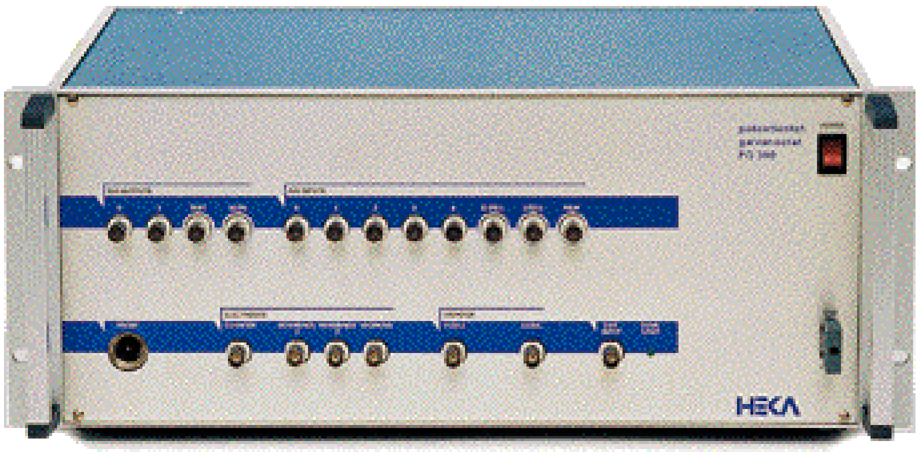
8.54
POTPULSE Manual



HEKA Elektronik Dr. Schulze GmbH

Wiesenstraße 71 • D-67466 Lambrecht • Germany
Tel: +49 (0) 6325 9553 0 • Fax: +49 (0) 6325 9553 50
Web Site: http://www.heka.com
E-mail: sales@heka.com • support@heka.com

Table of Contents

Introduction

Scope of the Program

The *POTPULSE* program provides versatile tools for electrochemical experiments. Pulse generation, data acquisition, storage, and analysis are among them.

POTPULSE is a data acquisition program as well as a driving program for potentiostats, especially with controlling features for those of the PG310/390 series by HEKA Elektronik and by use of an appropriate interface for all other potentiostats e.g. PG410 , PG490 , PG510 , PG 590.

Supported System Software

POTPULSE is supported on all Windows Versions: Windows 95, Windows 98, Windows ME, Windows NT 3.51, Windows NT 4.0, Windows 2000 and Windows XP.

POTPULSE is supported on all MacOS Versions above 7.6 *except* MacOS X.

Naming Conventions

Windows versions

Throughout the present manual we will address all the above Windows versions as "Windows". We will explicitly mention the particular Windows versions, whenever it is required.

Syntax

POTPULSE	Capitel italic letters denote a Heka program.
'F9'	Small Capitels denote keys on the keyboard.
Tree	Denotes menu entries, options and buttons.
Italic	Window names, Chapter names and Emphasis.
• Bullet list	Lists points.

1. Numbered list	Lists actions to be done chronologically.
& or (Windows) & (Mac)	Denotes system-specific keys and actions.

Installing POTPULSE

The first thing is to install the hardware and the software. If a PG310/390 potentiostat/galvanostat is used, connect it to the AT-16- or PCI-16-board as described (see PG310/390 Manual). For the detailed installation, render to the Installation Guide.

Starting POTPULSE

Upon clicking on *POTPULSE* the software starts and various controls become available:

- The drop-down menus (File, Edit, POTPULSE, PGF-Editor, Tree, Buffer, Marks, Display, Notebook, and Help).
- The *POTPULSE* windows (e.g., *Oscilloscope, Potentiostat, Replay, Pulse Generator, Configuration, Parameters, Online Analysis*).
- A scrolling protocol window at the bottom (*Notebook*).

Most of these windows can be iconized; they can be re-opened by clicking on the icon or by selecting them in the drop-down menu POTPULSE.

Note: The *Oscilloscope* window is the master window and cannot be closed.

All open windows are updated whenever the program encounters some changes; program speed can therefore be accelerated by keeping only the needed windows open.

Get Online Help

The option Show Keys in the drop-down menu Help displays the keys that are assigned to various controls of the active windows.

Closing POTPULSE

To exit from *POTPULSE*, do this:

Choose Quit from the drop-down menu File or press ‘ALT‘ + ‘Q‘.

Press ‘CMD‘ + ‘Q‘.

You will be asked whether you want to store the configuration file or not. If you choose Yes, acquired data are stored and files are closed before exit.

Support Hotline

If you have any question, suggestion, or improvement, please contact HEKA. The best way is to sent us an e-mail or a fax specifying:

- Your postal and e-mail address (or fax number)
- The program name:
  POTPULSE, etc.
- The program version number:
  v8.50, v8.53, etc.
- Your operating system and its version:
  MacOS 7.6.1, MacOS 8.5,
  Windows 98, Windows NT 4.0, etc.
- Your type of computer:
  Mac PPC 8500, Pentium II 300 MHz, etc.
- Your acquisition hardware, if applicable:
  ITC–16, ITC–18
- Your amplifier, if applicable:
  PG310, PG 390, etc.
- The serial number and version of your PG310/390, if applicable:
  EPC9 single, version "280552 D".
- The questions, problems, or suggestions you have.
- Under which conditions and how often the problem occurs.

We will address the problem as soon as possible.

HEKA Elektronik	*phone:*	*+49 (0) 6325 9553 0*
Wiesenstrasse 71	*fax:*	*+49 (0) 6325 9553 50*
D-67466 Lambrecht/Pfalz	*e-mail:*	*support@heka.com*
Germany	*web:*	*http://www.heka.com*

POTPULSE Tutorial: The first Experiment

This chapter will guide you briefly through the main features of the *POTPULSE* program and should take you a maximum of about 2 hours to read it. It briefly describs how a very simple first experiment with *POTPULSE* could look like. This section is thought for users that can't wait to get something done with *POTPULSE.* The basic requirements for starting the program and for doing a simple experiment are outlined. For more detailed descriptions of the features, refer to the later sections in this manual.

Note: In the following it is assumed that the hard- and software has already been set up correctly. Please refer to the "Installation Guide" for the installation of *POTPULSE.*

Starting POTPULSE

Turn on the interface and the computer and start *POTPULSE*, either via Programs/Heka/Potpulse (Windows) or by clicking on the icon POTPULSE (Mac). While starting the program, these things can happen:

- You may not have the hardware key correctly installed. *POTPULSE* will continue to run in Demo mode, with a stimulus simulation of the AD board and without continuous aquisition. For installation of the hardware key, please refer to the Installation Guide. Note that in the Demo mode it is *not* possible to save files!
- You may not have connected any AD/DA hardware. *POTPULSE* will recognize this and will ask you to abort, to continue, or to try again. If you just forgot to turn on the power of the PG310/390 or the ITC-16, do so and select Retry. If you don't have either of them, you still can select Continue, i.e., *POTPULSE* will continue to run. Data acquisition will be disabled but opening, saving and printing files will be possible.
- You may not yet have a so-called "Configuration File" (i.e., a file with the extension ".set " that contains all of your individual program settings). *POTPULSE* will then come up with "Cannot find configuration file: use defaults or find file". If you do not have a configuration file, select Use Defaults. In this case, *POTPULSE* will generate default settings and will come up with a selection of windows.

Based on your configuration file, *POTPULSE* will now look for file paths and a default Pulse Generator File (PGF), that contains your stimulus protocols. The factory default is the file "DefPot.pgf" in the Data folder inside the HEKA folder.

If *POTPULSE* cannot find your .pgf file, it will write a message into the Notebook window and will create a default file with only one entry called "Test". There may be other paths missing and *POTPULSE* will put up an alert to that effect. You can safely ignore that error message, we will setup these paths next in the Configuration window (see below).

After loading its configuration and pulse generator file, *POTPULSE* will ask, whether you wish to create a new experiment or just want to analyze some data:

There are five possibilities:

- **Update current data to disk** writes data from RAM to disk that was not yet stored.
- **Modify an existing file** opens an existing experiment for modification, i.e. you can delete or add further experimental data to a file.
- **Read and display old file** opens an existing experiment. The file will be write protected, so that modification (or loss) of the data is prevented.
- **Quit without change** cancels the dialog.
- **Create a new file** allows you to create a new experiment file.

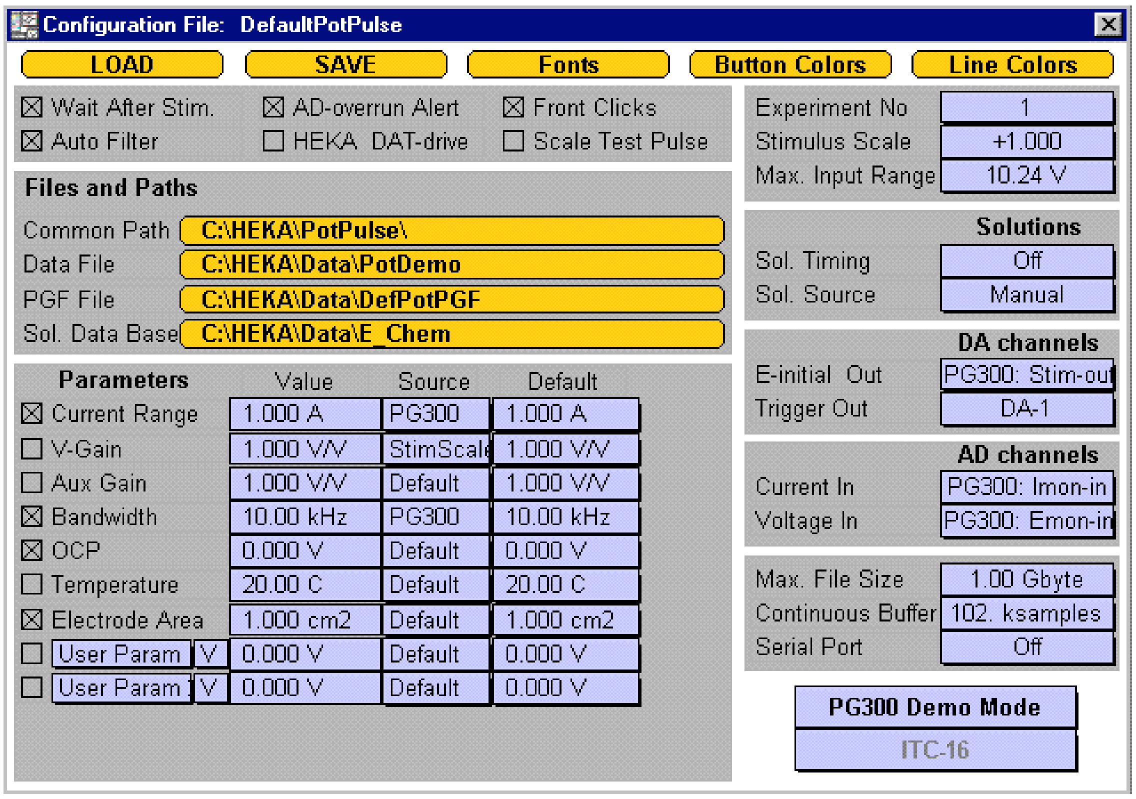
Select the **Create** option to start with a new experiment. You can call the file whatever you like, e.g. "Tutorial.dat".

Note: A *POTPULSE* experiment consists of at least 3 files, the raw data (file name extension: *.dat), the pulse protocols used (*.pgf) and a file that contains the amplifier settings and structure of your experiment (*.pul). You don't have to create all files by yourself and you can also ignore the file extensions. If you create a new experiment, simply type the name of the experiment, e.g. "Tutorial".

POTPULSE will open some windows: the most obvious ones are the *Potentiostat* and the *Oscilloscope* window. We will deal with these windows soon; however, first we have to make sure that the hardware is connected properly and that the software settings meet the requirements. The most important hardware settings are defined in the Configuration window.

Select Configuration from the drop-down menu POTPULSE or type 'F8' (Windows) or 'F11' (MacOS).

Configuring the Hardware


Configuration File: DefaultPotPulse

LOAD | SAVE | Fonts | Button Colors | Line Colors

☒ Wait After Stim. ☒ AD-overrun Alert ☒ Front Clicks
☒ Auto Filter ☐ HEKA DAT-drive ☐ Scale Test Pulse

Experiment No	1
Stimulus Scale	+1.000
Max. Input Range	10.24 V

Files and Paths

Common Path	C:\HEKA\PotPulse\
Data File	C:\HEKA\Data\PotDemo
PGF File	C:\HEKA\Data\DefPotPGF
Sol. Data Base	C:\HEKA\Data\E_Chem

Solutions

Sol. Timing	Off
Sol. Source	Manual

DA channels

E-initial Out	PG300: Stim-out
Trigger Out	DA-1

AD channels

Current In	PG300: Imon-in
Voltage In	PG300: Emon-in

Parameters	Value	Source	Default
☒ Current Range	1.000 A	PG300	1.000 A
☐ V-Gain	1.000 V/V	StimScale	1.000 V/V
☐ Aux Gain	1.000 V/V	Default	1.000 V/V
☒ Bandwidth	10.00 kHz	PG300	10.00 kHz
☒ OCP	0.000 V	Default	0.000 V
☐ Temperature	20.00 C	Default	20.00 C
☒ Electrode Area	1.000 cm2	Default	1.000 cm2
☐ User Param V	0.000 V	Default	0.000 V
☐ User Param V	0.000 V	Default	0.000 V

Max. File Size	1.00 Gbyte
Continuous Buffer	102. ksamples
Serial Port	Off

PG300 Demo Mode
ITC-16

AD/DA Channels

The *Configuration* window provides a variety of parameters that can be adjusted. First of all, *POTPULSE* has to know whether to use an PG310/390 potentiostat/galvanostat or another amplifier.

Click on the Amplifier Selection button in the lower right corner of the window and select the potentiostat/galvanostat or another amplifier from the list.

- If your particular potentiostat is not in the list, select the PG284
- If no amplifier is available, several Demo modes can be used.

The next step is to define the AD and DA channels to be used for test pulses and parameter input. For users of the PG310/390PG310/390, some of these channels are predefined:

Input/Output	Channel
voltage stimulus	DA-3
current input	AD-6
voltage	AD-5

Therefore, for a different potentiostat you must connect Emon-in to AD-5 (U-Cell). The other channels may be selected freely.

Note: There are 4 DA channels (0...3) and 8 AD channels (0...7). For the PG310/390, AD channel 7 is reserved for internal use (test and calibration) and should not be used for anything else.

Optionally, the output channels for up to three triggers can be specified. Triggers can alternatively be output via the digital port. The AD/DA channels used during stimulation and acquisition of a Series are determined in the Pulse Generator (see Chapter *Pulse Generator*).

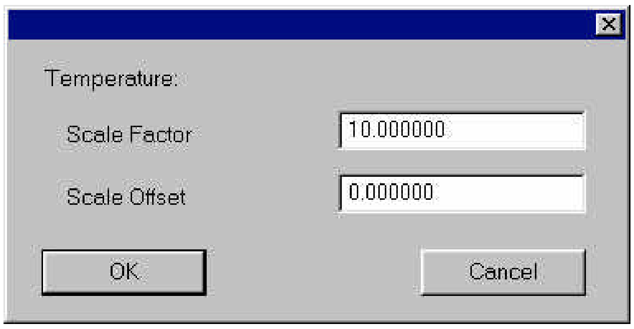
Parameter Input

So far we specified the most important parameters of the Configuration File. On the left-hand side – the Parameters section - there is a list of further values that are acquired and stored together with the experiment. These parameters can be input via different means: e.g., directly from the PG310/390, from default setting or from AD channels.

The checkboxes in the parameter list determine whether the parameter is shown in the *Parameters* window (see Chapter *Parameters*).

Note: The checkboxes to the left of each parameter only define, if the corresponding setting will be visible within the *POTPULSE* session. Despite of this setting, *POTPULSE* will always store every parameter with every block of data acquired. A separate block of data is called "sweep" in the *POTPULSE* terminology.

Go through all of these parameters and select the input source and maybe new default values. If an AD channel is used as input source, then the scaling can be specified.


Temperature:

Scale Factor 10.000000

Scale Offset 0.000000

Example: Let us suppose the Temperature is read from a temperature control unit via AD channel 4. Let's further assume that the control unit delivers an analog signal of 100 mV/°C and 0 V at 0 °C. Then the scaling factor to be entered is 10 (1V corresponds to 10 °C) and the offset is 0. Click the Scale button next to the Temperature parameter and fill in these settings.

Now the parameters concerning the interface to the setup hardware are set and the configuration file can be saved (Save button). A new name of such a file can be selected; when *POTPULSE* comes up, however, it looks at first for a file called "DefaultPOTPULSE.set". Thus, if one will have only one configuration file, it is useful to call it this way (see Chapter *Configuration*).

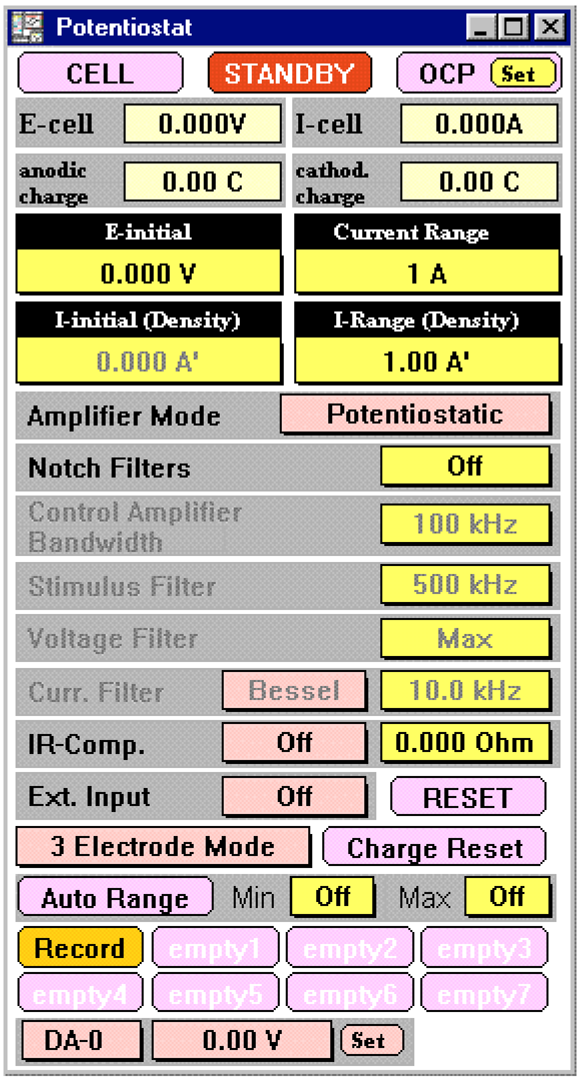
Note: If you are sampling from a so called "telegraphing" amplifier, the determination of the encoded amplifier gain and filter bandwidth settings is much more involved and is done via lookup tables that are provided as ASCII files (see Appendix IV). You can select the corresponding gain or bandwidth table with the Table button.

Potentiostat Control

If a PG310/390 potentiostat/galvanostat is used, the PG310/390 potentiostat window provides the potentiostat/galvanostat control functions (see Chapter *PG310/390 Potentiostat/Galvanostat*). For other potentiostats the user has to make sure that:

- the command potential at the potentiostat is set to zero,
- the stimulus scaling is set correctly (see Chapter *Configuration*),
- the DA channel for StimOut is connected to the stimulus input of the potentiostat,
- the current monitor of the potentiostat is connected to the I-Cell AD input channel, and
- that for a potentiostat with telegraphing capabilities for current range and/or bandwidth the corresponding analog outputs are connected to the assigned AD channels. For telegraphing potentiostats, current range and bandwidth lookup tables in ASCII format can be used (see Chapter *Configuration*).

The command potential is controlled by the program via the control E-Initial in the *Potentiostat* window.



E-cell 0.000V
I-cell 0.000A
anodic charge 0.00 C
cathod. charge 0.00 C
E-initial 0.000 V
Current Range 1 A
I-initial (Density) 0.000 A'
I-Range (Density) 1.00 A'
Amplifier Mode Potentiostatic
Notch Filters Off
Control Amplifier Bandwidth 100 kHz
Stimulus Filter 500 kHz
Voltage Filter Max
Curr. Filter Bessel 10.0 kHz
IR-Comp. Off 0.000 Ohm
Ext. Input Off

Auto Range Min Off Max Off

DA-0 0.00 V

Controlling the Hardware via POTPULSE

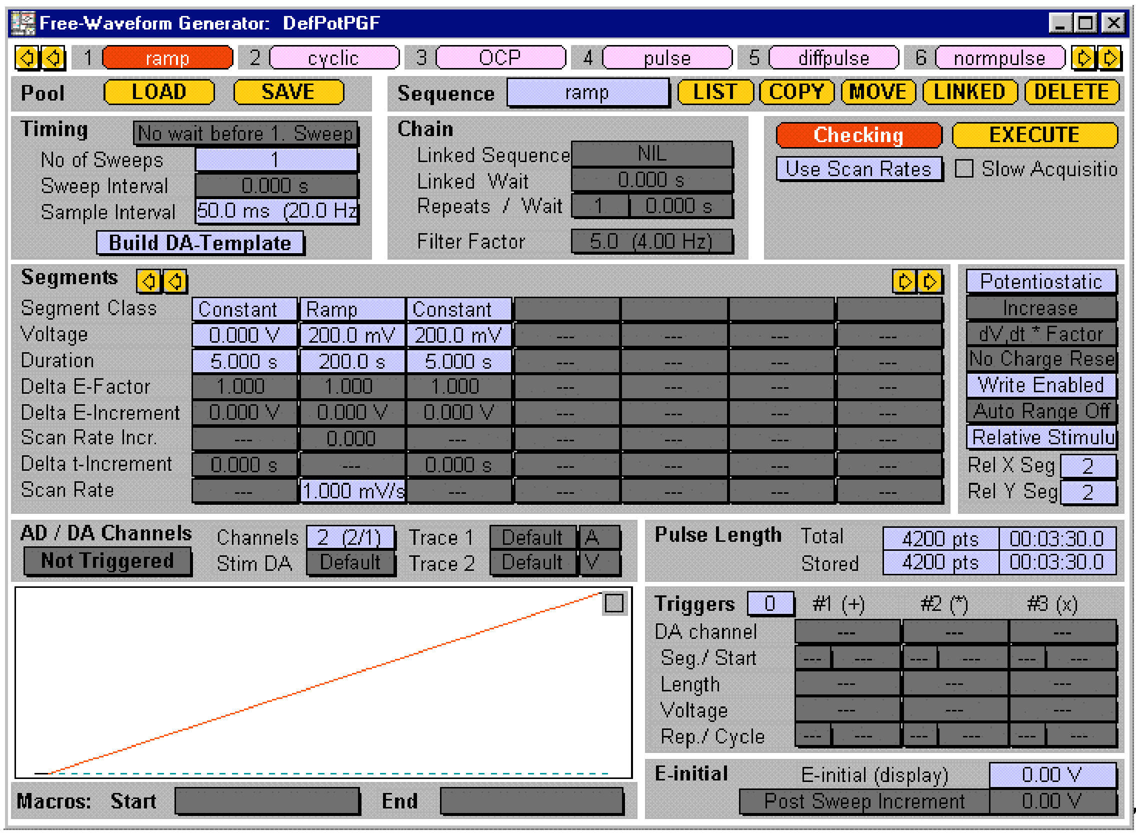
When we hit the space bar in the main dialog or when we click inside the *Potentiostat* window, the program will be able to control the potentiostat/galvanostat by selecting the desired settings in this window instead of manipulating real switches, buttons, etc., on the panel. Typical functions, such as setting current range, setting initial potential, connecting (Cell) and disconnecting (Standby) the cell, etc. are available.

To perform a cyclovoltammetric experiment, you have to generate a Sequence or to execute one of the basic patterns that can be found in the *Pulse* Window (e.g., ramp, cyclic...).

To open the *Pulse Generator* Window, select Pulse Generator from the drop-down menu POTPULSE or press ‘F9‘ (Windows) or ‘F12‘ (MacOS).

Note: “Pulse Generator window” and “Free-waveform Generator“ are synonymous.

Generating a Simple Pulse Protocol



Free-Waveform Generator: DefPotPGF

1 ramp | 2 cyclic | 3 OCP | 4 pulse | 5 diffpulse | 6 normpulse

Pool: LOAD | SAVE

Sequence: ramp | LIST | COPY | MOVE | LINKED | DELETE

Timing: No wait before 1. Sweep
No of Sweeps: 1
Sweep Interval: 0.000 s
Sample Interval: 50.0 ms (20.0 Hz)
Build DA-Template

Chain
Linked Sequence: NIL
Linked Wait: 0.000 s
Repeats / Wait: 1 | 0.000 s
Filter Factor: 5.0 (4.00 Hz)

Checking | EXECUTE
Use Scan Rates | ☐ Slow Acquisitio

Segments							
Segment Class	Constant	Ramp	Constant				
Voltage	0.000 V	200.0 mV	200.0 mV	---	---	---	---
Duration	5.000 s	200.0 s	5.000 s	---	---	---	---
Delta E-Factor	1.000	1.000	1.000	---	---	---	---
Delta E-Increment	0.000 V	0.000 V	0.000 V	---	---	---	---
Scan Rate Incr.	---	0.000	---	---	---	---	---
Delta t-Increment	0.000 s	---	0.000 s	---	---	---	---
Scan Rate	---	1.000 mV/s	---	---	---	---	---

Potentiostatic
Increase
dV,dt * Factor
No Charge Rese
Write Enabled
Auto Range Off
Relative Stimulu
Rel X Seg: 2
Rel Y Seg: 2

AD / DA Channels: Not Triggered
Channels: 2 (2/1) | Trace 1: Default | A
Stim DA: Default | Trace 2: Default | V

Pulse Length		
Total	4200 pts	00:03:30.0
Stored	4200 pts	00:03:30.0

Triggers 0	#1 (+)	#2 (*)	#3 (x)
DA channel	---	---	---
Seg./ Start	--- ---	--- ---	--- ---
Length	---	---	---
Voltage	---	---	---
Rep./ Cycle	--- ---	--- ---	--- ---

E-initial: E-initial (display) 0.00 V
Post Sweep Increment 0.00 V

Macros: Start | End

The *POTPULSE* software allows you to stimulate your cell with pulse pattern from a basic rectangular pulse to highly complicated stimulation patterns. Stimulus templates are edited in the *Pulse Generator* window.

A pulse pattern (also called *Sequence*) consists of an arbitrary number of pulse segments of constant, ramp, or sinusoidal voltage.

Note: The amount of all selectable sequences is called "Pulse Generator Pool". This Pool is also accessible in the *Oscilloscope* Window.

In the default Pulse Generator File (named "DefPotPGF.pgf") some simple step and cyclic sequences are given. You can modify these sequences and create more sequences by using the Copy command. The new, modified Pulse Generator File can now be stored to disk by clicking on Save.

It can have any name; *POTPULSE* automatically adds the file extension ".pgf".

If you want *POTPULSE* to come up with this PGF-file already loaded at the next launch, save the Configuration File before exit (this file also keeps track of the used file names and data paths).

Creating a new sequence

The easiest way to create a new sequence is to copy a given one.

1. Click on the sequence called "Cyclic" and then on Copy.
2. Name this copy “Test”. It is saved als last in row.
3. To bring this new protocol “Test” to a more handy position click the Move button and select “1” as the new position. Note that the first six protocols will be directly available from the Oscilloscope window.

Timing

Since we want to do just one single sweep, the parameter No. of Sweeps in the Timing section of the *Pulse Generator* window doesn`t have to be changed.

We also don`t want to wait before starting the pulse sequence, hence the setting No wait before 1. Sweep is correct.

Defining the Segments

The section Segments of the Pulse Generator window defines the actual stimulation to be applied.

In our case the sequence “Test” consists of two kinds of segments:

- **Constant Segments**: The cell will be held at the zero potential in the beginning and the end of the sequence.
- **Ramp Segments**: During a certain time (Duration**)** the voltage will be increased up to the target value, given in the setting Voltage.

This leads to the typical cyclovoltammetric triangles as you can see in the preview down right.

We now want to change the peak voltage. For that please double-click into the Voltage field of the first Ramp segment and enter “1” (for one Volt).

The upcoming alert warns you that you request more points per sweep than could be processed.

This you can correct by choosing a lower Sample Interval in the Timing section.

Double-click on the given Sample Interval and enter “0.001” (for 1 ms). Now you can enter the Voltage 1 V and –1 V into the corresponding fields of the first two ramps. The preview will be immediately updated.

Maybe you wondered why the third segment (ramp down) is drawn in red color in the preview while the rest is black. *POTPULSE* can perform an Online Analysis whenever you run or replay an experiment (see below). This is done by analyzing one segment (Rel. Y Seg.), e.g. determining its peak- or mean current, and plotting it against another parameter like the duration or potential of any other (or the same) segment (Rel. X Seg.). You can define which segment has to be analyzed by setting the so called Relevant Segment. This can be done separately for the segment that delivers the x- and the y-value.

Double-click on the fields next to Rel. X Seg. and Rel. Y Seg. and set both values to “2”. In the preview you can see that the first ramp up is red now.

Your later analysis will not be restricted to the segments you define here. In the *Online Analysis* window you can set a positive or negative Segment Offset that will be added to the relevant segment, thus allowing you to analyze also the other segments (e.g. the pre-trigger).

Other Settings in the Pulse Generator window

Macros: Start: Here you can input a macro which will be run before running the actual sequence. Therefore, the macro name in the field **End** will be run after the sequence. We will record a macro in the next section.

The option Write Enabled on the right side will make sure that the pulse sequence can be stored to disk. For some protocols it might not be required to save the data (such as during a pre-conditioning waiting period), so you can disable this feature in these cases.

The section AD / DA Channels defines which channel to stimulate and from which channel(s) to acquire the data. Stim DA is set to Default which means that you are stimulating through the PG310/390 stimulus channel. The number of input Channels is set to 2 (*POTPULSE* allows you to acquire two channels simultaneously), making a total of two input and one output channel (the bracketed number next to Channels).

The two input channels are also denoted as “traces” and by default are the current in A (trace 1) and voltage in V (trace 2).

Note: *POTPULSE* allows you per default a maximum of 16384 points per sweep. If you are sampling two input channels this number reduces to half (8192 points). If you want to be able to acquire more points in one sweep, please refer to the Chapter *POTPULSE Advanced*.

This new, modified Pulse Generator File should now be stored to disk by clicking on Save. It can have any name; *POTPULSE* automatically adds the file extension “.pgf”. If you want *POTPULSE* to come up with this PGF-file already loaded at the next launch, simply save the Configuration File before exit.

Recording a Macro

Macros can be generated in *POTPULSE* by letting the program record your commands, while you are executing them. Afterwards *POTPULSE* will convert these actions into a text file that you can easily edit with any ASCII editor (e.g. the Windows NotePad or Apple's SimpleText) in order to "fine tune" it.

One sensible automatic analysis would be the output of the peak current after the sequence was run.

The analysis itself you have to customize in the *Online Analysis* window.

To automatize that, do the following:

1. Start the macro recorder by selecting Start Recording from the PGF-Editor -> Macros sub-menu.

2. Bring the *Online Analysis* window to the top by either clicking on the window or selecting Online Analysis from the *POTPULSE* menu. Alternatively, you can type the keyboard equivalent 'F7' (Windows) or 'F8' (MacOS).

3. In order to define the first analysis you have to activate Range 1 from the Range popup.

4. From the Abscissa popup select Peak Voltage and from the Mode popup select Maximum. These settings tell *POTPULSE* to seek the Voltage of the peak current and plot it against the maximum current. From the Relevant Segment Offset popup choose "0" for both settings to analyze the relevant segment we defined in the Pulse Generator window.

5. The macro is finished by now, so select Stop Recording from the PGF-Editor->Macros sub-menu.

6. You will be asked for an index and a name for the macro. Use the index 1 and name the macro "Testmacro". Note that macros are internally recognized by their names rather than their index.

Finally, let us have a look at the macro by selecting List from the PGF-Editor -> Macros sub-menu. The fourth macro should give you in the Notebook window something like the following:

```
1 : Testmacro
  A  Range:            0; Range 1
  A  Abscissa:         6; Peak Voltage
  A  Mode:             2; Maximum
  A  RelXSeg:          0
  A  RelYSeg:          0
```

Note: *POTPULSE* has a built-in macro interpreter that executes command lines of the form "Window Control[: parameter; comment]". E.g., the line "A LeftB: 10%" would instruct *POTPULSE* to set the *left* boundary in the Online

Analysis window to *10%*. For this tutorial it is not necessary to know all possible commands and their syntax. Please, refer to the Chapter *Macros* for further details.

Running the Pulse Sequence

Bring the Oscilloscope window to the front. If the button Store is not highlighted, click on it to make it active. If you did not create a file yet, *POTPULSE* will ask you to do this now.

Note: *POTPULSE* will *only* save the data, when the Store button is active (red), otherwise the data is showen but *not* written to disk.

The bottom of the *Oscilloscope* window shows the currently available Pulse Generator Pool.

Customizing the Display

Before we execute the "Test" sequence (resp. Series in *POTPULSE* terminology, which describes a number of individual sweeps based upon the same Pulse Generator Protocol) we will set up the display.

In the Display menu activate

- **Show Zero Line**,
- **Show Potential**,
- **Dimmed Overlay**,
- **Labelling** ® **Labels Only** and
- **Display Mode** ® **I vs. t**.

To see all sweeps from the series activate the Overlay button, otherwise the *Oscilloscope* window will be cleared of data before every sweep.

Execute a Sequence

An individual Sequence from the Pulse Generator Pool can be executed in the *Oscilloscope* window in several ways:

- For the first sequences click on the sequence (1..6) or enter the corresponding number via the keyboard (1..9).
- Sequences with an index higher than 6 resp. 9 can be reached via the scroll arrows or directly by entering # followed by the number.
- Activated Sequences can be run by pressing 'E' on the Keyboard.

Note: You can also directly trigger the sequence in the *Pulse Generator* window by first clicking on the sequence to highlight it and then clicking on Execute or pressing 'E'.

Now click on the button Test or type "1", if „Test" is the sequence in the first position: The pulse pattern we defined above is output via the specified DA channel and the response is shown on the screen.

If you do more than one sweep, the last sweep of the series is shown in black color, the other sweeps are gray since you activated Dimmed Overlay.

The scaling bars are given in the lower left side of the Oscilloscope. The two red lines mark the range of the Online Analysis (see next section). You can make this range visible by activating the Keep button.

If a Sequence is executed from an active *PG310/390 Potentiostat* window by typing either E or the number of the corresponding sequence, the test pulses are interrupted, the Sequence is executed and displayed. The Wait control in *the Oscilloscope* window is then flashing "Continue?". Any key will resume the test pulses and thereby erase the displayed traces from the screen.

Replay window with data tree

The structure of the stored data - if the Store button was active - will be shown in the Data Tree of the *Replay* window.

To open it select Pulse → Replay or type 'F5' (Windows) or 'F13' (MacOS).

Double-click the "Test 1" entry to replay the just recorded sequence; double-click a single sweep to watch it in the *Oscilloscope* window.

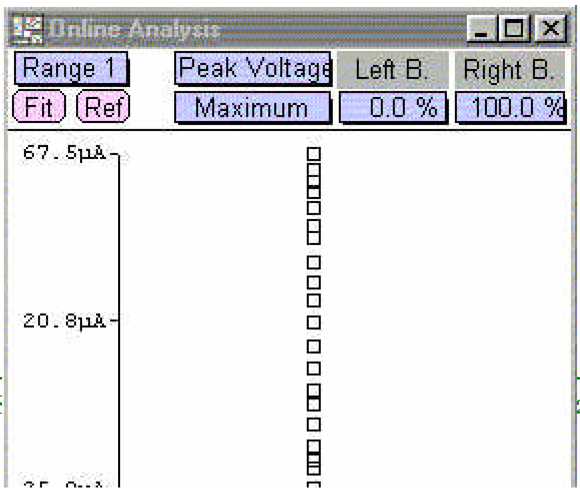
You might use the cursor keys ('UP', 'DOWN', 'LEFT' and 'RIGHT') to walk through the data tree. If you press 'RETURN', the currently active group, series or sweep will be displayed in the *Oscilloscope* and the *Online* Analysis will be calculated.

Using the options from the Tree menu it is possible to modify the entries. E.g. a single sweep, a series, or a whole Group of series can be removed by activating the item and then selecting Tree → Delete.

To write the recorded data to disk select File → Update File or close the experiment (File → Close), this will automatically store all files associated with the experiment. To create a new file for data acquisition select File → New... , *POTPULSE* will close the running experiment and then come up with an empty new one.

Online Analysis

Based on the relevant segments as specified in the Pulse Generator, a quick online analysis of the acquired (or replayed) data is performed immediately after the series has been collected entirely. The criteria for this analysis (i.e., type of analysis, time range, format of display) were specified before in the *Online Analysis* window, when we re-



corded the “Testmacro”; now the result - an maximum current I versus Peakvoltage V curve - is plotted in the graph inside the *Online Analysis* window.

Example: For a multi-sweep experiment with increasing maximum current, but without shift of the peak potential, you would get an Analysis result as in the picture.

The results are also sent to the Notebook as a table. If you bring the Notebook window to the front (e.g. by selecting Pulse → Notebook), you should see something like the following:

```
Execute: Test
#   V(2)[mV]   t[ms]     i[A]
    1   760.00m   9.9998    67.500µ
    2   760.00m   9.9998    63.125µ
    3   760.00m   9.9998    59.375µ
    4   760.00m   9.9998    56.562µ
    5   760.00m   9.9998    52.187µ
    6   760.00m   9.9998    47.187µ
    7   760.00m   9.9998    44.063µ
    8   760.00m   9.9998    37.187µ
    9   760.00m   9.9998    31.563µ
```

Exit

If *POTPULSE* is quit (File → Quit), you are asked whether you want to save the Configuration File. At least the first few times of running *POTPULSE*, after tuning the system, you should do that, since this file contains all of the settings that were adjusted as outlined above. Once you have a stable system that you don’t want to modify anymore, you can safely ignore this question. Finally, *POTPULSE* will automatically store the recorded data on the hard disk and close all open files.

User Interface

The following chapter describes the user interface of *POTPULSE*.

It consists of three chapters:

- "Dialog controls" explains all Controls used in the *POTPULSE* user interface.
- "Modifying the Dialogs and Controls" describes how size, color and arrangement can be changed.
- "Storing modified Dialogs and Controls" descibes how to save the Dialogs onto the hard disk.

Dialog Controls

In general, box items with a drop shadow enclose changeable values, either as a list item (or pop-up menu list), or as a Drag item. Rounded rectangles are items which perform some action, while simple rectangles (without drop shadow) display a measured value. Plain text is for titles only.

Drag: 63.5 % A number in a box with a drop shadow. The parameter value in a Drag item can be changed by clicking on it and dragging the mouse up and down. Alternatively, one can double-click on it, or 'SHIFT'-click, or 'RIGHT'-click (Windows), and then type in a new value. Terminate input with 'RETURN' or 'ENTER'. Using 'TAB' will cycle through all Drag items.

Note 1: When you enter a new value, do *never* include (or leave from the previous text) the unit character. The unit character (e.g. "m" for "meter") could conflict with the input of engineering units (e.g. "m" for "milli", see section "Numerical Input" below). Since the text is selected when the item gets selected, you can just type in the new value and the old text gets replaced.

Note 2: Sometimes the computer is very busy and there might be not enough processing time to handle a double mouse click (the operating system will report two separate mouse clicks to the program instead). In this case you still will be able to activate a control by keeping the 'SHIFT' key pressed and then click the control once with the mouse (right mouse click under Windows).

List: Voltage Similar in appearance to a Drag item. Clicking on it will pop up a menu list from which one can choose a setting.

Edit Text: User Param A text string in a box with a drop shadow. Clicking on it will allow to edit the displayed string.

Button: EXECUTE Rounded-corner rectangle. Clicking on it will cause the respective action to occur.

Switch: Show Rounded corner rectangle. Clicking on it toggles the parameter value. The switch is On or Activated if the item is highlighted Show. A Switch can optionally also execute some action.

Radio Button / Check Box: ☒ Pipette Resist. Identical to the standard dialog items. Clicking on them will toggle the respective parameters.

Framed Text / Number / Boolean: -70.0 Simple box with optionally some text. The Boolean value is indicated by its color, inactive controls are gray 0.0.

Enter: Pressing 'ENTER' on the extended keyboard always brings you back to edit the control that was edited last. The feature is very useful, when one edits very often the same control (e.g., a duration of a specific segment in the Pulse Generator or the Display Gain in the Oscilloscope window).

Background Color: The color that appears while the user is dragging or entering a value is set by the Highlight Color in the MacOS Control Panel. *Be careful if changing* - the user will not be able to read the edited number if the highlight color is set to a very dark color. The Windows version displays highlighted controls with white text on a black background.

Numerical Input: Numerical values can be entered either by dragging the value or by typing after double-click (or a 'SHIFT'-click) on the item. In the latter case the value can be entered in scientific notation (e.g., "2.3e-3", "2.3E-3") or in engineering format (e.g., "2.3m"). Numbers outside this range for engineering numbers (see table) are always displayed in scientific notation. The old value is erased as soon as the user starts to type. To preserve the old string, move the 'LEFT' or 'RIGHT' cursor first. To leave the previous value unchanged although a new one has been entered already, just clear the input by pressing 'ESC', then 'RETURN' or 'ENTER'.

Name	eng.	sci.
Exa	E	e18
Peta	P	e15
Tera	T	e12
Giga	G	e9
Mega	M	e6
kilo	k	e3
milli	m	e–3
micro	µ - u	e–6
nano	n	e–9
pico	p	e–12
femto	f	e–15
atto	a	e–18

Note: Never include (or leave from the previous text) the unit character when entering a new value. The unit character (e.g. "m" for "meter") could conflict with the input of the engineering units (e.g. "m" for "milli").

SI Units: *POTPULSE* expects most units to be SI-units, i.e., meters, seconds, amperes, volts, Hertz, etc. However, for convenience there are exceptions to that rule. In such cases the item title contains an identifier for what unit is to be used, e.g., mV (Post Sweep Increment [mV]) if a voltage is to be entered in milli-volts rather than in volts.

String Buffer: Whenever an edit process is finished with 'RETURN' the edited string is entered into a cyclic buffer of edit strings consisting of 10 entries. These strings can be accessed during editing using 'CURSOR UP' and 'CURSOR DOWN'. This feature is quite useful, when identical or similar strings have to be typed into various string items.

Modifying the Dialogs and Controls

There's a "hidden" feature in *POTPULSE*: Dialog items can be modified in many different ways: background or item color, text font, position of one item, position of all items in the window, etc.

First of all, engage 'CAPSLOCK'. Now you can customize the windows:

- To drag and resize an item, right-click on the item.
- To reposition all items within the dragged item rectangle, press 'CTRL' and right drag.
- To bring up the Dialog Control window, press 'CTRL' and click on the item. Here you can modify the item settings such as color, text font, dragging speed etc. (see below).

First of all, engage 'CAPSLOCK'. Now you can customize the windows:

- To drag and resize an item, press 'OPTION' and click on the item. The new item position will be ignored, if 'OPTION' is up when the mouse button is released.
- To reposition all items within the dragged item rectangle, press both 'OPTION' and 'COMMAND' while dragging.
- To bring up the Dialog Control window, press 'COMMAND' and click on the item. Here you can modify the item settings such as color, text font, dragging speed etc. (see below).

The following table summarizes all actions ('CAPSLOCK' has to be engaged!):

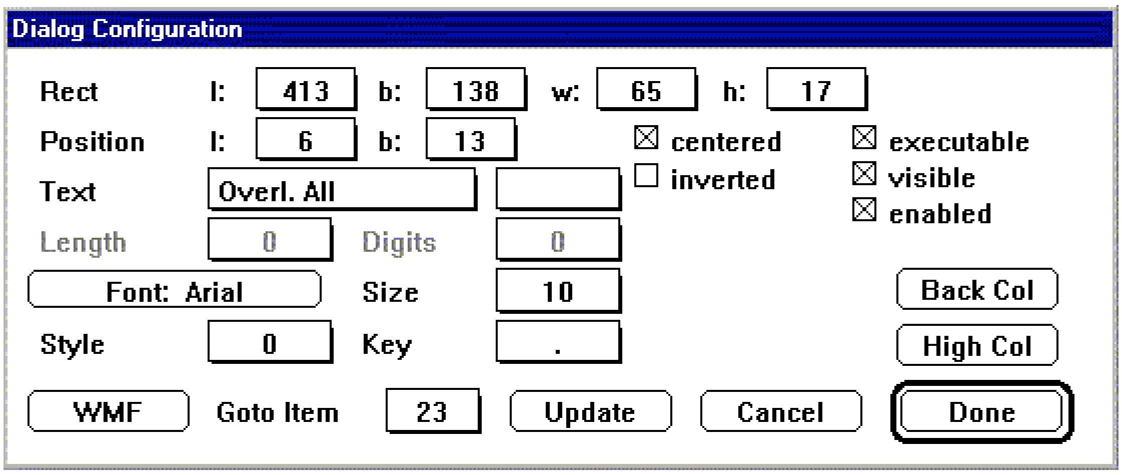
Action	MacOS	Windows
Open a configuration dialog	COMMAND + mouse click	CTRL + mouse click
Move one item	OPTION + mouse drag	right button mouse drag
Move group of items	COMMAND + OPTION + drag	CTRL + right button drag

All windows except the Configuration window can be iconized, i.e., reduced to a minimal size window. Such a window can be easily expanded to the original size (and shrunk again) by clicking in its zoom box.

Windows with a window bar can be moved and resized as any regular window.

Dialog Control window

In the Dialog Control window, you can control the settings of an item. Remember to disengage ‘CAPSLOCK’, if you want to enter numbers or small letters.



Dialog Configuration

Rect l: 413 b: 138 w: 65 h: 17

Position l: 6 b: 13

Text: Overl. All

Length: 0 Digits: 0

Font: Arial Size: 10

Style: 0 Key: .

Goto Item: 23

☒ centered ☐ inverted ☒ executable ☒ visible ☒ enabled

Here you can see the typical look of the Dialog Control window for a switch button with text. For other buttons, e.g., with numbers or execution commands, there may be more or less controls availables.

Here the complete list of controls:

Rectangle: Gives the position of the rectangle in the window (lenght **l**, width **b**) and the size of the item (**w**idth, **h**eight) in pixel.

Position: Gives the position of the text in the item (l, b) in pixel.

Text: Text which is displayed on the item. In this case **Length** and **Digits** are disabled.

Length: Lenght of the item in pixel.

Digits: Number of decimal places.

Font: Font choosen from the list of available fonts on the system.

Size: Font size of the text.

Style: Style of the text: 0 – normal, 1 – bold, 2 – italic, 3 – bold italic, 4 – underlined, 5 – bold underlined, 6 – italic underlined, 7 - bold italic underlined

Key: Character assigned to that item. This enables to execute the item from the keyboard.

WMF: The window can be saved as Windows Meta File (Windows). This is identical to the menu entry POTPULSE → Front Dialog → Save as WMF.

Goto Item: Focus in the activated window jumps to this function.

Centered: Centers the text on the item.

Inverted: Inverts the color of the text (black<>white)

integer: Value hast to be entered as integer, e.g., “5”.

fixed: Value has to be entered as floating point with maximum as many decimal places as given in **Digits**, e.g., “0.001”.

scientific: Value has to be entered in scientific format, e.g., “e-9” for “nano”.

engineering: Value has to be entered in engineering format, e.g., “n” for “nano”.

Executable: Button starts action by clicking. If not activated, button will turn blue when clicking, and its content cannot be edited.

Visible: Button is visible. Note: If not activated, the button will be invisible and cannot be edited anymore! To restore the original setting you have to delete the file “DefaultPotpulse.set”.

Enabled: Button can be activated or edited. If not enabled, the button will be grey and inactive.

Back Color: Color for a switch button that is *not* active (default: pink) or for any other button.

High Color: Color for a switch button that is active (default: red). For other buttons this feature is disabled.

To finish you input, do one of these:

- Click on Update to see the change in the item.
- Click on Cancel to leave the Dialog control window without changes.
- Click on Done to leave the Dialog control window and save all changes.

Storing modified Dialogs and Controls

The position, size, and iconized state of each window can be stored using the menu option Front Dialog → Save from the POTPULSE menu (see above). The settings of the Notebook window are saved in the Configuration File (e.g., DefaultPulse.set).

Note 1: When installing a new version of *POTPULSE* these customized dialogs are likely to become incompatible, because additional items will have been introduced in the new version. It is therefore best to trash these custom dialogs when upgrading.

Note 2: To sort a deranged window, do either close *POTPULSE* and restart it, or – if you already saved the window – delete “DefaultPotPulse.set” from the *POTPULSE* directory.

Toggling between windows

‘ESC’ can be used to switch to the *Oscilloscope* window. In addition, ‘ESC’ will cause some windows to be closed (e.g., *Pulse Generator* window, *Configuration* window, *Spectra* window).

The ‘SPACE BAR’ can be used to quickly toggle between the *Oscilloscope* window and the *Amplifier* window.

All dialog windows can be opened and closed using the function keys (see drop-down menu POTPULSE), except the *Configuration* window.

