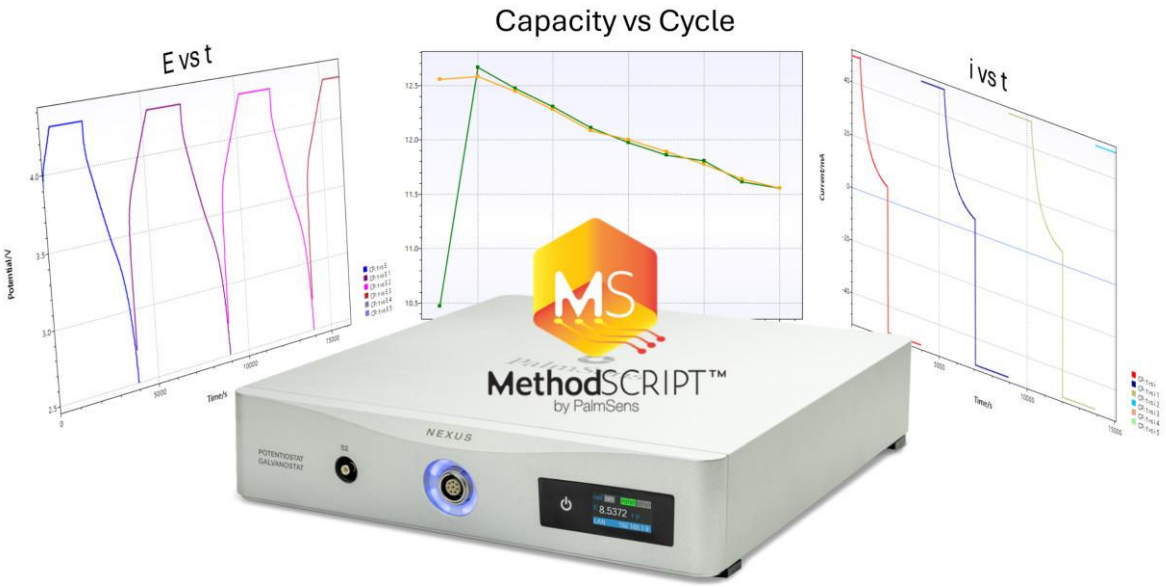
Last document update: March 2026

Application Note



1 Introduction

MethodSCRIPT™ is the communication language of our latest generation of instruments and modules. It was initially developed to facilitate custom software development.
As MethodSCRIPT enables full control over the instrument, it is a powerful tool for performing special electrochemical experiments that cannot be executed using the standard techniques available in our software (PSTrace or MultiTrace).
This application note is designed as a tutorial to use our pre-made MethodSCRIPT code for advanced battery cycling.
Therefore, it is intended for electrochemical researchers with no programming experience. If you want to start with advanced charge-discharge measurements as soon as possible, continue reading this document.
If you are looking for detailed information about MethodSCRIPT check this page, our developer portal and the MethodSCRIPT documentation.

1.1 Compatible Devices

The following potentiostat models support the MethodSCRIPT code for advanced battery cycling:

- Nexus
- EmStat4 series* (EmStat4S, EmStat4X, MultiEmStat4)

EmStat4 needs a small adjustment with the Cell ON codes. See section 2.5 for details. The version LR does not support the voltage used in this example.

The EmStat Pico and Sensit Series (Sensit Smart, Sensit BT, Sensit Wearable) support MethodSCRIPT, but their limited current and voltage aren't suitable for most battery cycling setups.

1.2 Compatible Software

This application note was written with MethodSCRIPT version 1.8 and PSTrace version 5.13. Using higher versions of MethodSCRIPT could require some changes due to commands that are deprecated code in that version.

MultiTrace supports MethodSCRIPT too, although we recommend PSTrace to generate the code due to higher convenience.

2 Setting up and Running the Experiment

2.1 Enabling MethodSCRIPT Editor

In PSTrace, the list of techniques includes one called '**MethodSCRIPT Sandbox**' which allows you to generate and run scripts. Make sure that this feature is enabled. Check the box 'Show MethodSCRIPT Editor' in the top menu *Tools* > *General Settings...* (see Figure 1). The box is not checked by default.

