUI is investigated. Due to this, multiple EIS measurements at different bias potentials are sometimes a necessity to understand the

behavior of the SUI. The sample script in this part measures EIS spectra at different bias potentials (from -0.5 V to 0.5 V – with a step of 0.1 V).

Here, only one rule file is required, which is measured at the parameters given below

Potential amplitude: 10 mV

Frequency range: 100 mHz to 100 kHz, starting frequency 1 kHz

Bias potential: -0.5 V

The rule file is opened before the first loop. In the first loop, the program has the parameters from the rule file and an EIS is measured. Afterward in the first loop, the bias potential is modified for the next loop whereas other EIS parameters stay the same as in the rule file. Then the next loop is measured and so on.

```
SCRIPT_PATH$="c:\thales\script\myscript\EIS-vs-Bias-potential"
SCRIPT1
GETPARAM ' get current configuration of potentiostat
Gal=0 ' potentiostatic mode
GAL=0 ' potentiostatic mode
Pset=-0.5  'set potential of -0.5 V
Pot=-1  'switch potentiostat i.e. cell state = on
SETUPECW ' apply the settings above
MEAS_OPEN_EIS(65,"c:\thales\script\myscript\EIS-vs-Bias-potential","eis-rule")    'rule
file (out of loop)
POTENTIAL  'measure potential
lprint("Bias potential"),Pact 'print potential
Number=-0.6     'Number defined for the loop
pause 500  'wait 500 ms
repeat     'start loop
Number=Number+0.1
POTENTIAL  'measure potential
lprint("New Bias potential"),Pact 'print potential with name, new bias potential
MEAS_EIS 'measure EIS
pack eis_NAME$, using"eis_data+@@@@mv",Number*1000 'file name
MEAS_SAVE_EIS(65,"@c:\thales\script\myscript\EIS-vs-Bias-potential\measured-
spectra",eis_NAME$) 'save EIS

Biaspotential=Pact 'give applied potential the name of "Bias potential"
GETPARAM
Pset=Biaspotential+0.1 'Increase the bias potential by 0.1 V
SETUPECW   'apply the settings above
pause 500  'wait 500 ms
get KEY$   'access keyboard
until KEY$="q" or Number>=0.5 'keep looping until key "q" is pressed or the number>=0.5
Pot=0 'turn off potentiostat
SETUPECW   'apply the settings above
gotoMENU   'go to menu
SCRIPT_END
```

In the folder **C:\THALES\script\myscript\EIS-vs-Bias-potential**, two scripts are provided which include a “**return-until**” loop and “**for-to (step), next**” loop. Here, In

`repeat` - `until` loop, an additional `or` function is introduced. So that loop can be stopped with one of two provided conditions.

For file names, a template of `eis_data+@@@@mv` is used. This allows the +/- sign in the name, with file numbers. The saved EIS files will have the name as

*eis_data-0500mv*

*eis_data-0400mv*

.

.

*eis_data+0400mv*

*eis_data+0500mv*

## 5.6 Potential vs time

The script shown on the next page illustrates the decay of potential with time. Here following steps are carried out

1. Select potentiostatic mode
2. Apply 1 V potential
3. Turn on potentiostat
4. Measure potential and current (and print in Z trace USB window)
5. Wait 1 second
6. Turn off potentiostat
7. Measure potential and current (and print in Z trace USB window)
8. Wait for 1 second
9. Start loop, measuring potential and current decay every second until the loop is interrupted with the “q” key
10. Save data in a text file

Measured potential is saved in a “text file” in a tabular format. For the table, a header is defined. Outmask defines the pattern in which the data will appear. Before the starting of a loop, a text file, named as test.txt is written and opened via the code

```
open1,65,2,"@c:\thales\script\myscript\potential-vs-time\test.txt,w"
```

Here, 65: PC Filesystem, 2: internal Zennium Hard disc

In folder **c:\thales\script\myscript\potential-vs-time**, two scripts file are present which contains multiple (slightly) different script codes. This is to illustrate that there are many ways to write a script for the same experiment.

```
SCRIPT1
HEADER$ ="TIME/sec     POTENTIAL/V"'header for output text file
OUTMASK$="######.#     -#.@@@@@^^^^"'code for the number format of the data file
CRLF$=hex$("0d0a") 'code for -> go to next line
      GETPARAM 'get active configuration of potentiostat
      Gal=0
      GAL=0'go to the potentiostatic mode
      Pset=1 'set potential to +1 V
      Pot=-1 'turn Potentiostat on
      SETUPECW 'apply the settings above
      POTENTIAL 'measure potential
      lprint"Pot-ON=",Pact 'print potential
      pause 1000 'wait for 1 seconds
            Pot=0 ' turn potentiostat off
            SETUPECW 'apply the setting above
      POTENTIAL 'measure potential
      lprint"Pot-OFF=",Pact 'print potential and current
      pause 1000 ' wait for 1 second
       open1,65,2,"@c:\thales\script\myscript\potential-vs-time\test.txt,w"
       'create text file for writing data, "w" is used for write able file
      print#1,tcode$ (1, HEADER$)+CRLF$
'print header, tcode$ manages font of the header, CRLF$ is used to go to next line
            for t%=0 to 1000     'loop starting (1000 cycles)
            POTENTIAL 'mesure potential
            lprint"POT=",Pact 'print POT
            pause 1000 ' wait for 1 second
            TIME=t% 'assign the TIME variable the same values at t%
            POT=Pact 'assign POT variable the value of measured potential
            packASCII$,usingOUTMASK$, TIME, POT
               'assign TIME, POT values the format defined in OUTMASK$
            print#1,ASCII$+CRLF$;'print TIME and POT value in the table
      get KEY$ 'access keyboard
            if KEY$="q" goto LOOPEND:'press key "q" to end loop prematurely
            next 'go to next loop

LOOPEND::       'loop end command
close1 'close the file test.txt
gotoMENU 'go to the main menu of the script software
SCRIPT_END
```