Tutorial: Changing the Size of the Oscilloscope

In the following we will use the “hidden” feature to increase the size of the oscilloscope display:

1. Increase the size of the *Oscilloscope* window. For that use the mouse on the lower end of the Oscilloscope window to pull it farther until you see the “hidden” features in the bottom.
2. Activate the ‘CAPSLOCK’ key.
3. Press ‘CTRL’ and click with the right mouse button (Windows) or while holding the ‘‘COMMAND’’ key down click (MacOS) into the right part of the window (somewhere between the E vs RE and Overlay button) and drag the whole group to right edge of the window. While moving you will see a gray rectangle underneath the set of controls.
4. Repeat the last step with the sequence pool.
5. Left click (Windows) or ‘OPTION’ click (MacOS) into the lower right edge of the oscilloscope and resize it.
6. Move it to the place you want to keep it.
7. Now you could save it using the menu Pulse → Front Dialog → Save.

Note: There is a shortcut to quickly resize the *Oscilloscope* window. Hold down the ‘SHIFT’ key while you resize the main window, and all controls will be moved automatically. But be careful – the controls may end somewhere you don`t want them. To sort a unsorted window, do either close *POTPULSE* and restart it, or – if you already saved it – delete “DefaultPotPulse.set” from the *POTPULSE* directory.

Menus

The following section describes the various drop-down menus in *POTPULSE*.

File Menu

The File menu has all options to handle *POTPULSE* experiment files. A single *POTPULSE* "Experiment", that can hold a variable number of single electrophysiological experiments, consist of at least three files (see Chapter *Data Format* for a detailed description):

- The *.pgf file (= **P**ulse **G**enerator **F**ile) has the stimulus templates used (Stim Tree).
- The *.pul file has the complete data tree (Pulsed Tree)
- The *.dat file has only the actual raw data without any timing or scaling information.

In addition there might be

- The *.sol file that has the solution database (if solution timing has been activated) and

If the X-CHART extension to *POTPULSE* was active, there are two more files:

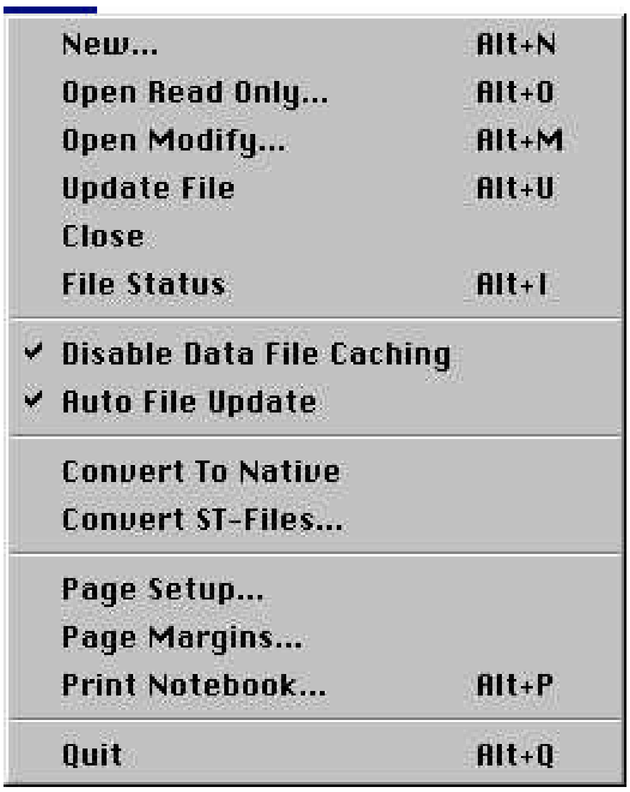
- The *.tree file with the X-CHART data tree and
- The *.grp file with the X-CHART data itself.

Raw data acquired by *POTPULSE* is only written to disk in the so called Store mode (i.e. if the Store button in the Oscilloscope window is engaged) and the pulse sequence executed is defined as write enabled in the Pulse Generator window. The data are written to disk *upon completion of a sweep or a series* or during the acquisition in case of a continuous sweep.

The structural information in the Stim Tree and Pulsed Tree are kept in RAM; they are stored to disk *only* when:

- a new file (File → New), a New Group (Pulse → New Group) or a New Experiment (Pulse → New Experiment) is created
- the Update File function is executed (File → Update File)
- the program is terminated (File → Quit)

New...: Creates a new, empty data file that is ready for data acquisition. The file has Read and Write permission.



Open Read Only...: Loads an existing file in Read Only mode. Modification of the file is not allowed. Use this option, when you want to analyze data and to make sure not to change or delete anything!

Open Modify...: Loads an existing file with Read and Write permission. Modification of the file such as appending data is allowed.

Note: Deleting entries in the data file is *not* reversible, once the file has been opened for modification. Make sure to always have a backup of the original files, when modifying an experiment. The exception is, of course, when you *really* want to delete a part of the stored data.

Update File: Updates the whole experiment to disk. This includes all files involved (see above). If you encounter computer crashes leading to data loss, use this option frequently (at least every time you go to get a fresh cup of coffee) or enable the setting File → Auto File Update. Chapter *Troubleshooting* tells you how to recreate a valid experiment from the raw data file if you loose the Stim and the Pulsed Tree due to a computer crash.

Close: Closes the actual experiment.

File Status: Prints information about the status of the currently opened file such as the path, length, etc. to the Notebook. A typical output could look as follows:

```
read-only file: "C:\HEKA\Data\PotDemo.dat"
length: 284 kb; free disk space: 1428 Mb.
```

Disable Data File Caching : Normally Windows writes all data into a file cache first. In case of a computer crash this cache would be lost, so here the Data File Cache is disabled by default. If you enable the cache, the data will be stored faster on the benefit of more safety.

Disk Write Options (MacOS only): Defines when the raw data are flushed to disk:

- **Write After Sweep** - Writes raw data to disk after each sweep.
- **Write After Series**- Writes raw data to disk after each acquired series.

The first option will make sure that the system file cache gets written to disk (i.e., "flushed") after acquiring a Sweep, and the second option performs a cache flush after acquiring a complete Series. Deselecting both options will suppress file cache flushing. In that case, the operating system will flush the file cache when the latter overflows. The file cache size can be set in the Control Panel Memory.

The flushing of the file cache may take few to many seconds, depending on its size. Thus, if one lets the system decide when to flush, it may occur at an inappropriate moment, such as in the middle of a series. On the other hand, if *POTPULSE* would always force file cache flushing (as it does when the option Write After Sweep is active), one could not take advantage of the file cache. The usefulness of the file cache is that writing to the file cache in RAM is faster than physically writing to disk.

Summarizing: It is safest to select the Write After Sweep option. This ensures that the data are immediately written to disk. Also, the timing between Sweeps is not interrupted by the system, when a possibly large file cache is written to disk. If one must get the fastest disk performance possible, one can de-select the options. In that case, data are written to RAM, not directly to disk. But this can only work as long as fewer data are acquired than there is space in the continuous buffer.

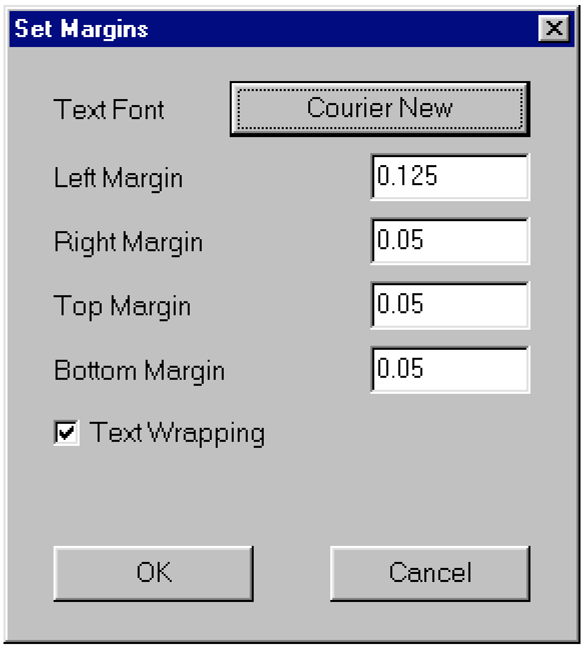
Auto File Update: Automatically updates all files after each series (including the Stim and Pulsed Tree). The previous options only specify when the *raw data* are written to disk. While these data are physically written, they are not really accessible without the appropriate tree structure. Thus, in case of a crash, these data are not retrievable, because the tree structure is only written to disk when the file is closed or updated. When selected, the option Auto File Update will do an automatic update of the file after each series.

Convert To Native: Converts the raw data of the opened experiment (the *.dat file) into the native format of the target machine (Intel format = Little Endian or Motorola format = Big Endian or Intel format). It is usually not necessary to convert the data into the native format, since all versions of *POTPULSE* (PPC, 68k and Intel) are able to read files generated on either computer platform.

Note: When closing a file after modification, the Stim and the Pulsed Tree will be completely written in the format of the target machine. Raw data, however, can originate from two different platforms, therefore it is possible that the internal data format varies from sweep to sweep.

Page Setup... : Calls the Printer/Page Setup dialog of the operating system.

Page Margins... : Calls a dialog to set the page margins (left, right, top and bottom) and the font for printing. These settings apply for both printing data and printing the Notebook.



Text Font: Courier New
Left Margin: 0.125
Right Margin: 0.05
Top Margin: 0.05
Bottom Margin: 0.05
[✓] Text Wrapping

Print Notebook... : Prints the Notebook content. If a text section is selected, i.e., highlighted, only that text section is printed.

Quit: Exits *POTPULSE.*

File Selectors (MacOS only): There are two general MacOS file selectors, one for the selection of existing files and one to enter new file names. *POTPULSE* has an additional option, which allows to select both, existing and non-existing files. Therefore, the selector has an additional Create button. The three file selectors used by *POTPULSE* function as follows:

- The standard Open file selector to select a file, which must exist.
- The standard New file selector to enter a name for a new, i.e., non-existing, file.
- The modified Open file selector with the Create option is presented when the user has to select a file, which is expected to replace an old one. If there is no such old file (or if you want to save it in another file), click on the Create button and enter the name as usual.

Edit Menu

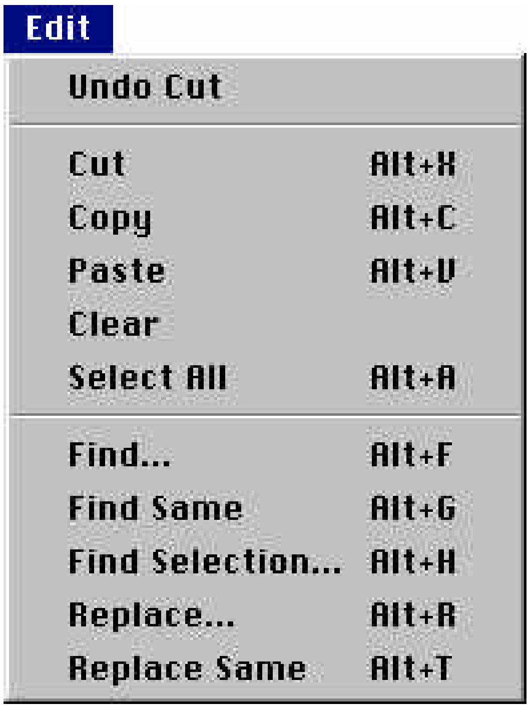
The Edit menu applies to text manipulation in the *Notebook* window (see Chapter *Notebook*).

The menu is disabled unless the *Notebook* window is in front. The menu entries conform to the typical functions of the actual operating system (MacOS or Windows).

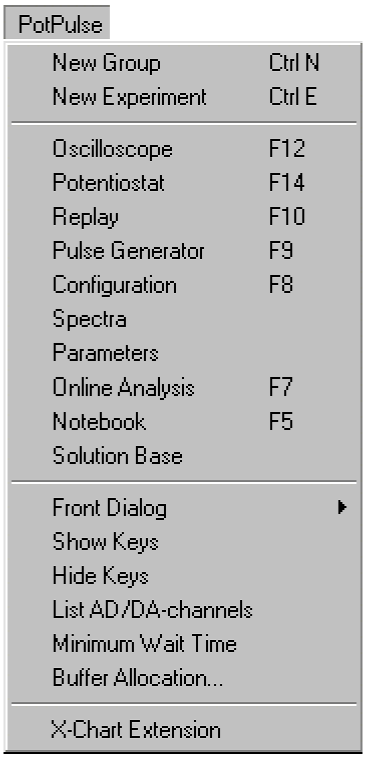
The Cut, Copy, and Paste commands copy the text selection to and from the clipboard. Clear removes the text. Select All selects the whole notebook.

Find... finds the entered search string, Find Same finds the next appearance and Find Selection... finds the search string that was marked (highlighted) in the *Notebook* window.

Replace... replaces the entered search string by some new string, and Replace Same finds and replaces the next appearance of it.



POTPULSE Menu



The Potpulse menu is the key menu to all windows of the *POTPULSE* application.

New Group: Generates a new Group in the output Data tree of the *Replay* window if the file is opened without write protection. After the addition of a new group a file update is automatically performed.

New Experiment: Generates a new Experiment (see above) and increments the experiment number.

Oscilloscope: Selects the *Oscilloscope* window.

Potentiostat: Selects the *Potentiostat* window.

Replay: Selects the *Replay* window.

Pulse Generator: Selects the *Pulse Generator* window.

Configuration: Selects the *Configuration* window.

Spectra: Selects the *Spectra* window.

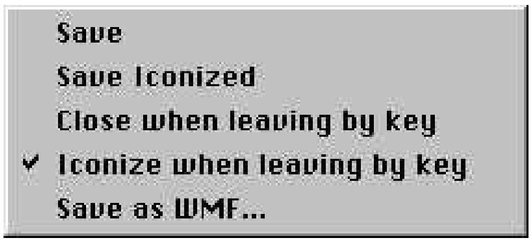
Parameters: Selects the *Parameters* window.

Online Analysis: Selects the *Online Analysis* window.

Notebook: Selects the *Notebook* window.

Solution Base: Selects the *Solution Base* base.

Front Dialog: Items in dialogs can be modified by the user (see Chapter *Dialog Controls*). These modifications may be stored. This submenu provides options for handling layout and display of the active window.



- **Save**: Stores the settings as resource file (e.g., “Default.Pulse_OsciDialog”) which are installed upon next restart of *POTPULSE*.
- **Save iconized**: Stores the settings as resource file but in iconized form (i.e., upon next restart of *POTPULSE* the window will appear iconized)
- **Close when leaving by key**: Closes the window when the dialog is exited using ESC or the close box is clicked.
- **Iconize when leaving by key**: Hides the window but leaves the window bar on the desktop.

- **Save as Pict** (Mac) **or Save as WMF** (Windows)...: Saves the window dialog as MacOS Pict file or as Windows Meta File (WMF).

Show keys: Shows the shortcuts (keys) behind the menu entries.

Hide keys: Hides the shortcuts (keys) behind the menu entries.

List AD/DA-channels: Prints the AD/DA assignments to the *Notebook*.

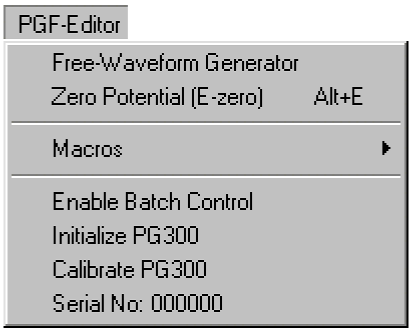
Minimum Wait Time: Opens a dialog window where the wait time as the time between pulses can be set. The actual wait time is displayed. Clicking on the minimum wait row permits a new wait time setting. During this time, X-Chart functions are executed, provided time is available. The minimum wait time defines how much time must be available. An entry of 0 ms will ensure that these functions are executed at least once inbetween sweeps. The drawback is that time overruns may occur when using closely spaced stimuli. The poll time increment technically sets the time between two reading cycles of the Windows event loop and was introduced to omit feasible freezing times. You can set both minimum wait time and poll time increment to the same value (as in the default setting).

Buffer Allocation: The acquisition buffer size is selected in this menu. Sample number (minimum 16000) can be enlarged and the required RAM acquisition buffer size can be adjusted, where it is noted that 48 bytes per sample is needed.

Note: The larger the acquisition buffer, the more time must be spent to process the data between sweeps. Virtual memory may be dangerous. Usually, it seems to work, but the page swapping increases the chance of getting a FIFO-overrun. Acquiring faster than effective 100 kHz will highly increase the probability of getting a FIFO-overrun. Moreover, one can effectively analyze data only with a program which has the same (or larger) acquisition buffer size.

X-Chart Extension: This activates options available within the X-Chart extension, which is a software implementation of a multi-channel chart recorder (see X-Chart Manual for details).

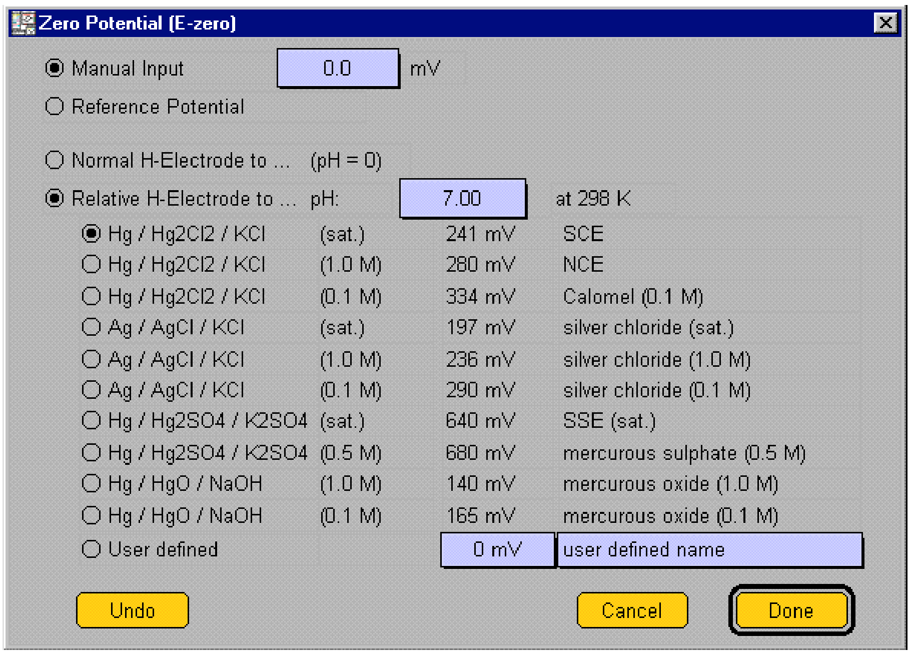
PGF - Editor Menu



The Pulse Generator drop-down menus provide functions to handle the potentiostat dialog, macros, and special functions (see Chapter *PG310/390 Potentiostat/Galvanostat*).

Free-waveform Generator: Selects the *Pulse* (also called Free-waveform) *Generator* window.

Zero Potential (E-zero): The reference electrode provides a fixed potential which does not vary during the experiment. In many cases, it is necessary to relate the potential of the reference electrode to other scales. The Zero Potential option allows selection of the desired potential scale. Clicking on Zero Potential will open a dialog window, which allows selection among the following options:



[✓ Manual Input] 0.0 mV

Reference Potential

Normal H-Electrode to ... (pH = 0)

[✓ Relative H-Electrode to ...] pH: 7.00 at 298 K

[✓ Hg / Hg2Cl2 / KCl]	(sat.)	241 mV	SCE
Hg / Hg2Cl2 / KCl	(1.0 M)	280 mV	NCE
Hg / Hg2Cl2 / KCl	(0.1 M)	334 mV	Calomel (0.1 M)
Ag / AgCl / KCl	(sat.)	197 mV	silver chloride (sat.)
Ag / AgCl / KCl	(1.0 M)	236 mV	silver chloride (1.0 M)
Ag / AgCl / KCl	(0.1 M)	290 mV	silver chloride (0.1 M)
Hg / Hg2SO4 / K2SO4	(sat.)	640 mV	SSE (sat.)
Hg / Hg2SO4 / K2SO4	(0.5 M)	680 mV	mercurous sulphate (0.5 M)
Hg / HgO / NaOH	(1.0 M)	140 mV	mercurous oxide (1.0 M)
Hg / HgO / NaOH	(0.1 M)	165 mV	mercurous oxide (0.1 M)
User defined		0 mV	user defined name

- **Manual Input** — Activating this option allows typing of a user-defined reference potential in mV, e.g., the potential of the Ferrocen/Ferrocinium pair. The short name of the actually tagged reference system will be stored as a comment.
- **Reference Potential** — Some of commonly used reference electrodes, which are related to the normal or relative hydrogen electrode, can be easily selected by tagging the radio button beside the desired reference system. The short name of the selected reference electrode will be written as a comment in the *Oscilloscope* window. The desciption of the reference systems shows the electrochemical chain, the concentration of the solution in mol/l (=M), the reference potential versus

normal or relative hydrogen electrode, and the respective short name. The User defined option enables the user of typing a user defined short name, which will be stored as a comment, and a reference potential.

Macros: Selects a submenu for macro functions.

- **Load**...: Loads a saved macro file.
- **Save**...: Saves a macro file.
- **List**: Prints the current macros into the Notebook.
- **Start Recording**: Starts macro recording. While recording a macro this function switches to Stop Recording, which stops macro recording.
- **Execute while recording**: If checked, the actions will be executed. Otherwise, the actions will be logged to the macro but no real changes will happen.
- **Execute**...: Executes a selected macro.

Enable Batch Control: Allows *POTPULSE* to be remotely controlled by another program (see Chapter *Appendix VI - Controlling POTPULSE*).

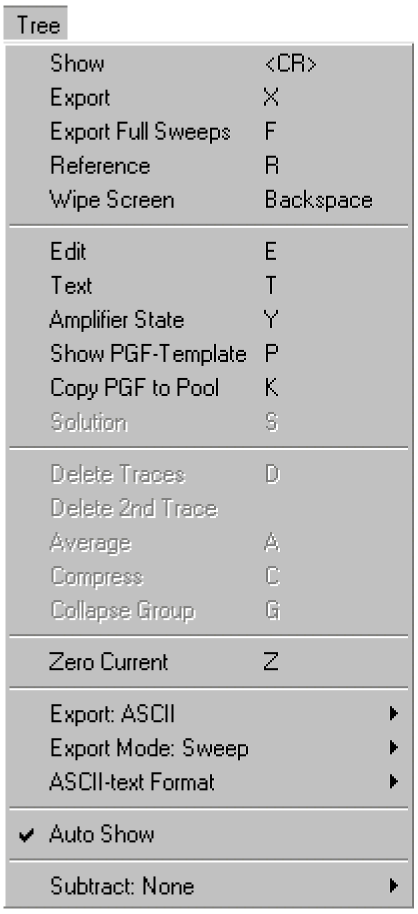
Initialize PG310/390: This is used to restart the AD/DA interface; e.g., in the case when *POTPULSE* was started with the interface being turned off.

Serial No.: While starting *POTPULSE* your instrument is identified by a serial number, which is exhibited as a six digit number. The board version is shown as a capital letter. “A” or “C“ means a PG 310 instrument (10V), and “B“ or “D“ means a PG 390 instrument (90V). For “A” or “C“ versions the 10 Hz setting of the Control Amplifier Bandwidth is available; the 300 kHz setting is unavailable. For “B“ or “D“ versions the 300 kHz setting of the Control Amplifier Bandwidth is available; the 10 Hz setting is unavailable.

Note: Your PG310/390 potentiostat/galvanostat will work properly only if *POTPULSE* can find the ”scale file” which belongs to the instrument. The “scale file” is named “SC[Serial#].PG3” (here SC280025.PG3) and located in a folder given by Common Path in the *Configuration* window.

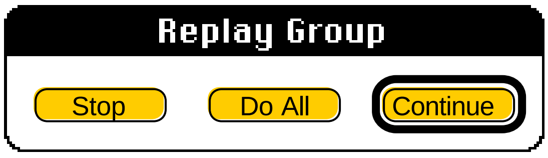
Tree Menu

The drop-down menu Tree provides functions that are active when the Replay window is selected. It allows to actually change the stored data, if the data file was opened with File → Open Modify.



Show: Displays the content of the selected target. Traces are displayed according to the settings specified in the Oscilloscope window.

If the target for the Show operation is a Group, the Replay Group dialog opens.



You will be asked to stop or to continue with the next series. Do All causes to go through all series of this group; it can be aborted by pressing CTRL B, CTRL S, or mouse click on the BREAK and STOP buttons in the *Oscilloscope* window. A corresponding dialog will come up, if the target is the Root or a group.

Export: Exports the content of selected target according to the **Export Format** and **Export Mode** (see below). This is the command you use to print traces or to output them in various formats. Export will be according to what is displayed in the *Oscilloscope*. Printing with Overlay all selected in the *Oscilloscope* window will print all sweeps superimposed. The printer will superimpose sweeps without consideration for changed amplifier gains. In such a case, multiple scaling labels will be superimposed as well (i.e. otput will be scaled in percentage of the amplifier range). To obtain a constant scaling in amperes, see option Fixed Scale in the Oscilloscope chapter.

Export Full Sweeps: Exports the full sweep as shown in the *Oscilloscope* window. However, display gain, filtering or zero-line subtraction will be applied. Every point of the data is exported.

Reference: Selects a target as Reference.

If the target is a *Sweep*, the reference sweep will be displayed in the background. It can then be compared to any other displayed sweep. The scaling of the screen is set according to the actually displayed Sweep, not according to the Reference Sweep. Thus, if the reference sweep is 10 ms long and the next displayed sweep is twice as long, the reference sweep is going to be compressed such as to match the new scaling of the time axis. The reference sweep is deactivated by turning the Background Trace in the *Display* menu off.

If the target is a *Series*, the results of the last Online Analysis of this series is shown as Reference Analysis. The scaling of the graph reflects the extreme values of both, the reference and actual series analysis (unless the Fixed Scaling option of the *Online Analysis* window is used). The reference series can be turned off by pressing the highlighted REF button in the *Online Analysis* window.

Wipe Screen: Clears the *Oscilloscope* screen.

Edit: Various functions for modification of entries of the Tree. For details see Chapter *Replay*.

Text: Allows editing text of selected target. This can be the Root Text, the Group and Series comment, or the Sweep label.

Amplifier State: Prints the content of the PG310/390 State Record (which reflects the hardware status for the current series) into the Notebook window.

Show PGF Template: Displays the stimulus protocol of a selected sweep or series.

Copy PGF to Pool: Copies the stimulus protocol of a selected sweep or series into the current Pulse Generator File.

Solution: Allows editing the solution stored for the current series (only active if the file is *not* write protected).

Delete Traces: Deletes all traces of target, i.e., 1st or 2nd Trace (only active if the file is *not* write protected).

Delete 2nd Traces: Deletes the 2nd Traces of target (only active if the file is *not* write protected).

Average: Performs averaging of target (only available if the file is *not* write protected):

- **Series**: Averages all sweeps of a series and stores them as series with one sweep. The sweeps have to have the same length.
- **Group**: It is assumed that all series within the selected group are of the same kind. One output series is created with the sweeps being the averages of all matching sweeps within the group (e.g., all first sweeps are used to generate the new first sweep, all second sweeps generate the new second sweep).

Compress: Compresses all sweeps of the selected target by a factor of two by taking the mean of each two successive data points (only available if the file is *not* write protected). A warning is given when the number of data points in a pulse segment is not a multiple of two.

Collapse Group: Allows to move the sweeps of all series in a group into the first series (only available if the file is *not* write protected). This is typically used when one acquired many series with one single sweep and one wants to combine them into one series for easier online analysis. A typical situation arises when one needs the "Start Macro" to perform some action like setting specific amplifier gains for every Sweep.

Zero Current: : Opens a window where Left bound / Right Bound / Segment can be inserted to specify a section of the first pulse segment where the zero current will be recalculated. Normally the zero current is calculated automatically within the full segment. This means that artifacts in the baseline will cause the zero current to be incorrect. This feature can therefore be used to recalculate the zero current in a section of the baseline without artifact. If there is no such section left, one can edit the entry Zero Current in the Tree Tree (see Chapter *Replay*) by taking the zero current of the next or previous sweep, for example.

Export Format: **:** This determines the output device and the type of output to be created. Output is generated in the way the data are displayed in the *Oscilloscope* window; e.g., if the digital filter is on, filtered data are output.

Note : The Export option will try to keep a "what-you-see-is-what-you-get" behavior. This means that the display options define the export options; e.g., when Second Trace is selected as background trace, the leak and second traces are also exported. The Overlay flag defines whether the sweeps are automatically displayed as a graph. The difference between the normal Export function and the Export Full Sweep option is that the former exports the actual data as displayed on the main window including filtering or reference etc., whereas the latter exports the data by reference to the original data file as the entire sweep (not considering Start- and End-times).

At present, the following options are implemented:

- **Printer:** Direct output to a connected printer. The page setup magnification determines the line width; usually, 50% gives good results.

  When Export or Export Full Sweeps is choosen, the number of columns and rows per page have to be entered. This determines how many items are placed on a page. In any case, a form feed is output after the selected target is output. Thus, if one prints a group with 3 series to a page with 2 columns and 2 rows, three quarters of the page will be filled, then the page is released from the printer. If one wants to have individual sweeps rather than a complete series plotted in the page sections, one has to turn off Overlay in the *Oscilloscope* window.

  Note: Remember that the *Replay* window has to be selected to use the Export function.

- **Log Book**: The information stored in the respective branch of the *Tree* is written to an ASCII file. Each entry is identified by an ASCII string. This is intended to replace (or at least complement) a conventional notebook.

  Example:

```
Group #   1: E-2

Series # 1:

WholeCell PipPot =-240.0 mV  CellPot =   0.0 mV  Bandw. = 10.00 kHz

Temp =  20.0 C  Sample Time = 10.00 ms

Temp =  20.0 C  Sample Time = 10.00 ms Ext. Solution

Sweeps :

#  Label   N   V( 2) [mV]   T( 2) [ms]   Anal.Value

1    1    -240.0       12000.00  2.29094E-003
```

- **ASCII**: Sweeps are output as columns of ASCII numbers representing time and current (both in the scientific format). Each sweep and series starts with an identifier.

  Note: This may create huge ASCII files if the output target is a group, for example. The separator can be modified (space, comma, or tab separators) by using the ASCII-text Format option in the Tree menu.

  Example:

```
Sweep 1_1_1
2002 points
"time[s]",   "trace1 [A]",   "trace2 [V]"
 1.95000E+000   6.56250E-006  -3.56250E-002
 1.96000E+000   8.43750E-006  -3.53125E-002
 1.97000E+000   6.56250E-006  -3.37500E-002
 1.98000E+000   8.75000E-006  -3.37500E-002
 1.99000E+000   7.50000E-006  -3.18750E-002
 2.00000E+000   7.81250E-006  -3.25000E-002
 2.01000E+000   7.18750E-006  -3.00000E-002 ...
```

- **PICT** or **WMF**: Sweeps are exported as Pict (under MacOS) or WMF (under Windows). Because the Pict files are limited to 32 kbytes, each file contains only a single sweep (plus second trace and other options). When a series is output, the sweep files are generated automatically with the same name convention as waves for *IGOR* files: indices of “Group_Series_Sweep” are appended to the name.
- **IGOR Text**: Export of sweeps as ASCII waves in “IGOR Text” format for the analysis and display program IGOR. Each wave is identified by indices “Group_Series_Sweep” (e.g., “Name2_4_3”).

  Example:

```
IGOR
WAVES PotD1_1_1
BEGIN
 6.56250E-006
 8.43750E-006
 6.56250E-006
 8.75000E-006
 7.50000E-006
 7.81250E-006...
```

If the file name starts with a number, a “W” is placed in front of it, because in IGOR waves are not allowed to start with a number. The created files have the extension “ITX”(Windows) resp. “IGO” (Mac) and are recognized by *IGOR*, i.e., double-click on this file will start *IGOR*, load and display the file content (not for Sweeps).

The waves will immediately be displayed in *IGOR* only if a series or group was exported, and the Overlay or Overlay All option was selected in the *Oscilloscope* window during data export.

Otherwise, the sweeps will be loaded, but must be displayed by IGOR's “Display Wave” command. To export the stimulus pattern, select the Show Stimulus option in the Display drop-down menu.

When loading IGOR Text output files, do not use the General Text import option in IGOR; always use the Load...IGOR Text option.

- **IGOR Layout**: Export of sweeps as ASCII waves in "IGOR Text" format, arranged on an *IGOR* layout page. The sweeps are displayed in *IGOR* graphs as they appear in *POTPULSE* in the *Oscilloscope* window. The created files have the extension " *.igl" and are recognized by *IGOR*, i.e., double-click on this file will start *IGOR*, load and display the file content.
- **IGOR Info**: Export of pulse protocols as ASCII waves in "IGOR Text" format. There are two waves generated per sweep: "Amp" and "Dur", concatenated to the sweep identifier. The created files have the extension " *.inf".

  Example:

```
IGOR
X | C:\HEKA\Data\PotDemo.dat

X | SWEEP
X | PotD1_1_1
X |  6.45576E+004; Sweep Time
X |  1.00000E-002; Sample Interval
X |  3.12500E-007; Amperes per ADC-unit
X |         2400; Total Points
X |            0; Bytes from beginning of file
X | A           ; Y-unit
X | -2.40000E-001; Holding Voltage

X | TRACE2
X | PotD1_1_1_2nd
X |  3.12500E-004; Amperes per ADC-unit
X |         4800; Bytes from beginning of file
X | V           ; Y-unit

WAVES PotD1_1_1_Dur PotD1_1_1_Amp
BEGIN
 0.00000E+000   -2.40000E-001
1.19800E+001    9.60000E-001 ...
```

- **IGOR Binary**: Export of sweeps as "IGOR binary" waves. A folder is created into which all binary waves are written. In addition, an *IGOR* macro file is written. It loads the waves into *IGOR* upon double-click. This macro file has the extension **"***.itx**".** The folder name is the command file name without file extension followed by ".f" for "folder". When you want to import data from within IGOR, use the option Load...IGOR Text. to load the macro file. Use the option Load...IGOR Binary only when you want to explicitly load one of the generated IGOR binary waves (file extensions *.ibw or *.bwav).

Note: It is much faster to work with "IGOR Binary" than with "IGOR Text" and the created files are considerably smaller.

When using the Export Full Sweeps option as *IGOR Binary*, the data are not really exported as waves inside a folder. The function will instead generate only a small command macro that uses the "GBLoadWave" IGOR extension to read the data directly from the *POTPULSE* raw data file, i.e., the ". dat " file.

- **IGOR for MacOS:** Choose this, if the IGOR processing will be under MacOS.
- **IGOT for Windows:** Choose this, if the IGOR processing will be under Windows.
- **Print compressed Vectors:** This option compresses the data without information loss by compressing the vectors. As when displaying on the screen, vectors with identical x-coordinates are compressed into one vector. For instance, if a sweep with 1000 points is drawn in a window with a width of 100 pixels, there will be 10 vectors to a pixel.

Export Mode: Shows a submenu which determines whether Sweeps, the results of the Online Analysis, or both are to be exported.

- **Sweep Data**: Only the sweep traces are exported.
- **Online Analysis**: Only the online analysis data are exported.
- **Both**: Traces and online analysis data are exported.

ASCII-text Format: Shows a submenu for selecting the format of the exported text.

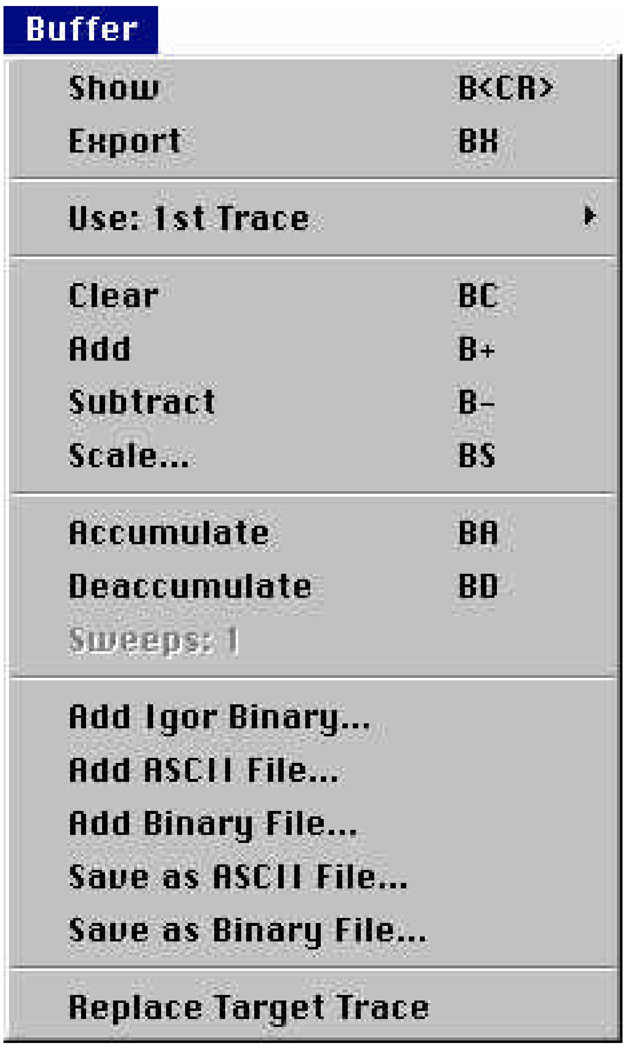
- **Space/Comma/Tab Separator**: Specifies how values are separated.
- **Include Headers**: If checked, a header which specifies various parameters of the exported data will precede the actual values.
- **MacOS Format (LF only):** Lines are terminated by line feeds.
- **Windows Format (CR+LF):** Lines are terminated by carriage returns and line feeds.

Auto Show: If this flag is on, sweeps are shown in the *Oscilloscope* window as soon as the corresponding sweep target in the *Tree* window is highlighted; RETURN is not required.

Subtraction Mode: This allows to perform a subtraction of the selected subtraction source from the target data.

- **None**: No subtraction.
- **Buffer**: The Buffer content is subtracted.
- **Reference Sweep**: The Reference Sweep is subtracted.
- **Reference Series**: The Reference Series are subtracted, i.e., corresponding Sweeps from the two Series are subtracted: the 1. Sweep of the Reference Series is subtracted from the 1. Sweep of the selected Series in the Tree, then the 2. Sweep of the Reference Series is subtracted from the 2. Sweep of the selected Series, etc.

Buffer Menu



This menu provides various functions for handling the Sweep Buffer. The buffer is automatically activated whenever a buffer command is executed. No assumption is made about the Sample Interval of data in the buffer; i.e., the sample interval (and all other parameters) of the sweep presently selected in the Tree is taken. The command interpreter keeps track of the absolute current sizes, however. The executed buffer commands are written as text lines to the "Notebook" so one has a record of what operations were performed with a given buffer.

Note: The displayed sweep data are used to compute the buffer sweep, i.e., filter setting affect the result. The buffer sweep is displayed using display Gain 1 and Offset 1. The user can offset the sweep (if it is hard to see it overlayed by a control sweep, for example) by adding an offset using the Buffer Scale option.

Show: Shows buffer content.

Export: Exports the buffer according to the settings from the menu Tree → Export. Thus, the user can transfer the sweep data to the sweep buffer and save it as ASCII file. Then he can load that text file in a text editor, perform the required modification, and store the text back to disk. Finally, he can load this modified file back into *POTPULSE* by using the menu option Buffer → Add.... This procedure can be useful, e.g., to edit a recorded sweep.

Use: Shows a submenu for selection of the trace to be used:

- 1st Trace
- 2nd Trace

Clear: Sets buffer to zero.

Add: Adds sweep to buffer.

Subtract: Subtracts sweep from buffer.

Scale... : Scales buffer content by a factor plus offset.

Accumulate: Adds a sweep to buffer and divides by number of sweeps accumulated. This averages sweeps in the buffer.

Deaccumulate: Subtracts a sweep from the buffer and divides by the number of sweeps remaining.

Sweeps: Number of sweeps currently accumulated in the buffer.

Add Igor Binary…: Adds an IGOR binary file (wave) to buffer.

Add ASCII File…: Adds an ASCII file to buffer (sequence of real values in absolute values (i.e., Amperes).

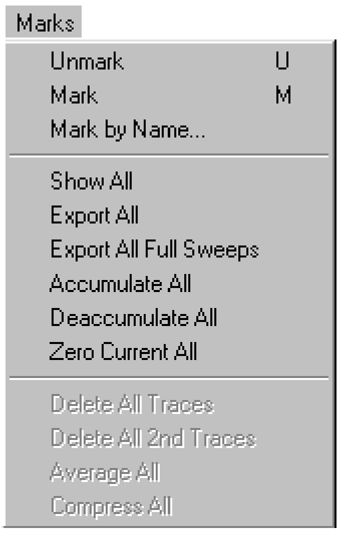
Add Binary File…: Adds a binary file to buffer.

Save as ASCII File…: Saves buffer to ASCII file.

Save as Binary File…: Saves buffer to binary file.

Replace Target Trace: Replaces the data in the data file with the content of the buffer. This option functions only when the selected target is a Sweep and the file has been opened with write permission.

Marks Menu



This menu is used for tagging Tree items for later analysis or buffer manipulations. All of the routines can be interrupted by click on the Break or Stop button in the main window or typing 'CTRL' + 'B' or 'CTRL' + 'S'.

Unmark: Removes the tag from the Tree target.

Mark: Attaches a tag to the Tree target.

Mark by Name...: Attaches a tag to a series which you have to name in a dialog window. If more than one series in this group with the specified name exist, every series with this name will be tagged.

Note: The name of the series is context sensitive.

Show All: Displays all marked entries.

Export All: Exports all marked entries. Only the part of the sweeps visible in the *Oscilloscope* window will be exported.

Export All Full Sweeps: Exports all marked entries. The whole sweeps will be exported not only the part visible in the *Oscilloscope* window.

Accumulate All: Builds the average of all marked entries to the buffer.

Deaccumulate All: Subtracts all marked entries from the buffer. Thus, this routine allows to compute the difference between two averages, the first being build by the Accumulate All option.

Zero Current All: Computes the zero current of all marked entries.

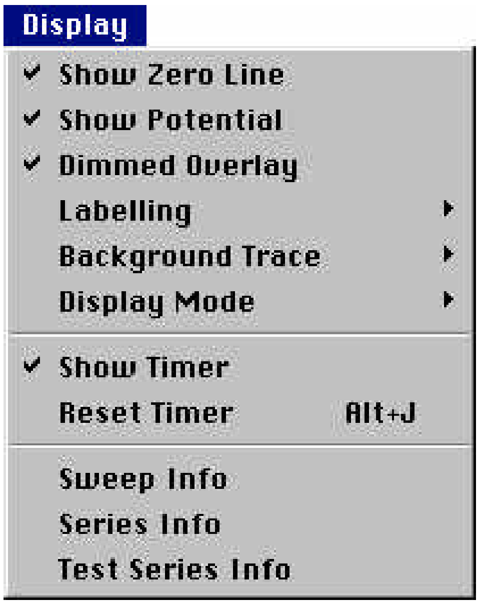
Delete All Traces: Deletes all traces and sweeps of all marked entries (only available when the file is *not* write protected).

Delete All 2nd Traces: Deletes the second traces of all marked entries (only available when the file is *not* write protected).

Average All: Averages all sweeps in a marked series or all series in a marked group (only available when the file is not write protected). The original sweeps will be replaced by this average. It is identical to the one found in the Tree menu, but will operate on all marked items.

Compress All: Compresses all sweeps in a series or all series in a group by averaging two adjacent data points (only available when the file is not write protected). Each compression reduces the data points by half. The original sweeps will be replaced by compressed sweeps. It is identical to the one found in the Tree menu, but will operate on all marked items.

Display Menu



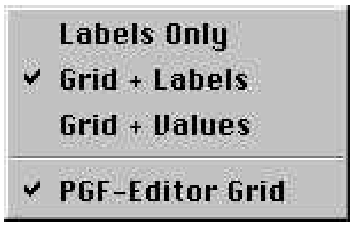
This menu sets some parameters for the display of data in the *Oscilloscope* window.

Show Zero Line: Draws a reference zero line.

Show Potential: Displays the stimulus template in the background.

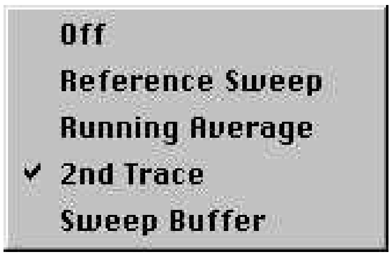
Dimmed Overlay: Turns the Dimmed Overlay mode on or off. If on, dense filled areas are grey.

Labelling: Determines the labels in the *Oscilloscope* window and the Pulse Generator preview.



- **Labels Only**: Draws calibration bars.
- **Grids + Labels**: Draws a grid and units/division.
- **Grids + Values**: Draws a labelled grid.
- **PGF-Editor Grid**: Draws a grid in the preview section of the *Pulse Generator* window.
- **No math on Grid-Values:** Option for the logarithmic display of data. When activated, the label on the x-axis will be displayed as absolute value, not as exponent only, e.g. 1ua , 10uA 100uA instead of – 6 ,-5, -4.

Background Trace: Shows a submenu for selection of a background trace.



- **Off**: No background trace.
- **Reference Sweep**: Draws a trace marked as reference.
- **Running Average**: Draws the running average of acquired sweeps.
- **2nd Trace**: Draws the second trace.
- **Sweep Buffer**: Draws the buffer contents.

Display Mode: Shows a submenu for selection of a display mode.

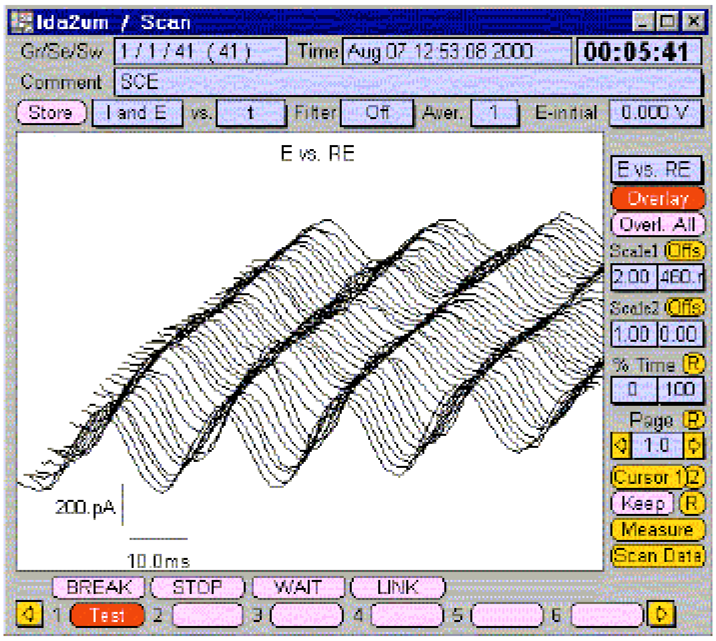
- **I vs. t**: Plots current versus time.
- **I vs. V**: Plots current versus measured voltage (i.e., 1st trace versus 2nd trace where the second trace is assumed to be the voltage trace).
- **I vs. V-ramp**: Plots current versus measured voltage during ramp segments. It is assumed that the second trace is the voltage trace.
- **I vs. V-ramp,theo**: Plots current versus theoretical ramp voltage.
- **I vs. sqrt(t)**: Plots current versus the square root of time.

3D Mode: Shows a submenu for selection of the pseudo three dimensional graphing. The 3D mode is useful for experiments where you want so see changes between a series of sweeps, e.g.:

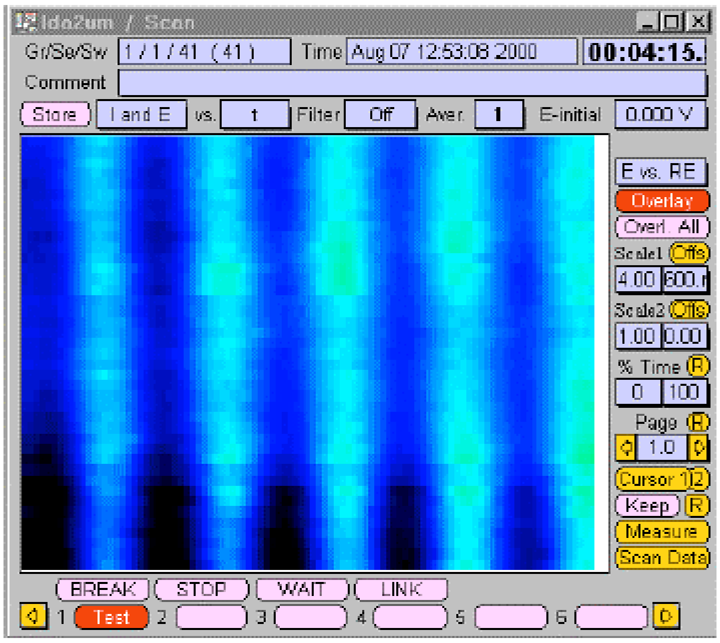
- Surface scans made with the SECM extension of POTPULSE. Normally a SECM scan will be made by positioning the electrode at a starting y-position and scanning in x-direction. For the next sweep the electrode will be positioned to the next y value and the second scan in x-direction will be done, etc., thus rastering the surface until a complete height profile is sampled.
- Peak changes in a multi-sweep experiment, e.g., on polymerizing monomers electrochemically. To display this you would probably choose "I vs t" with a dX value of "0".
- Peak changes - or none! - in a multi-sweep experiment for controlling the long time reproducibility of (electro-)chemically reversible electron transfers by the height of the first wave. Note that this is only a visual check, and use the Online Analysis for more complex correlations.