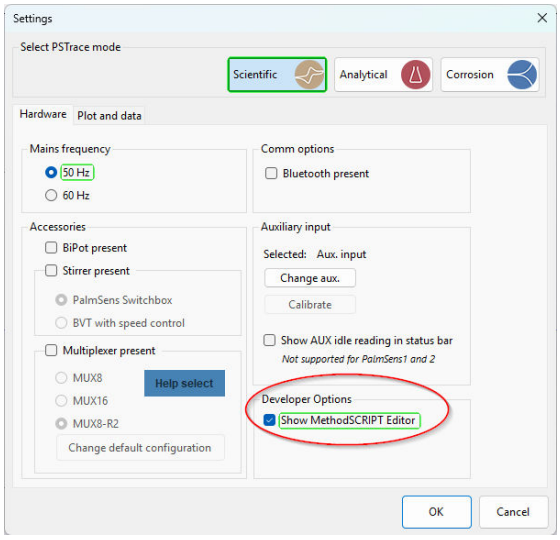


Figure 1 MethodSCRIPT option checked in General Settings

2.2 Load the Example

Go to the top menu *Method* > *Load* and choose the method 'CC-CV-CC Cycling with Delta-i-V and Qpass'. It is in the folder PSData\Battery testing examples. PSData is located in the Documents folder by default.
When a MethodSCRIPT is loaded, the MethodSCRIPT Sandbox is chosen as technique. The advanced options (…-button) will be open showing with the options for customized units (see Figure 2). Here you can assign variables type "AS" and "AT" custom units and symbols for plots in PSTrace.
For the MethodSCRIPT 'CC-CV-CC Cycling with Delta-i-V and Qpass' the passed charge plot uses a custom symbol and unit for the passed charge Qpass. This symbol and unite will be associated with any variable type AS. - The customized units will be custom labels in the resulting passed charge plot (Qpass vs Cycle).

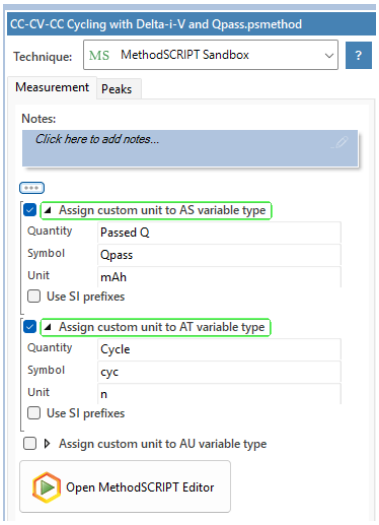


Figure 2 Battery Cycling method example loaded in PSTrace Method Editor

Instead of loading the psmethod file containing the MethodSCRIPT, you can copy a text with the referred MethodSCRIPT code into the editor of the Sandbox. However, this way the unit customization will not be loaded. However, you can customize the axes manually by filling in the *Assign custom unit…* fields.

2.3 Explore and Edit the Parameters of the CC-CV-CC

Click on 'Open MethodSCRIPT Editor' and to see the code. Scroll down to lines 52-63. These lines contain all the parameters that need to be set for this cycling experiment. See Figure 3.

```
51 store_var cycle 1i ja # defines which cycle will be the first. Sufix "i" after the number assign it as a floatin point.
52 # Set technique parameters
53 store_var potential_max 4300m ab # Maximum potential to charge to.
54 store_var current_min 5m ba # Minimum current to stop the CV charge step.
55 store_var potential_min 2500m ab # Minimum potential to discharge to.
56 store_var current_charge  100m ba # Constant current to charge with.
57 store_var current_discharge -100m ba # Constant current to discharge with.
58 store_var cycles 100i ja # Amount of charge and discharge cycles.
59 store_var interval 1000m eb # Interval time of each measurement point.
60 store_var max_time 4000 eb # Maximum duration of each step (if the cut-off is not met).
61 store_var delta_v 10m ia # Minimum potential variation criteria for plotting data during CC steps. It must be positive.
62 store_var delta_i 5m ia # Minimum current variation criteria for plotting data during CV step. It must be positive.
63 store_var delta_t 10 eb # Maximum time without plotting data.
64 # General instrument settings
```

Figure 3 MethodSCRIPT lines referring to the main experiment parameters

The code contains many comments (green) explaining the purpose of each code. Note that the MethodSCRIPT editor shows different color for text according to different functions:

- Dark yellow for variables
- Blue for codes that perform an action, create or adjust a setting or perform a logic or math operation. Example: **var** xx declares the variable xx
- Violet for numbers in general. It must be associated with an associated value, otherwise an error message will pop-up. When the value must be an integer, it is followed by an 'i'. Also, it can be followed by a metric SI suffix, i.e. -0.5 is represented as 500 and a prefix m, therefore -500m.

- Green for comments. Comments can be added to a line by inserting a # character followed by the comment. Comments do not affect the execution of the script.
- Red for string variables used for communication messages.