The following settings can be made:

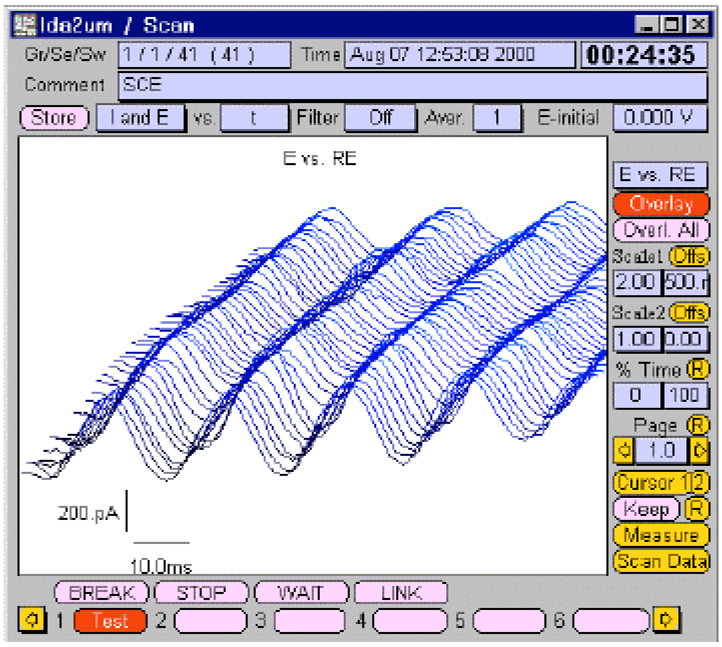
- **3D-Graph**: **Enter dX and dY**: Allows to specify the horizontal and vertical offset of subsequent sweeps. You can also enter "0" for one or both. Note that the 3D-Graph feature has to be On for this option (see below).
- **3D-Graph**: **On**: The results are displayed in black and white in pseudo three dimensional mode by displaying subsequent sweeps with a horizontal and vertical offset. You have to enter this offset as dX and dY (see above).


Ida2um / Scan
Gr/Se/Sw 1 / 1 / 41 (41) Time Aug 07 12:53:08 2000 00:05:41
Comment SCE
Store I and E vs. t Filter Off Aver. 1 E-initial 0.000 V
Scale1 Offs 2.00 460.r
Scale2 Offs 1.00 0.00
% Time R 0 100
Page R 1.0

- **3D-Graph: Color**: The results are displayed in pseudocolors leading to a two-dimensional height profile. You can set the color selection and the contrast by changing the values in Scale1 and Offset.



- **3D-Graph: On and 3D-Graph: Color active:** The results are displayed in pseudocolors and 3D mode to give the impression of contour lines, thus leading to a 3D height profile.



Show Timer: Enables the timer in the *Oscilloscope* window. The timer is displayed top right in the *Oscilloscope* window. The timer starts with each new *POTPULSE* start. The timer value at time of acquisition is stored in the series parameter block and is recalled during series replay.

Reset Timer: Resets the timer in the *Oscilloscope* window to zero.

Sweep Info: Toggles output of sweep parameters to *Notebook* window.

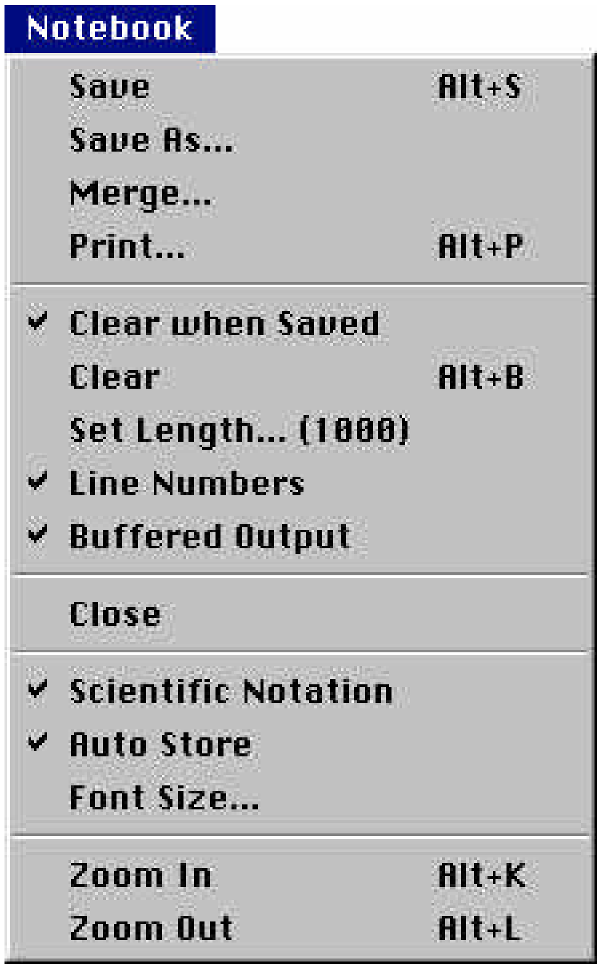
Series Info: Toggles output of series parameters to *Notebook* window.

Test Series Info: Toggles output of test series parameters to *Notebook* window.

Current Density: Either the current flowing through the cell or the current density can be displayed in the *Oscilloscope* window. If this option is tagged, the current density in [A/cm^2] is displayed. Instead of showing the unit A/cm^2, a A' (A prime) is plotted as the current density unit for avoiding overwhelming text in the display. The current density is calculated with the use of the Electrode Area specified in the *Oscilloscope* window.

Notebook Menu

The Notebook keeps track of information about the experiment. The ASCII-table separator setting of the Tree menu is used for the Notebook as well. This enables to directly “cut-and-paste” to spreadsheets which require a Tab separator, such as Microsoft Excel. The options in the Notebook menu are:



Save: Saves the Notebook under its default name: “Notebook_[Date].txt”.

Save as...: Asks for a filename before saving.

Merge...: Merges a text file to the content of the Notebook.

Print...: Outputs content of Notebook to a printer.

Clear when Saved: Automatically clears the Notebook after the present content is saved to disk.

Clear: Clears the *Notebook* window.

Set Length...: Specifies maximal number of text lines in the *Notebook* window. The maximal number of lines is given in parentheses.

Note: Large notebook buffers require a lot of CPU time for text handling. If execution time during acquisition is an issue, the buffer size should be kept small or the Buffered Output should be turned off. RAM requirement is 260 bytes per line.

Line Numbers: Shows line and column numbers.

Buffered Output: Keeps all text written to the Notebook window in the Notebook buffer. If not selected, information displayed in the Notebook window will not be saved.

Close: Closes the Notebook window.

Scientific Notation: If set, the results of the online analysis are written to the Notebook in scientific notation (e.g., 1.23e-12). The default is engineering format (e.g., 1.23p). The scientific notation is mostly used when the user wants to copy results from the Notebook to a spreadsheet program by copying to the clipboard.

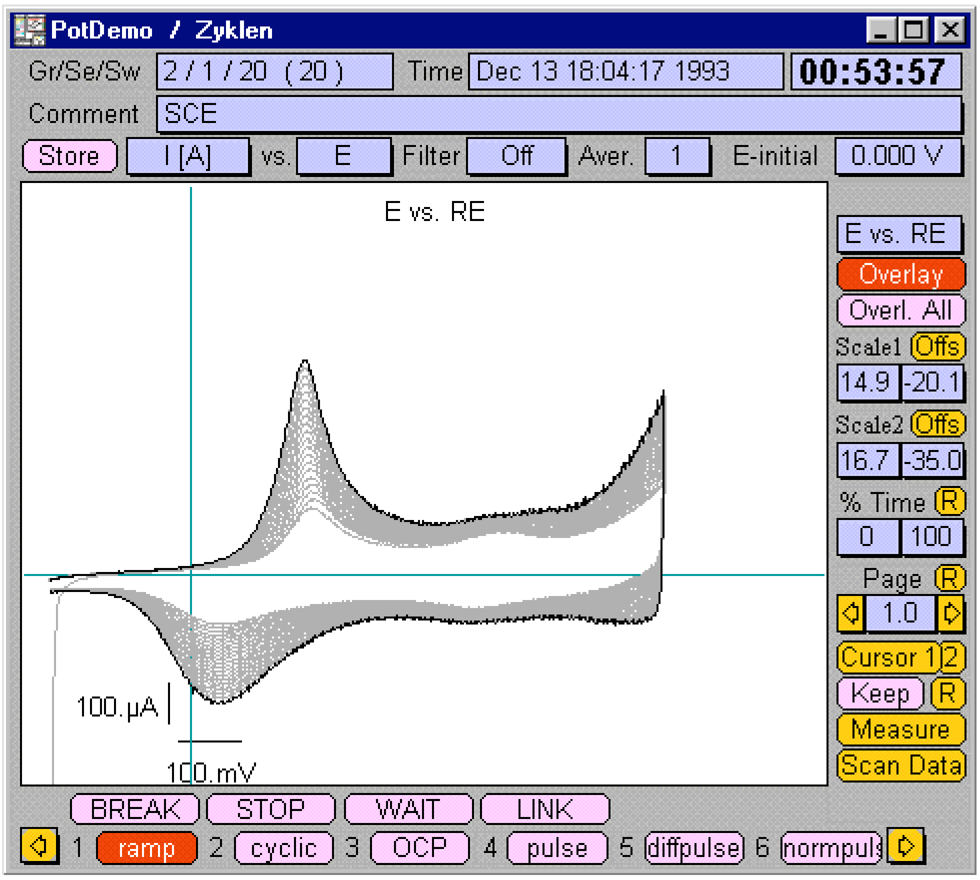
Auto Store: This option will automatically store the notebook together with the data file (“ [data file name].txt“). Upon opening a data file, its Notebook file will automatically be loaded as well.

Font Size…: Allows to select the font and font size of the Notebook text.

Zoom In / Zoom Out: Expands and shrinks the *Notebook* window.

Oscilloscope

The *Oscilloscope* window is mainly used for monitoring the data. Controls for display scaling and data handling are provided. The title of the window contains the information on the currently active data file and the currently active series. The Oscilloscope window can be wiped with the BACKSPACE key on your keyboard.



PotDemo / Zyklen

Gr/Se/Sw 2 / 1 / 20 (20) Time Dec 13 18:04:17 1993 00:53:57

Comment SCE

Store I [A] vs. E Filter Off Aver. 1 E-initial 0.000 V

Scale1 Offs 14.9 -20.1

Scale2 Offs 16.7 -35.0

% Time R 0 100

Page R 1.0

Some controls of this window serve a dual role. Time, Comment, Display Mode, Filter, Average, and E-Initial are used to control the experiment and to display the corresponding values of replayed data. During data replay these controls are replaced by the information of the replay data. As soon as the *Replay* window is deactivated (by clicking in any other window), the current values of the experiment will be restored.

It is possible to set the display scaling in the *Oscilloscope* window by "lasso-ing" a screen region while pressing the left mouse button. When you release the mouse button, the marked area will be set to fill the oscilloscope screen. Selecting while the 'OPTION' (MacOS) key or the right mouse button (Windows) is pressed will only change the display gain of the second trace (the X-scaling is not changed).

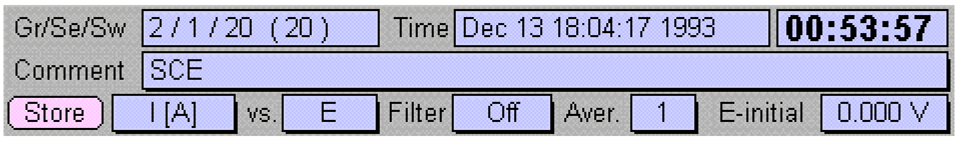
If you want to display data that is outside the active screen area, you can either enter some values into Scale1 or Scale2 or drag the mouse *outside* the active screen area. The display gains get reduced by 20% as long as the mouse is outside, and stops when it moves back on the active display area. When the mouse is below or above the screen, the X-scaling is changed, and when the mouse is to the left or right of the screen, the Y-scaling is changed.

All data outside the relevant Y-segment (Rel Y Seg button in the Pulse Generator window) are redrawn in dimmed color, when the Cursor function is called in any I vs. V display mode. This allows to unequivocally correlate which data belong to the relevant segment. This option requires the Dimmed Overlayto be selected, and only the ramp segments will be plotted.

It is possible to label sweeps during acquisition. Pressing 'Option' (MacOS) or 'ALT' (Windows) + '1'...'9' during the acquisition will mark the sweep which is going to be saved next. The labels are shown as the digits [1...9] in the Sweep Label in the *Replay* window. 'OPTION' / 'ALT' + '0' deletes a pending mark.

To accommodate different user preferences there are multiple cursor shapes available. Press the 'CTRL' key while the "cross" cursor is displayed (when setting cursors or selecting the Measure option), and you can cycle through all available cursor shapes.

Information about the Experiment


Gr/Se/Sw 2 / 1 / 20 (20) Time Dec 13 18:04:17 1993 00:53:57
Comment SCE
Store I [A] vs. E Filter Off Aver. 1 E-initial 0.000 V

Gr/Se/Sw (Group/Series/Sweep): Currently active group, series and sweep number within the Data Tree (visible in the *Replay* window). The total number of sweeps per series is given in parentheses. During an averaging record of a sweep, the displayed recording step is separated by a minus sign. (see below)

Time: Date and time of series (or test pulse) execution, e.g., actual system time for new experiments or original execution time for replays.

Timer: This item functions as a stopwatch or timer. It can be reset at any time by clicking on the Timer or pressing 'CMD' + 'J' (MacOS) or 'ALT' + 'J' (Windows) and may be useful to keep track of the experiment (e.g., to monitor the time spent in stand-by). The Timer is updated also during the series execution. The timer value is stored at the beginning of the sweep acquisition. The internal Timer tick corresponds to 1 ms.

Comment:: Comments to the currently active series. This field can be edited. It will result in a modification of the text of the present series, if the file was opened with write permission. The comment has to be entered to a series *after* it has been acquired. The new comment will be copied into all incoming new series until a new text is entered.

Store: This switch is used to enable or disable storing of data.

Note: There is no way to retrieve data that were acquired without the STORE control being on! If you are uncertain of whether to keep acquired data or not, just keep STORE on and remove unwanted data from the Tree later.

[ ] vs [ ] (Recording Mode): Displays the recording mode. The ordinate and abscissa can be selected separately to ensure the most convenient plot of the data. For the ordinate (x axis) you have these possibilities:

•	**I and E**	- The current density (first trace) and the cell voltage (2nd trace) are recorded.
•	**I [A]**	- current density recording
•	**E**	- cell potential recording
•	**Q**	- electric charge recording
•	**1/I and 1/E**	- inverted current density and potential recording
•	**1/I**	- inverted current density recording
•	**1/E**	- inverted potential recording
•	**1/Q**	- inverted charge recording
•	**ln(abs(I,E))**	- logarithmic recording of the current density and the potential
•	**ln(abs(I))**	- logarithmic recording of the current density
•	**ln(abs(E))**	- logarithmic recording of the potential
•	**ln(abs(Q))**	- logarithmic recording of the electric charge
•	**log(abs(I,E))**	- logarithmic (base 10) recording of the current density and the potential
•	**log(abs(I))**	- logarithmic (base 10) recording of the current density
•	**log(abs(E))**	- logarithmic (base 10) recording of the potential
•	**log(abs(Q))**	- logarithmic (base 10) recording of the electric charge

Note: With respect to the electrochemical notation in use, the cathodic current is negatively counted and the anodic current positively. Thus, the total charge will be negative, if the cathodic charge is greater than the anodic charge, and vice versa.

Five selections can be made for the abscissa:

- **t** - time for transient recording (e.g.: current-time transient) or the uptake of chronopotentiograms
- **E** - the cell potential (e.g.: in voltage sweep technique or cyclic voltammetry)
- **E-ramp** - measured ramp potential
- **E-ramp,. theo** – theoretical ramp potential
- **sqr(t)** - square root of the time (e.g. in chronoamperometry or chronocoulometry)

Filter: The currently selected bandwidth of a digital non-lagging Gaussian filter (i.e. software filter). The -3dB cutoff frequency is specified in Hertz. It must be larger than 0.01 times the sampling rate. The filter is used for *display purposes only*; no changes to the data are performed.

Note: Do not confuse this filter setting with the analog filter settings of the PG310/390.

Average: Only the average is stored to disk. During the data recording, the number of the individual acquiring steps is displayed in the Group/Series/Sweep field.

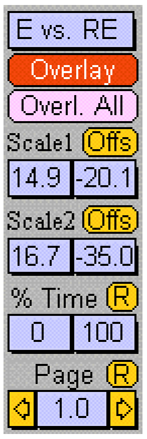
E-initial or **I-Initial**: The initial potential (in potentiostatic mode) or initial holding current (in the galvanostatic mode) as set in the *Potentiostat* window.

Overlay Options

Overlay: Displays all sweeps of a series without erasing the screen in-between sweeps. However, the next series will erase the screen.

Overlay All: Displays all incoming sweeps without erasing the screen. This allows to compare sweeps of different series. During data acquisition the screen can be wiped by pressing BACKSPACE.

Display Scaling



Potential Scale: The user can choose the appropriate potential scale in the oscilloscope display. Three options of the potential scale are provided:

- **E vs. RE** - The displayed cell potential is in accordance to the used reference electrode. That is, in the E vs. RE mode the shown potential is identical to the measured voltage between the reference and working electrodes.
- **E vs. HE** - The displayed cell potential is in accordance to the normal (NHE) or relative (RHE) hydrogen electrode. The offset of the potential scale is set in the Zero Potential window, which is opened via the drop-down menu PGF-Editor -> Zero Potential or 'ALT'+'E'.
- **E vs. OCP** - The displayed cell potential is in accordance to the measured open cell potential (OCP). Sometimes, the E vs. OCP potential is called polarisation or corrosion potential in corrosion experiments. If no OCP has been determined, the OCP value is set to 0 V.

Scale: For each trace (the current and the optional second trace) there is one control that determines the display scaling. The value of 1 corresponds to full scale of ±10.24 V. This scaling does not affect the display of test pulses.

'+' (from the numeric keypad, take care not to have 'NUMLOCK' activated!) increases the display scaling by a factor of 2 and '-' decreases it by a factor of 2.

'OPTION' + '+' and 'OPTION' + '-' (MacOS) or 'CTRL' + '+' and 'CTRL' + '-' (Windows) do the same for Scale 2.

Note: This display scaling does not affect the display of the test pulse current trace (the second, i.e., voltage trace can always be scaled). If you want to be able to scale the test pulse as well, activate the option Scale Test Pulse in the *Configuration* window. We do warn against activating that option because one can easily overlook that the amplifier gain is not correctly set, when the current trace is scaled by the display scaling.

Offset: For each trace (current and optional second trace) there is one control that determines the offset of the zero line. Offsets of traces may be between -1 and 1 relative to full scale of display (default = 0). 'SHIFT' + '+' and 'SHIFT' + '-' (from the numeric keypad) increase and decrease the display offset by 0.1. Pressing additionally 'OPTION' (MacOS) or 'CTRL' (Windows) do the same for the offset of the second trace. Option Shift + and Option Shift – (resp. CTRL Shift -/+) do the same for Offset 2.

Offs: Clicking on the Offs button (or '.' on the numeric keypad) automatically centers the respective trace on the screen.

The key command is '*' (from the numeric keypad) for the first trace, and '*' plus 'OPTION' (MacOS) or 'CTRL' (Windows) for the second trace.

Note: You can also zoom out a patch of the data display by dragging a rectangle with the mouse. Set the cursor at one corner of the desired section, press the mouse button (windows systems: left button) and hold it during dragging of the rectangle. After releasing the mouse button, the selected section will fill the whole display screen (for more features, see above).

% Time: Section of the sweep to be shown on the screen in % (*Start* - *End*). The reset button (R) sets the full sweep length (0 to 100 %).

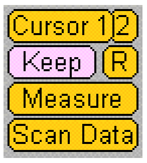
Note: The full time scale provided for sweep display is based on the longest sweep within a series. Alternatively, one can use the Fixed Scale option (see below).

Page: Page of display during replay of continuous data sweeps or when time axis is chosen to be less than 100%. Clicking on the right/left arrow control will display the next/previous page of the current sweep. Dragging the page number scrolls the data forward or backward; entering a page number will display that particular page. "Page" is highlighted whenever there is more than one page available. The reset button (R) resets display gains, display offsets, start and end times and is useful to quickly "undo" the performed scalings.

Cursors

Cursors: Two cursor ranges may be set independently:

- Cursor 1
- Cursor 2



Each cursor range is shown as two vertical lines. They are positioned as defined by Left Bound and Right Bound relative to the Relevant Y-Segment as specified in the Pulse Generator file. The values Left B. and Right B in the *Online Analysis* window are updated while the cursor lines are moved while dragging them with the mouse pointer. The cursor selection is terminated by clicking on the Cursors item once again or anywhere outside the trace window. One can also change the numeric values in the Online Window to define the cursor positions. The cursor bounds are used as search region for the online analysis functions.

Keep: This will keep the cursors displayed in the *Oscilloscope* window.

Reset (R) : This will reset the bounds of the cursor to 0% and 100%.

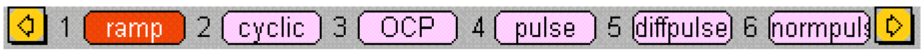
Measure: Two mouse-driven horizontal lines are provided to measure current differences. Point and drag the mouse for measuring the displayed data, which are continuously written to the *Notebook* window.

Scan Data: This option allows to scan the data points themselves. When selected, a marker is displayed on the trace. This marker can be moved forward and backward with the cursor key, while at the same time, the data values are displayed in the Notebook. Every time a cursor key is pressed, the speed with which the cursor moves over the trace is changed. Pressing twice the same cursor key increases the speed, while inverting the direction reduces the speed. The following commands are available:

Command	Action
Mouse clicks	clicking the mouse inside the display will set the cursor on the nearest data point; clicking outside the display aborts
‘SPACE’	stops the cursor if the cursor is moving, otherwise it moves to the next data point
'<' (or ',')	moves to the left
'>' (or '.')	moves to the right
‘RETURN’	stores the present data values in the notebook
‘ESC’	terminates the function
Cursor left	move left
Cursor right	move right
Cursor up	move fast to the left
Cursor down	move fast to the right
‘Z’	find the next zero crossing in the active direction
‘P’	find the next peak in the active direction
‘M’	find the next minimum in the active direction
‘1’ to ‘9’	defines the number of data points used for the running average of functions “z”, "p", and "m"

Note: In many instances, using heavy digital filtering (i.e. typing a high average value 7 or 9) improves the finding of zero current and peaks.

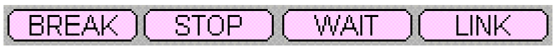
Controlling the Pulse Generator



Sequence Pool: Pool of available pulse templates. Six template controls are displayed at a time; paging is done by clicking on the arrows at the ends of the pool section. The keys ‘1’...’9’ are used to execute a template with that number. The number of a specific sequence with higher index than 9 can be entered after typing the pound key ‘#’. The highlighted sequence can alternatively be executed by typing ‘E’.

Note: Most of the key assignments are not fixed, i.e., the user can change them by modifying and storing the dialogs. In order to avoid confusion we do not recommend that you change already assigned controls like the ones above.

While a series is acquired, the cursor changes into a rotating wait cursor, i.e., into the watch icon. This indicates that some activity is still going on, even if *POTPULSE* is waiting between two sweeps. During such a period the cursor is disabled except for the purpose of clicking on one of the controls BREAK, STOP, WAIT and LINK, i.e., to interrupt acquisition.



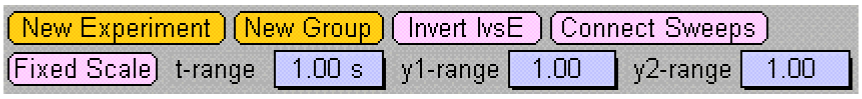
Break: Break is used to stop series execution. The Break flag is reset, when the next target is displayed or executed. If Break is pressed during acquisition of a sweep, this particular sweep will not be completed and the data acquired for this sweep thus far will be discarded. All previous sweeps of the series will be saved; thus there can be a series with less sweeps than specified in the Pulse Generator. The Break button can also be used in Replay mode when performing a lengthy operation, such as the Export All function in the Marks menu.

Stop: This button is used similarly to Break. However, during acquisition of a sweep, this particular sweep will be completed. The Stop button can also be used in Replay mode when performing a lengthy operation, such as the Export All function in the Marks menu.

Wait: : This button is used to pause series execution. Its command key is 'CTRL' + 'I'. Series execution is resumed by another click on the button or by 'CTRL' + 'Q'. Wait will become effective after completion of the currently acquired sweep. The Wait button can also be used in Replay mode when performing a lengthy operation; e.g., it allows to inspect a particular sweep, when replaying a collection of sweeps.

Link: : This button is used similarly to Break. However, acquisition is continued with a linked series or another repeat of the actual series, if present. This button can be used to toggle saving data during continuous recording. To do so, one has to create two identical series in the Pulse (Free-waveform) Generator with each one linked to the other one. One of the series has to be write-disabled (see Chapter Pulse Generator). With the Link button one can now switch between the two series.

Experiment Control and Display



Some further options are available after expanding the *Oscilloscope* window. Clicking on the upper right corner (MacOS) or the full screen button of the window (Windows systems) will expand the window and show the next options in the lower part of the window.

New Experiment: Allows to create a new "Experiment" target in the *Replay* window and increments the experiment number (if the file is opened without write protection). After the addition of a new group an automatic file update is performed (see File menu). This option has been added to allow a macro to create a new "Experiment".

New Group: Allows to create a new "Group" target in the *Tree* window and increments the group number. This option has been added to allow a macro to create a new "Group".

Invert IvsV: Inverts these axis of the *Oscilloscope* display. Note that you have first to select I as the y-axis and E as the x-axis and then to click on the Invert I vs. E button to display E versus I.

Connect Sweeps: Allows to draw a connecting line from the last point of a sweep to the first point of the next sweep, when the following conditions are met:

- the display mode is set to I vs. E
- the sweep has both, current and voltage traces
- the sweeps are plotted in sequential order from first to last
- the sweeps are within one group

Fixed Scale: Activating this function will fix the scale of the Oscilloscope window with the three parameters t-range, y1-range, and y2-range.

• t-range	-	defines the time range in seconds that should be displayed along the x-axis
• y1-range	-	defines the range of the first trace; normally the current in the potentiostatic mode or the potential in the galvanostatic mode. Example: A value of "200n"would result in a total range from -100 to +100 nA.
• y2-range	-	defines the range of the second trace; normally the potential in the potentiostatic mode or the current in the galvanostatic mode. Example: A value of "20" will result in a total range from -10 +10 V. Please note that the second trace is only available if 2 traces are acquired (see also "Trace" in Chapter *Pulse Generator*).

If this function is not activated, the scale of the *Oscilloscope* display depends on the selected current range. That is, in the case of current display full range of the display corresponds to the selected current range and in the case of voltage display full range corresponds to +/- 10 V.

Note: The Fixed Scale function may be useful if different current ranges are used in one experiment and the data of the whole experiment should be displayed with the Overlay option.

Changing the Size of the Oscilloscope Window

There are two ways to change the size of the Oscilloscope window:

1. The shortcut: Hold down the 'SHIFT' key while you resize the window, and all controls will be automatically moved down or right to make space for the Trace window. Note: This can lead to misplaced buttons. To sort a deranged window, do either close *POTPULSE* and restart it, or – if you already saved it – delete "DefaultPotPulse.set" from the *POTPULSE* directory.
2. The procedure described in the chapter User Interface, subchapter 'Tutorial: Changing the Size of the Oscilloscope Window'. That feature is not as easy,

but it allows to define the position and graphic appearance of each single control in the window.

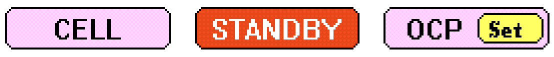
If you want to make the change permanent, save the new window organization with the menu command Pulse → Front Dialog → Save.

PG310/390 Potentiostat/Galvanostat

The *PG310/390 Potentiostat* window is used for controlling, adjusting and displaying the PG310/390 potentiostat/galvanostat operating modes.

Main window functions

Cell Connection



Cell/Standby/OCP: Via these buttons the connection of the PG310/390 potentiostat/galvanostat to the cell can be controlled.

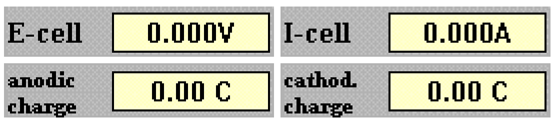
Cell connects all the electrodes to the respective inputs of the PG310/390 potentiostat/galvanostat. In the potentiostatic mode the cell potential is controlled by the PG310/390 potentiostat, and the current flowing through the cell is displayed as I-Cell. In the galvanostatic mode the current flow through the cell is controlled by the PG310/390 galvanostat, and the respective cell potential is shown in the E-Cell display.

In the **Standby** mode all the connections are switched off, and the potential can neither be read nor set.

In the **OCP** mode the counter-electrode is disconnected. The cell is in the zero current state, and the open cell potential is exhibited in the E-Cell display. The small Set insert button is used to set the Initial Potential value to the actual Open Cell Potential.

Note: It is also possible to switch to the Standby mode by pressing the '0' key of the numeric keypad.

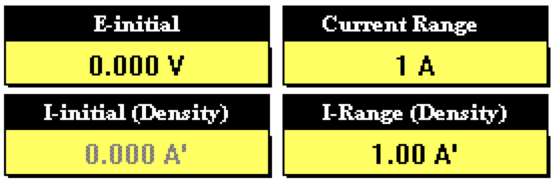
Potential / Current settings



E-cell: Cell potential monitor. The cell potential is displayed in relation to the chosen Zero Potential (E-zero) (see also Chapter *Pulse Generator*) .

I-cell: Direct current monitor. The current flowing through the cell is displayed.

Charge: The charge which is passing through the cell is subdivided into anodic charge and cathodic charge. With respect to electrochemical notation, the anodic charge is due to the positive portion of the current flow through the cell, and the cathodic charge, which is displayed as absolute value, is due to the negative current flow.



Initial Potential: Sets the desired initial value of the cell potential in the potentiostatic mode. The range is ±10 V and can be set by dragging the mouse or entering via the keyboard.

- Dragging with the mouse will change the potential in 1 mV steps.
- Pressing ← and → changes E-initial in steps of 10 mV.
- Pressing OPTION ? / ? (MacOS) or CTRL ? / ? (Windows) will change E-initial in 1 mV steps.

In the galvanostatic mode, the item will display the I-initial (Initial current). The current is calculated by multiplying the current density and the electrode area, which can be specified either in the Configuration or Parameter window. It is also possible to set I-initial (Density) instead of I-initial. In the potentiostatic mode, the I-initial (Density) field is *not* activated.

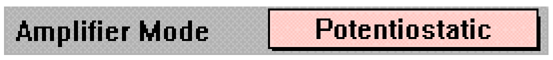
Current Range: Sets the scaling of the current monitor output (I-cell). The ranges are 1µA to 10A incremented by decades.

To set the current range, click with the mouse on the Current Range button and selecting the desired range in the pop-up window by tagging.

If the high gain pre-amplifier is connected to the PG310 Potentiostat/Galvanostat, the current ranges from 100pA to 100nA are selectable. The Auto Range mode is also available (see below).

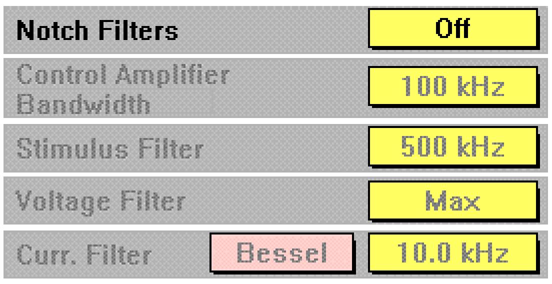
The current range is calculated by multiplying the current density range and the electrode area, which can be specified either in the *Configuration* window or the Parameter window . Alternatively, the user can choose the range of current density instead of the current range. The unit of the current density is labeled with a prime, e.g., A?.

If there is a saturation of amplifiers in the current monitor circuitry, a blinking box labelled Over on the Current Range button is displayed. This is a warning that excess artifacts or noise may occur as a result of the saturation of amplifiers.



Amplifier Mode: Allows a selection between potentiostatic and galvanostatic mode.

Filter Settings



Notch Filters		Off
Control Amplifier Bandwidth		100 kHz
Stimulus Filter		500 kHz
Voltage Filter		Max
Curr. Filter	Bessel	10.0 kHz

Notch Filter: The PG310/390 potentiostat/galvanostat provides a notch filter for both the voltage and the current pathway. The notch frequency (50 Hz or 60 Hz) is pre-set at the factory as desired by the customer, and cannot be set by the user. If the notch filter is On, incoming signals at the power supply frequency will be effectively filtered out.

Control Amplifier Bandwidth, Stimulus Filter, Viltage Filter, Current Filter: The following filters can be set either manually or automatically. (The Notch Filter has to be switched manually by the user in any case.)

For automatic filter mode, please activate the Auto Filter option in the *Configuration* window. The automatic settings dependance is that way:

- The automatic settings of the Control Amplifier Bandwidth, the Voltage Filter, and the Current Filter depend on the user defined Sample Interval and Filter Factor in the *Pulse Generator* (Free waveform) window.
- The automatic setting of the Stimulus Filter depends on the Sample Interval.

If the Auto Filter mode is selected, the filter settings are unable for manual settings and displayed with grayish numbers.

Note: If you are using an electrochemical cell with a high capacity and low Ohmic drop, a problem may arise with the automatic setting of the Stimulus Filter. In combination with a high frequency Stimulus Filter a pulse-like stimulus may cause oscillations at the potential steps with the use of such cells. In this case, try to select the manual filter mode and reduce the frequency of the Stimulus Filter. The other three filter settings proposed in the Auto Filter mode can be left unchanged.

Control Amplifier Bandwidth: The potential control amplifier can be filtered to avoid oscillations of the control circuit. Eight of the nine possible settings of the bandwidth (plus Max) are available:

- **10 Hz**
- **100 Hz**
- **300 Hz**
- **1 kHz**
- **3 kHz**
- **10 kHz**
- **30 kHz**
- **100 kHz**
- **Max**

The possible settings depend on the PG 310/390 potentiostat version (see also Chapters *Menus, PGF-Editor Menu, Serial number*):

- If you are using a PG310/390 potentiostat with a "A" or "B" labeled serial number, the low frequency bandwidth of 10 Hz will not be available.
- If you are using a PG310/390 potentiostat with a "C" or "D" labeled serial number, the high frequency bandwidth of 300 kHz will not be available.

Note: A transient problem may arise if the potential change is very fast in combination with a low bandwidth setting. There will be a rise time lag in the cell potential with respect to the given potential pulse. The time constant as the inverse of the bandwidth defines the shortest time domain over which the cell will accept a significant perturbation.

Stimulus Filter: The stimulus can be filtered (2-pole Bessel) to reduce the amplitude of fast capacitance transients when the speed of potential changes is not critical. Four settings are available:

- **500 kHz** • **50 kHz** • **5 kHz** • **0.5 kHz**

Note: In a ramp segment the stimulus change is step-like rather than line-like. The number of steps needed for representing the ramp is defined by the quotient of the segment length and the "sample interval" (see also Chapter *Pulse Generator*). To improve the smoothness of the stimulus change, these steps are filtered to yield exponential, which reach about 63 percent of the step height during a term of one time constant. Hence, a filtered stimulus will become a chain of exponential, which exhibits a phase shift relative to the unfiltered stimulus. For instance, if you choose a filter time constant equal to the sample interval, the filtered stimulus will be a nearly smooth line with a phase shift of about 90 percent of the step length (i.e. of the sample interval).

Voltage Filter: The voltage output (U-cell monitor) can be filtered by a 3-pole Bessel filter to reduce high frequency noise. The bandwidth can be set in four steps:

- **10 Hz** • **100 Hz** • **1000 Hz** • **Max**

With the selection of Max no filtering is active and all frequencies are allowed to pass the AD channel.

Current Filter: Controls a switchable analog Bessel/Butterworth filter (4-pole) in the current monitor pathway. The menu provides the following settings:

- Bessel
- Butterworth
- Bypass (e.g., no filter active)

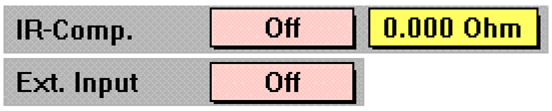
Dragging the mouse or entry on the keyboard allows bandwidth fine adjustment from 0.1-16 kHz in 0.1 kHz steps (Bessel) or 0.1-25.5 kHz (Butterworth). They differ in the following way:

- The Bessel setting is the best characteristic for general use. As such it is always used in automatic mode.
- The Butterworth response rolls off more rapidly with frequency and is useful mainly for power spectral analysis.

Under most conditions a 10 kHz bandwidth is more than ample, and the filtering reduces the high-frequency noise substantially.

Note: In automatic mode always the Bessel filter is used.

Additional Settings


IR-Comp. Off 0.000 Ohm
Ext. Input Off

IR-Comp.: The series resistance compensation corrects for the voltage drop between the reference and working electrodes under conditions of high access resistance or high current flow between counter- and working electrode. In the 4-Electrode Mode the serial resistance between both reference electrodes is corrected. The compensation is based on the value of R-series, which can be changed by dragging the mouse or entering on the keyboard (range: 0 to 1 M) and will be effective only when IR-Comp is On (see also *PG310/390 Manual – IR Compensation*).

Ext. Input: The *external input* (front panel of the PG310/390) is scaled by a selectable factor (range: 0.1x and 1x), to allow for different external stimulators. It is strongly recommended to set Ext. Input to Off (i.e., equal to zero), if no external stimulator is connected to Ext. Input. This will prevent pick-up of external noise.

Note: The E-Initial (the Initial Potential) is not affected by changing the external scale factor. The scaling will only affect the initial potential, if the user sets the *initial potential* externally (e.g., with a stimulator or another computer).

Resets

Reset: Selecting this button will reset the PG310/390 to its initial default configuration. Reset is very useful for defining the initial state of the PG310/390, when recording a macro. It will reset the DA channels to zero. E-Initial (the Initial Potential) will be unchanged by resetting the PG310/390.

Charge Reset: Resets both the anodic and the cathodic charge.

Cell mode

3 Electrode Mode

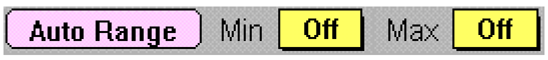
Cell Mode: Two separate cell setups are supported by the PG310/390 Potentiostat:

- **3 Electrode Mode**
- **4 Electrode Mode**

The **3 Electrode Mode** consists of a working, a counter and a reference electrode. In this mode the second reference electrode input (Reference II) on the front panel of the PG310/390 is short-circuited to the working electrode, so that the potentiostat controls the voltage between the working and reference electrodes. This commonly used mode is the default setting.

The **4 Electrode Mode** provides two distinct reference electrodes (Reference I and Reference II on the front panel of PG310/390), between which the potentiostat controls the voltage

Auto Range

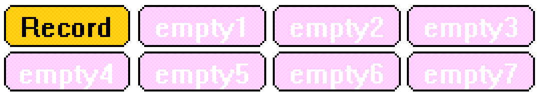

Auto Range Min Off Max Off

Auto Range: *POTPULSE* can adjust the current range after a sweep has been performed, if the current was out of bounds during this sweep. The Auto Range mode is available if:

- Min and Max are *not* set to Off in both these selection boxes.
- Auto Range is set in the *Pulse Generator* (Free waveform) window. There the Auto Range mode can be set to peak, mean, Yseg currents and Sweep as relevant values for auto ranging (see Chapter *Pulse Generator*).

The desired thresholds of current density within a given current range are defined then in the Min/Max boxes in percentages of full range. For instance, a Max setting with 90 percent will force *POTPULSE* to switch to the next higher current range if the current exceeds 90 percent of the actual range.

Macro Recording



The following buttons give you some of the features from the PGF-Editor menue Macros option (see Chapter *Macros*).

- Record corresponds to Start Recording.
- The macro buttons corresponds to Execute [1..7].
- Clicking on an empy macro after recording and choosing Assign and name recorded macro corresponds to Stop Recording. Here the index of the macro is directly the index number of the empty macro button.
- Clicking on Record at the end of recording aborts recording directly.

You can save 20 macros at once, whereas the macro items of macro 8..20 and higher are in that part of the window which gets exposed when you zoom out the PG310/390 Potentiostat window (see below in Chapter *Hidden Controls*).

You have the option of actually executing all actions as they are entered, or of disabling execution and only logging the actions to the macro. This option you have to set in the PGF-Editor menu Macro options before you record.

Record: To start macro recording, click on the Record button. Then, perform all desired actions. (The Notebook window will print a protocol of the macro actions.) You may record up to 50 actions in the *PG310/390 Potentiostat*, *Oscilloscope* and *Online Analysis* windows. To specify a parameter value, enter it as usual by dragging or keyboard entry. When clicking on a macro button you will see a dialog with the following options:

- **Delete Macro**: This will erase the selected macro.
- **Cancel**: This will disregard the macro call. You can continue recording.
- **Record call to macro itself**: This will execute the macro as part of the macro being recorded (embedded macro call).
- **Copy contents of macro**: This will copy each of the macro instructions of the selected macro into the macro being recorded. This avoids the problems of recursive macros (i.e., macros calling each other and causing an infinite loop).

- **Assign and name recorded macro**: This will prompt you to give a name to the macro. This name will become the button text. This is the function you need for saving a newly recorded macro!

To abort recording a macro, click again on **Record** and the sequence just recorded will be lost.

Normally, macros are only available until you leave the program. You have to save the macros explicitely in a file on disk (see PGF-Editor menu Macro options options), if you want to use it further.

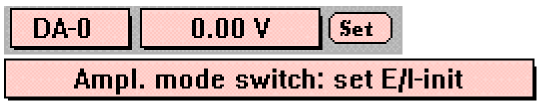
The default macro file is named “Default.PG310/390_Macros” and is automatically loaded the next time the program is started.

Macro 1..20: To start a recorded macro you can either click on the desired macro button or hit the the key with the same number on the numeric block of your keyboard (macros 1..9).

Hidden Controls

Some rarely used controls are on the right and bottom of the *Potentiostat* window. They can be accessed by clicking on the zoom box of the window or by pulling the window with the mouse on the right and lower sight.

The following features can be found at the bottom of the *Potentiostat* window.



DA Channels: These controls allow a command of the voltage output through the specified DA-channels. Three channels are available:

- **DA-0**
- **DA-1**
- **DA-2**

You can also choose, how the digital output is set:

- **Digital out (bits)**: One bit (one digital output) is set at each time.
- **Digital out (words)**: 8 bit (8 digital outputs) are set at each time.

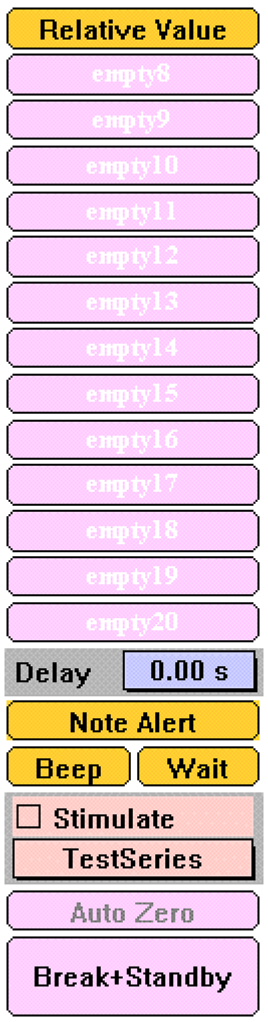
DA-Voltage: The voltage for the DA-channels may be adjusted here.

Set: Activating the Set button will output the voltage on the specified DA-channel.

Note: Trigger outputs are also fed to DA channels which may be specified in the *Pulse Generator* window (see Chapter *Pulse Generator*). Be sure that no undesired overlay of any trigger signal and command voltage will occur. On the other hand, you can create a pulse pattern on the desired DA channel with the use of the command voltage and trigger signals.

Ampl. Mode switch: set E/I-init: This feature ensures that by switching the Amplifier Mode between potentiostatic and galvanostatic, no current or voltage leap will occur. Thus, in the potentiostatic mode the actual current is measured and applied to the cell after the amplifier mode is switched to galvanostatic. This is the default setting.

If the function **Ampl. Mode switch: keep E/I-init** is choosen, current or voltage leaps may occur and damage the solution or the .



Features for Macros

These features can be found at the right side of the *Potentiostat* window.

Relative Value: This control button allows the performance of relative changes of control settings during macro recording.

Note: For instance, if you want to compare different data monitors because of different filter settings, you can start a sweep while recording a macro, change the current filter setting relative to the first one by 1 kHz (click on Relative Value and then set filter to 1 kHz), start the sweep again, etc., and save the macro.

Macros 8..20: Buttons for all macros with index numbers higher than 7.

Wait: This button is used to pause a macro execution. Pausing is indicated by a flashing Initial Potential field, which displays “Continue?” and the Wait button, as well as by a “Beep” sound. Macro execution is resumed by a click on the mouse. The Wait button can be used to inspect a particular sweep when playing a collection of sweeps. The time in seconds for the Wait function is set in the field above.

Note Alert: This opens an input window where one can enter a comment that will be automatically inserted into the *Notebook* window whenever this macro is executed.

Delay: This determines how many seconds the program has to wait before the next action is started by the PG310/390 potentiostat/galvanostat. When the Record button is activated, the label changes to Wait.

Beep: Plays the system sound. In recording a macro it is useful to acoustically indicate the end of an experiment or other events.

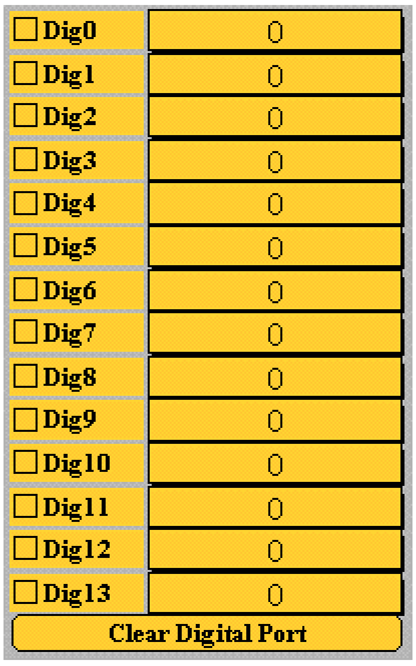
Stimulate / Test Series: The function Stimulate activates the test pulse, which is automatically executed as long as the *Potentiostat* window is active. You can define what pulse that shall be used as test pulse in the input field Test Series, e.g., “potstep” from the DefPotPGF generator file.

Break + Standby: Corresponds to the buttons Break and Standby, thus stopping all data aquitision immediately and turning the cell to Standby mode.

Digital Ports

Dig 0- Dig 13: Here fourteen digital output ports can be set. This feature can only be used with the additional Trigger Interface TIB14.

Clear Digital Port: All settings of digital ports will be cleared.



☐ Dig0	0
☐ Dig1	0
☐ Dig2	0
☐ Dig3	0
☐ Dig4	0
☐ Dig5	0
☐ Dig6	0
☐ Dig7	0
☐ Dig8	0
☐ Dig9	0
☐ Dig10	0
☐ Dig11	0
☐ Dig12	0
☐ Dig13	0

Special Function - Controlling the PG340

With the *POTPULSE* software you can also control the PG340 Bipotentiostat/Galvanostat .

The main feature of the PG340 ring/disc potentiostat is its design as a double potentiostat which allows to control two independent working-electrodes. This not only provides the tools to control rotating ring-disc electrodes, which are almost essential for the thorough study of mechanisms and kinetics of electrochemical reactions; it also allows the control of ultramicro-electrodes and the performance of generator- and collector- electrode techniques in such innovative technologies as the Scanning Electrochemical Microscope (SECM).

Therefore, the *Potentiostat* window is amended with some buttons and functions for the two working electrodes, in comparison to the normal *Potentiostat* window, (see Chapter *PG310/390 Potentiostat/Galvanostat*).

Electrode Conditions

Via the buttons **Disk** and **Ring** the active electrode can be set and controlled.

- **Disk** – all settings in the *Potentiostat* window apply to the DISK electrode
- **Ring** – all settings in the *Potentiostat* window apply to the RING electrode

Cell Connection

Cell/Standby/OCP: Via these buttons the connection of the PG310/390 potentiostat/galvanostat to the cell can be controlled.

Cell connects all the electrodes to the respective inputs of the PG340 potentiostat/galvanostat. In the potentiostatic mode the cell potential is controlled by the PG340 potentiostat, and the current flowing through the cell is displayed as I-Cell. In the galvanostatic mode the current flow through the cell is controlled by the PG340 galvanostat, and the respective cell potential is shown in the E-Cell display.

Note: The galvanostatic mode can only be set with the DISK electrode.

In the **Standby** mode all the connections are switched off, and the potential can neither be read nor set.

In the **OCP** mode the counter-electrode is disconnected. The cell is in the zero current state, and the open cell potential of the selected electroded – either DISK or RING - is exhibited in the E-Cell display. The small Set insert button is used to set the Initial Potential value to the actual Open Cell Potential.

Note: It is also possible to switch to the Standby mode by pressing the '0' key of the numeric keypad.

Potential/Current Settings

Almost all functions are the same as for the standard PG310/390 potentiostat/galvanostat, see Chapter *PG310/390 Potentiostat/Galvanostat*. However, there are some small differences to be taken into account.

All settings apply to the selected electrode, e.g., DISK or RING.

Exceptions are:

- **Standby**, **Cell** and **OCP**, which act on the DISC and the RING electrode.
- **Control Amplifier Bandwidth** Filtering, which acts on the DISC and the RING electrode.
- **Stimulus Filter**, which is active only with the DISK electrode.

Additional Settings

The IR-compensation in the 3-Electrode Mode as well as in the 4-Electrode Mode applies only to the DISC electrode.

Cell Mode

Cell Mode: Two separate cell setups are supported by the PG340 Potentiostat:

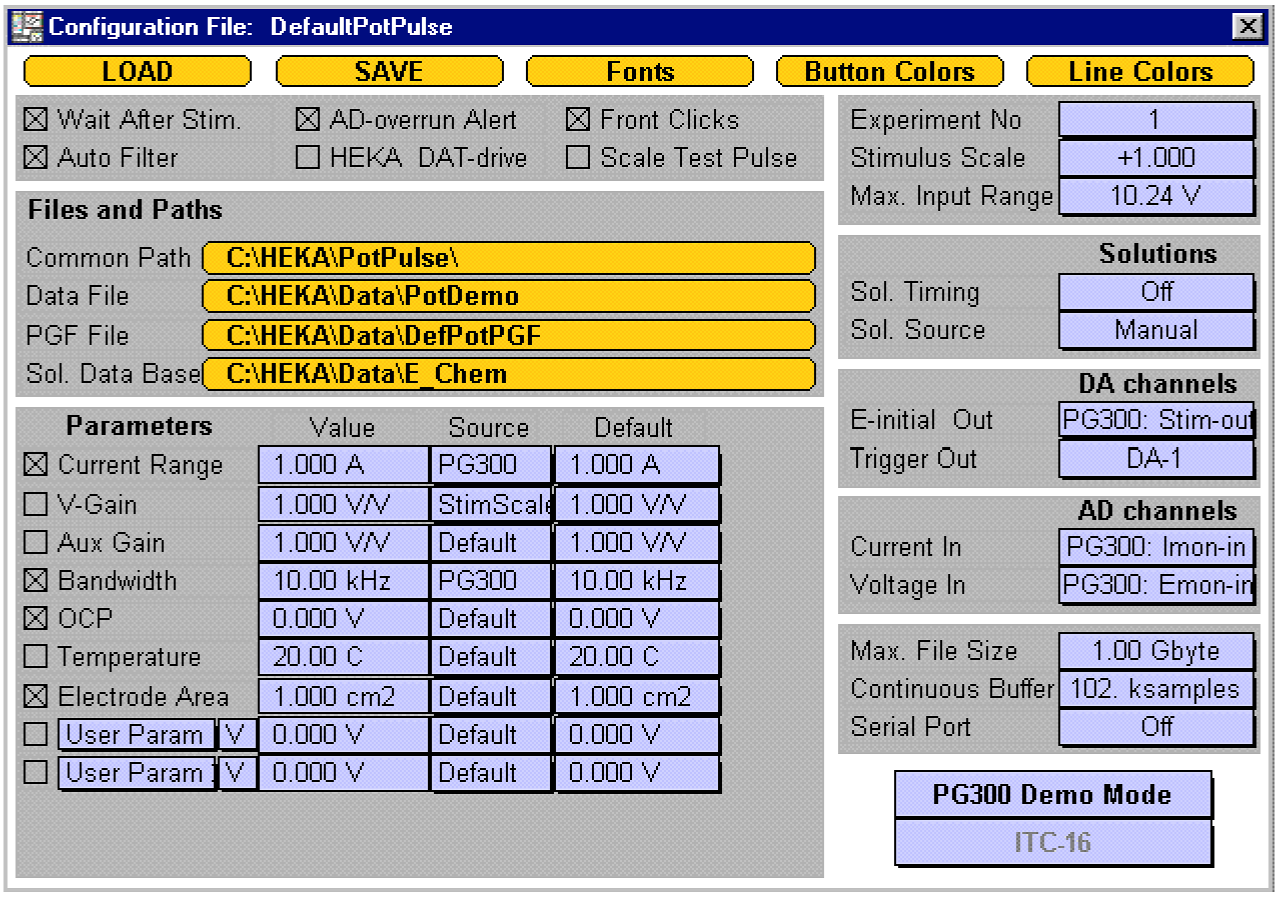
- **3 Electrode Mode**
- **4 Electrode Mode**

The **3 Electrode Mode** consists of a working (DISK and RING), a counter and a reference electrode. In this mode the second reference electrode input (Reference II) on the front panel of the PG340 is short-circuited to the working electrode (DISK), so that the potentiostat controls the voltage between the working (DISK and RING) and reference electrodes. This commonly used mode is the default setting.

The **4 Electrode Mode** provides two distinct reference electrodes (Reference I and Reference II on the front panel of PG340), between which the potentiostat controls the voltage.

Configuration

Settings like sources for external parameters, default values, display settings, colors, fonts, default files, etc. can be edited in the Configuration window. To access the Configuration window type 'F11' (MacOS) or 'F8' (Windows) or select the drop-down menu PotPulse → Configuration. These and other settings can be stored as a specified file (extension " .set "). That way every user can define her/his individual program layout to meet the specific requirements.


Configuration File: DefaultPotPulse

☒ Wait After Stim. ☒ AD-overrun Alert ☒ Front Clicks
☒ Auto Filter ☐ HEKA DAT-drive ☐ Scale Test Pulse

Experiment No	1
Stimulus Scale	+1.000
Max. Input Range	10.24 V

Files and Paths

Common Path	C:\HEKA\PotPulse\
Data File	C:\HEKA\Data\PotDemo
PGF File	C:\HEKA\Data\DefPotPGF
Sol. Data Base	C:\HEKA\Data\E_Chem

Solutions

Sol. Timing	Off
Sol. Source	Manual

DA channels

E-initial Out	PG300: Stim-out
Trigger Out	DA-1

AD channels

Current In	PG300: Imon-in
Voltage In	PG300: Emon-in

Parameters	Value	Source	Default
☒ Current Range	1.000 A	PG300	1.000 A
☐ V-Gain	1.000 V/V	StimScale	1.000 V/V
☐ Aux Gain	1.000 V/V	Default	1.000 V/V
☒ Bandwidth	10.00 kHz	PG300	10.00 kHz
☒ OCP	0.000 V	Default	0.000 V
☐ Temperature	20.00 C	Default	20.00 C
☒ Electrode Area	1.000 cm2	Default	1.000 cm2
☐ User Param V	0.000 V	Default	0.000 V
☐ User Param V	0.000 V	Default	0.000 V

Max. File Size	1.00 Gbyte
Continuous Buffer	102. ksamples
Serial Port	Off

Load / Save: Loads or saves a Configuration File (file name extension: *.set).

Fonts / Button Colors / Line Colors: Colors and text fonts for the program layout can be selected here. These are global settings for all window dialogs and they are installed upon restart of *POTPULSE*. The colors and fonts are stored in the configuration file (*.set), i.e., they are independent of the Dialog files (see Chapter *User Interface*). If dialog files are present, they will overwrite the Configuration settings.

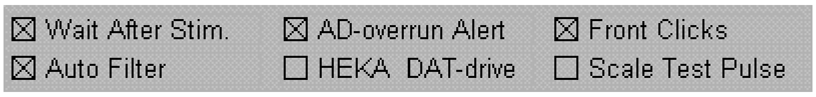
Note: You can make the background colors of the windows dark (useful when doing light- sensitive experiments) by selecting the option Button Colors. In this case you may also have to change the color of lines, like the Main Trace Color, for example, using the option Line Colors.

Under MacOS the window title bar color cannot be changed from within *POTPULSE*. To change it you will need to employ a resource editor like ResEdit. In ResEdit you can generate a window resource with a resource ID of *0* (zero). Then, open this window resource, select custom color, and set the content color to the color you want. The color can be dark but should not be black, because then you will not be able to read the window titles and the text in some alert boxes, windows, and dialogs. It is not recommended to modify the other color options, for it may cause odd results. Now, save the changes, and test the dialog by running *POTPULSE*.

Under Windows one can modify the windows and desktop elements via "Start > Settings > Control Panel > Appearance".

General Settings

The general settings can be found on the top right side of the configuration window.



☒ Wait After Stim. ☒ AD-overrun Alert ☒ Front Clicks
☒ Auto Filter ☐ HEKA DAT-drive ☐ Scale Test Pulse

Wait after Stim: This option determines if *POTPULSE* waits after executing a stimulus before proceeding with an operation which would erase the display of the just acquired traces. This occurs when a series is executed from the PG310/390 *Potentiostat* window or from the *Pulse Generator* window.

Auto Filter: The Pulse Generator provides the option to automatically set a hardware filter according to the chosen sample interval. Auto Filter enables/disables this automatic filter setting, which is only supported for the PG310/390 with its built-in hardware filters. In the test pulse mode the PG310/390 filter is set according to the sample interval if Auto Filter is on using a Filter Factor (see Chapter *Pulse Generator*).

Note: Short test pulses will require a high sampling rate, i.e., a high filter bandwidth.

AD-Overrun Alert: Prints an alert in the *Notebook* window in case of an AD-Overrun.

Front Clicks: Normally, applications with multiple windows only accept mouse clicks in the topmost or active window (the MacOS Finder is a notable exception). With the option Front Clicks, *POTPULSE* can be configured to accept a single click on a button or control of an inactive window to select or activate that button (there is no need to first bring the window to the front).

Note: Be aware that by enabling this option, there is also the danger of activating some function unwillingly when blindly clicking into an apparently inactive window.

Scale Test Pulse: Scaling factor of the Test signal. In case of the *POTPULSE* program the test signal is activated in the Hidden Options in the *Potentiostat* window as

Stimulate. The default setting is a factor of 1, and it is recommended not to change this value in common applications.

Experiment No: This number can be used to identify experiments. Whenever Potpulse → New Experiment is executed this number is incremented. It is stored at the group level, i.e., only one experiment is contained within one group but one can have multiple groups with the same experiment number.

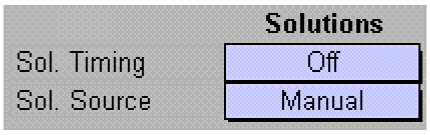
Stimulus Scale: Scaling factor of the stimulus signal.

Note: The PG310/390 has a fixed Stimulus. Scale factor of 1.

Max. Input Range: A potential delivered by an external voltage generator can be applied to the cell via the External Input on the front panel of the PG310/390. The maximal value of the external stimulus is defined in this menu.

Solutions

These two entries determine the time when information is written to a solution file and the source, i.e., how this information is obtained. For more details on how solution files are handled see Chapter *Solutions*.



Solutions

Sol. Timing	Off
Sol. Source	Manual

Sol. Timing: Specifies the time when information is written to a solution file.

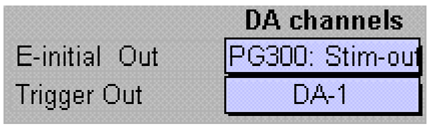
- **Off**: No solution file is created.
- **Before Series**: Inquire solution information before series acquisition.
- **After Series**: Inquire solution information after series acquisition.
- **After Group/Exp.**: Inquire solution information after group acquisition, i.e., whenever a new group is created.
- **End of File**: Inquire solution information whenever a file is written to disk.

Sol. Source: Specifies the source from which the solution information is obtained.

- **Manual**: Enter solution information via the Solution dialog.
- **Data Base**: Read solution information from Solution Data Base as specified in the control Sol.Data Base in the Files & Paths section.

DA-Channels

These items define the default output channel assignments used for initial potential and triggers (see also Chapter *Pulse Generator*).

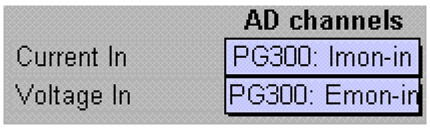


DA channels

E-initial Out	PG300: Stim-out
Trigger Out	DA-1

- **E-initial Out**: Assigns the output DA channel of the initial potential.
- **Trigger Out**: Assigns the output DA channel of the trigger pulses.

AD-Channels


AD channels

Current In	PG300: Imon-in
Voltage In	PG300: Emon-in

These items define the default input channel assignments used for current and voltage monitors.

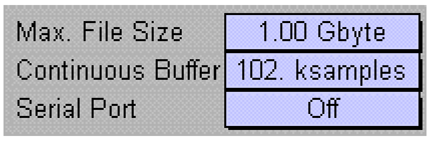
- **Current In**: Assigns the input AD channel for *I-cell Monitor*.
- **Voltage In**: Assigns the input AD for *U-cell Monitor*.

The units and gain scaling of the data read from the different AD-channels are:

- If PG310/390 channels are used, their data scaling and logical Y-units are applied.
- If the AD-channel is the Current In channel, then I-Gain (current range) and the Y-unit A are used.
- If the AD-channel is the Voltage In channel, then V-Gain and the Y-unit V are used.
- Otherwise, Aux-Gain and the Y-unit, which is exhibited as trace 2 unit in the DA/AD channels field of the Pulse Generator (default: V) are used.

Note: If you are using a PG310/390 potentiostat/galvanostat, the E-Initial out, Voltage In, and Current In channels are to be set to the channels labeled by PG310/390.

Options for data acquisition



Max. File Size	1.00 Gbyte
Continuous Buffer	102. ksamples
Serial Port	Off

Max. File Size: A warning is written to the Notebook if the size of the raw data file exceeds this value. This feature will not limit the size of any one file written to disk, but can be used as a reminder so that no files are created that do not fit onto a floppy disk or a zip drive, for example. Entering a very large number will practically suppress any warnings.

Continuous Buffer: For continuous data acquisition it may be necessary to temporarily write data to RAM rather than directly to disk, e.g., when acquiring continuously on two input channels (*POTPULSE* only allows you one continuous channel to be written directly to disk). The size of such a buffer in memory can be specified here.

Note: This option requires a buffer size large enough to accommodate the data in memory. A comprehensive description is given in the Chapter *POTPULSE Advanced > Continuous Recording*.

Serial Port Configuration

Serial Port: This control is used to set the serial port communication mode. If a so called Serial Communication has been established between the setup computer and another device, *POTPULSE* can send strings to other devices, but it will not receive instructions. After activating the protocol, the Oscilloscope window will show several additional buttons at the bottom in the usually hidden part. There are three options for the serial communication.

Off: No connected device.

User Defined: Any device that can receive strings through the serial port. Click on Open to get to the Serial Port Configuration window. *POTPULSE* allows the following settings:

- **Serial Port**: No Port, COM1 to COM4 (Windows) or Modem Port and Printer Port (MacOS).
- **Baud Rate**: 300, 600, 1200, 1880, 2400, 3600, 4800, 7200, 9600, 19200 or 57600 bps.
- **Stop Bit**: 1.0, 1.5 or 2.0.
- **Parity**: None, Even or Odd parity.
- **Data Bit**: 5, 6, 7 or 8 data bits.
- **XOn/XOff**: On or Off.
- **Rts/Cts**: On or Off.

To change all settings or leave the dialog, do one of the following:

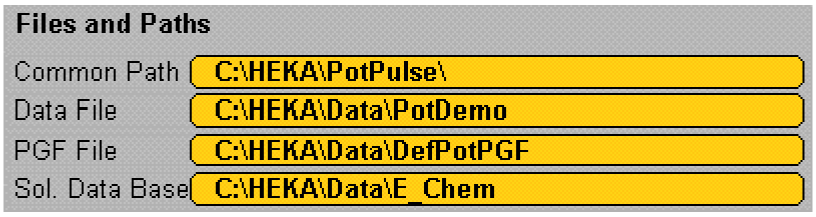
- **Get Defaults** sets the default settings given in *POTPULSE.*
- **Undo** cancels all changes without closing the dialog.
- **Cancel** cancels all changes and closes the dialog.
- **Done** saves all changes and closes the dialog.

Up to 16 strings can then be defined and triggered by buttons in the Oscilloscope window. These buttons are hidden at the bottom of the window and can be seen after expanding the Oscilloscope window (clicking on the zoom box). The button Serial is used to open the serial port (as well as sending an initialization string) or to edit the Send strings (*Send 2...16*). The Send buttons can be custom-labeled by editing the buttons (see Chapter *User Interface*).

To X-Chart: Special protocol to control the standalone version of X-Chart running on another computer. The communication strings are preset and mimic some of the controls of X-Chart. The new buttons correspond to the buttons from X-CHART's Control window and allow you to "remote control" the program running on a second machine from *POTPULSE* (see the X-Chart Manual for more details).

Files and Paths

In this section, the user has to specify the paths for certain files. *POTPULSE* will use these paths when storing or retrieving files.



Files and Paths

Common Path	C:\HEKA\PotPulse\
Data File	C:\HEKA\Data\PotDemo
PGF File	C:\HEKA\Data\DefPotPGF
Sol. Data Base	C:\HEKA\Data\E_Chem

You can name the following pathes:

Common path: The title of the Configuration window contains the name of the file that holds the information shown in the *Configuration*, *Oscilloscope*, *PG310/390 Potentiostat*, and *Online Analysis* windows. This file is read from or written to disk relative to the specified Common Path (Default: \Heka\Potpulse\).

Data File : Sets the path to the PGF file (Default: \Heka\Data\PotDemo)

PGF File: Sets the path to the PGF file (Default: \Heka\Data\DefPotPGF).

Sol. Data Base: Provides the name of the common solution data base (see Chapter *Solutions*). (Default: \Heka\Data\E_Chem)

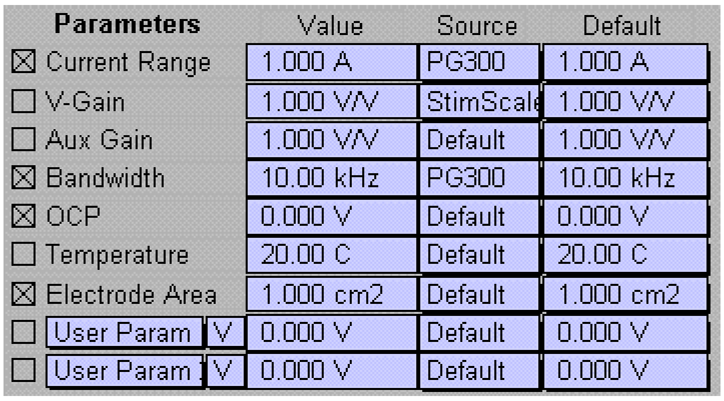
To rename a path, click on the button and set a new path.

Note: (MacOS only) For this option to work you must activate the setting Folder that is set by the application in the General Controls control panel of the MacOS 7.5 and higher. Otherwise the system overrides the *POTPULSE* path settings by its own ones.

Here is some advice to keep in mind when naming folders and files:

- The use of invisible characters and spaces is not recommended (a blank in a filename is very often overlooked).
- Names should not begin with a digit, because some other applications, e.g., IGOR, do not allow names with a digit as the first character (exported IGOR waves inherit the first 3 characters of the data file).
- (MacOS only) The first few letters of a name are the more important ones, because they ease file selection with the file selector. The file selector continuously selects the files while the user types. In addition, *POTPULSE* only shows the first 14 letters of the file name in the title of the *Oscilloscope* window.

Parameters



Parameters		Value	Source	Default
☒ Current Range		1.000 A	PG300	1.000 A
☐ V-Gain		1.000 V/V	StimScal	1.000 V/V
☐ Aux Gain		1.000 V/V	Default	1.000 V/V
☒ Bandwidth		10.00 kHz	PG300	10.00 kHz
☒ OCP		0.000 V	Default	0.000 V
☐ Temperature		20.00 C	Default	20.00 C
☒ Electrode Area		1.000 cm2	Default	1.000 cm2
☐ User Param	V	0.000 V	Default	0.000 V
☐ User Param	V	0.000 V	Default	0.000 V

Parameters such as temperature, current range (I-gain), electrode area, etc. are entered here. These are used to customize the recording environment of a given experimental setup. The settings are automatically pre-set when using an PG310/390 amplifier. Other amplifiers will require specific settings in order for *POTPULSE* to know how to scale an input or output voltage. The checkbox on the left determines whether the parameter is displayed (and updated) in the *Parameters* window.

- **Current Range**: Current range of the current monitor input.
- **V-Gain**: Gain of the stimulus voltage input.
- **Aux Gain**: Gain of an auxiliary voltage input.
- **Bandwidth**: Recording bandwidth of the current monitor.
- **OCP**: Open cell potential.
- **Temperature**: Temperature in °C (from a recording device).
- **Electrode Area**: Electrode area in cm^2 (from a recording device or entered in default section).
- **User Parameter 1**: Parameter from a custom device (e.g., pH).
- **User Parameter 2**: Parameter from a custom device.

For more details on the use of some of the above parameters, see below.

Value: Shows the actual value of the corresponding parameter.

Source: Determines how the value is obtained. For some of the parameters there are three alternatives:

- **Default**: The specified default value is taken
- **PG310/390**: Parameters are obtained from the PG310/390
- **AD-Channels**: Parameters are sampled via a specified AD-channel.

Default: The default value can be edited here. If the source is **Default** the value can be also changed in the *Parameters* window.

User Parameters: There are two parameter fields reserved for user-specific assignments. These fields act like other parameter fields with the addition that parameter name and unit can be specified (e.g., “pH” and “U”). Name and unit are also stored in the output “*.pul” file on the level of a series, so they can be changed during an experiment, if required. Text, unit, and value are stored in the Tree in the *Replay* window.

Amplifier and Digital Data Converter Section

Here the kind of amplifier (potentiostat) that is used has to be selected.

POTPULSE supports the following amplifiers:

- **PG 300 Amplifier (whole PG3xx family!)**
- **PG 400 Remote**
- **PG 400 Local**
- **PG 285 Amplifier**
- **PG 284 Amplifier**
- **Telegraphing Amplifier**

Digitizer Selection: Below the amplifier list one can select the AD/DA-converter used, when the amplifier is not an PG310/390. *POTPULSE* supports the following AD/DA-converters:

- **ITC-16**
- **ITC-18**

The option Telegraphing Amplifier is used if the gain (or current range) and/or bandwidth settings for the amplifier is read from a lookup table. Upon selection of this amplifier type, the user is asked to load first a gain table and then a bandwidth table. Some lookup tables for common amplifiers are already supplied (see folder “POTPULSE/Lookup Tables”).

If your amplifier is not in the list, create your own table. Lookup tables can be created easily because they are ASCII files with a simple structure with a series of text lines each containing:

`Voltage  Gain (mV/pA)` or `Voltage Bandwidth (Hz).`

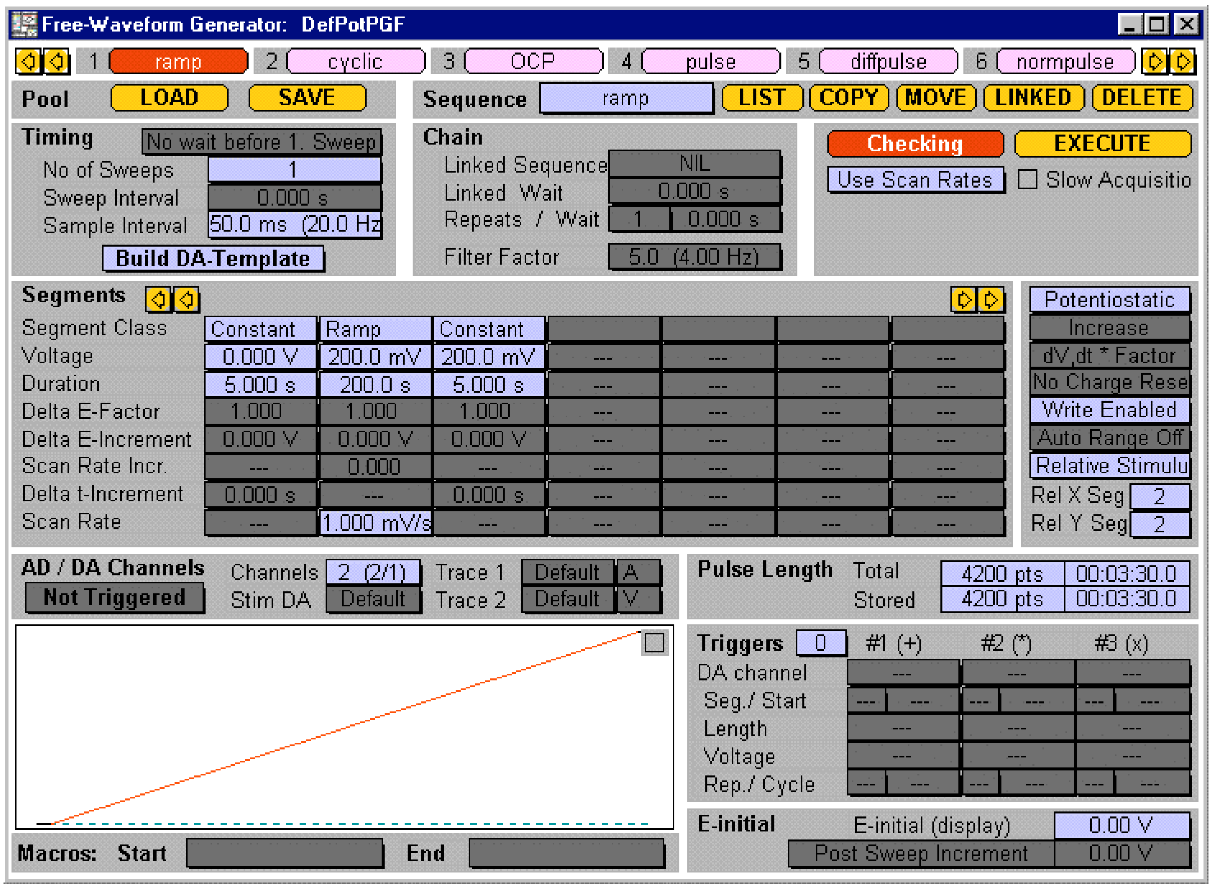
The voltage is the threshold above which the following gain or bandwidth setting applies. The voltage values are scanned beginning with the first value. Thus, voltages must be in descending order! The voltage thresholds for discrimination between adjacent settings should be halfway between the nominal telegraph voltages.

Demo Modes: If *POTPULSE* is only used for data review or demo purposes, the AD/DA hardware is not necessarily required. In this case *POTPULSE* will come with the message “AD/DA initialization failed: check power!” or “PG310/390 initialization failed! Please, check power and connections”. One can now click on Continue and *POTPULSE* will start in the PG310/390 Demo mode. This ensures that the program does not try to access the hardware.

Note: In the Demo mode no data can be stored to disk! The program behaves like every file is opened read-only, thus deactivating all functions that would lead to changes in the data, e.g., Delete Traces in the Tree menu.

Pulse Generator

The *Pulse Generator* (Free-waveform Generator) window provides the layout of a stimulation pulse sequence. Entries that are not default (e.g. changing parameters) are shown highlighted (*active* parameters), the rest is shown in dark (*inactive* parameters). Because of this automatic feature, the color of individual editable controls cannot be changed in this dialog, i.e., they are reset to the color assigned to their function upon the next window update. All other controls can be modified.



Pool

Sequence: ramp

Timing: No wait before 1. Sweep
No of Sweeps: 1
Sweep Interval: 0.000 s
Sample Interval: 50.0 ms (20.0 Hz

Chain
Linked Sequence: NIL
Linked Wait: 0.000 s
Repeats / Wait: 1 | 0.000 s
Filter Factor: 5.0 (4.00 Hz)

Use Scan Rates
☐ Slow Acquisitio

Segments							
Segment Class	Constant	Ramp	Constant				
Voltage	0.000 V	200.0 mV	200.0 mV	---	---	---	---
Duration	5.000 s	200.0 s	5.000 s	---	---	---	---
Delta E-Factor	1.000	1.000	1.000	---	---	---	---
Delta E-Increment	0.000 V	0.000 V	0.000 V	---	---	---	---
Scan Rate Incr.	---	0.000	---	---	---	---	---
Delta t-Increment	0.000 s	---	0.000 s	---	---	---	---
Scan Rate	---	1.000 mV/s	---	---	---	---	---

Rel X Seg: 2
Rel Y Seg: 2

AD / DA Channels
Channels: 2 (2/1)
Stim DA: Default
Trace 1: Default | A
Trace 2: Default | V

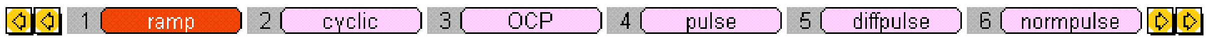
Pulse Length			
Total	4200 pts	00:03:30.0
Stored	4200 pts	00:03:30.0

Triggers 0	#1 (+)	#2 (*)	#3 (x)
DA channel	---	---	---
Seg./ Start	--- ---	--- ---	--- ---
Length	---	---	---
Voltage	---	---	---
Rep./ Cycle	--- ---	--- ---	--- ---

E-initial
E-initial (display): 0.00 V
Post Sweep Increment: 0.00 V

Macros: Start | End

Sequence Pool



Sequence Pool: The first row displays a section of the pool of available sequences. It is a paging bar in units of six sequences. Two arrows at each side allow to scroll through the available pulse protocols (the innermost arrows move in increments of one page, i.e., six sequences; the outermost arrows move to the start/end of the sequence list). A sequence is selected by clicking on it.

A copy of this pool of sequences is shown at the bottom of the *Oscilloscope* window. From there the sequences can be executed, i.e., the corresponding pulse pattern is output and the responses are sampled and displayed.

If no Pulse Generator File is available, *POTPULSE* creates a default file ("DefPotPGF.pgf"). This file only contains one stimulation sequence, called "Test". This sequence can be edited.

More sequences are added to the pool by copying an already existing sequence (Copy button) or by clicking a free position in the pool.

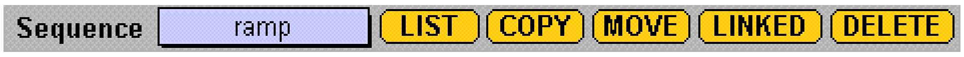
After modification of the existing pool of sequences, the entire Pulse Generator File (PGF) should be saved to disk (Save button). It can be saved under any name. *POTPULSE* automatically appends the extension .pgf to the file name. If this new PGF file should be loaded into the *Pulse Generator* as a default, the new name of the PGF file has to be specified in the *Configuration* window and the configuration file has to be saved.

Pool

Load/Save : Loads or saves the pool of available stimulation sequences (.pgf file). The present file name is indicated in the title bar of the Pulse Generator window, e.g., "Free waveform Generator File: DefPotPGF".

Sequence

Sequence: Editable name of the present sequence.



List: Writes the settings of the actual sequence into the *Notebook* window. Use this option, if you want to create a listing of your sequence to be able to recreate it on another machine. A PGF listing could look as follows:

```
PGF-File:,  Sequence 1: IV
EXTERNAL TRIGGER MODE: TrigNone
TIMING:
NumberSweeps: 9, SweepInterval: 0.000 s, SampleInterval: 20.00µs
Wait before 1. sweep:TRUE, FilterFactor: 3.0
CHAIN:
LinkedSequence: NIL, LinkedWait: 0.0 s, Repeats: 1, RepeatWait: 0.0 s
TRIGGERS:
1: Segment: 1, Channel: default, Time: 5.000ms, Length: 2.500ms, Am-
plitude: 5.000 V
RELEVANT SEGMENTS : X = 2, Y = 2, MaxSweepLength: 1250
Write Mode: Enabled, Absolute Stimulus, Post Sweep Inc.: 0.000 V
G and C Update: No Update
INCREMENT:   Mode : Increase,  LogIncrement: on
SEGMENTS:      Volt        Dur        VFact    VIncr      TFact    TIncr
1: Const.,     HOLD,      10.00ms,  1.00,    0.000 V,   1.00,    0.000 s
2: Const.,   -60.00mV,    10.00ms,  1.00,    10.00mV,   1.00,    0.000 s
3: Const.,     HOLD,      10.00ms,  1.00,    0.000 V,   1.00,    0.000 s
CHANNELS:
StimDA: default, Channel_1: default, Channel_2: default
Amplifier mode: VoltageClamp only
Macros: SetIV, none
```

Copy: Duplicates the actual sequence into the first free position. A new name has to be entered.

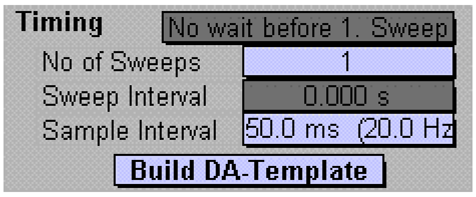
Note: Copy is one way of creating a new sequence. Another way is to activate an unassigned sequence button and name the new sequence in the prompted window. The pulse pattern can be edited in the Segments section of the *Pulse Generator* window.

Move: Moves the sequence to a new position. A number for the new position has to be entered. Use this option to move the most commonly used sequences to one of the handy first six positions, or to rearrange your pool.

Linked: Switches to the linked sequence, if there is another sequence linked to the current one.

Delete: Removes the present sequence from the pool.

Timing



Timing: No wait before 1. Sweep
No of Sweeps: 1
Sweep Interval: 0.000 s
Sample Interval: 50.0 ms (20.0 Hz

Timing: Wait before 1. Sweep will force *POTPULSE* to wait the time indicated by Sweep Interval before executing the series after activating it. Use No Wait before 1.Sweep if you want the sequence to start immediately without this delay.

No. of Sweeps: Determines the number of sweeps within one sequence, e.g., how many times the specified segments are run.

Sweep interval: The first sweep of a sequence has a waiting period during which the so-called DAC (digital analog converter) template is computed and loaded. Then the first sweep is acquired. The next sweep will start Sweep Interval milliseconds after the beginning of the first sweep.

Note: *POTPULSE* will try to do the timing as exactly as possible, however there is a minimum waiting period, the so called "Sweep Gap", that depends on the speed of your computer system and the complexity and duration of the stimulus to be calculated and output.

Sample Interval: The timing of data acquisition is given as Sampling Interval (in seconds) and as Sampling Frequency (in Hz). The shortest sample interval (in case no triggers are used) is 5 µs (200 kHz); the longest interval is 60 s (0.017 Hz).

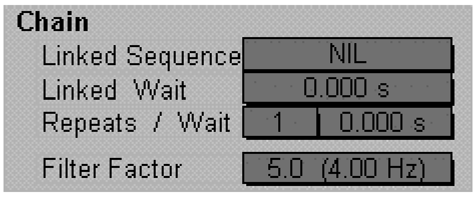
Note 1: If two or more channels are sampled simultaneously, the shortest interval has to be multiplied by the number of channels, e.g., for 2 channels the minimal sample interval is 10µs.

Note 2: If because of the sampling rate only one point per sweep would be taken, *POTPULSE* asks whether the segment duration should be fixed, e.g., set to higher durations.

Build DA-Template: This is the default setting which uses the entries in the Pulse Generator window for building the stimulus.

Get File Template: Selecting **Get File Template** causes the DAC-stimulus template to be loaded from a file instead of being computed. For more information please read the Chapter *POTPULSE Advanced > The "Get File Template" Feature*.

Chain



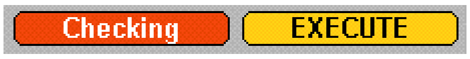
Linked Sequence: Stimulation sequences can be linked to other sequences within the pool or to themselves (Repeats). You have to enter the name of the sequence. An entry of "NIL" means no linked sequence.

Linked Wait: The delay between the end of one sequence and the beginning of the Sweep Interval before the first sweep of the linked sequence is given as Linked Wait in seconds.

Repeats / Wait: If a sequence is linked to itself, the number of repeats as well as their interval has to be specified. The user is notified when a timing overrun occurs. To prevent overruns, the Repeat Wait and Linked Wait values are checked for consistency with the rest of the template when entered: Repeat Wait and Linked Wait must be either zero or at least as long as a complete sweep.

Filter Factor: The Filter Factor (or *Nyquist factor*) is implemented for the PG310/390. It is used to define the automatic filter setting relative to the sample rate (activated by Auto Filter in the *Configuration* window). The default value of the Filter Factor is set to 5. The cutoff frequency is calculated by inverting the product of the sampling interval and the Nyquist factor. For the above example of Sample Interval = 20 µs and Filter Factor = 3, a filter cutoff frequency closest to 16.7 kHz will be selected. The suggested filter frequency is shown in parentheses.

Check and Execute

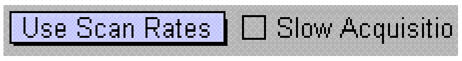


Not Checking / Checking : This option determines, whether a check is performed for any inconsistencies that might occur when entering values.

- When **No Checking** is enabled, the validity checking of the sequence editing is suspended. This is convenient when one wants to perform multiple changes, especially when some intermediate steps would result in a (temporarily) faulty sequence.
- When **Checking** is enabled, the active sequence is checked after any modification of segments and when storing, linking, switching, or leaving the PGF-editor. If the input is faulty, the user is notified and the last operation is canceled, until the sequence is valid. The Checking should be done at least at the end of the sequence input, before executing or storing a sequence.

Execute: Allows to output the presently active stimulation sequence. Upon termination of data acquisition the program returns to the *Pulse Generator* window. In this way pulse patterns can be adjusted interactively without changing windows until they yield the desired responses.

Time Mode



Ramp segments can be specified either by their duration or by their scan rate. If the scan rate is given by the operator, the duration will be calculated, and vice versa. The desired mode can be selected by clicking on the button.

Use Duration: The time between the ending and beginning of a ramp is set in seconds in the Duration field of the sequence.

Use Scan Rates: The scan rate is set in Volt per seconds in the Scan Rate field of the sequence and is calculated as *Voltage/Duration*.

Note: Using scan rates causes the E-initial checkboxes to disappear in the row above the segment.

Slow acquisition Mode: The Sample Interval has to be below 1s for that mode. Only in the Slow Aquisition mode Auto Range can be activated.

Wave Parameters

This option allows specification of the wave characteristics for segments of Sine wave and Square wave type by Wave Cycle and Amplitude. It will be active *only* if any of those segment types is selected. The parameters are:

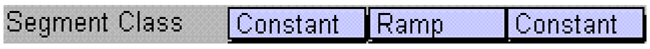
Wave Cycle: Wave length in seconds.

points/cycle: Number of sample points per full wave length. This number is calculated by Wave Cycle over Sample Interval and is not editable.

Wave Ampl.: The amplitude of the wave in volts. Note that the peak-to-peak amplitude is twice this value.

Segments

A pulse pattern consists of an arbitrary number of segments. Segments are shown as a horizontally scrolling matrix; scrolling is done by clicking on the arrows.



Segment Class: Segments can be the following:

- **Constant**: Segment of constant voltage.
- **Ramp**: Segment with ramp from the voltage of the previous segment to the voltage of this segment.
- **Conditioning**: Conditioning segment of arbitrary length. Triggers and continuous data acquisition are not supported for such a segment. The timing accuracy of conditioning segments is limited to the clock ticks of the computer clock (i.e., 1 ms).
- **Continuous**: Identifier for continuous data acquisition. Note that only the last segment can be of that class.

- **Sinewave**: Segment with sine characteristics. Amplitude and cycle length are defined in Wave Parameters (see above).
- **Squarewave**: Wave segment with rectangular characteristics. Amplitude and cycle length are defined in Wave Parameters (see above).

The list entries Insert, Duplicate, and Delete are used to create or remove segments.

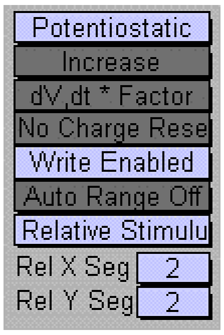
- **Insert**: Inserts a constant segment (default duration = 0) at the actual location.
- **Duplicate**: Creates a copy of the selected segment and inserts it at the actual location.
- **Delete**: Deletes the selected segment at once.
- **Duplicate…**: Creates multiple copies of a number of segments. You have to enter How many segments you want to copy (inclusive the actual segment) and How often these segments shall be copied. They will be inserted at the actual location.
- **Delete…**: Deletes a specified number of segments at once.

Voltage: The voltage of a segment is either a numeric value usually entered in V (will be recalculated to mV, if necessary) or the initial potential (E-initial) at the time of sequence execution. The value may be adjusted by dragging the mouse, or typing the number. E-initial can be activated by the checkbox in the upper row, if Duration is used. In the galvanostatic mode, voltage is replaced by current or current density.

Duration: Duration of a segment, usually entered in seconds (will be recalculated to ms or µs, if necessary). The value may be adjusted by dragging the mouse, or typing the number. Make sure that the durations of segments are even multiples of the sample interval. A warning is given if they are not, in case Checking is activated.

Delta E-Factor/E-Incr./t-incr.: Voltage and duration of a segment can be incremented between *successive* sweeps. There are two increment modes (see *Increment Mode* below). The magnitude of the increment is entered here.

Recording Settings



Recording Mode: Stimulation sequences can be restricted to potentiostatic or galvanostatic modes. The following options are available:

- **Pot. + Galv. Modes** : Allows stimulation in the potentiostatic and galvanostatic mode. (Is currently never active!)
- **Potentiostatic**: Allows stimulation in the potentiostatic mode only (default).
- **Galvanostatic**: Allows stimulation in the galvanostatic mode only.
- **Galv. (I-Density)**: Allows stimulation in the galvanostatic mode only and outputs the result as current density.

Increment Mode: This determines the order of incrementing the segment voltage and/or duration. The options are as follows (the numbers give an example for a series with 6 sweeps):

- **Increase**: first sweep comes first:
  1, 2, 3, 4, 5, 6.
- **Decrease**: last sweep comes first:
  6, 5, 4, 3, 2, 1.
- **Interleave incr.** : ascending interleaved:
  1, 3, 2, 5, 4, 6.
- **Interleave decr.**: descending interleaved: 6, 4, 5, 2, 3, 1.
- **Alternate**: first, last, second, penultimate...: 1, 6, 2, 5, 3, 4.

Computational background: The conversion from the logical (1, 2, 3, 4, 5, 6) to the physical order (e.g. 6, 4, 5, 2, 3, 1, in Interleave decr. mode) is calculated the following way (in 'C'-code):

```
int GetStimIndex (// returns the physical sweep index
int sweepCount,   // logical sweep index
int totalSweeps,  // total number of sweeps in series
int incrementMode)
{
 int  i;

 switch (incrementMode) {
   case increase:   // first sweep comes first
     return (sweepCount);
   case decrease:   // last sweep comes first
     return (totalSweeps - sweepCount + 1);
   case interleaveIncr:
   case interleaveDecr:
     if (sweepCount == 1)
       i = 1;
     else {
       i = sweepCount + (1 - (sweepCount % 2) * 2);
       if (i > numberSweeps)
         i = numberSweeps;
     }
     if (incrementMode == interleaveIncr)
       return i;
     else
       return (totalSweeps - i + 1);
   case alternate:    // first, last, second, ...
     if (sweepCount % 2)
       return (sweepCount / 2 + 1);
     else
       return (totalSweeps - sweepCount / 2 + 1);
 }
}
```

Example: If you apply a series of 6 sweeps at an increment of 10 mV starting at -40 mV the logical sequence will be: -40, -30, -20, -10, 0 and +10 mV. With the mode *Interleave decr* activated the pulses will be output in the following (physical) order: +10, -10, 0, -30, -20 and -40 mV.

Note: The expressions "V" and "E" for "Voltage" and "Potential" are used synonymously in the software and the documentation.

There are two increment modes for some potentiostatic pulse pattern (e.g., not for Continuous):

- **V, t * Factor**
- D**V,** D**t * Factor**

The two respective increment modes for some galvano-static pulse pattern are:

- **I, t * Factor**
- D**I,** D**t * Factor**

In mode *V, t * Factor* the logarithmic increment is based on the voltage (duration) of the *first sweep*, therefore it cannot be zero. The segment's voltage (duration) of the ith sweep is then calculated as:

$$t_i = \text{Duration} * \text{t-Factor}^{i-1} + (i - 1) * \Delta\text{t-incr}$$

$$V_i = \text{Voltage} * \text{V-Factor}^{i-1} + (i - 1) * \Delta\text{V-incr}$$

In mode *V, t * Factor* the increment may be zero. Let *Duration* be *10 ms*, *ΔtIncr = 0 ms*, and *tFactor = 2* then the series 10, 20, 40, 80 ms, ... is obtained.

In mode *ΔV, Δt * Factor* the logarithmic increment is based on the *linear increment step*, calculated as follows:

for *Vfactor, tFactor = 1*:

$$t_i = \text{Duration} + (i - 1) * \Delta\text{t-incr}$$

$$V_i = \text{Voltage} + (i - 1) * \Delta\text{V-incr}$$

for *Vfactor, tFactor ≠ 1*:

$$t_1 = \text{Duration}; V_1 = \text{Voltage}$$

$$t_{i,i>1} = \text{Duration} + \Delta\text{t-incr} * \text{t-Factor}^{i-2}$$

$$V_{i,i>1} = \text{Voltage} + \Delta\text{V-incr} * \text{V-Factor}^{i-2}$$

In mode *DV, Dt * Factor* the first segment may be zero and is then logarithmically incremented. Let *Voltage = 0 mV*, *DV-Increment = 1 mV*, and *DV-Factor = 2* then the series 0, 1, 2, 4 mV, ... is obtained.

In the galvanostatic mode, the analogous increment methods are possible but *Voltage*, *VFactor*, *ΔV*, have to be replaced by *current*, *I-Factor* and *ΔI*.

Charge Reset: This list specifies if and how the value of integrated current is to be reset. The options are:

- **No Charge Reset**
- **Q-Reset before Sweep**: Charge is reset before starting the following Sweep.
- **Q-Reset before Series**: Charge is reset before starting the following Series.

Write/Show Options: These settings determine saving and display options during stimulation:

- **Write Enabled**: Data are saved to file.
- **Write Disabled**: If this flag is set, no data are written to disk even if the Store control in the *Oscilloscope* window is activated. This feature allows to execute test sequences or conditioning series that should not be stored (without turning Store off).
- **No Write no Show**: If this flag is set, no data are written to disk even if the Store control in the *Oscilloscope* window is activated, and the data are not displayed on the oscilloscope. This option might be useful, e.g., to apply conditioning pulses followed by a test scan.
- **Write no Show**: If this flag is set, data are written to disk but the data are not displayed on the *Oscilloscope* window. This option will increase the maximal repetition rate of acquisition by neither displaying nor analyzing a sweep during the acquisition.
- **Inactive**: If this flag is set, no stimulation at all will occur.

Auto Range: There are four options which can be selected for each series (note that auto ranging applies to the current):

- **Auto Range Off**: No auto ranging .
- **Auto Range Peak**: *POTPULSE* switches, if 10 contiguous points or 50 points total are above or below the threshold.
- **Auto Range Mean**: *POTPULSE* switches, if the mean of the complete sweep is above or below the threshold.
- **Auto Range Yseg**: *POTPULSE*switches, if the mean of the selected range in the relevant segment is above or below the threshold.
- **Auto Range Sweep**: *POTPULSE* switches, if any part of the sweep is above or below the threshold.

Auto Range thresholds Min and Max values have to be set in the *PG310/390 Potentiostat* window. Note that Auto Range is only possible in Slow Aquisition Mode**.**

Amplitude Mode: The relation of stimulated voltage to initial potential can be defined:

- **Absolute Stimulus:** The default case is Absolute Stimulus, the stimulus is built up with the use of the segment voltages as the output voltage, and the stimulus is scaled by the Stimulus Scale given in the *Configuration* window. For instance, a segment of +100 mV will in fact be output as this value.

$$V_{out} = \frac{V_{segment}}{Stimulus\ Scale}$$

- **Relative Stimulus:** If Relative Stimulus is activated, the stimulus is built up with the use of the segment voltages as relative changes from the initial potential or current. Moreover, the stimulus is scaled by the Stimulus Scale given in the *Configuration* window. If an initial potential of 80 mV and a segment voltage of +100 mV are assumed, an output yielding +20 mV is created.

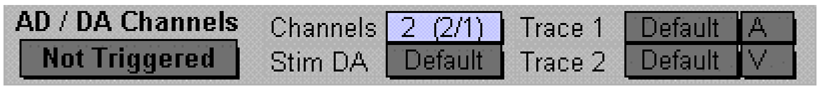
$$V_{out} = \frac{V_{segment} - V_{E-initial}}{Stimulus Scale}$$

- **Absolute Voltage:** If Absolute Voltage is selected, voltage output will be the segment voltage (same as Absolute Stimulus) and the stimulus will not be scaled by Stimulus Scale. When this option is active, the display of E-Initial (display) as well as the E-Initial checkboxes above the segments are suppressed.

$$V_{out} = V_{segment}$$

Relevant Segments: The Relevant X-Segment specifies a segment of interest, which is mainly used as X-axis reference for later analysis. The Relevant X-Segment specifies the segment where the analysis is performed (e.g., determination of peak current). The relative segment is displayed in red color.

AD/DA Channels



Trigger Mode: Determines if and how data acquisition is triggered by an external TTL pulse. The trigger input is located at the rear of the PG310/390 Potentiostat. The default is Not triggered, which means that stimulation is immediately elicited by the user within *POTPULSE*. Otherwise you have to activate the sequence and then one or more external triggers have to be applied.

- **Not Triggered**: No external triggering.
- **Trigger Series**: One trigger at start of series.
- **Trigger Sweeps**: One trigger at start of Sweeps.

Channels: Number of channels acquired. The default is *1*, the second channel (Channels = *2*) may be used to simultaneously record the potential, for example. For recording the second channel the sample interval has to be larger than 5µs. Note that with multiple channels, each channel is sampled consecutively, not simultaneously.