In its simplest form, an experiment in MethodSCRIPT consists of a *measurement loop* (meas_loop_xxx). For example, an OCP experiment with 1 s interval and 1 hour duration can be defined with the command:
meas_loop_ocp e 1 3601
In this command, the measured potential is stored in the variable e, which can later be plotted using the pck commands.
In this example, however, variables were used instead of fixed numerical values to define the measurement loop parameters. These variables are declared at the beginning of the script using the store_var command. This approach makes the script easier to modify, as the same variable can be reused in multiple measurement loops. It also improves readability by grouping all key parameters in one place.

2.3.1 Dynamic Point Intervals

Line 61, 62, and 63 define which events lead to recording of a point (see Figure 3).
During a current controlled step, a point will be recorded if the potential has changed at least by the value entered as delta_v (line 61).
During a potential controlled step, a point will be recorded if the current has changed at least by the value entered as delta_i (line 62).
After the time that is entered as delta_t (line 63) has elapsed since the last recording of a point, a point will be recorded no matter if delta_v or delta_i respectively was reached.

2.4 Run the Experiment

After setting the parameters, simply close the MethodSCRIPT Editor window and click Run.

HINT
Before the full experiment, run a short preliminary experiment to verify the setup. For example, reduce the cycle times and, if necessary, connect a dummy cell. This allows you to quickly confirm that the response behaves as expected before running the full experiment.

This experiment will generate 3 plots:

1. Voltage versus time as ‘Plot E vs t’
2. Current versus time as ‘Plot i vs t’
3. Passed charge versus cycle number as ‘Plot Qpass vs Cyc’

See result examples obtained with a real Li-ion battery in Figure 4 and Figure 5.

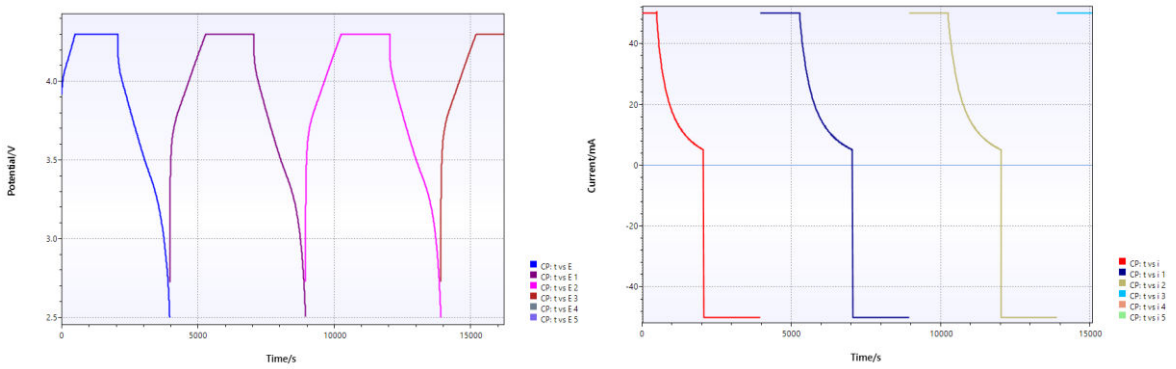


Figure 4 Plot examples, Voltage vs. Time (left) and Current vs. Time (right) for the first cycles of a real Li-ion battery. Each color corresponds to a complete charge-discharge cycle.