The values in parentheses indicate the number of input and output channels (inputs/outputs). The user can select two A/D channels (inputs) for sampling (one is normally Imon); hence, the number of inputs is in the range of 1-2. In addition to the Estim output, three triggers are available as D/A channels (outputs); therefore, the number of output channels is in the range of 1-4.

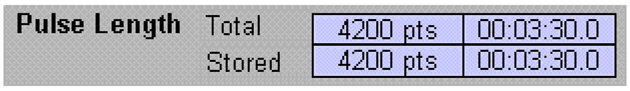
The second trace is shown on the screen when *2nd Trace* is selected from the list of Background Traces in the Display menu.

Stim DA: The DA channel for stimulation has to be specified. If Default is selected, the channel specified in the *Configuration* window is chosen.

Trace 1 / Trace 2: Assignment and units for input channels have to be provided. Defaults are *"A"* for current and *"V"* for voltage. In the shown example, stimulation is output via the default PG310/390Stim-Out channel (DA 3). Trace 1 is recorded from the Default PG310/390Imon-in channel (AD 6). The unit *"A"* is dimmed, because it depends on the recording mode and will be set automatically (e.g., "V" if galvanostatic is selected as recording mode). The second trace is to be read via AD-5 *(*PG310/390 Emon-in*)* and has the unit *"V"*.

Note: Each stimulation sequence can be output and sampled via different DA/AD channels to allow for high flexibility in experiment design. However, it is recommended to always use the same channels in all stimulation sequences except for special purpose applications. Alternatively, use the Default setting for the channels.

Pulse Length



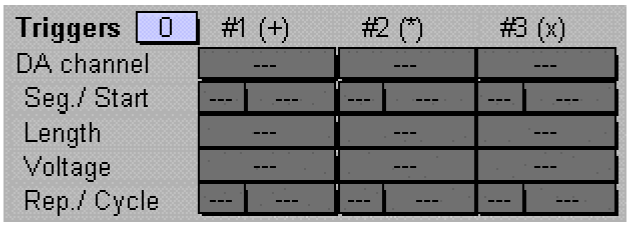
Pulse Length			
	Total	4200 pts	00:03:30.0
	Stored	4200 pts	00:03:30.0

Total: The number of total points and the total time of a sequence is displayed. For the determination of these values the entire sequence is considered, taking into account that segments can change in duration. The maximal time of a sweep of a given sequence is used as 100% time of the display. A possible continuous segment is *not* taken into account when computing this number.

Note: The maximal possible length of a sweep is 1'000'000'000 samples (1 GSample). That would be almost 28 hours at 10kHz!

Stored: Data points saved to disk and length of the sequence segment after the first trigger (see below).

Triggers



Triggers 0	#1 (+)		#2 (*)		#3 (x)	
DA channel	---		---		---	
Seg./ Start	---	---	---	---	---	---
Length	---		---		---	
Voltage	---		---		---	
Rep./ Cycle	---	---	---	---	---	---

The Triggers can be used to control external devices such as oscilloscopes, valves, flash lights, etc., and they are also useful as additional stimulation output, because they can be output via analog channels with specified amplitude and duration.

There are up to three triggers supported. The number of used triggers is defined by the entry Triggers. A trigger can be output via DA channels (DA-1, ... DA-2) or via one of the 8 digital trigger lines (Dig-0, ... Dig-7) available at the DIGITAL-TRIGGER-OUT or DIGITAL-I/O interface on the backside of the PG310/390 or via the TTL OUTPUTs on the front side of the ITC-16 and ITC-18. The trigger DA channel, time, duration, and amplitude are to be specified within one segment of the pulse; it can extend up to the end of the pulse template. If the trigger actually exceeds the pulse template, it is not reset after the pulse. This feature can be used to e.g. turn valves on or off.

The first trigger is used to determine the start of data display and storage; data before the first trigger are *not* stored. For this purpose the DA channel for a trigger can be set to Off; in this case no signal is output but the location of the trigger (if it is the first one) is used to set the start time for data storage.

Using the same DA channel for more than one trigger can create simple pulse patterns in addition to the main output channel. The convention for these patterns is that the DA template for the first trigger is loaded first. The non-zero values for the following triggers are then added, i.e., in case of overlap the DA values will become additive.

The symbols of the triggers as used in the sequence cartoon are given in parentheses.

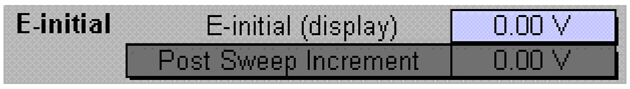
How to set the first trigger correctly

The first of the three triggers is a special one: it is used to determine the start of data display and storage; data before the first trigger are not stored! Therefore, the position of the first trigger has implications to the recorded data.

The position of the first trigger is of particular importance when the voltage of the first segment is not equal to the initial potential. In this case the step from the initial potential to the potential of the first segment will cause a transient capacitive current, so that a perfect digital capacitance cancellation is not possible.

To prevent this, one can create a first segment with a duration such that this initial capacitive transient is over before the actual test pulse starts. The first trigger is then used to define this time, i.e., the time when the baseline is stable enough to be used as first segment (note that the zero current is calculated from the segment after the first trigger). Typical values for the decay of the transients while recording from small cells are a few ms. If a baseline of 5 ms is desired, a first segment of duration 15 ms should be created and the time for the first trigger should be set to 10 ms.

E-initial



E-initial (display): This control shows the currently selected initial potential. It is used for the sequence cartoon as reference and can be changed without actually affecting the initial potential in the Oscilloscope window (E-initial) or of the PG310/390 Potentiostat window (E-initial).

Post Increment: This is used to change the initial potential after sequence execution. The new initial potential is indicated by a small bar in the sequence cartoon at the end of the stimulation. The options are:

- **Post Sweep Increment**: Increments the initial potentials after each sweep.
- **E-Initial**: Changes the initial potentials after a sweep to a specific value.
- **Post Series Increment**: Changes the initial potential after a series.
- **Last Segment Amplitude**: After running a sweep, the initial potential (or initial current in galvanostatic mode) remains the potential (or current in galvanostatic mode) of the last segment of the sweep.

Note: Using the modes other than Last Segment Amplitude, the potential or current will be 0 Volt or 0 A for a short while before the new potential or current is applied. Therefore, the option Last Segment Amplitude is useful in galvanostatic experiments combining several galvanostatic steps, where a 0 A situation between the steps should be avoided.

Macros

For each sequence you can enter the names of two macros. The first macro is executed before starting the Series and the second macro after every sweep. When you enter the name of the macro you will be notified by an alert, if the name refers to an non-existing macro. Of course, only existing macros will be executed. The macro may be used to set specific settings for a given series, such as changing the Online Analysis type.

Stimulus Template Preview

After each editing operation, the stimulus template preview in the lower left corner is refreshed to reflect the changes made. The horizontal dashed line indicates the initial potential level. Triggers are indicated by markers with a horizontal line indicating the length of the trigger pulse.

Sweeps other than the first one are shown as dashed lines. The next initial potential after application of Post Increment is indicated by a bar a the end of the stimulation. Segments drawn in red color indicate the relevant segment as given by Rel Y Seg.

Note: You can disable the display of the stimulus template preview (make it "invisible") by clicking on the small box in the upper left corner of the picture. The cartoon will be replaced by a checkbox which allows you to bring back the drawing of the stimulus template. This is useful if your computer is slow and the pulse protocol is very complex.

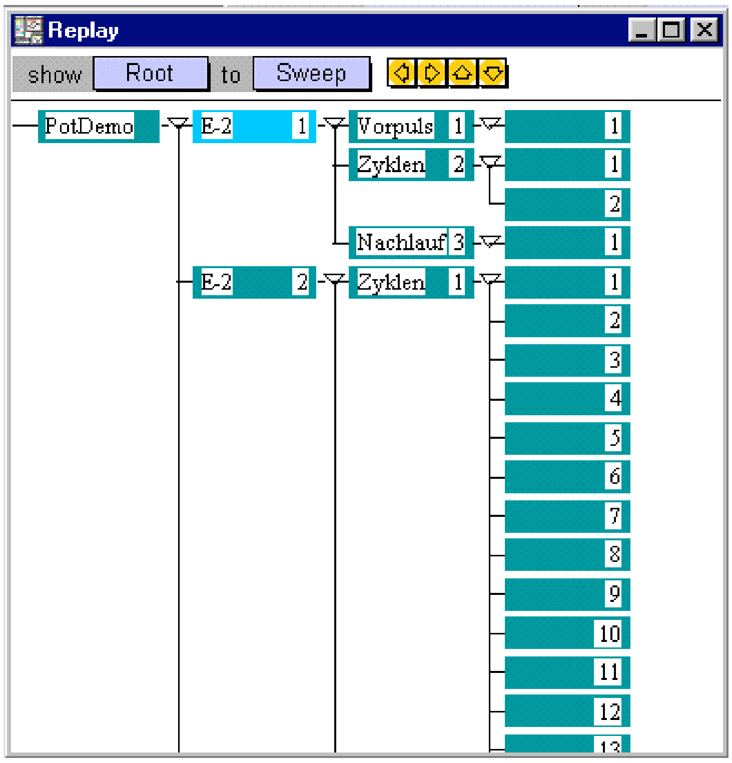
Error Handling

If *POTPULSE* encounters an unreasonable value in the Pulse Generator dialog, the user is requested to change the corresponding parameter before proceeding. In case of multiple errors, one cannot exit the dialog until all parameters are set correctly. If it seems impossible to solve the error situation and you get recursively error messages, you can load a valid PGF file from disk.

Values in the Pulse Generator controls are rounded to the exact values as displayed. This prevents unexpected results caused by rounding problems, such as the Sample Interval being an odd number.

The drawback is that it can thereby happen that a segment duration cannot be both, a multiple of the Sample Interval and the exact value displayed. However, this can only occur if the Sample Interval is not a multiple of 10 µs.

Replay



If data have been acquired in the Store mode or if an old data file has been opened, they can be reviewed and edited in the *Replay* window. To open it select Pulse → Replay or type 'F10' (Windows) or 'F13' (MacOS). Up to four levels of the data tree are displayed (see also Chapter *Data Format*).

The controls show (Root) and to (Sweep) define which part of the data tree has to be displayed. The highlighted part of the data tree is referred to as *Target* throughout this manual.

The 4 "arrow" icons allow to scroll through the data structure of the Tree using the mouse. Alternatively the cursor keys ('LEFT', 'RIGHT', 'UP' and 'DOWN') of the keyboard can be used. 'PAGE UP' and 'PAGE DOWN' can be used to scroll one window up or down. 'HOME' and 'END' will move to the start or end of the Tree. You can also click any entry directly to make it the active one. The following table shows how to step through the data tree in the *Replay* window:

MOUSE / KEY	Function
mouse click on group, series, sweep	makes the selected item the active one
SHIFT + mouse click	marks/unmarks selected item
mouse click on show / hide icon ▷ and ▽	toggles between showing and hiding the children (similar to the behavior of folders in the MacOS Finder and Windows Explorer)
CONTROL + mouse Click	selects the first child to the right, showing it and its sisters if they are hidden
COMMAND + mouse click	toggles the visible state of the selection and sets the state of all children recursively to the selected one, i.e. showing and hiding all children, respectively
CONTROL + CURSOR RIGHT	show children if they are hidden
CONTROL + CURSOR LEFT	hide children if they are visible

MOUSE / KEY	Function
CONTROL + CURSOR UP / DOWN	move up/down and show the next target if it is hidden

The function of keys in this mode are shown below:

KEY	Function	Applicable to
A	Average	Group, Series
C	Compress by factor of two	Root, Group, Series
D	Delete	Group, Series, Sweep
E	Edit entry	Root, Group, Series, Sweep
P	Show PGF template	Group, Series, Sweep
R	Select as reference	Series, Sweep
S	Edit solution	Series
T	Enter text	Root, Group, Series, Sweep
Y	Show PG310/390 state in *Notebook*	Series, Sweep
X	Export according to *Export Mode*	Root, Group, Series, Sweep
Z	Recalculate *Zero Current* value	Root, Group, Series, Sweep
RETURN	Display and analyze	Root, Group, Series, Sweep
CURSOR	Move in data tree	Root, Group, Series, Sweep

Some of these functions only apply if the data file was opened read/write (Modify or Create in File menu), some are restricted to certain kinds of targets. These functions are also accessible via the drop-down menu Tree that becomes active when the *Replay* window is selected (see below).

While the *Replay* window is active, the controls of the upper three rows of the *Oscilloscope* window as well as the list of parameters in the *Parameters* window are used for replay of information. Leaving the *Replay* window causes a restoration of the presently active values.

Besides a running index, text is displayed in the icons of the tree entries. Usually the Root icon contains the file name, the Group icon contains the experiment number, e.g., “E-2”, the Series icon shows the name of the Stimulation Sequence (e.g., “Cyclic”).

During data acquisition, the Sweep icon is empty. It holds the Sweep Label, which can be entered with the Text option from the Tree menu. If, for example, a sweep with a channel opening should be identified among other sweeps with blanks, you can select this sweep, then press “T” for text (or go via the entry “Text” of the Tree Menu,) and enter a sweep label, e.g., “Channel”. This way it is easy to find certain sweeps for analysis or printout.

Note: MacOS and Windows functions for text editing, like ’COMMAND’ ’C’ or ’CTRL’ ’C’ for copy and ’COMMAND ’ ’V ’ or ’CTRL ’ ’V ’ for paste can be used.

The raw current traces as well as the individual entries of the Tree data structure can be edited. In addition, the Buffer and Mark commands allow to perform quick data analysis without the need of further analysis programs.

All editing operations are invoked inside the active *Replay* window by keys or by the entries of the drop-down menu Tree. The following section describes how to edit stored data and how to extract and manipulate information from a data file using the sweep buffer commands. If the active data file was opened using New or Modify in the File menu, stored data can be modified by several commands.

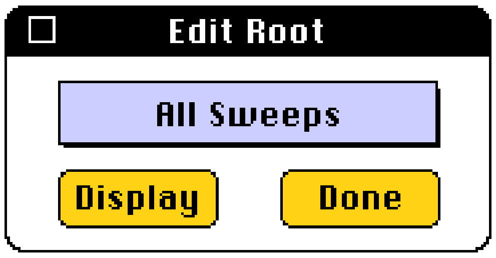
Editing Raw Data

Besides deleting an entire tree branch (Tree → Delete Traces), data can be reduced in size by the Compress option. A time compression by a factor of two is performed and the Sample Interval in the corresponding entry in the stimulation file is doubled. Therefore, Compress cannot be applied to individual sweeps.

The option Average allows to pool compatible data sets. The average of a series of identical sweeps will result in a series with one sweep only. The entry Averages in the Series Record will be updated accordingly. Averaging a Group will result in a group with one Series only. Be sure that all series in this group were generated with the same template! It is assumed that individual bad sweeps have been deleted before averaging.

Editing the Pulsed Tree

Even if one is careful, it will happen that one forgets to enter some parameters during an experiment. Therefore, each entry in the Series Record can be edited at any instant. Select a target in the tree and type ‘E’ or select Edit from the drop-down menu Tree.



A small dialog will appear that holds a list of parameters to be edited, and the controls Modify and Done. If the file was opened as read only, then one can only Display the entries. Among the parameters are entries that point to higher branches in the tree, e.g., All Sweeps. By using such an entry, many parameters of the same kind can be edited at once. The entries that can be displayed or modified for each tree element are listed below:

Edit Root:

- All Sweeps
- All Series
- All Groups

Edit Group:

- All Sweeps
- All Series
- Text
- Label
- Experiment Number
- Extra Value

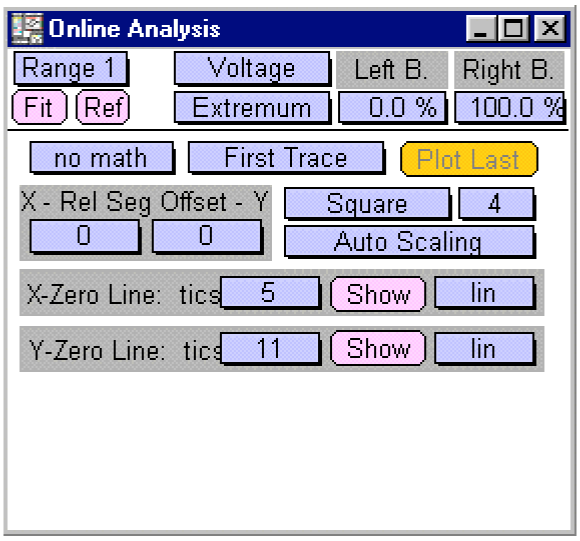
Edit Series:

- All Sweeps
- Comment
- Time
- Timer
- Bandwidth
- Background Noise
- Y-unit 1
- Y-unit 2
- UserParam 1: Name
- UserParam 1: Value
- UserParam 1: Unit
- UserParam 2: Name
- UserParam 2: Value
- UserParam 2: Unit
- External Phase
- Temperature
- Extra Value 1
- Extra Value 2
- Extra Value 3
- Extra Value 4
- E-zero
- Electrode Area
- Solution

Editing the Stim Tree

The stimulation file of any experiment can be shown in the *Pulse Generator* window with the option Show PGF Template from the Tree menu or by typing ‘P’. It will display the stimulation corresponding to the selected target series (or the first series, if a group is the target). The only parameters that can be changed afterwards are the Relevant Segments.

Online Analysis



The online analysis is thought to give a quick overview of some characteristics of the just acquired or replayed data on the level of a series. For the determination of values for the abscissa and the ordinate, the Relevant X-Segment and the Relevant Y-Segment as specified in the Pulse Generator plus the respective relative segment offsets are used.

POTPULSE will automatically plot the analysis to this window after or during execution of a series (based on the settings made in the various controls inside this window). To access additional plot options, click into the graph plot area. The current plot will be erased and the plot options will be shown.

Type of Online Analysis

Range: Two time windows for analysis can be set independently, they correspond to two calculations:

- **Range 1** (Online analysis based on the first cursor pair)
- **Range 2** (Online analysis based on the second cursor pair)

The default range is determined by the relevant segment as specified in the *Pulse Generator* plus the respective relative segment offsets. The analysis of the selected range will be shown in the graph.

Fit: When selected, the numerical analysis values are determined and shown in the *Notebook* window whenever a new sweep is taken or a group/ series / sweep is replayed. Extremum, Maximum, Minimum, and Time to Peak are determined by a cubic polynomial fit; a section of the fit function is superimposed on the data trace.

Ref: This control is highlighted when a reference analysis was selected (for selecting a series as reference see Chapter *Menus>Tree Menu* and Chapter *Replay*). Click on this control results in deactivation of the reference analysis.

Abscissa (X) : The analysis result is shown in the graph as function of a parameter as specified by the variable Abscissa. The source for these parameters is determined by the X-Rel Seg Offset relative to Relevant X-Segment. Currently the following Abscissa are supported:

- **Voltage**: Relevant segment voltage.
- **Duration**: Relevant segment duration.

- **Time**: Time from start of series.
- **Timer Time**: Time of the timer in the *Oscilloscope*.
- **Realtime**: Real time of sweep execution.
- **Index**: Index of sweep within series.
- **Peak Voltage**: The voltage applied at the position of peak current. It may be useful when analyzing ramps.
- **Scan Rate:** Scan Rate of the relevant segment.
- **Y1 versus Y2**: Plots the Y-result of Range 1 versus the Y-result of Range 2.

Ordinate (Y) : The type of analysis is specified by the variable Ordinate. Currently the following types are supported:

- **No Analysis**: Skips data analysis.
- **Extremum**: Extreme of current: may be the maximum or minimum.
- **Maximum**: Maximum of current.
- **Minimum**: Minimum of current.
- **Time to Peak**: Time to peak of current within specified segment.
- **Mean**: Mean of current.
- **Charge**: Integral of current.
- **Variance**: Variance of current (square of standard deviation).
- **Slope**: Slope of current (by linear regression).
- **Reversal (theo)**: Finds next zero crossing, starting at the left bound and searching to the right, using an average of 3 data points. Vref is then computed from the stimulus template.
- **Reversal (2.trace)**: Equivalent to Reversal (theo) above, but gets Vref from the voltage at the corresponding position in the 2. Trace (which is usually the voltage trace).
- **C-Slow**: Slow capacitance taken from each sweep (not relevant for POTPULSE).
- **G-Series**: Series conductance taken from each sweep (not relevant for POTPULSE).
- **Anodic Charge**: Integral of positive current.
- **Cathodic Charge**: Integral of negative current.

Note: The total charge is calculated as the difference between anodic and cathodic charge ($Q_{total} = Q_{anodic} - Q_{cathodic}$). The partial charges are calculated as integrated values of the respective positive or negative current densities:

$$Q_{anodic} = \int_{t_1, i \geq 0}^{t_2} i dt \text{ and } Q_{cathodic} = \int_{t_1, i < 0}^{t_2} |i| dt$$

Left / Right Bound: Within a segment, Left Bound and Right Bound (in %) determine the actual time limits. These parameters can also be set using the cursors of the *Oscilloscope* window. They can be outside the limits of the selected segment, i.e., they can be smaller than 0% and greater than 100%.

Note: Online analysis will always be performed between the right and left bounds. Although a relevant segment is given by the Pulse Generator, the time range is set by the cursors in the *Oscilloscope* window. That means that if you move any cursor in the *Oscilloscope* window, this action will change the bounds of Online Analysis in the respective range and exhibit these in relation to the relevant segment. To ensure analysis of the desired range, activate the cursors in the *Oscilloscope* window .

Display Options

To access additional plot options, click into the graph plot area. The current plot will be erased and the plot options will be shown.

Math: Allows different calculations on the two Y-results. If a further calculation is required (i.e. the Math Type is set to anything else than no math), the *Online Analysis* window will show the calculation instead of the two online results. A pop-up window allows selection to determine whether

- **no math**: No further calculations, both Y-results are plotted.

  or one Y-result is computed as

- **y = y1 + y2**: Calculates and shows the sum of the two Online Analysis results.
- **y = y1 - y2**: Calculates and shows the difference Range 1 minus Range 2.
- **y = y1 * y2**: Calculates and shows the product.
- **y = y1 / y2**: Calculates and shows the quotient Range 1 divided by Range 2.

Plot Last / Clear: This will redraw the last plot using the modified plot parameters or erase the current plot.

Rel Seg Offset: The relevant segment specified in the PGF file sets the default for the analyzed segment. You can change that here by setting an x- and y-offset relative to the relevant segment.

Note: An online analysis with the use of the peak voltage must be performed in the segment, in which the peak occurs. Therefore, use the Relevant Segment Offset option or set the relevant segment in the Pulse Generator to this segment, which should be analyzed; do not set the cursors out of bounds with respect to the relevant segment in the case of analyzing peak voltages.

Symbols / Size: The shape and size of the plot symbols can be set:

- Point Symbol .
- Plus Symbol +
- Star Symbol *

- Diamond Symbol ?
- Cross Symbol x
- Square Symbol ?

Zero Line - X-Y Tics: Sets the number of tics plotted on the axis in case Show is activated. Decimals are rounded. A value smaller than 0.4 suppresses the display of axis tics.

Zero Line - Show: Selects if the zero-line is drawn.

Zero Line - lin / log / exp / sqr: Defines if the respective axis is drawn linear (default), logarithmic, exponential or as square root.

Scaling

The scaling of the online analysis data can be fixed or automatic (i.e., auto ranging).

Fixed Scaling: Online analysis results can be shown in the graph immediately after acquisition or replay of a sweep, if the scaling is known from the beginning. Therefore, a Fixed Scaling option is provided and the X-Y-scaling of the analysis graph can be specified. Now, the program knows the scaling before completion of a series, i.e., the online analysis results can be displayed in the graph after each sweep, which makes it a true online analysis.

If Fixed Scaling is selected, a further set of graphing options will appear:

- **Copy Last**: This will copy the scaling of the last plot range into the fixed min/max. controls. The scaling of Range 1 will be used, if two ranges are available.
- **Copy Other**: Copies the “other” scaling, i.e., if Range 1 is currently active, the settings of Range 2 will be copied and vice versa.
- **Overlay Options**: If axes are determined by fixed scaling, the time evolution of the online analysis plot can be selected. There are three options that determine how often the graph window is wiped out:
    - **No Overlay**: This is the default mode: whenever performing a new analysis, the graph window will be cleared.
    - **Time Wrap:** Time axis wraps around, when time is selected as abscissa. The plot is cleared when it wraps. However, the values of the time axes will be held.
    - **Overlay + “T”-Wrap**: Overlay analysis is possible for all analysis types, i.e. the plot is not cleared when it wraps. The time axis wraps as in option Time Wrap.

Auto Scaling: After all sweeps of a series have been acquired or replayed, the maximal and minimal values of the currently selected abscissa and ordinate values are determined and used to scale the graph.

Cont. Auto Scale: This selection allows to display the Online results with auto scaling immediately after every acquired Sweep (beginning with the second Sweep).

Note: If a timer function ("Time", "Timer", or "Real Time") is selected as abscissa, data points are plotted in the given bounds until the time exceeds the high end of the abscissa scale: this causes a wrap-around of the axis scaling. Thus, the graph is cleared and redrawn from the left end.

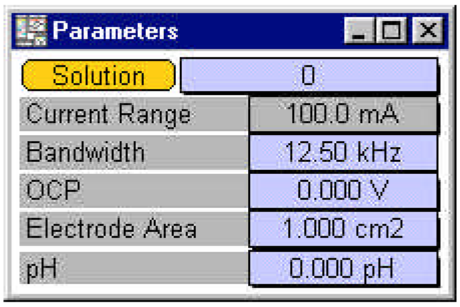
Axis Scaling: The axis ranges for the fixed scaling can be entered as min/max values.

Exporting Data

With the online analysis features classical analyses like current-voltage relationships can be performed easily during data replay and acquisition. The result of the Online Analysis is also displayed in the *Notebook*. From the Notebook these data columns can be written to disk or copied to the clipboard.

Alternatively, the information written to the *Online Analysis* window can be output by using the Tree → Export function if the Export Mode includes Online Analysis.

Parameters



Solution	0
Current Range	100.0 mA
Bandwidth	12.50 kHz
OCP	0.000 V
Electrode Area	1.000 cm2
pH	0.000 pH

The parameters as selected in the *Configuration* window (see *Chapter Configuration*) are shown in a separate window. They are determined before acquisition of each series or each sweep depending on the parameter. In the test pulse mode (i.e. if the option Stimulate in the *Potentiostat* window is active) the parameters are determined before each pulse, otherwise they are updated continuously, i.e., the repetition interval is just limited by the CPU time used.

Solution Numbers:

Solution Number: The top row displays the currently used identifiers for internal and external solutions. The solutions are given as integer numbers (0 to 2’147’483’648), when the “solution source” is set to “manual” in the *Configuration* window. The solution names are listed in a pop-up list, when the “solution source” is set to “DataBase” in the *Configuration* window.

Solution: This button was designed to access the Solution menu, but is actually not functioning, so to open the solution menu choose POTPULSE → Solution base instead. In this menu new solutions can be specified or can be selected from the common solution data base. (For more information see Chapter *Solutions*).

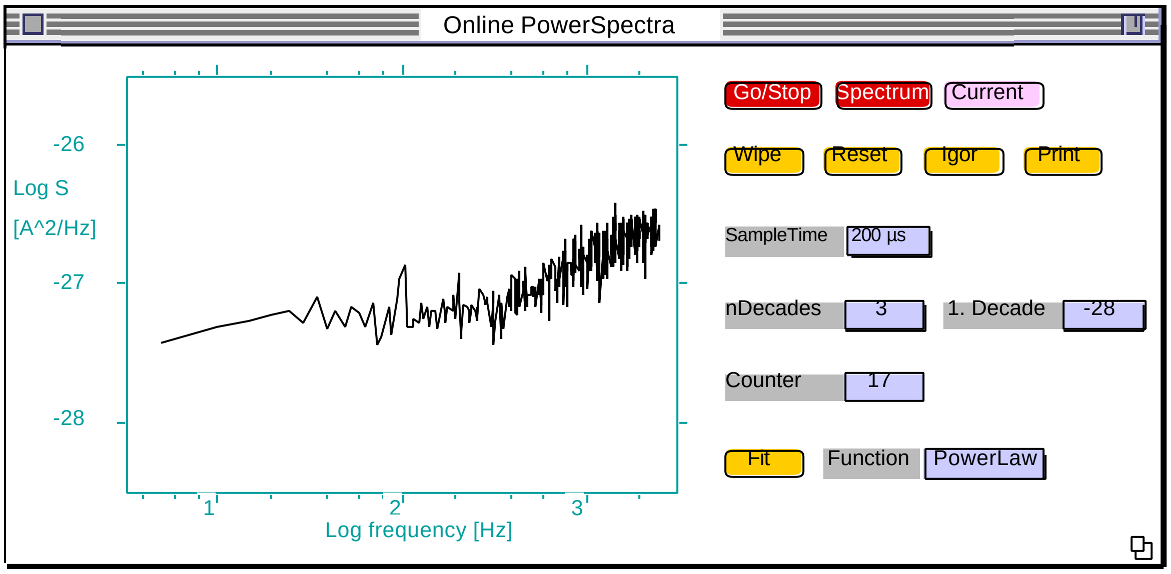
Parameter Values

Parameters whose source is specified to be Default can be edited in this window (in the case shown above: pH (pH is a user parameter); i.e., one does not need to edit the corresponding entries in the *Configuration* window. Parameters that are read from other sources cannot be edited and they are shown in a different color.

After a replay action the parameter list is updated according to the replayed data. A mouse click in a window other than the *Replay* window restores the actual parameter settings of the experiment.

Power Spectra

Power Spectra can be measured, displayed, and averaged online. This mode is accessed via the drop-down menu POTPULSE → Spectra or by pressing 'F10' (MacOS). It is helpful for troubleshooting and for testing the frequency response of the recording system. The power spectrum is repetitively acquired, averaged and displayed.



SampleTime 200 µs

nDecades 3 1. Decade -28

Counter 17

Function PowerLaw

Go / Stop: Starts or stops acquisition and accumulation of power spectra.

Spectrum: If enabled, the display of the spectrum will be updated after every FFT cycle.

Current: Displays the current trace in red color in addition to the power spectrum.

Wipe: Clears the display.

Reset: Restarts accumulation of power spectra.

Igor: Outputs the spectrum in "IGOR Text" format.

Print: Prints the spectrum.

Sample Time: Sets the sample interval. This determines the frequency band because a fixed number of 1,024 data points is always used for FFT calculation. Available sample intervals are:

- 2 ms
- 200 µs
- 20 µs
- 5 µs

Counter: This is the number of spectra that were accumulated and averaged.

n Decades: Number of spectral density decades to be displayed.

1. Decade: First displayed decade.

Power Spectra Fit

Fit: Draws the fit function. Clicking on it allows to enter the parameters for the fit function: zero-limit, sloop, exponent.

Function: Simple functions can be drawn; no automatic fit is performed. In the functions to follow f denotes the frequency, and S the spectral density. The values of the free parameters have to be input via the *Notebook* terminal.

- **Power Law**

  $$S(f) = S_0 + \text{slope} * f^{\text{exponent}}$$

- **One over ¦**

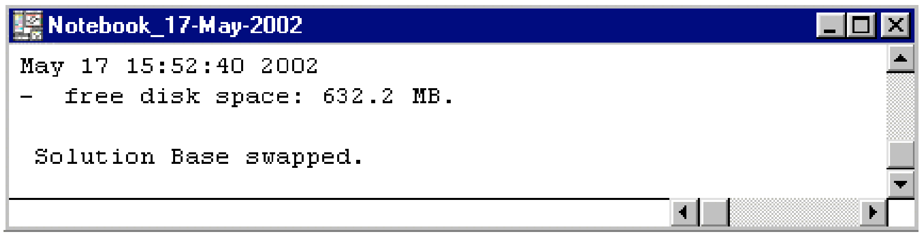
  $$S(f) = S_0 + \sum_i \frac{a_i}{1 + f / fc_i}$$

- **Lorentzian**

  $$S(f) = S_0 + \sum_i \frac{a_i}{1 + (f / fc_i)^2}$$

Notebook

The *Notebook* window is used to display messages and warnings of the program, such as error messages, analysis results, and information about displayed data. The content of the Notebook can be stored on disk; its maximal size has to be specified in the drop-down menu Notebook → Set Length. To keep a better log of an experiment, the names of opened files and of executed series are written to the *Notebook* window.



```
Notebook_17-May-2002
May 17 15:52:40 2002
-  free disk space: 632.2 MB.

 Solution Base swapped.
```

When the window is activated (by clicking with the mouse pointer into it), the text editing functions are activated and applicable in the *Notebook* window. Thus, the *Notebook* window is basically an editor window of the memory-resident text file Notebook. Therefore, one can modify text in the *Notebook* window just as in any other text file. This option can be used to add further information to the text file, or to get rid of messages that should not be stored to the disk file.

The applicable menu commands are described in the Edit and Notebook menus (see Chapter *Menus*).

Note: Text written by *POTPULSE* to the Notebook is only stored, when the option Buffered Output in the Notebook menu is selected.

Note: The Cut, Copy and Paste commands copy to and from the clipboard.

Solutions

In many electrochemical experiments solutions are changed during the experiment. Thus, it is of great importance to keep track of the solutions which have been used.

Besides using the text lines on all Tree levels (Root, Group, Series, Sweep) for the identification of the solutions, *POTPULSE* provides two additional ways to do so. A simple way is using the entry Solution in the *Parameters* window. There is a Longint number that identifies a certain solution. This number can be the entry of an external list of solutions, or the index of a specific solution in a *POTPULSE* Solution Data Base (see below).

For each data file a solution data base (*.sol) can be created. Such a data base is a *Tree* of solution entries ordered by the given identifying index.

Creating a solution entry

At first the Solution Timing in the *Configuration* window has to be set to something other than Off. This means that for a new index an entry in the solution data base is created.

Then the options in Solution Timing are used to control when the input of the solution data is necessary.

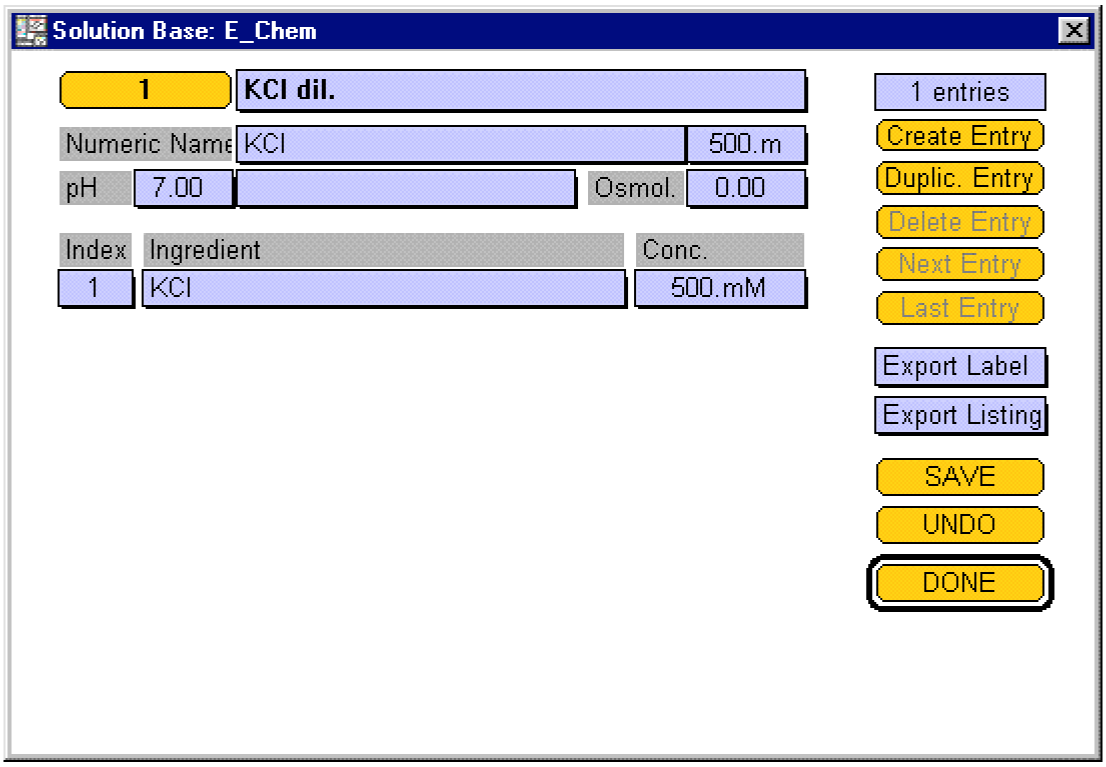
- When Before Series is set and the option for Solution Source in the *Configuration* window is set to Data Base, *POTPULSE* will first search the common solution data base for this entry. The name of that common data base is defined in the *Configuration* window. If the desired solution is not found in the data base, you have to enter the new solution manually.
- During a time-critical experiment in which delays due to entering new solutions are not desired, the options After Series, After Groups and End of File are recommended, since the data is entered only at the end of a series, at the closure of the active group or the active file, respectively. This means that after a series has been acquired with an unidentified solution index, the solution dialog will come up and will request entry of the specifications of this new solution.

Using Solution Indices

The indices can be given to solutions in any arbitrary way but it certainly is of advantage if one sticks to some consistent concept in order to be able to identify solutions by their index easily. Since the solution indices as LONGINT numbers can range from 0 to 2'147'483'648 (many more than you will find bottles in the laboratory), there is plenty of freedom to organize them. Here is an example:

Usually one has several standard solutions, which are frequently used and modified slightly for various experiments. One could assign numbers divisible by *1000* or *10000* to them. Then one has *999* or *9999* possibilities for modifications of this solution, respectively. An example would be that certain concentrations of a corrosion inhibitor are added to the standard solution 1000, thus yielding numbers 1001 through 1099. Another corrosion inhibitors could occupy the numbers 1100 through 1199, etc. .

Solution Data Base


Solution Base: E_Chem

1	KCl dil.		
Numeric Name	KCl		500.m
pH	7.00	Osmol.	0.00

Index	Ingredient	Conc.
1	KCl	500.mM

The solution data base can be edited using the POTPULSE drop-down menu entry Solution Base. A subset of this dialog appears if one is requested to enter a solution into the *POTPULSE* data structure.

Solution Index: Index number for the solution.

Name: Name of the solution.

Numeric Name: An editable field that may hold a feature of the solution that is not easily determined from its ingredients (e.g., free calcium concentration, etc.).

pH: Holds the pH-value of the solution and the substance used to adjust it.

Osmol. : Holds the value of osmolarity of the solution.

Index / Ingredient / Conc.: Each ingredient is defined by its index number, ingredient name and concentration in the solution.

- **Index:** Index number of the ingredient in this solution. To add a new ingredient click on Index and choose Insert for inserting at the actual position or Append for appending a new ingredient at the end of the list. With Delete the active ingredient is removed from the list.

- **Ingredient:** Name of the ingredient. The maximum length is 28 digits**.**
- **Conc.:** Concentration of the ingredient in Mol (the unit is adjusted to mM, if necessary).

Entries: Number of entries in the solution data base.

Create / Duplicate / Delete Entry: Used to generate, copy, and remove solutions in the data base.

Next / Last Entry: This option moves through the data base by selecting the next or last available solution.

Export Label: This is used to output labels of the shown solution:

- **ASCII File**: Exports the solution information as ASCII text.
- **Igor Text**: Exports the solution information as IGOR text.
- **Printer**: Prints the solution information. Two labels are created, a bigger one which can be used to label a bottle, and a smaller one which fits on a syringe.

Export Listing: This is used to output a list of the entire solution file:

- **ASCII File**: Exports the solution file as ASCII text.
- **Printer**: Prints the solution file.

Save: Saves the file to disk.

Undo: Reverts the edited solution to its original form.

Done: Exits the dialog.

Note: A user can generate the complete missing solution tree as follows. In the *Configuration* window one sets the Sol.Source to Data Base and Sol.Timing to anything but Off. Then one has to open the data file in modify mode. Finally, he can clicks on the root in the *Replay* window, and select the option Tree → Solution.

Keys

Most controls within windows can be accessed from the keyboard. Menu functions are called by a combination of 'CMD' (MacOS) or 'ALT' (Windows) plus a further key (e.g., 'CMD' / 'ALT' + 'Q' for File → Quit); they are always accessible. Some global functions are called by a combination of 'CTRL' and a key (e.g., 'CTRL' + 'B' for Break). Some single keys are also implemented in *POTPULSE* (e.g., 'E' for Execute).

Window-dependent Keys

In addition to these global keys, each individual window can have further keys assigned to each control:

KEY	Function
SPACE	Toggle between *Potentiostat* and *Oscilloscope* window
1, 2, 3, ... 9	Execute sequence with this index number
E	Execute selected sequence
BACKSPACE	Wipe screen
ESC	Switch to *Oscilloscope*, closing the front window
CURSOR + LEFT / RIGHT	Change E-initial by 10 mV
SHIFT + CURSOR LEFT / RIGHT	Change E-initial by 1 mV
Z	Perform *Zero Subtraction*
CTRL + Z	Wipe screen
CURSOR UP / DOWN	Change PG 310/390 current rate
HELP	Show *Help* window (not supported yet)
OPT + HELP	Show *Key* assignment (not supported yet)
CTRL + CURSOR	Move in *Tree* (from any active window) MacOS only
CTRL + RETURN	Display *Tree* target (from any active window) MacOS only
	Edit *Initial Potential* Windows only
CTRL + B	Interrupt sequence execution (Break)
CTRL + I	Pause sequence execution (Wait)
CTRL + E	New experiment
CTRL + L	Stop and continue with linked sequence (Link)

KEY	Function
CTRL + N	New group
CTRL + Q	Resume paused sequence execution
CTRL + R	Show remaining buffer space
CTRL + S	Stop sequence execution (Stop)
CTRL + T	Edit Root text (Text)
CTRL + W	Toggle Store control (write/no write)

The option Help → Show Keys displays the key assignments of the various windows. Many key commands are active in all windows (e.g., the key assigned to the Overlay option will toggle Overlay from all windows).

Keys accepted during Data Aquisition

During data acquisition, *POTPULSE* will not accept mouse clicks (except for the Break button). However, *POTPULSE* will accept a limited set of keyboard events during data acquisition:

KEY	*Function*
'.'	Overlay all
'T'	Reset Timer
'BACKSPACE'	Wipe screen
'CTRL' + 'W'	Toggle Store before next series
'CTRL' + 'B'	Interrupt sequence execution (Break)
'CTRL' + 'I'	Pause sequence execution (Wait)
'CTRL' + 'L'	Stop and continue with linked sequence (Link)
'CTRL' + 'Q'	Resume paused sequence execution
'CTRL' + 'R'	Show remaining buffer space
'CTRL' + 'S'	Stop after sequence execution (Stop)
'CMD' / 'ALT' + 'I'	Show file status info
'OPT' / 'ALT' + '1', '2', '3', ...	Mark a sweep in the tree with the corresponding number
'SHIFT' + ANY KEY	Executes the Amplifier control with the corresponding letter

KEY	*Function*
NUM '+' OR '-'	Change display scaling
NUM '*'	Center sweep in display
CURSOR UP OR DOWN	Change Current range before next series
SHIFT + CURSOR UP OR DOWN	Change Current range before next sweep
CURSOR LEFT OR RIGHT	Change E-initial before next series

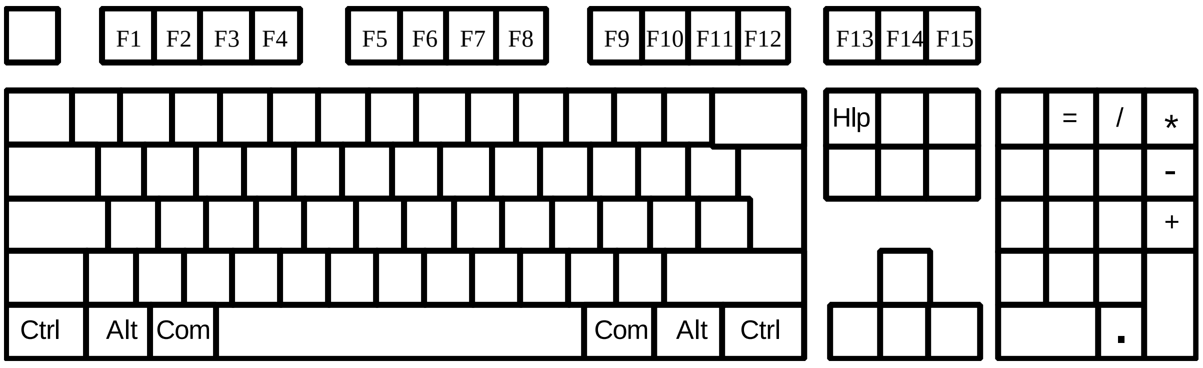
One can send key strokes to the Amplifier window during the waiting periods between sweeps, if one holds down the 'SHIFT' key. Yet, be aware that changing the amplifier state within the acquisition of a series may make the stored information inconsistent, e.g., when one changes the holding potential within one series. Even worse, one can call a macro which switches modes or gets into a recursive macro call. This may cause the program to crash!

Differences between MacOS and Windows Version

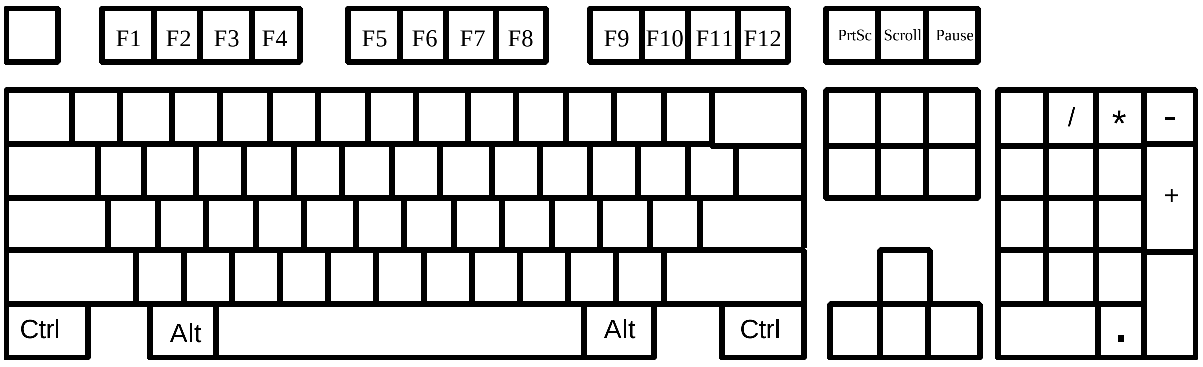
Command Key Differences

The differences are due to the windows keyboard, which has fewer functions, modifier keys, and numeric keys.

This sketch shows schematically the MacOS extended keyboard:



The Windows keyboard is sketched here:



The labeled keys are the ones which differ. They are discussed below.

Function Keys

The windows keyboard has function key F1 through F12, as compared to F1 through F15 for the MacOS extended keyboard.

Option	MacOS	Windows	Alternative
Switch to Window			
Oscilloscope	F 15	F 12	Esc
Potentiostat	F 14	F 11	Space
Replay	F 13	F 10	
PGF	F 12	F 9	
Configuration	F 11	F 8	
Power Spectra	F 10	-	
Parameters	F 9	-	
Analysis	F 8	F 7	
Notebook	F 5	F 5	
in Notebook:			
Copy One Line	F 3	F 4	
Cut One Line	F 2	F 3	
Paste One Line	F 1	F 2	

Modifier Keys

Under Windows the “Alternate” key replaces the MacOS “Command” key. It is used for the menu shortcut keys (in Windows lingo called “accelerator” keys).

Therefore, the functions of the MacOS “Alternate” key (also called the “Option” key) are not available under Windows.

Option	MacOS	Windows
Increase Display Scaling 1	Num +	Num +
Decrease Display Scaling 1	Num –	Num –
Auto Display Offset 1	Num *	Num .
Increase Display Scaling 2	Option Num +	Control Num + [1]
Decrease Display Scaling 2	Option Num –	Control Num – [1]

Option	MacOS	Windows
Auto Display Offset 2	Option Num *	Control Num * [1]
Move Page Left	Num =	Num /
Move Page Right	Num /	Num *
Increase Initial Potential by 1 mV	Option >	Control > [1,2]
Decrease Initial Potential by 1 mV	Option <	Control < [1,2]

[1] These key combinations are not available under Windows 3.1.

[2] These key combinations are available in the Potentiostat window only.

Configuring the Dialogs

These functions are available with Capslock down only.

Configuring Dialog	Command mouse click	Control left button click
Move Item	Alt drag	Alt right button drag
Move Group	Control Alt drag	Control right button drag

Data Format

In this chapter we describe the general structure of the files generated by *POTPULSE*.

Data Files

POTPULSE generates up to five files when you create a data file:

1. The *Data* file (file extension *.dat).
2. The *Stimulus Templates* file (file extension *.pgf).
3. The *Acquisition Parameters* file (file extension *.pul).
4. The *Notebook* file (file extension *.txt).
5. The *Solution Data Base* file (file extension *.sol).

Except for the raw data file and the Notebook file, all other files have a Tree structure. The entire trees are kept in memory, whereas the raw data traces are always loaded from disk, when needed.

Raw Data

This raw data file (extension "dat") is a continuous data stream. Each data point is a 16-bit signed integer (exceptions are explicitly mentioned below). When a sweep is stored, *POTPULSE* stores the various single traces, if available.

Inside the program, the data are processed as *Real* numbers. This helps to avoid round-off errors during data averaging procedures, for example.

Stimulation Template: Stim

Stores the stimulation protocol. The structure of the *Stimulation File* (extension "pgf") is defined by the *Definition Module* "Stim.de" (see *Appendix I*). The *Stimulation File* has a tree structure:

Record	Description
Root	Version number
Stimulation	Description of an ensemble of pulse patterns; e.g., I-V curve
Segment	Individual segment of a pulse pattern

Stimulation files can be loaded into the *Pulse Generator*. In fact, the *Pulse Generator* files for the stimulation protocols used during the experiments have the same data structure as the PGF-files, which belong to the recorded data. In this way it is possible to exactly repeat an experiment by using a copy of a PGF-file as *Pulse Generator* file.

Data Description: Pulsed

Stores parameters, such as current range, IR-Compensation, etc. The pointer to the data stored in the raw data file is also contained in this file. The structure of the *Pulsed File* (extension "pul") is defined by the *Definition Module* "Pulsed.de" (see *Appendix I*). The *Pulsed File* has a tree structure:

Record	Description
Root	Version number, text, time, file name
Group	Larger section of an experiment; e.g., cell
Series	Description of an ensemble of traces; e.g., I-V curve
Sweep	Description of an individual data trace

A graphical template of the *Pulsed File* (*Tree*) is shown in the *Replay* window. It contains information necessary to reconstruct the experimental conditions as the data were recorded.

Notebook File

The notebook file is a standard ASCII text file with line breaks.

Solution Data Base: Solution

The fourth file (optional) that *POTPULSE* writes contains a data base of the solutions used during the experiment. For each series, a solution can be specified by values between 0 and 2'147'483'648 (see variable in *Series*: *InternalSolution* and *ExternalSolution*). A more detailed description of these solutions is stored in the *Solution File* (extension "sol"). This file has a tree structure as defined by the *Definition Module* "Solution.de" (see *Appendix I*):

Record	Description
Root	Version number
Sol	Description of a solution
Chemical	Description of each ingredient

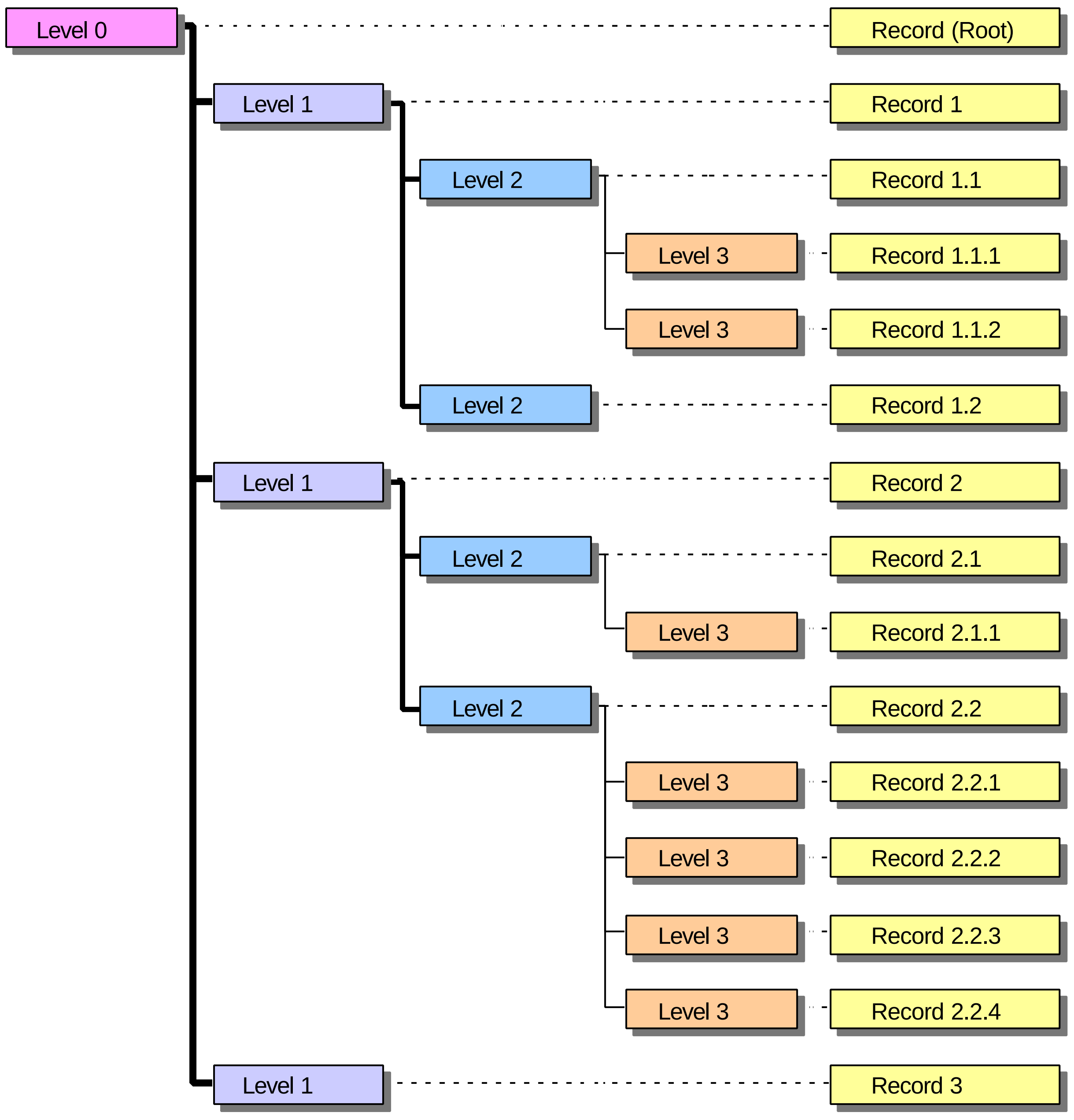
One entry in the *Sol record* is *NumericalValue*. It specifies a parameter of the solution that is not easily determined from the ingredients, like the free calcium concentration of the solution, for example. It can be used later for analysis purposes, such as to plot current as function of the calcium concentration, for example. The assignment of numbers to solutions must be unambiguous within one data file. It is recommended, however, that unambiguous assignments are used within one laboratory to make life easier. To support this, *POTPULSE* makes use of a default common solution data base which is called "E_chem.sol". This data base is a file of the same format as the other “*.sol” files. It is thought to be a generally accessible data base of all solutions used in the laboratory (or at least within one project). During the experiment the user can select solutions directly from this base to store them in the current data file; this saves a lot of typing and reduces the number of errors. More solution data bases can be used; the requested name can be specified and saved in the *Configuration* window.

The Tree Format

The idea of *POTPULSE* is to order the data of an experiment in *Trees*. The trunk of the tree (*Root*) is the main descriptor of a data file (it could, for example, correspond to one cell or the entire experiments of one day). The next level is the *Group*. This level can be defined by the user to identify data that belong together. An example would be to open a new group for each temperature. The group may contain several families of records. Such a family (e.g., records of a current–voltage relationship) is called *Series*. The individual records of a family are called *Sweeps*. Finally, each sweep is composed of two traces. A copy of this data tree is accessible to the user throughout the experiment (so one has an overview of what was recorded, and one can immediately edit the entries (e.g., discard bad records).

The following is a description of the *Tree* format. *Appendix III* has a source code listing for a program that shows how to scan and load a *Tree* file. It can be compiled and executed by the *PowerMod* environment (MacOS only). The source code is commented and can easily be translated to other languages.

An example tree can be diagrammed as follows:



There is only one record of level 0, the root record. The above tree has four levels, the root (level 0), levels 1 and 2, and the leaf nodes (level 3).

The format of a tree stored to a file is as follows:

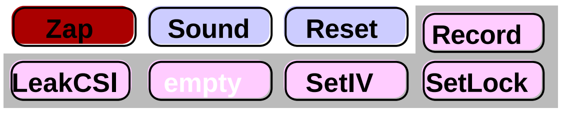
1. Magic number: 054726565H
2. Number of levels
3. Level sizes, one per level
4. Tree records, top down, left-to-right. Each record has the format:
    - Record contents

- Number of children

All of the values (except the record contents) are INT32 values, i.e., 32-bit (4 bytes) values.

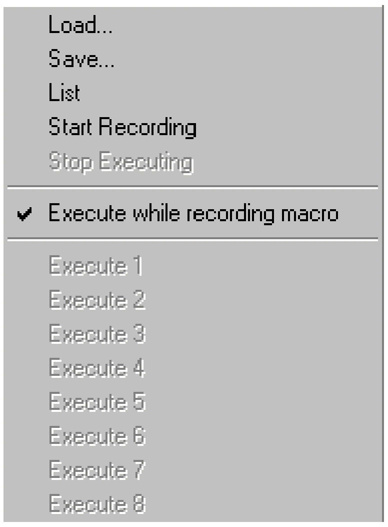
Note: Check the record sizes in the file headers. The record sizes may differ from what you are expecting, e.g., because the file has been created by an older program version which used fewer fields than it is currently using, or a newer version with additional fields. You have to use the record sizes stored in the files themselves.

Macros



One of the great advantages of the PG310/390 is, that in combination with the *POTPULSE* software you are able to greatly automate your experiments. Using Macros instead of directly interacting with the program allows you to increase this automation to a higher level. *POTPULSE* can handle up to 20 macros available through the Potentiostat window and a separate menu. If you used the program you might already have noticed three built in macros: the three buttons CELL, STANDBY and OCP represent three predefined macros that are distributed together with the software. The freely definable macros *#1* to *#7* are located in the bottom of the *Potentiostat* window, whereas macros *#8* to *#20* are hidden in the right part of the *Potentiostat* window. To make them visible you have to expand the *Potentiostat* window to the right.

Macros Menu



All macros are organized within the sub-menu Macros in the PGF-Editor menu.

Load...: Merges a macro file (extension *.mac) to the current macro pool. This will only redefine those macros that are available in the file. If a macro position already used is not defined in the macro file, it will not be cleared.

Save...: Writes the current macro pool to a file.

List: Lists the current macro pool to the *Notebook* window.

Start Recording: Starts (or stops) recording all key strokes and actions the user performs in the *Potentiostat*, *Oscilloscope*, *Online Analysis* or *Parameters* window. During recording a macro the name of the menu item changes into Stop Recording .

Stop Executing: If a macro is running, this will stop its execution.

Execute while recording macro: If this option is not enabled, the user commands will only be recorded but not executed while recording a macro. This is useful if you e.g. want to create a macro that changes amplifier settings but without affecting them while you are recording them.

Execute 1 ... 20: If the corresponding macro has been defined, the menu item will be enabled and entitled by the macro name (e.g. Execute “cyclo1“). Selecting the menu item will execute the associated macro. This is analogous to clicking the corresponding macro button in the Potentiostat window.

Creating Macros

One can easily generate simple macros in *POTPULSE* through the button Record in the *Potentiostat* window and via the PGF-Editor drop-down menu, Macros → Start Recording. If you start recording a macro, the Record button and the menu bar start blinking to notify you about this fact. If you want to finish macro recording click the Record button again to cancel the recording or click on any macro button to assign the recorded actions to that button. Alternatively you can select Macros → Stop Recording from the Potentiostat menu. In this case you will be asked for an index (1 to 20) and an unambiguous name for the new macro.

While the above method works fine for simple macros it might be very inconvenient for more complex macros where you easily make a mistake during recording such as clicking the same button twice or in the wrong order or entering a wrong value. In this case you can store the macro file and modify it using any word processor which can generate plain ASCII text files (*Notepad, BBEdit, SimpleText*). E.g. you can use *Microsoft Word* and save the document as Text Only with Line Breaks, or you can use the editor of *POTPULSE* (see below).

The Editor of POTPULSE

One can use the built-in editor of *POTPULSE* to create or modify macro files. The following is a step-by-step description how to modify an example. This macro sets the amplifier mode to potentiostatic, defines an $E_{initial}$ of 1 and an IR-Compensation, finally executes the waveform "cyclic" and goes to standby.

```
MACRO-FILE        820
  E  Standby:                 TRUE
  E  AmplMode:                 0; Potentiostatic
  E  IinitU:                   1.000V
  E  IRCompOn:                 1;On
  E  IRComp:                 200.0mOhm
  E  Cell:                    TRUE
Execute: cyclic
     # V(3) [mV] t[ms]  iExt[A]
     1  -200.00m  400.00m   199.69m
E  Standby:                   TRUE
```

We want to make sure that the OCP is taken and set before the sweep. This can be done by adding a OCP command into the macro. In addition, we want to save the sweeps This is how the macro should look like (the changed parts are italic):

```
MACRO-FILE        820
  E  Standby:                 TRUE
  0  WriteDate:               TRUE
  E  AmplMode:                 0; Potentiostatic
  E  OCP:                     TRUE
  E  OCPSet
  E  IinitU:                   1.000V
  E  IRCompOn:                 1;On
  E  IRComp:                 200.0mOhm
  E  Cell:                    TRUE
Execute: cyclic
     # V(3) [mV] t[ms]  iExt[A]
```

```
      1  -200.00m  400.00m   199.69m
   E  Standby:               TRUE
```

The first step is to load the macro text from the macro file and display it in the *Notebook* window. This allows us to use the editor of the *Notebook* to modify the macro instructions.

1. First we clear the Notebook by selecting Clear from the Notebook drop-down menu (MacOS CMD+B, Windows ALT+B). We also have to delete the header of the "blank" *Notebook*.
2. Second, we load the text of the macro file with the Notebook → Merge... command.
3. Next, we delete any text which does not belong to the macro text itself, such as the Notebook date, etc. For that, we select any text from the beginning of the *Notebook* down to the beginning of the line starting with `MACRO-FILE`. The text is then deleted by hitting the DELETE or BACKSPACE key.
4. We are now ready to insert the `OCP` command. It is best to use an existing text line as a template which can easily be modified. For our example, we duplicate the line containing the `"Standby:"` string. To duplicate that text line, triple click on it. This will select the complete line. Now we perform a Edit → Cut operation (MacOS CMD+X, Windows ALT+X) followed by two Edit → Paste commands (MacOS CMD+V, Windows ALT+V).
5. We can now replace the `"Standby"` string by the string `"OCP"`. For that we double click on the word `"Standby"`. This will select the entire word. We now can type `"OCP"`. Make sure the colon does not get lost.
6. Finally, we save the modified macro file to disk with the Save As... command from the Notebook drop-down menu. Enter Test.mac as the file name, and make sure to have selected the correct directory. Then click on Save. The new macro file will be saved and can subsequently be loaded into *POTPULSE* with the option Macros → Load of the Potentiostat menu.

Parsing Rules for Macros Files

The following is a description of the rules which are used to interpret a macro file.

I. A macro file is a plain text file in ASCII. The first text line is the identifier `MACRO-FILE [version]`. The version number, if used, must start after position [10].

II. When a line has a `digit` in position [2], it contains the macro index plus the macro name. The macro index can start at position zero, but has to end at position [2]. The `macro name` is the text from position [6] to the end of the line.

III. A line is interpreted as a macro item, if it does not have a digit in position [2], and is longer than 6 letters.

   A. The letter at position [2] defines the owning window, e.g.:
   `"o"` and `"O"` → **O**scilloscope window

"e" and "E" → Amplifier (Pot**e**ntiostat) window (“**PG310**”)
"a" and "A" → **A**nalysis window

B. Item values start after the first colon up to the second colon or the line end.

1. A Boolean value is considered TRUE, if the value string begins with "t" or "T", or is a digit ("1", "2", ... "9").
2. A list selection has to be the list index.
3. A floating point number can be in fixed, scientific, or engineering format (i.e., with unit such as "p" for pico). Values are scaled with the same factors as within the program.
4. Integer values should be within the expected range.
5. A string can be any text, but cannot contain a colon!

IV. Whatever comes after the second colon is ignored. Thus, comments can be added after the second colon.

Macros Reference

The following gives an overview of all macro commands used within *POTPULSE*. The description corresponds to the various windows one can control by macros. If you want to build up your own custom macros, you can simply copy the necessary lines into your macro file.

Note: A macro can invoke also other macros, and can call itself as the last command in the macro. A macro which calls itself, or calls another macro which finally calls the original macro again would run indefinitely. Thus, to interrupt such a macro execution, one can use the Break command (‘CTRL’ + ‘B’) to interrupt immediately, and the Stop command (‘CTRL’ + ‘S’) to stop execution at the end of the momentarily running macro (i.e., when the macro would restart).

Potentiostat (Amplifier) Window (E)

Amplifier Gain: `Gain 0, 1, ... 19` sets the amplifier gain. The amplifier gain can be increased or decreased stepwise by simulating the up cursor (`CursorUp`) or down cursor (`CursorDown`) key.

```
E  Gain:            0; 0.005 mV/pA
E  Gain:            1; 0.010 mV/pA
E  Gain:            2; 0.020 mV/pA
E  Gain:            3; 0.050 mV/pA
E  Gain:            4; 0.1 mV/pA
E  Gain:            5; 0.2 mV/pA
E  Gain:            7; 0.5 mV/pA
E  Gain:            8; 1.0 mV/pA
E  Gain:            9; 2.0 mV/pA
E  Gain:           10; 5.0 mV/pA
E  Gain:           11; 10 mV/pA
E  Gain:           12; 20 mV/pA
```

```
   E  Gain:            14; 50 mV/pA
   E  Gain:            15; 100 mV/pA
   E  Gain:            16; 200 mV/pA
   E  Gain:            17; 500 mV/pA
   E  Gain:            18; 1000 mV/pA
   E  Gain:            19; 2000 mV/pA

   e  CursorUp
   e  CursorDown
```

Holding Potential: `VHold` sets the holding potential of the active amplifier. If the button Relative Value has been activated (`E Relative: TRUE`) the new holding potential will be changed relatively. The holding potential can be increased or decreased in steps of 10 mV by simulating the right cursor (`CursorRight`) or left cursor (`CursorLeft`) key

```
   E  VHold:            10.0mV; sets holding potential to 10 mV
   E  VHold:           -50.0mV

   E  Relative:         TRUE; the following depolarizes VHold by 10 mV
   E  VHold:            10.0mV
   E  Relative:         FALSE

   e  CursorRight
   e  CursorLeft
```

AD-Input Channel: `TestAdc 0, 1, ... 12` sets the default input channel. Arguments `5` to `12` define the AD-channels 1 to 7.

```
   E  TestAdc:          0; F2-Ext
   E  TestAdc:          1; Vmon
   E  TestAdc:          2; Imon1
   E  TestAdc:          3; Imon2
   E  TestAdc:          5; AD-0
   E  TestAdc:          6; AD-1
   E  TestAdc:          7; AD-2
   E  TestAdc:          8; AD-3
   E  TestAdc:          9; AD-4
   E  TestAdc:         10; AD-5
   E  TestAdc:         11; AD-6
   E  TestAdc:         12; AD-7
```

Recording Mode: `Mode 0, 1, ... 5` sets the recording mode (*inside-out*, *on-cell*, ...).

```
   E  Mode:             0; In Out
   E  Mode:             0; In Out
   E  Mode:             1; On Cell
   E  Mode:             2; Outside Out
   E  Mode:             3; Whole Cell
   E  Mode:             4; C-Clamp
   E  Mode:             5; V-Clamp
```

Test Pulse: `PulseDur` sets the duration and `PulseAmp` sets the amplitude of the test pulse. Both values can be changed relatively. `PulseOff, PulseOn` and `NoiseOn` turn the test pulse off/on or activate the noise mode. `PulseMode 0, 1, 2` defines the type of test pulse (single or double pulse or execution of the test series).

```
E  PulseDur:         5.0ms; test pulse 5 ms long
E  PulseAmp:         5.0mV; test pulse 5 mV in voltage clamp modes
E  PulseAmp:        10.0pA; test pulse 10 pA in current clamp mode
E  PulseOff:         TRUE
E  PulseOn:          TRUE
E  NoiseOn:          TRUE
E  PulseMode:        0; double
E  PulseMode:        1; single
E  PulseMode:        2; run the test series
```

Offsets: `Ljunc` sets the liquid junction potential and `Vzero` the pipette offset potential. Both values can be changed relatively. `AutoZero` automatically zeroes the pipette current, while `SearchMode` tracks the pipette offset by repetitively calling the `AutoZero` function.

```
E  Ljunc:            5.0mV;
E  Vzero:            5.0mV
E  AutoZero
E  SearchMode:       TRUE
E  SearchMode:       FALSE
```

C-Fast Compensation: `CFastTot` sets the value, `CFastTau` the time constant and `CFastPerc` the %-value of the fast capacitance compensation. `AutoCFast` performs an automatic C-Fast compensation. All values can be changed relatively.

```
E  CFastTot:         6.00pF
E  CFastTau:         0.5µs
E  CFastPerc:        80 %
E  AutoCFast
```

C-Slow Compensation: `CSlowRange` sets the range of the slow capacitance compensation circuit (off, 30, 100 or 1000 pF). `CSlow` sets the slow capacitance and `RSeries` the series resistance value. `AutoCSlow` performs a single and `CapTrack` a repetitive automatic C-Slow compensation. For the later you will have to set a `Delay`.

```
E  CSlowRange:       0; Off
E  CSlowRange:       1; 30 pF
E  CSlowRange:       2; 100 pF
E  CSlowRange:       3; 1000 pF
E  CSlow:           20.00pF
E  RSeries:          5.0MO

E  AutoCSlow
E  Delay:            0.10s; set frequency of CapTrack and TrackGLeak
E  CapTrack:         TRUE; turn CapTrack on
E  CapTrack:         FALSE; turn CapTrack off again
```

RS Compensation: `RsMode 0 ... 3` sets the range and `RsComp` sets the %-value of the RS compensation. The %-value can also be changed relatively.

```
E  RsMode:           0; Off
E  RsMode:           1; 100 µs
E  RsMode:           2; 10 µs
E  RsMode:           3; 2 µs
E  RsComp:          93%

E  Relative:         TRUE; the following decreases RsComp by 10%
E  RsComp:         -10%
```

```
  E  Relative:         FALSE
```

Leak Compensation: `GLeak` sets the size of the hardware leak compensation. This value can also be changed relatively. `AutoGLeak` performs a single and `TrackGLeak` a repetitive automatic leak compensation. For the later you will have to set a `Delay`.

```
  E  GLeak:            1.00nS

  E  AutoGLeak
  E  Delay:            0.10s
  E  TrackGLeak:       TRUE
  E  TrackGLeak:       FALSE
```

Filter: `Filter1 0 ... 3` sets the type of *Filter 1* (Bessel 100, 30 and 10 kHz, HQ 30 kHz). `Filter2Response` sets the response of *Filter 2* (Bessel or Butterworth). The bandwidth of Filter 2 can be set by `Filter2`.

```
  E  Filter1:          0; Bessel 100 kHz
  E  Filter1:          1; Bessel 30 kHz
  E  Filter1:          2; Bessel 10 kHz
  E  Filter1:          3; HQ 30 kHz
  E  F2Response:       0; Bessel
  E  F2Response:       1; Butterworth
  E  Filter2:         10.0kHz
```

Stimulus: `StimFilter` sets the *Stimulus Filter* (2 or 20 µs). `ExtScale` sets the scaling of the external *Stimulus Input*. `TstDacToStim1/2/3` redirects the stimulus to the output of amplifier 1, 2 or 3 (EPC9/2 and EPC9/3 only).

```
  E  StimFilter:       0; 2 µs
  E  StimFilter:       1; 20 µs
  E  ExtScale:         1.000x
  E  TstDacToStim1:    0; DA-3 to Stim-1: OFF
  E  TstDacToStim1:    1; DA-3 to Stim-1: ON
  E  TstDacToStim1:    2; Ext. Stim. Input: ON
  E  TstDacToStim1:    4(    1); AutoCSlow -> DA-3
  E  TstDacToStim2:    0; DA-3 to Stim-2: OFF
  E  TstDacToStim3:    0; DA-3 to Stim-3: OFF
```

Current-Clamp mode: `CCSpeed` toggles the *Fast Current Clamp* mode of the EPC9 board version ‘C’. `CCRange` sets the scaling of the output in the *Current-Clamp* mode. This value is fix for every EPC9. `GentleCCSwitch` toggles the *Gentle CC-Switching*.

```
  E  CCSpeed:           TRUE; activate fast current clamp mode
  E  CCSpeed:           FALSE; deactivate fast current clamp mode
  E  CCRange:           0; CC-Range: 1pA/mV
  E  CCRange:           1; CC-Range: 10pA/mV
  E  GentleCCSwitch:    0; Gentle CC-Switch: OFF
  E  GentleCCSwitch:    1; Gentle CC-Switch: ON
```

DA-Channels: With these commands you can define and set the analog and digital outputs. `DAChannel` defines the output channel to be used (`0..3` = analog channels, `5` = digital output), `DAValue` defines the output value either as voltage (analog channels) or as unsigned byte (digital trigger lines). The digital lines are interpreted binary: 1 corresponds to the first trigger, 2 to the second, 4 to the third, 8 to the fourth, and so on. To set multiple triggers you have to add the binary values, e.g. 7 (= 1 + 2 +

4) activates the first, second and third trigger. To output the last value that was defined use `DASet`.

```
E  DAChannel:        0; DA-0
E  DAChannel:        1; DA-1
E  DAChannel:        2; DA-2
E  DAChannel:        3; DA-3
E  DAValue:          5.00V

E  DAChannel:        5; Digital out (word)
E  DAValue:          16; set trigger 5 high
E  DASet
E  DAValue:          27; set triggers 1, 2, 4 and 5 high
E  DASet
```

Calling a macro from a macro: `Macro1...Macro20` calls Macro #1 to #20 from a running macro. Note: there is no separator between `Macro` and `Index`. Take care to not create an endless macro by calling the same macro recursively!

```
E  Macro1
E  Macro19
```

Sound: The commands `SoundOn`, `SoundHz` and `SoundVol` activate the sound and define its frequency and volume.

```
E  SoundHz:        100
E  SoundVol:       100%
E  SoundOn:        FALSE
```

Various Commands: `LastVHold` restores the last holding potential before switching into *Current-Clamp* mode. `Relative` defines the next value to be a *relative* change instead of an *absolute* setting. This *relative* input feature is automatically deactivated after one value is entered. `Wait` interrupts the current macro execution and asks the user to continue. `Bell` gives an acoustic signal, e.g. after a macro has been executed. `Zap` applies a *Zap* pulse. `Reset` resets the active amplifier. `E9Board1/2/3` sets the active amplifier.

```
E  LastVHold
E  Relative:       TRUE
E  Wait
E  Bell
E  Zap
E  Reset
E  E9Board1:       TRUE
E  E9Board2:       TRUE
E  E9Board3:       TRUE
```

Oscilloscope Window (O)

Sweep Settings: `WriteData` toggles the Store button. `Mode` defines the *Recording* mode. `Filter` sets the *Oscilloscope* filter, `MemPot` sets the *Oscilloscope* holding potential. `Averages` defines the number of averages acquired for one sweep.

```
O  Comment:        This is a Comment
O  WriteData:      TRUE
```

```
O  Mode:           0; In Out
O  Mode:           1; On Cell
O  Mode:           2; Outside Out
O  Mode:           3; Whole Cell
O  Mode:           4; C-Clamp
O  Mode:           5; V-Clamp
O  Filter:         500. Hz
O  Filter:         2.00kHz
O  Averages:       4
O  MemPot:         -50.0mV
```

Display Flags: `ShowPn` turns the display of P/n data on or off. `LeakSubtract` turns the display of leak subtracted data on or off. `ZeroSubtract` turns the display of zero subtracted data on or off. `Superimpose` turns the overlay mode on or off. `StoreAll` in-/activates the Overl. All button.

```
O  ShowPn:         TRUE
O  LeakSubtract:   TRUE
O  ZeroSubtract:   TRUE
O  Superimpose:    TRUE
O  StoreAll:       TRUE
```

Display Scaling: `DisplayGain1/2` defines the scaling and `DisplayOffset1/2` the offset of display 1 or 2. To reset the display (gain = 1, offset = 0) use `DisplayZero1/2`. The scaling of the oscilloscope can be fixed to an absolute range using `FixedScaling`. The ranges are set with `XRange`, `Y1Range` and `Y2Range`.

```
O  DisplayGain1:   4.00
O  DisplayOffset1: 200.m
O  DisplayZero1
O  DisplayGain2:   1.00
O  DisplayOffset2: -500.m
O  DisplayZero2

O  FixedScaling:   TRUE
O  XRange:         1.00 s
O  Y1Range:        500.p
O  Y2Range:        1.00n
```

Display Mode: `Xmode` sets the x-axis of the *Oscilloscope* (t, V, V-ramp, ...). `InvertIV` is used to plot *V* versus *I*. With `ConnectSweeps` different sweeps in a series are connected.

```
O  Xmode:          0; I vs. t
O  Xmode:          2; I vs. V
O  Xmode:          3; I vs. V-ramp
O  Xmode:          4; I vs. V-ramp,theo
O  Xmode:          6; I vs. t + LockIn
O  InvertIV:       TRUE
O  ConnectSweeps:  TRUE
```

Display Timing: `StartTime` and `EndTime` define the left and right margin of the display (in % of the full scale). To reset the timing (start = 0%, end = 100%) use `TimeReset`. To reset the whole *Oscilloscope* window (scaling and timing) use `DisplayReset`. `PageLeft` and `PageRight` scroll one page to the left or right. To scroll to any page use `Page`.

```
O  StartTime:      10
O  EndTime:        90
O  TimeReset
O  DisplayReset

O  PageLeft
O  PageRight
O  Page:           5.0
```

Cursors: `ShowCursor` toggles display of the cursors, `ResetCursor` resets the cursors of the actual analysis range (i.e. sets them to 0 and 100%).

```
O  ShowCursor:     TRUE
O  ResetCursor
```

Controlling the Experiment: `Pool1...Pool6` starts the pulse sequence 1 ... 6 from the actual PGF pool. To change the pool (i.e. shift it by 6 to the left or right) use `LeftSeq` or `RightSeq`. `Break`, `Stop`, `Wait` and `Link` activate the corresponding button. `Timer` resets the *POTPULSE* timer. `User1...User16` sends corresponding the user defined serial string to the serial port.

```
O  Pool1:          TRUE
O  Pool6:          TRUE
O  LeftSeq
O  RightSeq
O  Break:          FALSE
O  Stop:           FALSE
O  Wait:           FALSE
O  Link:           TRUE
O  Timer
O  NewExperiment
O  NewGroup
O  User5:          TRUE
O  User12:         TRUE
```

Online Analysis Window (A)

Analysis Type: `Range` sets the actual analysis (1 or 2). `Abscissa` (0...8) sets the x- and `Mode` (0...20) sets the y-value.

```
A  Range:            0; Range 1
A  Range:            1; Range 2
A  Abscissa:         0; Voltage
A  Abscissa:         1; Duration
A  Abscissa:         2; Time
A  Abscissa:         3; Timer Time
A  Abscissa:         4; Realtime
A  Abscissa:         5; Index
A  Abscissa:         6; Peak Voltage
A  Abscissa:         8; Y1 vs. Y2
A  Mode:             0; No Analysis
A  Mode:             1; Extremum
A  Mode:             2; Maximum
A  Mode:             3; Minimum
A  Mode:             4; Time to Peak
A  Mode:             5; Mean
A  Mode:             6; Charge
A  Mode:             7; Variance
```

```
A  Mode:              8; Slope
A  Mode:              9; Reversal
A  Mode:             10; C-Slow
A  Mode:             11; G-Series
A  Mode:             12; Anodic Charge
A  Mode:             13; Cathodic Charge
A  Mode:             14; LockIn CM
A  Mode:             15; LockIn GM
A  Mode:             16; LockIn GS
A  Mode:             17; Fura Ratio
A  Mode:             18; Fura Ca
A  Mode:             19; Fura F1
A  Mode:             20; Fura F2

A  Fit:              TRUE
A  Reference:        FALSE
```

Analysis Settings: The online analysis will be calculated over the range defined by the two cursors (`LeftB`, `RightB`) for the segment defined as the *relevant* one in the *Pulse Generator*. Change this segment by adding an offset (`RelXSeg`, `RelYSeg`). `Trace` defines which trace is used for the calculation (first or second trace), `Math` enables you to do further calculations with the values obtained by the two ranges.

```
A  LeftB:            10.0%
A  RightB:           90.0%

A  RelXSeg:           1
A  RelYSeg:           1

A  Trace:             0; First Trace
A  Trace:             1; Second Trace

A  Math:              0; no math
A  Math:              1; y = y1 + y2
A  Math:              2; y = y1 - y2
A  Math:              3; y = y1 * y2
A  Math:              4; y = y1 / y2
```

Display Settings: `MarkerKind` sets the symbol type (0...5) and `MarkerSize` (1...12) sets the size of the makers in the *Online Analysis* plot. `Scale` sets the scaling to *automatic* or *fixed*. In the later case you can define the range of the graph with `Xmin`, `Xmax`, `Ymin` and `Ymax`. `X/YZeroLine` defines whether to display the x-/y-axis and `X/YZeroTics` the number of ticks. Each axis can be linear, logarithmic or exponential (`X/YTransform`). Use `CopyLast` to copy the last used settings or `CopyOther` to copy the range settings of the other analysis into the actual ones. `PlotLast` plots the last *Online Analysis*. `Overlay` sets the *Overlay* mode.

```
A  MarkerKind:       1; Plus
A  MarkerKind:       2; Star
A  MarkerKind:       3; Diamond
A  MarkerKind:       4; Cross
A  MarkerKind:       5; Square
A  MarkerSize:       2

A  Scale:            0; Fixed Scaling
A  Scale:            1; Auto Scaling

A  Xmin:            -50.000m
```

```
A   Xmax:             70.000m
A   Ymin:             0.000
A   Ymax:             80.000p

A   XZeroLine:        TRUE
A   XZeroTics:        5
A   YZeroLine:        TRUE
A   YZeroTics:        11

A   XTransform:       0; lin
A   XTransform:       1; log
A   XTransform:       2; exp
A   YTransform:       0; lin
A   YTransform:       1; log
A   YTransform:       2; exp

A   CopyLast
A   CopyOther
A   PlotLast

A   Overlay:          0; No Overlay
A   Overlay:          1; "Time" Wrap
A   Overlay:          2; Overlay + "T"-Wrap
```

Parameters Window (P)

Parameters: The commands `Gain`, `VGain`, `AuxGain`, `CSlow`, `GSeries`, `RsValue`, `Bandwidth`, `CellPotential`, `PipPressure`, `Temperature`, `LockinExtPhase`, `UserParam1`, `UserParam2`, `PipResistance`, `SealResistance`, and `RMSNoise` allow you to set any of the parameters stored together with the sweep.

```
P   Gain:             2.000mV/pA
P   VGain:            10.00 V/V
P   AuxGain:          1.000 V/V
P   CSlow:            22.00pF
P   GSeries:          5.000MOhm
P   RsValue:          5.000MOhm
P   Bandwidth:        3.000kHz
P   CellPotential:   -50.00mV
P   PipPressure:      5.000 cm
P   Temperature:      20.00 C
P   LockinExtPhase:   0.000 °
P   UserParam1:       2.500 V
P   UserParam2:      -1.300 V
P   PipResistance:    5.000MOhm
P   SealResistance:   5.000GOhm
P   RMSNoise:         110.0fA
```

Solution Timing: `IntSol` sets the internal and `ExtSol` sets the external solution used.

```
P   IntSol:           1
P   ExtSol:           3
P   IntSolEdit
P   ExtSolEdit
```

Key Translations

The following macros simulate key strokes during the execution of a macro:

```
Char: [character]  separated by space
DeleteLeft, DeleteRight
END, Enter
F1, F2, ... F15, Help, HOME
Numeric *, Numeric +, Numeric -, Numeric ., Numeric /, Numeric =, Nu-
meric Clear
Numeric 0, Numeric 1, ... Numeric 9
PageDown, PageUp
```

Input of a Relative Increment

It is possible to change values by a given increment instead of having to specify the absolute value. Thus, one would have to click first on the button Relative Value in the *Amplifier* window and then enter the new value in the target control. The macro recorder will calculate the difference between the new and the old setting and will store only this increment. E.g. if you change the holding potential from -60 to -70 mV while recording a macro, a step of *-10 mV* will be recorded. The *"relative"* input feature is automatically deactivated after one value is entered.

POTPULSE Advanced

Continuous Recording

POTPULSE treats continuous recording just as pulsed recording with a pulse template having a very long segment. These segments are called Continuous segments in the Free-waveform Generator, allowing the user to execute pulsed recording followed by continuous recording in the same run (this is often also designated as open end stimulation). Most of the functions for purely pulsed data are available. In other words, a *Continuous* segment can be incremented in duration and voltage in the same way as the other segments, repeats and links of sequences are allowed.

The restrictions for the continuous segment are:

- It has to be the last segment of a template.
- Duration has to be used instead of scan rate.
- No averages are supported for continuous segments.

The timing is done in the same way as for the other segments in ms. If a quasi infinite recording is desired (note that a hard disk is filled up very quickly), this duration is to be set to an accordingly large value.

You can interrupt a continuous recording at any time during execution with 'CTRL' + 'B' or the Break control. In this case the data acquired up to this point are saved, if at least "one page" of data were acquired. The Stop key ('CTRL' + 'S') is used to stop the acquisition at the end of the running sweep.

Note: Holding the Option (MAC) or Shift (Windows) key pressed will apply the functions to the second trace!

The keyboard functions that are active during continuous data acquisition are:

Key	Function
'CTRL'+'B' 'BREAK'	Cancel and save data (min. one page)
'CTRL'+'S'	Stop aquisition after the end of the running sweep
Numeric '+'	Increase display gain for trace 1
Numeric '-'	Decrease display gain for trace 1
Shift + Numeric '+'	Shift the trace up for trace 1
Shift + Numeric '-'	Shift the trace down for trace 1

Key	Function
Numeric ‘*’	Center the 1. trace in the display
‘O’	Toggle *Overlay* mode
‘.’	Toggle *Overlay All* mode
‘BACKSPACE	Clear display
‘T’	Reset the Timer clock

Some key strokes will be buffer until the end of the ongoing acquisition of a series. The corresponding functions will be applied before the next series is started. These keys are:

Key	Function
Cursor Right	Increases Holding by 10mV or 10pA
Cursor Left	Decrease Holding by 10mV or 10pA
Cursor Up	Increase the amplifier gain by one step
Cursor Down	Decrease the amplifier gain by one step
CTRL + ‘W’	Toggle writing next series

These functions allow to easily switch to a new continuous record, when e.g. the series with the continuous acquisition is linked to itself. Thus, one can start acquiring till one decides to change one parameters such as storing, the amplifier gain, or the holding potential. Then one presses the appropriate key, and clicks on the Link button (or hits ‘CTRL‘+ ‘L’).

For replay the initial part of the template (without conditioning and continuous segments) is shown as one display or page (0-100%) by default. Thus, by creating an initial constant segment of a certain length one can specify the default display time resolution. However, the minimum time basis per page is 200 ms. The rest of the trace can be reviewed by paging through the data.

The output functions take either the section of the data as shown on the screen or the entire sweep.

Continuous (“gap-free”) Data Acquisition

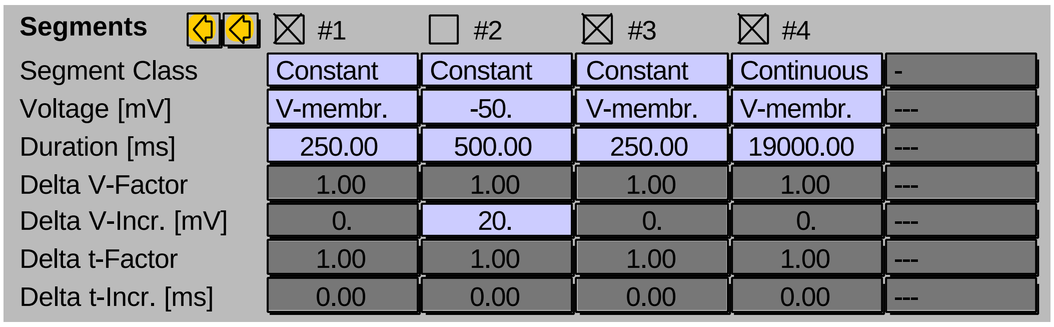
POTPULSE can acquire very long sweeps. There exist two completely different approaches which we explain in the following:

- Using the “Continuous” Segment Type
- Stimulating with more than 16 kSamples in one Sweep

Before describing them there is a one advice to be recalled: In most situations where a repetitive stimulus is applied - be it a voltage or a chemical stimulus - it is very important to know how the data are to be analyzed. E.g. an current-voltage experiment could be acquired by applying all voltage steps within one sweep. Yet, it will be very difficult to get the online analysis to correctly and *easily* analyze that train of pulses. In that example it is clear that it would be much preferable to acquire one voltage pulse per sweep. Then the online analysis gets obvious while it would be very tedious to analyze the data, when all voltage pulses were within one sweep. Please keep that in mind when designing your sweep structure.

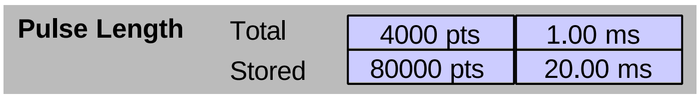
Using the “Continuous” Segment Type

The segment type Continuous signalizes to *POTPULSE* that acquisition of long data stretches are asked for. The maximal segment duration of such a “continuous” segment can be as high as 10^7 samples, which is more than 2.5 hours at 100 kHz sampling rate. Please keep in mind that only the last segment can be “continuous” and that there has to be at least one “non-continuous” segment before that.



Segments	☒ #1	☐ #2	☒ #3	☒ #4	
Segment Class	Constant	Constant	Constant	Continuous	-
Voltage [mV]	V-membr.	-50.	V-membr.	V-membr.	---
Duration [ms]	250.00	500.00	250.00	19000.00	---
Delta V-Factor	1.00	1.00	1.00	1.00	---
Delta V-Incr. [mV]	0.	20.	0.	0.	---
Delta t-Factor	1.00	1.00	1.00	1.00	---
Delta t-Incr. [ms]	0.00	0.00	0.00	0.00	---

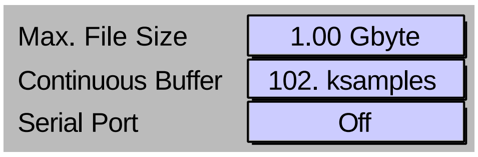
Note: The complete sweep before the “continuous” segment has to fit into the allocated stimulus buffer, which is 16384 points (= 16 kSamples) by default (see below, how to surpass this limitation). The size of all “non-continuous” segments is the Total Pulse Length in the *Pulse Generator* window. The Stored Pulse Length is the physical size of the whole sweep as written to disk, i.e. the size off all segments including the last “continuos” one.



Pulse Length			
	Total	4000 pts	1.00 ms
	Stored	80000 pts	20.00 ms

The continuous segment can either be stored directly to disk (*“hard disk recording”*), or the data can temporarily be stored in memory in the allocated *“Continuous Buffer”*:

- Storing directly to disk is possible when one acquires from one AD-channel only. To store directly to disk, enter zero for the size of the Continuous Buffer in the *Configuration* window. Keep in mind that storing to disk requires your computer system and hard disk to be fast enough. A contemporary Pentium II system with a SCSI hard disk can record data directly to disk at a rate of up to 100 kHz. If you notice repetitive messages about *"AD-overrun"* you should decrease the sampling rate or you will have to store the data in memory (see below).
- To store continuous data in memory, you have to reserve an appropriate amount of memory for the Continuous Buffer in the *Configuration* window.



Max. File Size	1.00 Gbyte
Continuous Buffer	102. ksamples
Serial Port	Off

The total required space of the "continuous" data (on disk and in RAM) can easily be computed with the following formula:

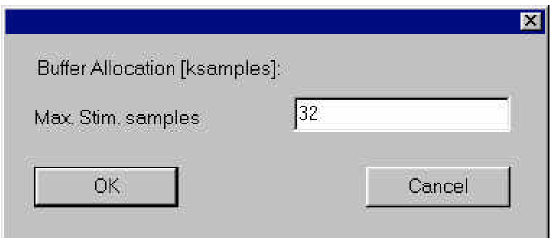
$$Bytes_{total} = \frac{2\ Bytes}{Sample} \cdot Input\ Channels \cdot Frequency \cdot Duration$$

For example, sampling 2 input channels at 10 kHz will produce 2.3 Mbytes of data every minute or 137 Mbytes every hour, respectively (1Mbyte = 1024^2 bytes).

Under MacOS, you can enable Virtual Memory in the Memory control panel, when the physical memory of your computer system is too limited. However, the required acquisition rate should not be too fast and the hard disk speed should not be too slow. Only practical tests can determine, whether your system will satisfactorily work with *"virtual"* memory enabled.

Under Windows, virtual memory is always active and cannot be turned off.

Stimulating with more than 16 kSamples in one Sweep

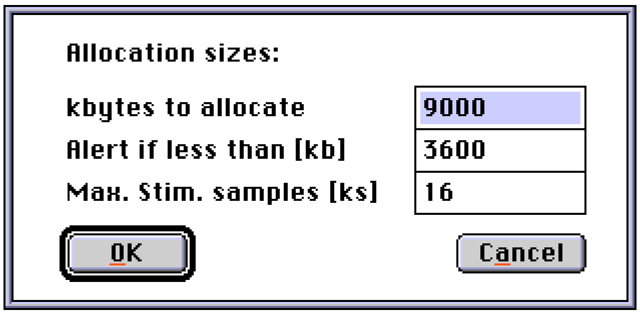


Buffer Allocation [ksamples]:

Max. Stim. samples: 32

While the "continuous" segment type allows you to acquire sweeps longer than the default stimulus length, it does not allow you to stimulate with complex voltage patterns during the "continuous" part of the stimulus. If you are limited by the default maximal sweep length of 16 kSamples and you need to acquire longer sweeps, you can increase the size of the stimulus buffer in the Buffer Allocation... dialog. You can call this dialog from the Pulse drop-down menu and enter the new value into the field Max. Stim. samples [ks] . Please be warned that increasing the stimulus buffer size will consume plenty of computer memory, approximately 48 bytes per additional stimulus sample.

Under MacOS (see picture) you will also have to increase the values given in the kbytes to allocate and Alert if less than [kb] fields accordingly. Those changes are drastically. If it should happen that there is not enough memory left, start *POTPULSE* again from the Finder with the selected memory allocation. Then, perform the following:

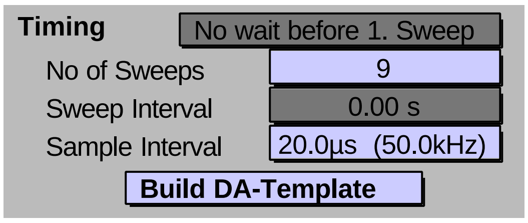

Allocation sizes:

kbytes to allocate	9000
Alert if less than [kb]	3600
Max. Stim. samples [ks]	16

1. Remove the file DefaultPulse.set from the Pulse folder.
2. Remove the files Pulse.Settings and PulseFit.Settings from the Preferences folder inside the System Folder.
3. Increase and/or enable Virtual Memory in the Memory control panel.
4. Restart *POTPULSE*.
5. Enter an new allocation size which is appropriate to you computer configu-ration.

Minimizing the Sweep Interval

To make the gap between two sweeps as short as possible set the Sweep Interval of the stimulus template to zero - this tells *POTPULSE* never to wait before starting the next Sweep - and turn off as many options as the experiment al-lows. If possible, acquire only one input chan-nel. Deactivate any background programs, such as networking, screen savers, virus scanner and so on. Close all windows that are not essential, such as the *Notebook*, the *Pulse Generator*, the *Amplifier* window, the *Parameter* window, the *Replay* window and the *Online Analysis* window. Check for the following options:


Timing: No wait before 1. Sweep

No of Sweeps	9
Sweep Interval	0.00 s
Sample Interval	20.0µs (50.0kHz)

- Both online analysis ranges should be set to No Analysis.
- Deactivate Notebook → Buffered Output.
- MacOS computers: reduce the number of colors to 256. On some Power Macs the lowest setting is Thousands of Colors; use that setting.
- Keep the *Oscilloscope* window small.
- Turn Display → Dimmed Overlay off . Dimming is performed by redrawing with gray.
- Turn the Overlay mode in the *Oscilloscope* window off, since clearing the screen in-between sweeps uses time!
- MacOS: Set File → Disk Write Options to Write after Series.

- Windows: Turn File → Disable Data File Caching off. Note: usually Windows NT is faster than Windows 95 which is faster than Windows 3.1.
- If possible, use a large fast SCSI hard disk to store data. If you are using Windows NT consider using the NTSF format which is faster than the FAT system.
- Use the smallest reasonable screen resolution. The program needs more time to draw on a higher resolution screen for the same physical size of the oscilloscope.

Minimal Sweep Intervals in Test

For the closely related programm *PULSE+PULSEFIT* a test series was made, comparing the achieved minimal sweep intervals of three typical computers: The PPC8500 and the Pentium used the PCI-bus cards, while the Quadra used the NUBUS card. The MacOS computers had a SCSI hard drive, while the Pentium computer had an EIDE hard drive.