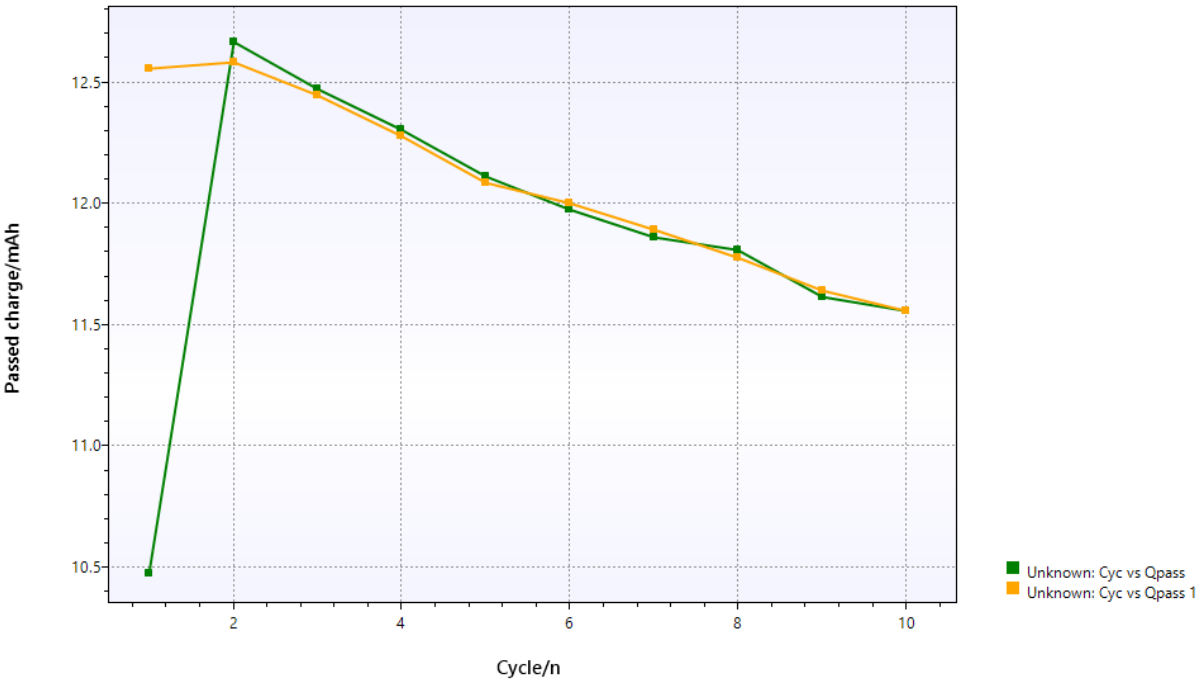


Figure 5 Plot example of Passed Charge versus Cycle number of a real Li-ion battery. Green = charged capacity, yellow = discharged capacity.

2.5 Running this Example on EmStat4 series

The provided code is fully compatible with the Nexus. For the EmStat4 series a small change is required. One command is not supported by the EmStat4 series: the use of ocp(potential) together with the cell_on command. To adapt the script for EmStat4, simply remove the ocp(potential) command and use the cell_on command without any additions.
In other words, simply replace all lines containing:

- cell_on ocp(potential)

with:

- cell_on

2.6 Further Customization

If you need to modify the code for a different application, you can edit the script accordingly. For detailed information about available commands and syntax, refer to the MethodSCRIPT manual.
When an invalid script is executed, an error message is usually displayed including the line that produces the error.
The MethodSCRIPT manual contains a list of error messages along with explanations of their causes. However, some configuration issues may not generate an error message but can still lead to unexpected behavior. Pay special attention to the topics below.

2.6.1 Potentiostat Settings

The settings listed below are automatically reset to default values whenever the potentiostat mode is changed using the set_pgstat_mode command:

- set_max_bandwidth
- set_acquisition_frac_autoadjust
- set_range_minmax
- set_autoranging
- set_range

Therefore, you must declare them again every time you change the potentiostat mode. If these parameters are not explicitly set after a set_pgstat_mode command, they will revert to factory values instead of the ones declared before. For example, the selected current range may be set to 100 µA (factory default for Nexus), rather than the previous values declared with set_range.

2.6.2 Bandwidth

PSTrace automatically calculates an appropriate bandwidth based on the selected technique, its parameters and the connected device. In this example script, we calculate the bandwidth using an equivalent formula used by PSTrace. Some techniques may require a different calculation.
If you are unsure which bandwidth to use, you can generate a MethodSCRIPT code from a PSTrace method using the same technique parameters. This provides a reliable starting point. In some cases, reducing the bandwidth can help stabilize measurements with noisy or unstable samples. However, excessively low bandwidth may result in inaccurate data.

2.6.3 Potential and Current Ranges

There are 3 different MethodSCRIPT commands associated with potential and current ranges.

- set_range_minmax
- set_autoranging
- set_range

Remember to set them accordingly, as a bad setting may lead to bad data collection due to overload or underload. If no range is selected, the potentiostat may select a default value or apply a previous setting. Refer to the MethodSCRIPT manual to see more details about the ranges syntax.

2.6.4 Concatenating Plots

In this example, the set_script_output command is used to concatenate the Constant Current (CC) and Constant Voltage (CV) charging steps into a single continuous plot. Without this command, each step would be displayed as a separate dataset, typically shown in different colors.
This behavior is related to how MethodSCRIPT transmits data and how PSTrace displays it. To concatenate datasets, data transmission is temporarily disabled using set_script_output 0 before the endloop command. It is then re-enabled with set_script_output 1 before starting the next dataset. This ensures that the data appears as a single continuous plot.