The "default" test uses the default *PULSE+PULSEFIT* configuration as set by the installation procedure. The "minimal" test uses the above described optimizations and the "Write NoShow" test shows the times when data is not stored using the option Write NoShow in the *Pulse Generator* window. We measured the time used to acquire the series IV from the DefPGF.pgf pool, subtracted the sweep time and divided that value by the number of gaps in the series. The following table shows the minimum sweep gap for these three tests:

Computer	default	minimal	WriteNoShow
Mac Quadra 650 33 MHz	201	173	84
Mac PPC 8500 120 MHz	130	93	34
Pentium (NT) 90 MHz	191	111	70

These minimal processing times depend of course on the speed of the complete computer system, such as the CPU, the hard disk, the graphics card and the operating system.

The Sweep Gap on a Pentium II Computers

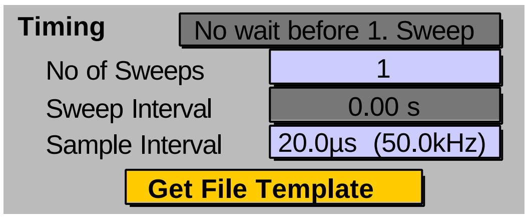
Using a Pentium II computer 300 MHz (Intel 82443LX/EX, 64 MB, 60 ns DIMM RAM, 4 GB UW-SCSI hard drive, Winner 1000/T2D S3 Trio64V2/DX 2 Mb) running at a resolution of 1024x768 high color under Windows 95B, the minimum duration between two sweeps could be as low as 30 ms. With all extensions (X-CHART and LockIn) and the *Online Analysis* enabled the mean sweep gap was about 90 ms.

The following is a benchmark that gives you a better feeling what effects the sweep gap. For every test we ran the PULSE+PULSEFIT series IV curve from the DefPGF.pgf pool 5 times. The sequence was slightly modified in a way that the first trigger was set to 0 ms and 13 instead of 9 sweeps were used. This protocol produces a total of 60 (= 12 x 5) gaps per test. The complete results are as follows:

OS	configuration	Gap [ms]
Windows 95	no show, caching	29 ± 13
	+ show	28 ± 15
	+ no caching	31 ± 21
	+ show everything, 2x analysis	47 ± 21
	+ LockIn and X-Chart extension	91 ± 15
Windows NT 4	no show, caching	33 ± 9
	show everything, no caching, analysis	44 ± 10
PULSE 8.11, Win 95	no show, caching	295 ± 42

The last line is the same test as the first one, however, *PULSE* v8.11 instead of v8.31 was used. The later version has been "hand optimized" with respect to the Intel code, making the gap 270 ms (or ten times) shorter. As can be seen from the table, the only thing that really affects the minimum duration are the *Online Analysis* and the extensions. Therefore, if you need maximal speed, you should turn off the *Online Analysis* first.

The "Get File Template" Feature


Timing
No wait before 1. Sweep
No of Sweeps: 1
Sweep Interval: 0.00 s
Sample Interval: 20.0µs (50.0kHz)
Get File Template

The DAC-stimulus template of *POTPULSE* can be either computed by the program or loaded from a file instead, when you activate Get File Template in the Timing section of the *Pulse Generator* window. This way you can stimulate any complex pulse pattern that *POTPULSE* otherwise can not offer. You can even stimulate a prerecorded voltage trace such as an action potential. There are the following things to consider, when using the File Template feature:

1. The template file must be in the folder where the "pgf"-files are. Alternatively, you can put the files into a sub-folder inside the folder where the "pgf"-files are. In this case, the folder name must be the name of the stimulus.

2. The name of the template file must be "[stimulus name]_[sweep number]". E.g., if the stimulus name is "IV", then *POTPULSE* looks for the template file "IV_1" to be the first sweep, "IV_2" for the second sweep, etc.

3. The file must contain one voltage value per stimulus point. The voltage value must be a "short" (4 byte), binary IEEE-floating point format number. All values must be in volt, i.e., if a voltage of -80 mV has to be output, then the required value is -0.080.

The total number of output samples must be equal to the total number of input samples. Thus the stimulus length is defined by:

$$DA-Length = AD-Length \times \frac{AD-Channels}{DA-Channels}$$

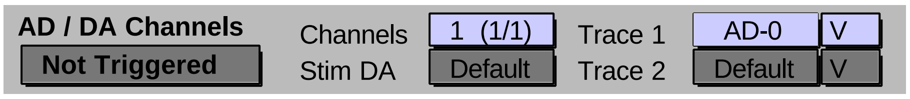
where:

DA-Length	total number of samples per output channel
AD-Length	total number of samples per input channel
DA-Channels	number of physical DA-channels used
AD-Channels	number of physical AD-channels used

The number of physical output channels depends on the used acquisition option, and is computed as follows:

$$Stimulus + Number\ of\ \ physical\ Triggers + FURA-Channels$$

You can direct more than one *logical* trigger to the same DA-channel. In this case the triggers are output to one *physical* DA-channel. If a trigger is set to off, then it does not use a physical DA-channel at all. To help comprehend the acquisition conditions of *POTPULSE*, the number of physical AD- and DA-channels are indicated in the *Pulse Generator* window, field AD / DA-Channels, option Channels between brackets.



The first value is the number of physical *input* channels, the second is the number of physical *output* channel. The total number of samples per input channel is displayed in the fields Pulse Length, option Total.

It is advisable to begin using but one output DA-channel and only one input AD-channel, and not to use any triggers until one is familiar with this option. The following describes the conditions to be considered when building a file template using multiple input and output channels, and how to implement the triggers.

Here is an example that demonstrate how the File Template works to stimulate a prerecorded pulse pattern. You can easily test this using the model circuit:

1. Generate a simple stimulus named “Get” with one sweep per series (No of Sweeps = 1) and three Constant segments (Duration: 20, 10, and 50 ms, Voltage: -80, 0 and -80 mV). There should be no trigger (Triggers = 0), the Sample Interval should be 0.1 ms and one input channel has to be acquired (Channels = 1). The sequence should not be linked (Linked Sequence = NIL). Stim DA and input Channels are set to Default.
2. In the oscilloscope window, execute the “Get” stimulus. You should see the current response with its corresponding “pulse” shape. Let's assume that the response is 50 pA in amplitude. Note: the Store button must be on in the oscilloscope, otherwise the sweep would not be stored.
3. To generate a file template, clear the “sweep buffer” first (Buffer → Clear) and then add the acquired sweep into it (Buffer → Add).
4. The sweep data has been acquired as pA and now has to be scaled to mV. This is done by calling Buffer → Scale and entering “1e9” as the Scale factor.
5. Store the sweep buffer to disk as “Get_1” (Buffer → Save as binary File...).

6. Now, go back to the Pulse Generator window, select the "Get" stimulus and activate Get File Template.
7. Finally, execute the "Get" stimulus again in the Oscilloscope window. The file template is read and used as the template, and you should see the corresponding current response.

Alternatively, you can use third party programs such as *IGOR Pro* (see next section) to generate the stimulus template file.

Using IGOR Pro together with POTPULSE

POTPULSE tightly cooperates with *IGOR Pro* from *WaveMetrics*. You can export sweeps and results from the *Online Analysis* to *IGOR Pro* for further analysis and you can import *IGOR Binary Files* into the sweep buffer.

Creating a Stimulus Template with IGOR Pro

One can import a template generated or modified by *IGOR Pro*. Similarly to the procedure described in the preceding paragraph, one uses the Buffer → Add Igor Binary option to import an IGOR binary wave into the "sweep buffer". One creates an appropriate "IGOR binary wave" with the IGOR command Data → Make Waves. In the *Make Wave* dialog one should not select the Double Precision check-box. Double precision generates 8 byte long variables which correspond to the 'C'-type *"double"*, while single precision generates 4 byte long variables which correspond to the 'C'-type *"float"*. The later format is the one used by *POTPULSE*. One can export the template generated in the example of the preceding paragraph with the command Buffer → Save as ASCII and import the template in *IGOR Pro* using the command File → Open File as a text wave.

The total number of output samples is computed by the formula given in the preceding section "The Get File Template Feature".

The following is an *IGOR* macro that allows you to save any wave within *IGOR Pro* as a file suitable to be loaded by the *Pulse Generator* as a stimulus template:

```
// ****************************************************
// Save an IGOR wave as PULSE stimulus template
// ****************************************************

Macro SaveStimulusTemplate (waveName, platform)
  String waveName
  Variable platform
  Prompt waveName, "Select Wave to save", popup, WaveList ("*", ";", "")
  Prompt platform, "Select Target", popup, "Native;MacOS;Windows"

Silent 1

  if (! waveExists ($waveName))
    Abort "Please, select an existing wave!"
  endif

  DoSaveStimulusTemplate ($waveName, platform)
End Macro
```

```
// Coded as function to improve speed

Function DoSaveStimulusTemplate (waveName, platform)
  Wave waveName
  Variable platform
  Variable fRefNum, V_Flag, V_filePos, V_logEOF
  String S_fileName, S_path, S_info
  Variable byteOrder, num, inx, value

  if (platform == 1)
    byteOrder = 0           // native format
  else if (platform == 2)
    byteOrder = 2           // BigEndian MacOS
  else
    byteOrder = 3           // LittleEndian Windows
  endif

  inx = 0
  num = numPnts (waveName)
  Open fRefNum as NameOfWave (waveName)
  FStatus fRefNum           // check if file has been opened
  if (V_Flag)
    do
      value = waveName [inx]
      FBinWrite /B=(byteOrder)/F=4 fRefNum, value
      inx += 1
    while (inx < num)
    Close fRefNum
  endif
End Function
```

Loading an IGOR Macro File generated by POTPULSE

When you export data from *POTPULSE* to *IGOR* Pro, a so called “recreation macro” is created. The data itself may be stored into a sub-directory or may be loaded directly from the data file, while the macro in the main directory has the instructions how to import these data and display them in *IGOR Pro*o. It is easiest to load such a macro file by simply opening it from the MacOS Finder or the Windows Explorer by double-clicking on it. *IGOR Pro* will start-up and automatically load and execute the macro. On a Windows computer the macro file is recognized by its file name extension *.itx.

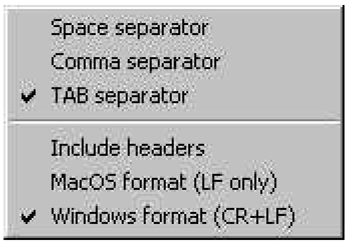
To load the “recreation macro” within *IGOR Pro* itself, select the menu option Data → Load Wave → Load Igor Text. This will bring-up a file selector, allowing you to select the desired macro file to be loaded and thereby executed. Sometimes it can happen that the folder with the data files is no longer accessible, or its name has changed. This can happen, e.g., when one accesses the macro file via a network or when the data are transferred by a floppy disk. When the macro file is executed in such a case, *IGOR Pro* will put an alert on the screen telling you that the volume or folder has not been found and asking you, whether you want to help. In this case *IGOR Pro* will bring up a file selector allowing you to select the target folder in which the data files are located. Once the correct folder has been selected, *IGOR Pro* will proceed executing the macro file.

Note: All macro files generated by *POTPULSE* are text files and must be loaded by Load Igor Text, not Load Igor Binary, even when the export format in *POTPULSE* was set to IGOR Binary. The macro file does never contain the binary data. If required, the binary data are separately stored in IGOR Pro wave files, while the macro file contains the instructions how to load and scale the binary data.

Copying a Table from the Notebook into a Table in IGOR Pro

First, you have to configure *POTPULSE* for the special requirements for table data in *IGOR Pro*, as for any program which can import tables. *IGOR Pro,* for example, requires that the values in a table are separated by tabs - other programs may request commas or blank spaces. Also, most programs are not aware of the "engineering" representation of data like "2.345µ". Thus, you must define some settings before copying the notebook and pasting it into *IGOR Pro*:

1. Select the option Scientific Notation in the Notebook drop-down menu.
2. Select the option TAB Separator in the menu Tree → ASCII-text Format.



1. Select the appropriate MacOS Format (LF only) or Windows Format (CR+LF) option in the menu Tree → ASCII-text Format.
2. Generate the table in the *Notebook* the usual way in *POTPULSE* by replaying the data and performing the desired analysis.
3. Switch to the *Notebook* window and select the table by dragging the text selection with the mouse.
4. Perform a Copy command from the Edit drop-down menu.
5. Switch now to *IGOR Pro* and activate the Table window into which you want to paste the data.
6. Finally, issue a Paste command from the Edit menu of *IGOR Pro*.

Timing Schedule of the Digitizer Board ITC-16

Although modern digitizers such as the ITC-16 which is built into the PG310/390 seem to stimulate up to 4 and acquire up to 8 channels simultaneously, in reality this happens sequentially. In principle, the digitizer reads one AD- and writes one DA-channel at a time. This happens at a minimum of every 5 µs or a multiple of that. First, the AD-channel will be read, then the DA-channel is output with a small delay (1 µs) so that it cannot affect the reading. If several channels have to be read and written, this occurs one after the other. Please note, that the output of a stimulus is always coupled to the acquisition of an input, the ITC-16 can not stimulate a DA-channel without reading an AD-channel.

One example shall clarify the timing scheduling: Let's assume we are sampling 2 channels and stimulating 2 channels. The sampling interval per input channel as defined within *POTPULSE* is 200 µs (5 kHz). Thus, the digitizer will acquire and stimulate one channel every 100 µs (= sampling interval per input channel divided by the number of input channels). Please note, that this value can not be smaller than 5 µs. In the example the second DA should be set “high” during the time of the first sample (see column “Trigger”).

Time [µs]	Input		Output		Trigger
	Channel	Sample	Channel	Sample	
0	AD-1	1			
1			DA-1	1	
100	AD-2	1			
101			DA-2	1	high
200	AD-1	2			high
201			DA-1	2	high
300	AD-2	2			high
301			DA-2	2	
400	AD-1	3			
401			DA-1	3	
500	AD-2	3			
501			DA-2	3	

As long as the number of input and output channels is the same, things are pretty easy, the minimum duration of a trigger can be as long as the sampling interval, which is 200 µs in the example above: the channel is “high” from 101 to 301 µs. However, this changes, when a the number of read and write differ. The same example as above with 3 instead of 2 output channels gives a different scheme:

Time [µs]	Input		Output		Trigger
	Channel	Sample	Channel	Sample	
0	AD-1	1			
1			DA-1	1	
100	AD-2	1			
101			DA-2	1	high
200	AD-1	2			high
201			DA-3	1	high
300	AD-2	2			high
301			DA-1	2	high
400	AD-1	3			high
401			DA-2	2	
500	AD-2	3			
501			DA-3	2	
600	AD-1	4			
601			DA-1	3	

Thus, the minimum trigger length increases from 200 to 300 µs: the second DA-channel is “high” from 101 to 401 µs. One can easily calculate the minimum duration of a trigger by the following formula:

$$\mathit{Minimal\ Output\ Time} = \frac{\mathit{Output\ Channels}}{\mathit{Input\ Channels}} \times \mathit{Sampling\ Time\ per\ Input\ Channel}$$

One can get a misleading trigger length, when one uses the AD-channels to measure the duration during which a trigger is high. Although the trigger time is different in the two examples above (200 versus 300 µs) AD-channel 2 reports 200 µs in both cases (low at 100 and 500 µs, high at 300 µs → 500 - 300 = 200 µs). In addition, AD-channel 1 reports 200 (= 400 - 200) µs in the first and 400 (= 600 - 200) µs in the second example (low at 0 and 600, high at 200 and 400 µs).

Comments to the Data Format of POTPULSE

The data format is extensively described in Chapter *Data Format* and the appendices I, II, and III. *Appendix III* contains a simple program depicting how to correctly load an experiment file. Here some additional tips for *programmers:*

Never assume you know how large a record is in a file. You *must* always use the sizes as stored in the file header itself. Otherwise the files may not be readable. E.g., newer files may add fields to a record. This is the way *POTPULSE* had never required a file conversion program, because it could add newly required fields to the end of the respective records.

Remember that many compilers handle *variable alignments* in memory, which differ from the one our compilers use. On a 680x0 MacOS computer, memory alignment is typically on even addresses for fields of 2 or more bytes. PowerPCs run better on 4 byte alignments, and programs for Pentium processors use 4 or 8 byte alignments. Thus, do not use the given record structures, but load the structures in your program byte-wise!

The byte offsets of the record fields are identical for both platforms, although the alignment in memory differs. Thus, one has to de-compress the Tree files upon loading them. The byte offsets and field descriptors are listed in the *POTPULSE* manual, appendix 1, Data Structure. The record fields define all bytes positions by using filler bytes, where necessary.

The *size of variables* and other common definitions are listed, e.g., in Appendix II. Remember: one BYTE is 1 byte; in 'C' one *"char"* can be 1 byte (the old default) or 2 bytes when using Unicode support!

Platform specific issues: there are two byte orders of storing variables in RAM: *big- and little-endians*. "Intel" processors use the "little-endian" format while "Motorola" processors are "big-endians". "Little-endian" means that the last byte (the one with the highest address in RAM) contains the "little" part of a variable. Please, refer to Appendix IV. The following explains how to correctly interpret the *LittleEndianBit* of the sweep structure:

- All sweeps written under MacOS have the *LittleEndianBit* cleared. When the MacOS version of *POTPULSE* reads a sweep with the LittleEndianBit set, it swaps the bytes.
- All sweeps written under Windows have the *LittleEndianBit* set. When the Windows version of *POTPULSE* reads a sweep with the *LittleEndianBit* cleared, it swaps the bytes, otherwise no swapping takes place.

Be prepared to byte-swap the Tree file as well. The *"Tree"* identifier in the first 4 bytes of a "Tree" file is written as INT32. Thus, when a program reads in the first 4 bytes, and that number is equal to "Tree" (hex 054726565H) no swapping is required. If that "magic number" however is "eerT" (hex 065657254H), then byte-swapping is required. Please note, that the swapping requirement for a Tree file (*.tree) always applies to the complete file, while the raw data file (*.dat) must be swapped on a per-sweep basis. The reason is that Tree files are always written as one "native" file, while the raw data can originate from two different platforms, e.g., acquired on a Windows machine, and modified or analyzed under MacOS.

Note: For C/C++ programmers we offer a package called "DataAccess" which includes libraries that can be linked to your projects to read *POTPULSE* files under MacOS as well as Windows. If you need further information, please contact HEKA or your local distributor.

Experimental Examples

The *POTPULSE* software is a powerful and versatile tool for performing a wide variety of electrochemical measurements. The functions of the program are delegated to the various parts described in the preceding sections and can be categorized as follows:

- Pulse generation
- Data acquisition
- Data manipulation
- Data export

The *Pulse Generator* allows the generation of potential and current scans in the potentiostatic or galvanostatic mode in an arbitrary way, including:

- Wave form programming
- Timing
- Sweep compilation to series

A sweep is the base unit, which is defined by a pulse pattern to stimulate the cell. The wave form of the sweep is created by combining segments of ramps, constant values, sine waves, square waves, continuous, or conditional segments. For instance, a triangular voltage scan is represented by two ramps, which are defined by their starting voltages, vertex voltages, and durations or scan rates. Full flexibility of the relation with an *Initial Voltage* or *Initial Current* (in the galvanostatic mode) is achieved with the possibility of locking the sweep as an absolute or relative stimulus with respect to the initial value, or even of running it as an absolute voltage.

The durations of the segments determine the pulse length and together with the sample interval (that is the period between successive data points) they determine the number of recorded data points. The available buffer size that is allocated to data acquisition can be adjusted in the *POTPULSE* drop-down menu of the main program by choosing "Buffer Allocation...". For instance, the default value of 16 000 samples (16 ksamples) and a sample interval of 5 µs allow a sweep length of 80 ms. For the purpose of long-term experiments, both the buffer allocation and the sample interval should be adjusted.

Moreover, these individual sweeps can be compiled to form a series of sweeps. This option is particularly important for changing a segment parameter, such as the height of a constant segment in steps with the use of increments. The waiting time before the first sweep as well as between the individual sweeps can be defined. Thus, the duration of a series depends on the length of the individual sweeps, the number of sweeps, and the waiting time before each sweep. Note that not only the voltage (or current), but also the duration parameters of the segments, can be incremented, and thus the durations of the individual sweeps can be different within one series.

The setting of input and output triggers allows a correlation of the scan (i.e. a sweep or a series) timing with external events. The start of a scan can be triggered by an external trigger pulse. (The trigger input is at the rear of the PG310/390 potentiostat or ITC-16 board.) On the other hand, up to three triggers can be set and varied both in length and signal height by the Pulse Generator. For instance, these output triggers are very helpful in polarographic experiments for triggering the fall of the mercury drop.

A digital-to-analog (D/A) converter with 16-bit resolution of the ITC-16 data acquisition interface transmits the software-controlled *Initial Potential* (*Initial Current*) and generated scan to the cell. The signal is applied to the cell by means of a D/A output connector chosen by the operator.

On the other hand, one 16-bit analog-to-digital (A/D) converter at the interface allows data acquisition with 14-bit resolution (linearity up to 100 kHz, or 12-bit linearity up to 200 kHz) via eight different input channels. In the potentiostatic (galvanostatic) mode, the cell current (potential) will commonly be the first trace of the recorded data. Beside this one-channel recording, it is also possible to monitor two arbitrary traces. Although most electrochemical applications are operated by recording potential, current, and time, *POTPULSE* provides the possibility of recording any signal as the first or second trace, simply by defining the input channel and connecting the experimental setup in the appropriate way.

The recorded data are displayed on a digital oscilloscope (*Oscilloscope* window). Separate selection of the abscissa and ordinate provides the desired plot format. Please note that plotting of measured currents and potentials requires two-channel recording of both signals. The possible plot formats include many commonly used formats for electrochemical techniques, for instance:

i vs. E	all voltammetric techniques, voltage sweep techniques
i vs. t	chronoamperometry, coulometry
E vs. t	chronopotentiometry
i vs. $t^{1/2}$	chronoamperometry
1/i vs. $t^{1/2}$	chronoamperometry
Q vs. t	coulometry, chronocoulometry
Q vs. $t^{1/2}$	chronocoulometry
ln(i) vs. t	coulometry
ln(i) vs. E	Tafel plot

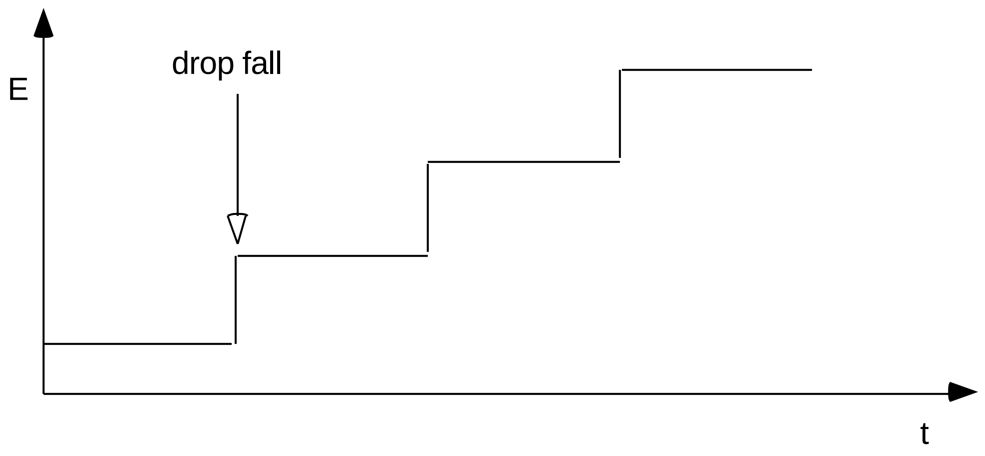
If several sweeps are compiled to form a series, some distinct characteristics of the sweeps, such as maximum current or mean current etc., can be analyzed and displayed in the *Online Analysis* window. *Online Analysis* display is particularly important if the electrochemical scan signal is built-up by a series of sweeps, and just a few selected properties or the current (potential) at appointed times are of interest. For instance, in normal pulse polarographic experiments the current at the end of a potential pulse is supposed to be recorded.

For further analysis of the data, it is possible to export both the sweep and the *Online Analysis* data to any graphing and data analysis programs. The Export mode in the Tree drop-down menu permits a decision on whether the sweep data, the *Online Analysis* data or both should be exported.

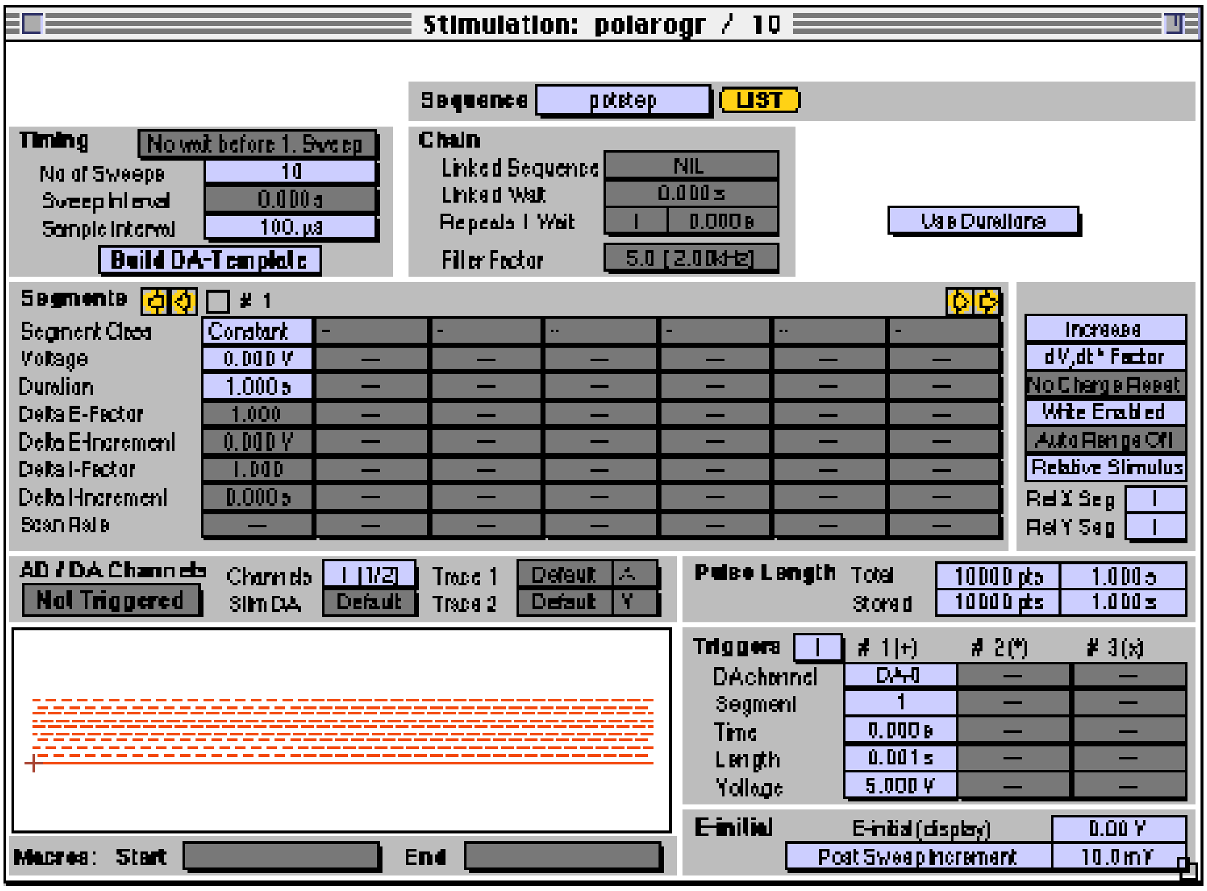
For every data acquisition, either opening an existing data file for modification or creating a new file is imperative before starting any intended data recording. Please make sure that the Store button in the *Oscilloscope* window is activated before starting an experiment.

In the following, some examples of *POTPULSE* applications are presented for frequently used electrochemical techniques. The wave form programming in these examples should not be regarded as the only possible way to attain the experimental goal, but rather as examples which illustrate the use of *POTPULSE* as a versatile electrochemical tool.

Potential Step Polarography



In potential step polarographic experiments the potential is changed stepwise. The polarogram is the display of the currents at the end of the steps versus the actual step potentials. Moreover, a potential step should occur when the mercury drop of the dropping mercury electrode (DME) is dislodged.



Stimulation: polarogr / 10

Sequence	potstep	LIST					

Timing	No wait before 1. Sweep
No of Sweeps	10
Sweep Interval	0.000 s
Sample Interval	100. µs
Build DA-Template	

Chain		
Linked Sequence	NIL	
Linked Wait	0.000 s	
Repeats / Wait	1	0.000 s
Filter Factor	5.0 [2.00kHz]	

Use Durations

Segments # 1							
Segment Class	Constant	-	-	-	-	-	-
Voltage	0.000 V	—	—	—	—	—	—
Duration	1.000 s	—	—	—	—	—	—
Delta E-Factor	1.000	—	—	—	—	—	—
Delta E-Increment	0.000 V	—	—	—	—	—	—
Delta t-Factor	1.000	—	—	—	—	—	—
Delta t-Increment	0.000 s	—	—	—	—	—	—
Scan Rate	—	—	—	—	—	—	—

Increase
dV,dt * Factor
No Charge Reset
Write Enabled
Auto Range Off
Relative Stimulus

Rel X Seg	1
Rel Y Seg	1

AD / DA Channels				
Not Triggered	Channels	1 [1/2]	Trace 1	Default A
	Stim DA	Default	Trace 2	Default V

Pulse Length			
	Total	10000 pts	1.000 s
	Stored	10000 pts	1.000 s

Triggers 1	# 1(+)	# 2(*)	# 3(x)
DA channel	DA-0	—	—
Segment	1	—	—
Time	0.000 s	—	—
Length	0.001 s	—	—
Voltage	5.000 V	—	—

E-initial		
	E-initial (display)	0.00 V
	Post Sweep Increment	10.0 mV

Macros: Start End

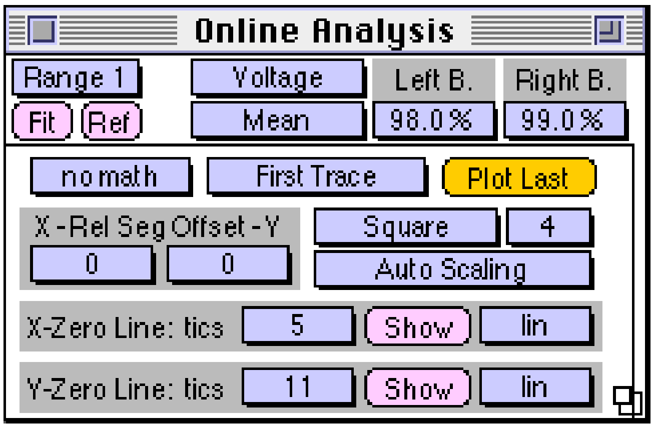
The pulse scheme can be realized as a series of individual sweeps, which contain one constant segment of appropriate duration. The potential is linearly incremented in the Initial Potential value after each sweep. That is, the potential is calculated as follows:

$$V_i = Voltage + (i\text{-}1)\ ?Initial\ Potential$$

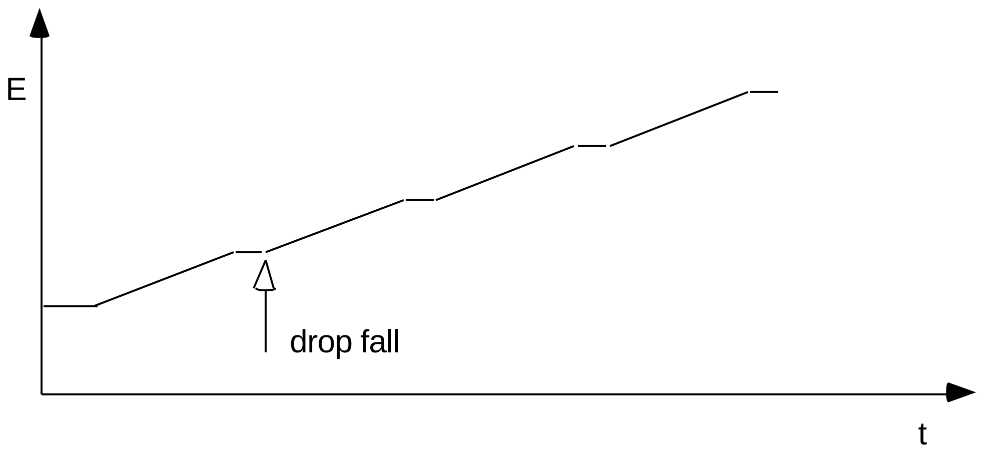
Voltage is set to 0 V, and the pulse is a Relative Stimulus with respect to the Initial Potential, which is displayed in the *Pulse Generator* window. The first applied potential will therefore be 0 mV, the second 10 mV, etc. up to the tenth of 90 mV relative to the first sweep Initial Potential. The x- and y-segments, which are relevant to the *Online Analysis*, can only be the first segments.

The number of sweeps is set to 10, and the sample interval to 100 µs. There is no waiting time before the first sweep and between the sweeps. The pulse length of the sweep is equal to the segment length of 1 s. Thus, the experiment will continue for 10 s, and a total of 100 000 data points are stored (each sweep comprises 10 000 data points). The recorded trace is defined as the default value in the potentiostatic mode, that is, the cell current. The drop fall is triggered by an output voltage pulse to the DA-0 output channel at the beginning of each sweep. The length and magnitude of the trigger signal should be adjusted to the DME in use and are set to some arbitrary values (pulse length 1 ms, pulse magnitude 5 V) in this case. The pulse template of the series is displayed as a time-wrapped overlay of ten sweeps. The time scale of the display is reduced to the length of a sweep. The trigger pulse is indicated by the cross at the beginning of the sweep.

The task of building up a polarogram is performed with the use of the *Online Analysis* function of *POTPULSE*. Since no offset is desired, the relevant segment for the *Online Analysis* is identical to that which has been chosen in the *Pulse Generator*. In this case, the first track of the recorded data, i.e. the current, is analyzed between the limits of 98 and 99 percent of the full time range of the relevant segment of each sweep. The mean value of the current between those limits will be plotted versus the applied voltage. With regard to the scan timing and sample interval, one percent of the full time range leads to an averaging of 100 data points, which are acquired during a period of 10 ms.



Tast Pulse Polarography



In tast pulse polarographic experiments the potential is changed ramp wise during a period that is large in comparison with a constant potential phase after the end of the ramp. The polarogram is the display of the currents at the end of the constant segments versus the actual step potentials. Moreover, the mercury drop of the dropping mercury electrode (DME) should be dislodged immediately after a new potential ramp is started.

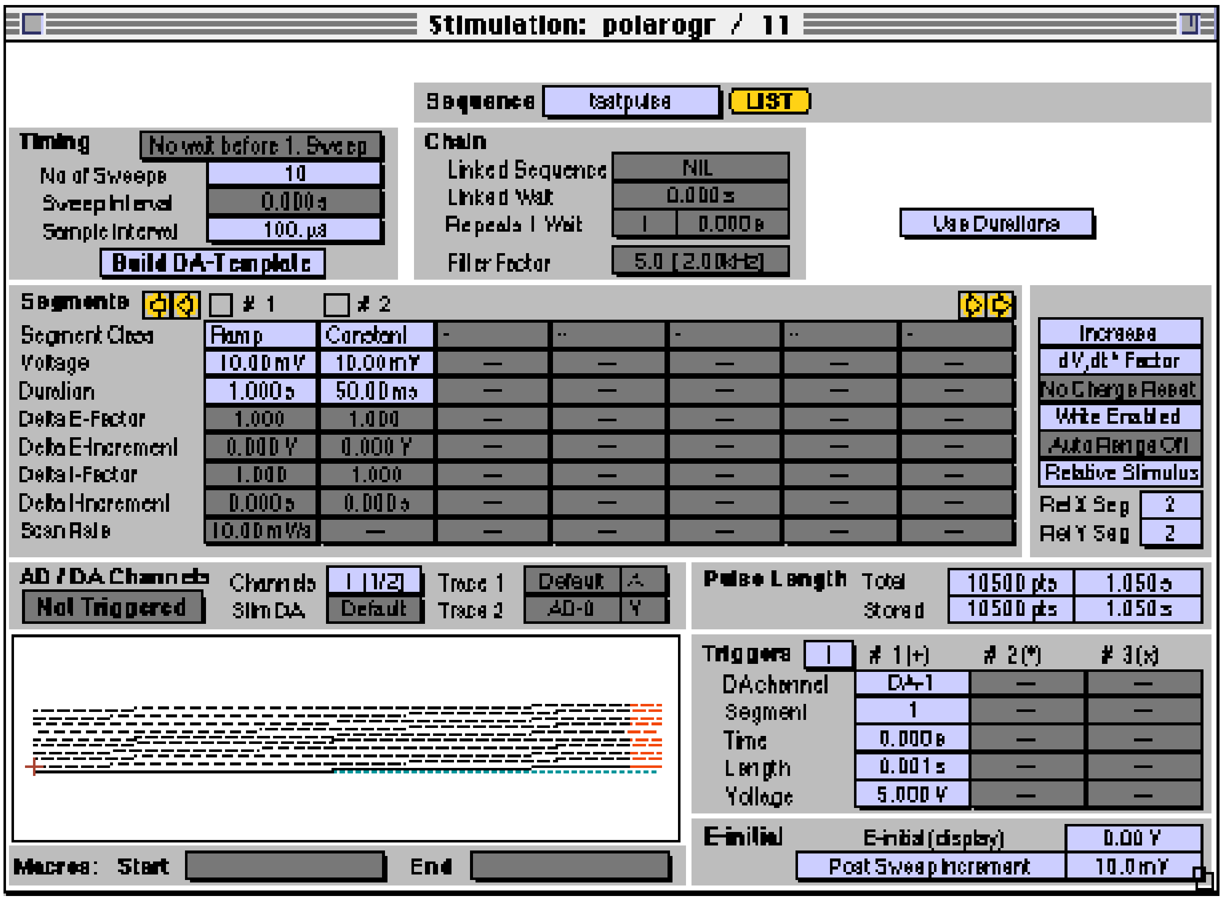
The pulse scheme can be realized as a series of individual sweeps, which are a combination of a ramp followed by a constant segment. The ramp segment is characterized by the vertex potential of 10 mV and the ramp duration of 1 s. The scan rate of 10 mV/s is calculated by the program and shown in the lowest row of the segment. The following constant segment fixes the potential at 10 mV, that is, the potential identical to the ramp vertex, during a period of 50 µs. The Initial Potential is linearly incremented with 10 mV after each sweep, and the sweep stimulates the cell in relation to the Initial Potential. That is, the potential at the end of the individual sweeps is calculated as follows:

$$V_i = Voltage(constant\ segment) + (i\text{-}1)\ ?Initial\ Potential$$

The applied potential of the first constant segment will therefore be 10 mV, the second 20 mV, etc. up to the tenth of 100 mV relative to the first sweep Initial Potential. The succeeding sweep will start at the potential of the preceding constant segment. The x- and y-segments, which are relevant to the *Online Analysis* are fixed to the constant, that is, the second segment.

The number of sweeps is set to 10, and the sample interval to 100 µs. There is no waiting time before the first sweep and between the sweeps. The pulse length of a sweep is equal to the sum of the segment lengths and is calculated to 1.05 s. Thus, the experiment will continue for 10.5 s, and a total of 105 000 data points are stored (each sweep comprises 10 500 data points). The recorded trace is defined as the default value in the potentiostatic mode, i.e. the cell current.

The drop fall is triggered by an output voltage pulse to the DA-1 output channel at the beginning of each ramp. The length and magnitude of the trigger signal should be adjusted to the DME in use, and are set to some arbitrary values (pulse length 1 ms, pulse magnitude 5 V) in this case. The pulse template of the series is displayed as a time-wrapped overlay of ten sweeps. The time scale of the display is reduced to the length of a sweep. The trigger pulse is indicated by the cross at the beginning of the sweep.



Stimulation: polarogr / 11

Sequence: testpulse — LIST

Timing: No wait before 1. Sweep

No of Sweeps	10
Sweep Interval	0.000 s
Sample Interval	100. µs

Build DA-Template

Chain

Linked Sequence	NIL
Linked Wait	0.000 s
Repeats / Wait	1 / 0.000 s
Filter Factor	5.0 [2.00kHz]

Use Durations

Segments — # 1 — # 2

Segment Class	Ramp	Constant	-	..	-	..	-
Voltage	10.00 mV	10.00 mV	—	—	—	—	—
Duration	1.000 s	50.00 ms	—	—	—	—	—
Delta E-Factor	1.000	1.000	—	—	—	—	—
Delta E-Increment	0.000 V	0.000 V	—	—	—	—	—
Delta t-Factor	1.000	1.000	—	—	—	—	—
Delta t-Increment	0.000 s	0.000 s	—	—	—	—	—
Scan Rate	10.00 mV/s	—	—	—	—	—	—

Increase
dV,dt* Factor
No Charge Reset
Write Enabled
Auto Range Off
Relative Stimulus

Rel X Seg	2
Rel Y Seg	2

AD / DA Channels: Not Triggered

Channels	1 [1/2]	Trace 1	Default	A
Stim DA	Default	Trace 2	AD-0	V

Pulse Length

Total	10500 pts	1.050 s
Stored	10500 pts	1.050 s

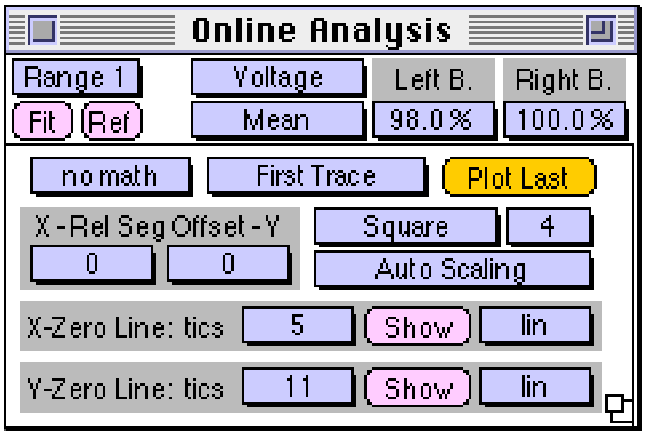
Triggers: 1

	# 1 (+)	# 2 (*)	# 3 (x)
DA channel	DA-1	—	—
Segment	1	—	—
Time	0.000 s	—	—
Length	0.001 s	—	—
Voltage	5.000 V	—	—

E-initial

E-initial (display)	0.00 V
Post Sweep Increment	10.0 mV

Macros: Start — End


Online Analysis

Range 1 | Voltage | Left B. | Right B.

Fit | Ref | Mean | 98.0 % | 100.0 %

no math | First Trace | Plot Last

X-Rel Seg Offset-Y: 0 | 0

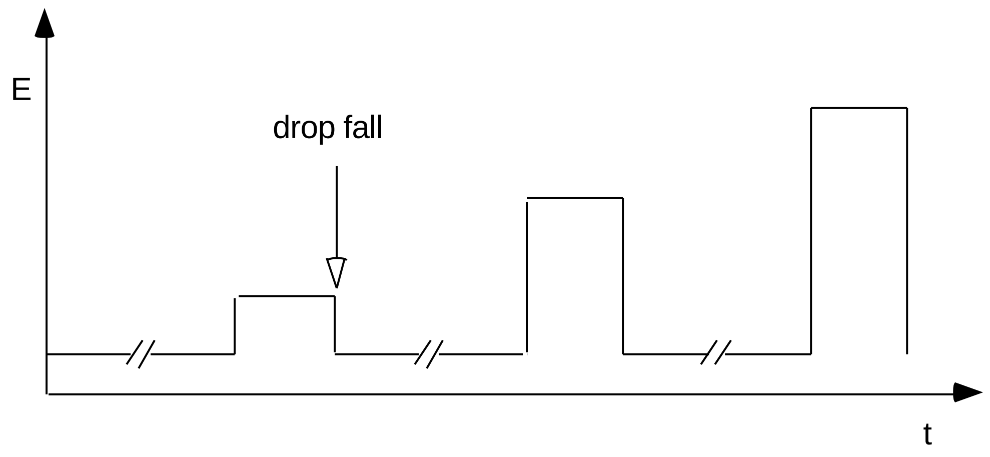
Square | 4 | Auto Scaling

X-Zero Line: tics | 5 | Show | lin

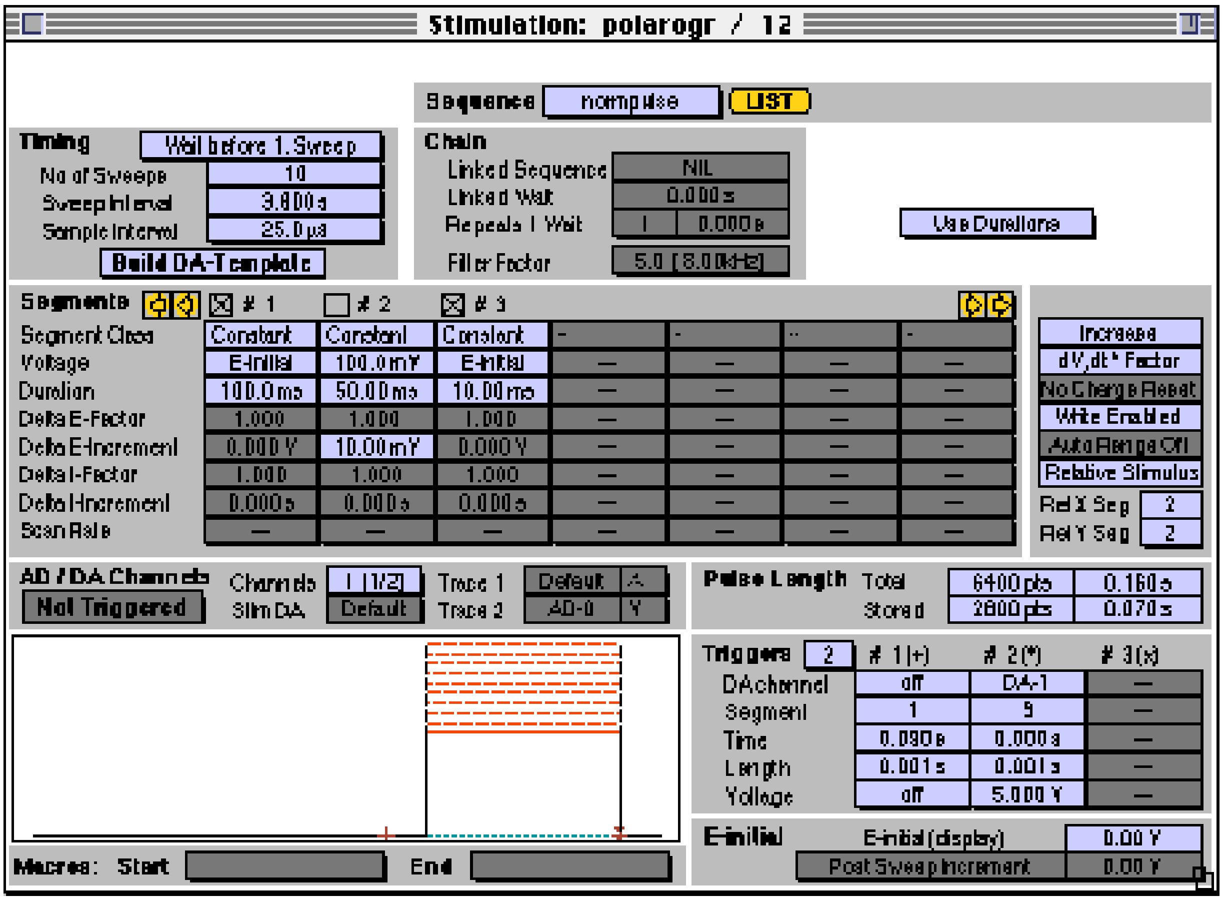
Y-Zero Line: tics | 11 | Show | lin

The task of building up a polarogram is performed with the use of the *Online Analysis* function of *POTPULSE*. Since no offset is desired, the relevant segment for the *Online Analysis* is identical to that which has been chosen in the *Pulse Generator*. In this case, the first track of the recorded data, i.e. the current, is analyzed between the limits of 98 and 100 percent of the full time range of the relevant segment of each sweep. The mean value of the current between those limits will be plotted versus the applied voltage. With regarding to the sweep timing and sample interval, 2 percent of the full time range leads to an averaging of 10 data points, which are acquired during a period of 1 ms.

Normal Pulse Polarography



With the use of the normal pulse polarographic technique, some short pulses with growing magnitudes are superimposed on an Initial Potential. The polarogram is the display of the currents at the end of a potential pulse versus the actual step potentials. Moreover, the mercury drop of the dropping mercury electrode (DME) should be dislodged immediately after the potential pulse.


Stimulation: polarogr / 12

Sequence: nompulse — LIST

Timing	Wait before 1. Sweep
No of Sweeps	10
Sweep Interval	3.800 s
Sample Interval	25.0 µs

Build DA-Template

Chain	
Linked Sequence	NIL
Linked Wait	0.000 s
Repeats / Wait	1 / 0.000 s
Filter Factor	5.0 [8.00kHz]

Use Durations

Segments	☒ # 1	☐ # 2	☒ # 3				
Segment Class	Constant	Constant	Constant	-	-	-	-
Voltage	E-Initial	100.0 mV	E-Initial	—	—	—	—
Duration	100.0 ms	50.00 ms	10.00 ms	—	—	—	—
Delta E-Factor	1.000	1.000	1.000	—	—	—	—
Delta E-Increment	0.000 V	10.00 mV	0.000 V	—	—	—	—
Delta t-Factor	1.000	1.000	1.000	—	—	—	—
Delta t-Increment	0.000 s	0.000 s	0.000 s	—	—	—	—
Scan Rate	—	—	—	—	—	—	—

Increase
dV/dt * Factor
No Charge Reset
Write Enabled
Auto Range Off
Relative Stimulus

Rel X Seg	2
Rel Y Seg	2

AD / DA Channels — Not Triggered

Channels	1 (1/2)	Trace 1	Default	A
Stim DA	Default	Trace 2	AD-0	V

Pulse Length		
Total	6400 pts	0.160 s
Stored	2800 pts	0.070 s

Triggers 2	# 1(+)	# 2(*)	# 3(x)
DA channel	off	DA-1	—
Segment	1	3	—
Time	0.090 s	0.000 s	—
Length	0.001 s	0.001 s	—
Voltage	off	5.000 V	—

E-initial		
	E-initial (display)	0.00 V
	Post Sweep Increment	0.00 V

Macros: Start — End

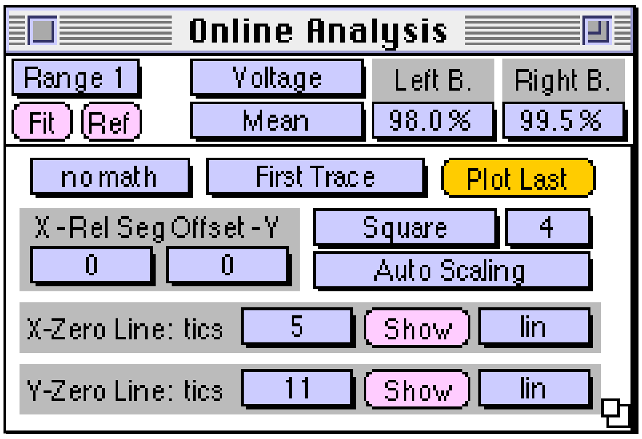
The pulse scheme can be realized as a series of individual sweeps, which are an assembly of an Initial Potential segment followed by a constant segment and a finishing Initial Potential segment. The first segment is characterized by its fixed potential and the duration of 100 ms. The following constant segment causes a potential pulse with a pulse length of 50 ms. The magnitude is linearly incremented by 10 mV after each sweep and the sweep stimulates the cell in relation to the Initial Potential. That is, the magnitude of the potential pulse of the individual sweeps is calculated as follows:

$$V_i = Voltage(2nd\ segment) + (i-1)\ ?V$$

The applied potential of the first pulse will therefore be 100 mV, the second 110 mV, etc. up to the tenth of 190 mV relative to the Initial Potential. Each sweep is finished with a constant Initial Potential segment. The x- and y-segments, which are relevant to the *Online Analysis*, are fixed to the potential pulse, that is, the second segment.

The number of sweeps is set to 10, and the sample interval to 25 µs. There is a waiting time of 3.8 s before the first sweep and between the sweeps. The pulse length of a sweep is equal to the sum of the segment lengths and is calculated to 160 ms. Thus, the experiment will continue for 39.6 s, and the number of data points is calculated to 6400 for each sweep. However, only 2800 data points are stored for each sweep, and a total of 28 000 data points are stored. The reason for the reduction in data recording is the setting of the first trigger to 90 ms; thus, the storing of data will start at this time. ***Bear in mind that the program will never store any data before the first trigger signal.*** In this case, the long waiting time of 3.9 s between two pulses is divided into a waiting time between the sweeps and a 100 ms Initial Potential segment. The advantages of this programming style are the flexibility in setting the first trigger, if one is interested in data points before the pulse starts, and the time-stretching feature without high memory consumption, which are needed for a huge number of sweep points. The first trigger should affect only the data storage; its output channel and its magnitude are set to Off. The recorded trace is defined as the default value in the potentiostatic mode, i.e. the cell current.

The drop fall is triggered (2nd trigger) by an output voltage pulse to the DA-1 output channel at the beginning of the third segment, that is, immediately after each pulse. The length and magnitude of the trigger signal should be adjusted to the DME in use and are set to some arbitrary values (pulse length 1 ms, pulse magnitude 5 V) in this case. The pulse template of the series is displayed as a time-wrapped overlay of ten sweeps. The time scale of the display is reduced to the length of a sweep. The trigger pulses are indicated by the crosses at 90 ms in the first segment and at the beginning of the third segment.


Online Analysis

Range 1 | Voltage | Left B. | Right B.
Fit | Ref | Mean | 98.0 % | 99.5 %

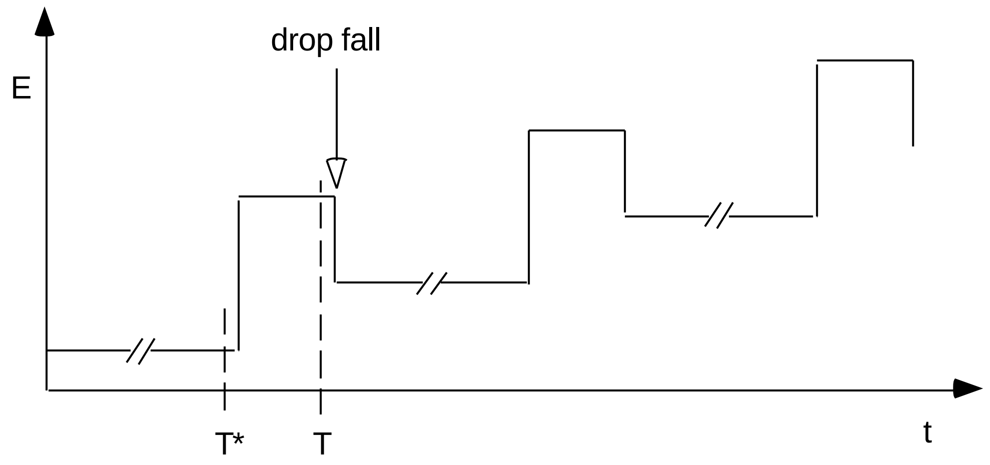
no math | First Trace | Plot Last

X-Rel Seg Offset-Y: 0 | 0
Square | 4
Auto Scaling

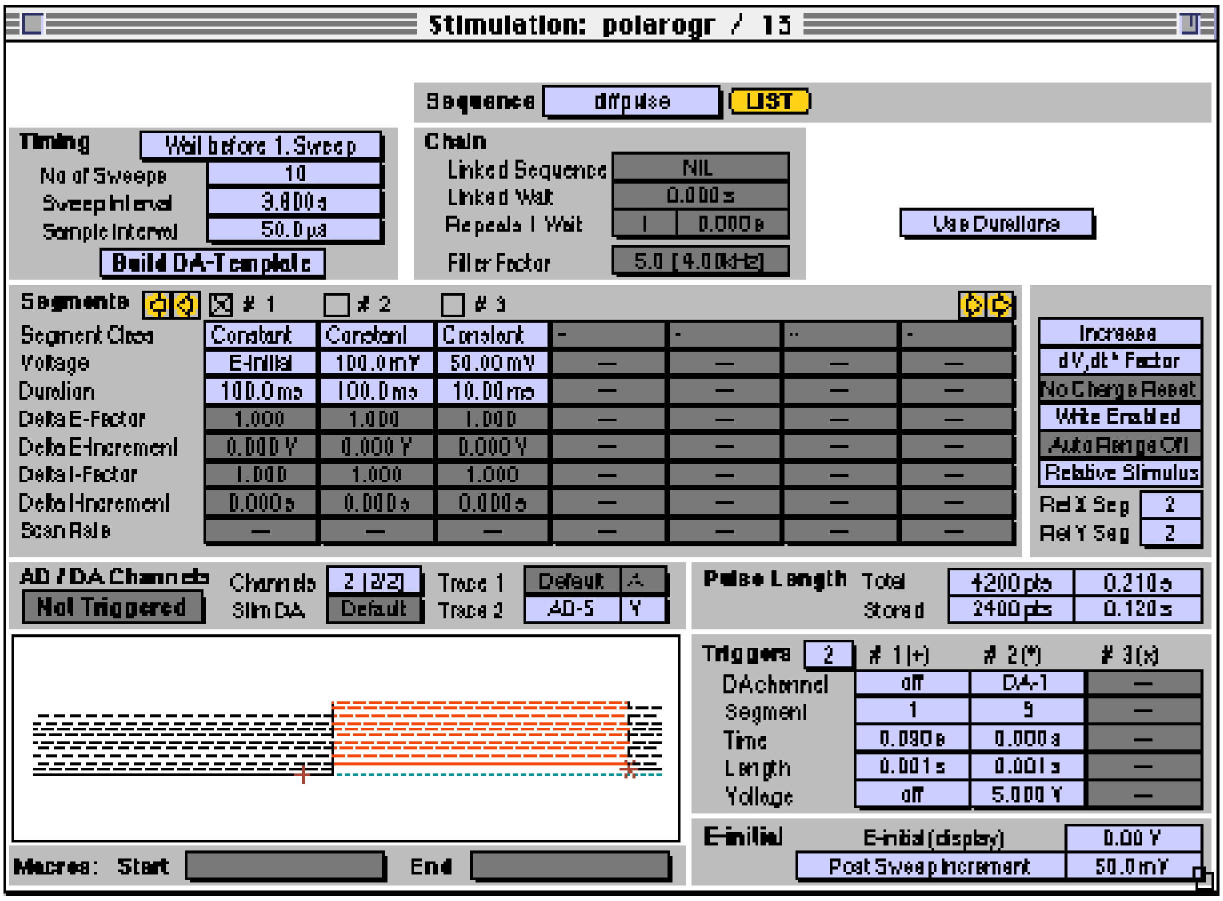
X-Zero Line: tics | 5 | Show | lin
Y-Zero Line: tics | 11 | Show | lin

The task of building up the desired polarogram is accomplished with the use of the *Online Analysis* function of *POTPULSE*. Since no offset is given, the relevant segment for the *Online Analysis* is identical to that which has been chosen in the *Pulse Generator*. In this case, the first track of the recorded data, i.e. the current, is analyzed between the limits of 98 and 99.5 percent of the full time range of the relevant segment of each sweep. The mean value of the current between those limits will be plotted versus the applied voltage. With regarding to the scan timing and sample interval, 1.5 percent of the full time range leads to an averaging of 30 data points, which are acquired during a period of 750 µs.

Differential Pulse Polarography



Differential pulse polarographic experiments involve long periods at constant potential followed by a short potential pulse. Typically, this pattern is repeated with successive growth of both potential magnitudes. The polarogram is the display of the differences in current at the end of the short pulses and at a time just before the pulses (i(T)-i(T*)) versus the actual pulse potentials. Moreover, the mercury drop of the dropping mercury electrode (DME) should be dislodged immediately after the end of a pulse.



Stimulation: polarogr / 13

Sequence: difpulse — LIST

Timing: Wait before 1. Sweep

No of Sweeps	10
Sweep Interval	3.800 s
Sample Interval	50.0 µs

Build DA-Template

Chain

Linked Sequence	NIL
Linked Wait	0.000 s
Repeats / Wait	1 / 0.000 s
Filter Factor	5.0 [4.00kHz]

Use Durations

Segments ☒ # 1 ☐ # 2 ☐ # 3

Segment Class	Constant	Constant	Constant	-	-	-	-
Voltage	E-initial	100.0 mV	50.00 mV	—	—	—	—
Duration	100.0 ms	100.0 ms	10.00 ms	—	—	—	—
Delta E-Factor	1.000	1.000	1.000	—	—	—	—
Delta E-Increment	0.000 V	0.000 V	0.000 V	—	—	—	—
Delta t-Factor	1.000	1.000	1.000	—	—	—	—
Delta t-Increment	0.000 s	0.000 s	0.000 s	—	—	—	—
Scan Rate	—	—	—	—	—	—	—

Increase
dV,dt* Factor
No Charge Reset
Write Enabled
Auto Range Off
Relative Stimulus

Rel X Seg	2
Rel Y Seg	2

AD / DA Channels — Not Triggered

Channels	2 [2/2]	Trace 1	Default	A
Stim DA	Default	Trace 2	AD-5	V

Pulse Length

Total	4200 pts	0.210 s
Stored	2400 pts	0.120 s

Triggers 2

	# 1 (+)	# 2 (*)	# 3 (x)
DA channel	off	DA-1	—
Segment	1	3	—
Time	0.090 s	0.000 s	—
Length	0.001 s	0.001 s	—
Voltage	off	5.000 V	—

E-initial

E-initial (display)	0.00 V
Post Sweep Increment	50.0 mV

Macros: Start — End

The pulse scheme can be realized as a series of individual sweeps, which assembles an Initial Potential segment followed by a constant segment and a second and finishing constant segment. The first segment is characterized by its fixed potential to the value of Initial Potential and the duration of 100 ms. The following constant segment causes a potential pulse with a pulse length of 100 ms and a potential difference of 100 mV with respect to the first segment (relative stimulus). The third segment reduces the potential by the amount of 50 mV and remains at this constant value during a period of 10 ms. Since the Initial Potential is incremented by 50 mV, the following sweep of the series will start at the potential which is determined in the last segment of the actual sweep.

That is, the magnitude of the potential pulse (2nd segment) of each individual sweep is calculated as follows:

$$V_i = \text{Initial Potential}_i + \text{Voltage(2nd segment)}$$

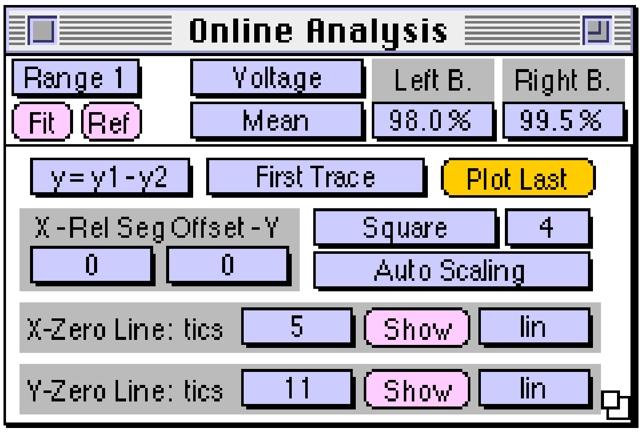
with

$$\text{Initial Potential}_i = \text{Initial Potential(1st sweep)} + (i\text{-}1)\ ?\text{Initial Potential}$$

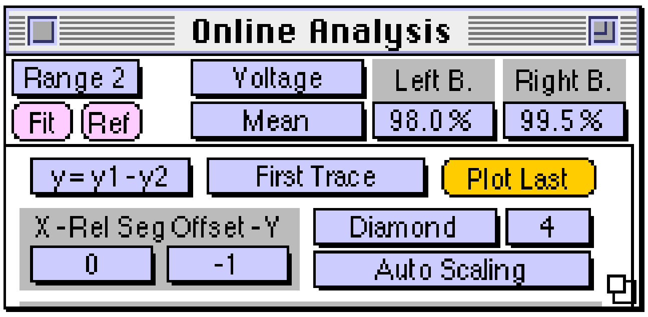
The applied potential of the first pulse will therefore be 100 mV, the second 150 mV, etc. up to the tenth of 550 mV relative to the Initial Potential of the first sweep. The x- and y-segments, which are relevant to the *Online Analysis*, are fixed to the potential pulse, that is, the second segment.

The number of sweeps is set to 10, and the sample interval to 50 µs. There is a waiting time of 3.8 s before the first sweep and between the sweeps. The pulse length of a sweep is equal to the sum of the segment lengths and is calculated to 210 ms. Thus, the experiment will continue for 40.01 s, and the number of data points is calculated to 4200 for each sweep. However, only a 2.400 data points are stored for each sweep and a total of 24 000 data points are stored. The reason for the reduction in data recording is the setting of the first trigger to 90 ms; thus, the storing of the data will start at this time. ***Bear in mind that the program will never store any data before the first trigger signal.*** In this case, the long waiting time of 3.91 s between two pulses is divided into a constant segment (10 ms), a waiting time between the sweeps (3.8 s) and a 100 ms Initial Potential segment. During this period, the potential remains at a value 50 mV lower than the last pulse potential (pulse height 100 mV) and 50 mV higher than the preceding Initial Potential. The advantages of this programming style are the flexibility in setting the first trigger, since one is interested in data points before the pulse starts, and the time stretching-feature without high memory consumption, which are needed for a huge number of sweep points. The first trigger should affect only the data storage; its output channel and its magnitude are set to Off. The recorded traces are defined as defaults in the potentiostatic mode, i.e. the cell current for the first trace, and the potential for the second trace. The potential is supported at the AD-5 channel of the ITC-16 interface, that is, the U-Cell Monitor channel of a PG 300 potentiostat/galvanostat.

The drop fall is triggered (2nd trigger) by an output voltage pulse at the DA-1 output channel at the beginning of the third segment, that is, immediately after each pulse. The length and magnitude of the trigger signal should be adjusted to the DME in use, and are set to some arbitrary values (pulse length 1 ms, pulse magnitude 5 V) in this case. The pulse template of the series is displayed as a time wrapped overlay of ten sweeps. The time scale of the display is reduced to the length of a sweep. The trigger pulses are indicated by the crosses at 90 ms in the first segment and at the beginning of the third segment.


Online Analysis
Range 1 | Voltage | Left B. | Right B.
Fit | Ref | Mean | 98.0 % | 99.5 %
y = y1 - y2 | First Trace | Plot Last
X - Rel Seg Offset - Y: 0 | 0
Square | 4
Auto Scaling
X-Zero Line: tics 5 | Show | lin
Y-Zero Line: tics 11 | Show | lin

The task of building up the desired polarogram is accomplished with the use of the *Online Analysis* function of *POTPULSE*. Since no offset is set, the relevant segment for Range 1 of the *Online Analysis* is identical to that which has been chosen in the *Pulse Generator*. In this case, the first track of the recorded data, i.e. the current, is analyzed between the limits of 98 and 99.5 percent of the full time range of the relevant segment of each sweep. With regarding to the scan timing and sample interval, 1.5 percent of the full time range leads to an averaging of 30 data points, which are acquired during a period of 1.5 ms.

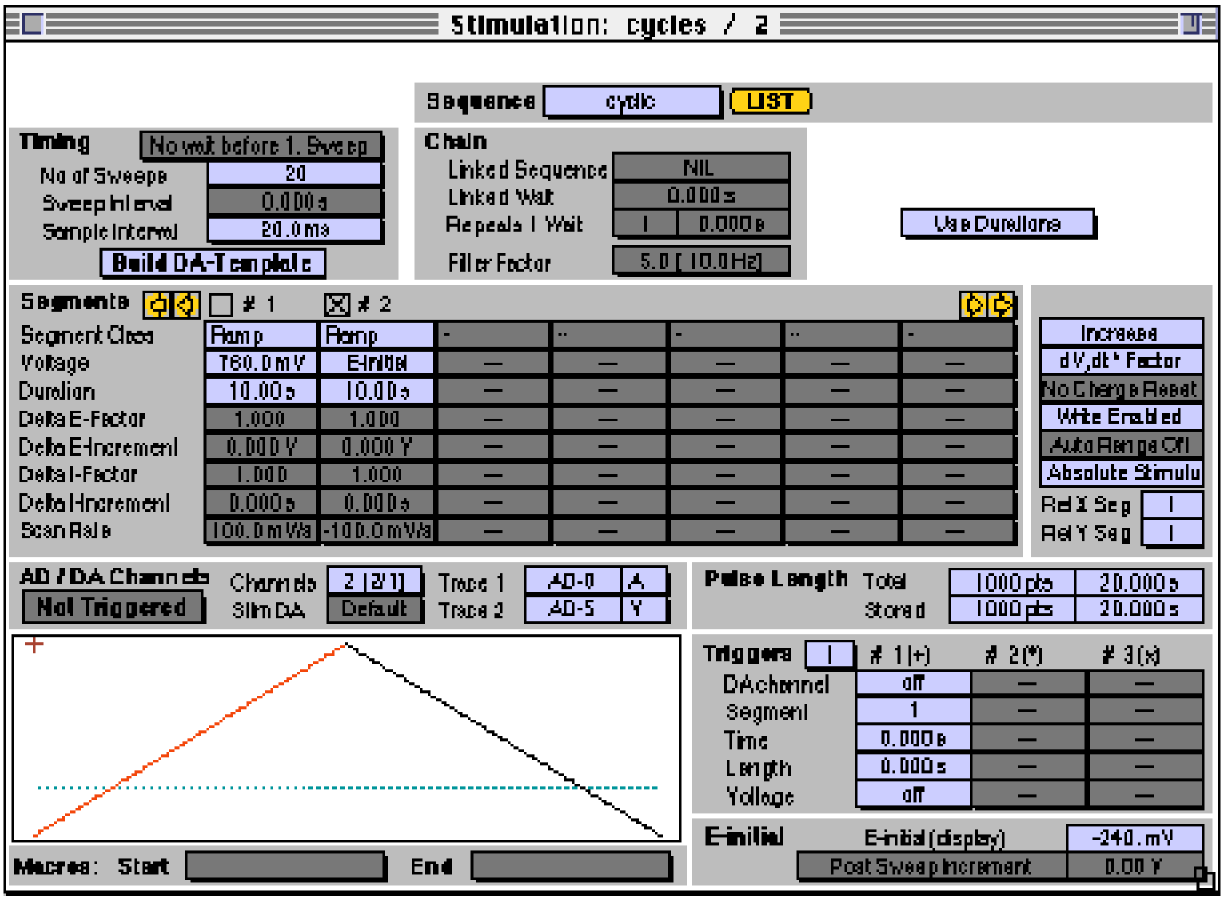

Online Analysis
Range 2 | Voltage | Left B. | Right B.
Fit | Ref | Mean | 98.0 % | 99.5 %
y = y1 - y2 | First Trace | Plot Last
X - Rel Seg Offset - Y: 0 | -1
Diamond | 4
Auto Scaling

The relevant x-segment of Range 2 is identical with that of Range 1. However, the relevant y-segment number is reduced by one with respect to the value set in the *Pulse Generator*. In this case, the current of Range 2 is analyzed in the initial potential segment, which runs before the pulse. Moreover, an average of 30 data points, sampled during a period of 1.5 ms, is calculated in this segment. Thus, with the help of the arithmetic function "Y=Y1-Y2" the *Online Analysis* calculates the desired difference of the two currents, taken at the end of a pulse and a short time before the pulse starts:

$$Y = i(T) - i(T^*).$$

This calculated value of the current difference will be plotted versus the applied pulse voltage.

Cyclic Voltammetry


Stimulation: cycles / 2

Sequence: cyclic — LIST

Timing: No wait before 1. Sweep

No of Sweeps	20
Sweep Interval	0.000 s
Sample Interval	20.0 ms

Build DA-Template

Chain

Linked Sequence	NIL
Linked Wait	0.000 s
Repeats / Wait	1 / 0.000 s
Filter Factor	5.0 [10.0 Hz]

Use Durations

Segments ☐ # 1 ☒ # 2

Segment Class	Ramp	Ramp	-	..	-	..	-
Voltage	760.0 mV	E-initial	—	—	—	—	—
Duration	10.00 s	10.00 s	—	—	—	—	—
Delta E-Factor	1.000	1.000	—	—	—	—	—
Delta E-Increment	0.000 V	0.000 V	—	—	—	—	—
Delta t-Factor	1.000	1.000	—	—	—	—	—
Delta t-Increment	0.000 s	0.000 s	—	—	—	—	—
Scan Rate	100.0 mV/s	-100.0 mV/s	—	—	—	—	—

Increase
dV,dt * Factor
No Charge Reset
Write Enabled
Auto Range Off
Absolute Stimulu

Rel X Seg	1
Rel Y Seg	1

AD / DA Channels: Not Triggered

Channels	2 [2/1]	Trace 1	AD-0	A
Stim DA	Default	Trace 2	AD-5	V

Pulse Length

Total	1000 pts	20.000 s
Stored	1000 pts	20.000 s

Triggers 1

	# 1 (+)	# 2 (*)	# 3 (x)
DA channel	off	—	—
Segment	1	—	—
Time	0.000 s	—	—
Length	0.000 s	—	—
Voltage	off	—	—

E-initial

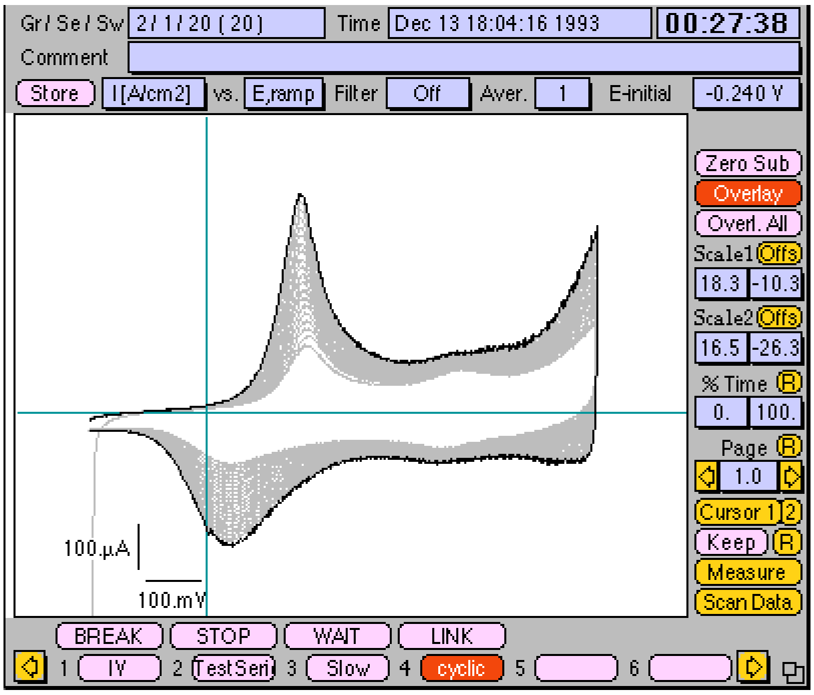
E-initial (display)	-240. mV
Post Sweep Increment	0.00 V

Macros: Start End

In cyclic voltammetric experiments two linear voltage ramps are combined to form a triangular potential pattern.

The first ramp starts at the Initial Potential of -240 mV and ends at the vertex voltage of 760 mV (absolute voltage). The end point of the second ramp is identical with the starting point of the first ramp of the succeeding sweep: the initial potential. The duration of each ramp is set to 10 s. The scan rate of 100 mV/s is calculated by the program and shown in the lowest row of each segment. Data points are sampled at 20 ms intervals, with a resulting sweep of 1000 data points. Since 20 individual sweeps are compiled to form a series, a total of 20 000 data points are sampled and stored. The data points contain information on the time as well as the recorded current and potential, because two-channel data acquisition has been chosen. The current is recorded through the AD-0 channel, and the potential through the AD-5 channel of the *ITC-16* interface board. The first trigger is set at the beginning of the sweep, and the output channel is set to Off. Therefore, no trigger pulse is emitted, but the data storage will start at the first data point of the sweep. In the pulse template the trigger is indicated by a cross.

The commonly used data display "i vs. E" can be realized within the digital *Oscilloscope*.


Gr / Se / Sw 2 / 1 / 20 (20) Time Dec 13 18:04:16 1993 00:27:38

Comment

Store I[A/cm2] vs. E,ramp Filter Off Aver. 1 E-initial -0.240 V

Scale1 Offs 18.3 -10.3

Scale2 Offs 16.5 -26.3

% Time R 0. 100.

Page R 1.0

The current density is plotted as a function of the applied potential. By means of the Overlay function, every sweep of the series can be displayed. The curve expansion can be easily changed by outlining the boundaries with the mouse and clicking. The Scan Data and Measure functions are helpful in analyzing the curves. For instance, peak currents or minimum current findings are supported by the Scan Data function. The display mode is not limited to the "i vs. E" mode; further modes like "i vs. t" or "Q vs. t" are also possible by choosing the desired ordinate and abscissa above the display window.

Questions & Answers

Notebook information lost

Q: In the course of a normal experiment I have annotated the notebook (detailing drug additions etc.). When I print the notebook this information is also printed. However, despite saving the notebook, when I re-open the associated file I find that whereas any online analysis has been stored, all the things which I typed myself into the notebook have been lost - what am I doing wrong?

A: When Buffered output in the Notebook Menu is selected, the online analysis and the comments typed via keyboard are written to the notebook. All contents of the notebook is saved as a text file if

a) Auto Store is selected or

b) The user saves it manually (Notebook → Save / Nootebook → Save as).

Note: With b), the notebook is by default stored in a directory different from the one with your data file. Specifiy the directory when saving the notebook. The size of the notebook is set to 1000 lines by default. It can be changed with Menu: Notebook/set length. The notebook can also be printed (File → print Notebook).

Troubleshooting

Restoring POTPULSE Data after a Computer Crash

POTPULSE stores every experiment into three files:

- The file with the extension *.pgf contains the stimulus templates used in a given experiment. The templates will be copied from your PGF pool into the running experiment.
- The *.pul file has the complete data tree as visualized in the Replay window. It includes e.g. all the amplifier settings such as filters or gain and the timing of the data.
- The *.dat file has only the actual raw data without any timing or scaling information.

With the option Auto File Update enabled *POTPULSE* writes the raw data after every sweep. The other two files are written only when you close or update the experiment by selecting Update File or Close from the POTPULSE menu. If the computer crashes during a running experiment, you will lose the *.pgf and the *.pul file. *POTPULSE* will not be able to reopen this experiment, although the *.dat file contains valid data with a high probability. However, there's a good chance to restore the two corrupted files in order to be able to access the experiment:

Backup the *.dat file to a safe place.

In *POTPULSE* create a new experiment. Call it whatever you want, e.g. something like Test.dat. With the new experiment opened, simulate your lost experiment by calling all the pgf protocols in the same order as you did during the experiment. You can use the model circuit or you may shield the probe.

Now you can throw away the new Test.dat file to replace it by your old experiment. Rename your raw data file into Test.dat or the two files Test.pul and Test.pgf using the name of your experiment. It is important, that all three files have the same name and only differ in their file name extension. This is the only criterion *POTPULSE* checks for files that belong to one experiment.

Now you can reopen the data file!

Unfortunately, the time information is completely useless, however, the most important thing - your data - should still be there.

Printing Problems

Printing Problems under MacOS

Most printing problems on MacOS computers are caused by three types of problems:

You changed the printer in the Chooser while running *POTPULSE* version 8.09 and earlier. Thus, if a printing problem occurs, check whether it does occur when the printer selection is made before you start *POTPULSE*

Another source of problems may be the printer driver itself. Some of Apple's printer drivers show peculiar incompatibilities. All our programs do print with the LaserWriter printer driver 8.51. Printer driver 7.0 and 8.1 were the cause of some incompatibilities with *POTPULSE.*

The most frequent printing problem type is due to *out-of-memory* situations of the printer drivers. The default memory settings are usually too tight especially when you print to a laser printer. This produces misleading error messages, such as *"printing 312 jobs"* and the like. The printer driver is located in the Extensions folder within the System Folder and called PrintMonitor, in case of a HP-printer its name is HP PrintMonitor. If you have the *Desktop Printing Extension* installed that comes with System 7.5 and higher, there is a second printer driver called Desktop PrintMonitor in the same place. The default memory assignments is 160 kB for the PrintMonitor and 117 kB for the Desktop PrintMonitor. To relieve out-of-memory situations you should increase the memory allocation of **both** drivers by about 400 kB using the Finder's File → Get Info menu.

Here some further steps you can try to relieve persisting printing problems:

- Turn Virtual Memory on in the Memory Control Panel and make sure that the Modern memory manager is selected and 32-bit memory is enabled (if your computer allows to turn it off). This may cure the general lack of memory.
- Assign 500 kB more memory to *POTPULSE* itself in the Get Info window. This may cure those drivers which grab memory directly from the application instead of the system heap like the HP-drivers!
- Install more physical RAM. It may not only cure the printing problems, but also speed up the performance of your computer. 16 MB of physical RAM is the minimum to run *POTPULSE* with System 7.5, 24 MB is at least required with System 8.x.
- Print to a PostScript file and send this file to the printer after quitting *POTPULSE.*

Finally, does your printer have enough RAM to print? The typical RAM-requirement is at least 2 MB for a 300x300 dpi printer, 4 MB for 600x300, and 8 MB for 600x600 dpi, plus about 1 MB.

Printing Problems under Windows 3.1 and 95

It can happen that *POTPULSE* begins to crash after one printed something. This can be caused by a printer driver which enables math-exceptions for its own use. By that, *POTPULSE* will not be able to handle correctly math-exceptions, and may crash, e.g., when performing power functions during fitting routines. There is no solution to that problem, besides finding another printer driver which does not show that problem. You can switch to Windows NT where this problem does not occur.

Printing Troubleshooter

Try the following to identify what the source of the problems may be:

Q: Can you print the Notebook?

A: This is a small and fast job. It tests the printer connection, initialization and page ejection. If that test is working it indicates that the *POTPULSE* printer routines are working properly.

Q: Can you print one Series on one page? Are you sure you waited long enough?

A: We measured the time it took to print the “Tail → IV 2” series of the Demo.dat experiment. On an otherwise speedy PowerPC 8500 printing to a Personal Laser Writer NT took **five** minutes without and **three** minutes with Tree → Export → Print Compressed Vectors selected. The same using a Pentium II (300 MHz) on a HP LaserJet 6P took about 15 seconds in both cases.

Q: Does it make a difference, if you print with or without the option Tree → Export → Print Compressed Vectors *selected?*

A: If these tests perform correctly, then most probably the problems are located outside *POTPULSE*!

MacOS Specific Problems

”Offscreen” Error Message after starting POTPULSE

The so called “offscreen” is a feature to accelerate the redrawing of windows. *POTPULSE* reserves some memory to redraw the window in the background and updates it after the redraw has finished. The amount of necessary memory depends on the size of the window. Sometimes however, *POTPULSE* or one of its extensions stores a wrong (negative) size of windows. When the program restarts again it tries to allocate the buffer for that window according to the dimensions, which become tremendously huge because the negative values are converted automatically into huge positive ones. E.g. -1 x -1 becomes 32767 x 32767 which would result in a memory requirement of 1 GB at a resolution of 256 colors. Of course, you don't have that amount of memory, so you get this error message. You should get rid of the problem by deleting all *POTPULSE* preferences: DefaultPulse.set in the POTPULSE folder and the files Pulse.Settings and PulseFit.Settings in the Preferences folder inside the System Folder. Take care, you will have to reconfigure *POTPULSE* again after throwing away its preferences!

Wrong Paths when storing Files

The MacOS gives you three possibilities, which path will be set by default when calling the *Open* or *Save* dialog. This setting is available through the General Controls control panel. Usually, you should keep the Folder that is set by the application setting. This will allow *POTPULSE* to preset the paths you specified within the *Configuration* dialog, i.e., if you wish to save or open a PGF file, *POTPULSE* will take you to your PGF-folder by default. If you, however, enable the option Last folder used in the application, *POTPULSE* will always take you to the last folder used. E.g., if you save an experiment into your data folder and then wish to open a PGF file, *POTPULSE* will take you to the last data folder instead of the PGF folder from the *Configuration*.

The Default Configuration File on a PowerPC

One can starts *POTPULSE* by double clicking on any *.set file or on an alias pointing to the *.set file. On a Quadra computer this works as expected, and *POTPULSE* will boot up and load the configuration settings from that selected file. Yet, on a PowerPC computer, *POTPULSE* will always load the configuration file named DefaultPulse.set.

Thus, on a PowerPC one must proceed differently to be able to use multiple configurations. The easiest way is to store the *Configuration* file with the default file name in the Common Path folder. Then, you can move that file to any other folder you want! You should create a special folder for each individual *Configuration* file. Starting *POTPULSE* by double clicking on this Configuration file, even when it is not in the Common Path, will load the settings from that file.

Tip: If you remove the files *DefaultPulse.set* and *DefaultFit.set* from the *Pulse* folder, *POTPULSE* will ask you during startup whether to use the default configuration or to look for a **.set* file. In the later case you can run the program with any configuration thus allowing you to quickly change experimental conditions.

Resolving Minor Problems

Problem: POTPULSE cannot read the protection key (dongle):

- Install the latest version of EvE Init. The latest version is 1.9 and is supplied on the HEKA CD.

Problem: Computer is slow and "Time Overrun" messages appear:

- The first reason may be "memory starvation". If you have only 8 Mbyte of RAM and you installed memory-hungry system extensions, there may not be enough memory for the system itself to run. The operating system needs additional memory, e.g., for every new font and font size used, for every new created file, etc. Thus the system will perform a "memory compaction" every time the used font size changes. If your "System Software" memory allocation is 2.5 Mbytes, there will be no memory left for the memory requirement of the operating system when *POTPULSE* is running. To solve this problem, remove unnecessary system extensions, control panels, INITs, and the like, until the About this Macintosh dialog reports some free memory when *POTPULSE* is running.
- A second source are other applications running in the background, such as networking. You should turn them off during any important experiment.
- Another possibility is that you may have activated one or both of the options Calculate folder sizes and Show disk info in header in the Views control panel. This will continuously compute the exact folder sizes and waste a lot of computing power, especially when you are writing to disk. Thus, always deselect these options.

Resolving Major Problems

If you encounter more severe problems when trying to start or run *POTPULSE* there may be a problem with any of the following:

- *POTPULSE* itself (i.e., software bugs or incompatibilities)
- The computer hardware used (RAM-chips, SCSI-connections, etc.)
- The EPC9 hardware, including the Mac-23 board
- The settings of the computer used
- System extensions, utilities, and INITs installed on the user computer
- Viruses

First off, you should check the following:

- **Connections**: Check, if the Mac-23 board is properly installed and the connecting cable is plugged in. Check, if the EPC9 is powered up.

- **Control Panel settings**. These are the suggested defaults of the Control Panels:
  - Memory: File Cache Size 128 kbytes, Virtual memory OFF; 32-bit Addressing; RAM-disk OFF.
  - Cache Control: CPU cache ON
  - Monitors: 256 colors, more colors will slow down the graphics
  - Views: Calculate folder sizes OFF, Show disk info in header OFF
  - Sharing Setup: File Sharing OFF, Program Sharing OFF
  - General Controls: Documents...Folder which contains the Application.

If the problem persists, you should perform the following:

- **Rebuild the desktop file**. Rebuilding the desktop is often necessary when documents fail to launch after double-clicking, or when custom icons are replaced with generic document or application icons. Traditionally, pressing 'OPT' + 'CMD' before the Finder loads will force the invisible desktop file to "rebuild" itself.

Note: Quite often, this is not completely effective because the original desktop file was corrupted, so rebuilding it only yields an updated desktop file which is still damaged. "TechTool" (a utility program which can be found on Info-Mac or any of its numerous mirror sites) provides a better solution to rebuilding the desktop. It actually deletes the original desktop. The next time the Finder loads, it will create a brand new desktop file.

- **System Extension, Control Panel, and INIT conflicts**. To test this, restart your computer while keeping 'SHIFT' + 'SPACE' pressed. This will disable any extension from loading.
- **Network driver or printer driver conflicts**. Inactivate networks and printers in the “Chooser”, then restart your computer.

And finally:

- **Zap the PRAM**. The parameter RAM (PRAM) contains user-definable settings that must be retained after the computer has been deactivated. Settings such as time of day, mouse scaling, keyboard repeat rate, and startup drive preferences are all stored in the upper 64 bytes of PRAM. Traditionally, one could clear or "zap" these upper 64 bytes of PRAM by holding a special key combination ('OPT' + 'CMD' + 'P' + 'R') at startup. This often cures behavioral anomalies which cannot be remedied with software replacement.

Note: Below the standard 64 bytes of PRAM lies another 192 bytes of memory which are, for the most part, publicly undocumented. These are secret storage areas that Apple uses for such things as Manufacture Date and Factory Service settings. When these portions of the PRAM become corrupted with invalid data, odd problems can occur and sometimes the machine will fail to work at all. Traditional PRAM zapping does not clear the lower 192 bytes of PRAM. The only alternative is to remove the PRAM battery, which is often soldered to the logic board. TechTool clears all 256 bytes of PRAM memory without the need to remove the battery. Once the system is rebooted, the MacOS ROMs will replace the PRAM contents with its default or factory settings. Some settings will revert to their factory defaults (color-capable Macs will revert to 1-bit B&W, the printer port will revert to AppleTalk Active, mouse speed will revert to super-slow). Adjusting Control Panel and Chooser settings will correct the above problems.

- **Virus disinfection**. Scan all your system and disks with a virus-checker (e.g., “Disinfectant"; which can be found on Info-Mac or any of its numerous mirror sites).

If these procedures do not cure the problem, there might be hardware incompatibilities:

- Hardware conflicts. Remove all installed boards, including the Mac23 board. Does POTPULSE now start? Re-install the boards one after the other, testing the performance of POTPULSE each time.
- Bad RAM chips. RAM chips are never checked during bootup. It can occur that a bad RAM chip is failing only when it is hot, or only sporadically. Thus, it can happen that, e.g., E9Screen is running (it uses only 2 Mbyte of space) but POTPULSE does not (it needs at least 5.5 Mbyte).

If none of this helps, call our customer hotline.

Windows Specific Problems

Crash with "Page Fault" Error under Windows 95

This error message is issued by the memory manager of Windows 95. The error **page fault** means that it run out off virtual memory. Probably the swap file on disk is too small, or too many programs are simultaneously executed. If the second condition does not hold, check the size of the Windows swap-file, and increase it, if required. More physical RAM may also be very beneficial, 16 MB is too limited for Windows 95 anyway. 32 MB are more advisable. But RAM itself cannot be the problem: *POTPULSE* did run without problems under Windows 3.1 on a 33 MHz 86486, with as little as 4 MB (four!) of RAM. It was slow, though, acquiring a sweep once in three seconds.

Appendix I: Data Structure

In this chapter we describe the general structure of the files generated by *POTPULSE*.

Data Files

PULSE generates up to 4 files when you create a data file:

1. The *Data* file (file extension *.dat).
2. The *Stimulus Templates* file (file extension *.pgf).
3. The *Acquisition Parameters* file (file extension *.pul).
4. The *Notebook* file (file extension *.txt).
5. The *Solution Data Base* file (file extension *.sol).

Except for the raw data file and the Notebook file, all other files have a *Tree* structure. The entire trees are kept in memory, whereas the raw data traces are always loaded from disk, when needed.

Stimulation Template: Stim

Stores the stimulation protocol. The structure of the *Stimulation File* (extension "pgf") is defined by the Definition Module"Stim.de" (see Chapter *Appendix I: Data Structure*). The Stimulation File has a tree structure:

Record	Description
Root	Version number
Stimulation	Description of an ensemble of pulse patterns; e.g., I-V curve
Segment	Individual segment of a pulse pattern

Stimulation files can be loaded into the *Pulse Generator*. In fact, the *Pulse Generator* files for the stimulation protocols used during the experiments have the same data structure as the PGF-files, which belong to the recorded data. In this way it is possible to exactly repeat an experiment by using a copy of a PGF-file as *Pulse Generator* file.

Data Description: Pulsed

Stores parameters, such as current range, IR-Compensation, etc. The pointer to the data stored in the raw data file is also contained in this file. The structure of the *Pulsed File* (extension "*.pul") is defined by the Definition Module "Pulsed.de" (see Chapter *Appendix I: Data Structure*). The *Pulsed File* has a tree structure:

Record	Description
Root	Version number, text, time, file name
Group	Larger section of an experiment; e.g., cell
Series	Description of an ensemble of traces; e.g., I-V curve
Sweep	Description of an individual data trace

A graphical template of the *Pulsed File* (*Tree*) is shown in the *Replay* window. It contains information necessary for reconstructing the experimental conditions as the data were recorded.

Solution Database: Solution

The fourth file (optional) that *POTPULSE* writes contains a data base for the solutions used during the experiment. For each series, a solution can be specified by a Longint value (see variable in *Series: Solution*). A more detailed description of this solution is stored in the *Solution File* (extension "*.sol"). This file has a tree structure as defined by the Definition Module "Solution.de" (see Chapter *Appendix I: Data Structure*):

Record	Description
Root	Version number
Sol	Description of a solution
Chemical	Description of each ingredient

One entry in the *Sol record* is *NumericalValue*. It specifies a parameter of the solution that is not easily determined from the ingredients, like the free calcium concentration of the solution, for example. It can be used later for analysis purposes, such as to plot current as function of the calcium concentration, for example. The assignment of numbers to solutions must be unambiguous within one data file. It is recommended, however, that unambiguous assignments are used within one laboratory to make life easier. To support this, *POTPULSE* makes use of a default common solution data base which is called "E_chem.sol". This data base is a file of the same format as the other "*.sol" files. It is thought to be a generally accessible data base of all solutions used in the laboratory (or at least within one project). During the experiment the user can select solutions directly from this base to store them in the current data file; this saves a lot of typing and reduces the number of errors. More solution data bases can be used; the requested name can be specified and saved in the *Configuration* window (see below).

Analyzed Data: Analysis

Stores the results of data analysis. The structure of the *Analysis File* (extension "ana") is defined by the Definition Module "Analysis.de" (see Chapter *Appendix I: Data Structure*). The *Analysis File* has a tree structure:

Record	Description
Root	Version number, description of the tree size
Group	Larger section of an experiment; e.g., cell
Series	Analysis of an ensemble of traces; e.g., I-V curve
Sweep	Parameter of an individual data trace; e.g., peak current

Raw Data

This raw data file is a continuous data stream. Each data point is a 16-bit signed integer (exceptions are explicitly mentioned below). When a sweep is stored, *POTPULSE* stores the various single traces (if available).

Inside the program, the data are processed as *Real* numbers. This helps to avoid round-off errors during data averaging procedures, for example.

Stim.de

```
DEFINITION MODULE Stim;

(*
 * Stim data file format
 *
 * This module defines the data types to be used for the stimulation in
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'StimTree'.
 *)

FROM SYSTEMp1 IMPORT INT16, INT32, SET16;

CONST
   VersionNumber  = 7;

(* Structure of the trees *)

CONST
   TreeLevels     = 3;   (* 3 Levels: SHeader, Stimulation, StimSegment *)

VAR
   RamSizes       : ARRAY [0..TreeLevels-1] OF INT32;

(*
 * RecordTypes for the StimTree:  Root, Stimulation, Segment
 *
 * Stimulation
 * StimulationRecord
 *
 * A StimulationRecord describes a stimulation pulse made up of one or
 * more segments. Each segment has a voltage and a duration. The segment
 * can be a constant output voltage (SegmentConstant) or a ramp from the
 * previous segment voltage to the present segment voltage (SegmentRamp).
 * For conditioning pulses a segment "SegmentConditioning" is provided.
 * No P/n is performed for such segments. For long conditioning pulses
 * a possibly less accurate timing is applied.
 * Incrementing voltages or times in a ramped record is allowed, but be
 * sure to think about what will happen; incrementing is done before the
 * ramp parameters are evaluated.
 *
 * When a series of pulses are output, the Delta entries in each
 * StimulationSegment describe the parameter change to perform for each
 * pulse in the series. We have:
 *
 *       Voltage(n)  := Voltage(0) * DeltaVFactor^n + DeltaVIncrement * n;
 *       Duration(n) := Duration(0) * DeltaTFactor^n + DeltaTIncrement * n;
 *
 * The Leak sweeps usually follow the test sweeps with a delay LeakDelay.
 * If LeakDelay is negative then leak sweeps precede test sweeps and Leak
 * Delay is the time between the end of the last leak sweep and the
 * beginning of the test sweep
 *)

TYPE SegmentClass = ( SegmentConstant,
                      SegmentRamp,
                      SegmentConditioning,
                      SegmentContinuous,
                      SegmentConstSine,
                      SegmentRampSine );

TYPE StimSegmentRecord = RECORD
   Class                 : SegmentClass; (* constant, ramp, or condit.   *)
   IsHolding             : BOOLEAN;      (* TRUE if Voltage is to be set *)
   Voltage               : LONGREAL;     (* to holding                   *)
   Duration              : LONGREAL;
   DeltaVFactor          : LONGREAL;
   DeltaVIncrement       : LONGREAL;
   DeltaTFactor          : LONGREAL;
   DeltaTIncrement       : LONGREAL;
END (* RECORD *);

TYPE StimSegment = POINTER TO StimSegmentRecord;

TYPE Trigger = RECORD
   TriggerSegment        : INT16;
   TriggerTime           : LONGREAL;   (* Seconds, within trig.Segment *)
   TriggerLength         : LONGREAL;   (* Length of Trig.Pulse, sec    *)
   TriggerAmplitude      : LONGREAL;   (* Amplitude of Trig.Pulse      *)
   TriggerDac            : INT16;
END; (* RECORD *)

TYPE IncrementModeType  = ( ModeInc,
                            ModeDec,
                            ModeIncInterleaved,
                            ModeDecInterleaved,
                            ModeAlternate );

TYPE GUpdateType        = ( NoGUpdate,
                            SwGSlow,
                            SwGFast,
                            SwGBoth,
                            SeGSlow,
                            SeGFast,
                            SeGBoth );

TYPE WriteModeType      = ( WriteEnabled,
                            WriteDisabled,
                            NoWriteNoShow,
                            WriteButNoShow );

TYPE ExtTriggerType     = ( TrigNone, TrigSeries, TrigSweep );

TYPE PostVmembType      = ( VmembSweepIncr,
                            VmembValue,
                            VmembSeriesIncr );
TYPE LockInType         = ( Loff, Lnormal, Lpwlinear );

TYPE AmplModeType       = ( AllAmplModes, VCAmplMode, CCAmplMode );

TYPE StimulationRecord  = RECORD
   FileName           : ARRAY [0..13] OF CHAR;  (* Source File           *)
   EntryName          : ARRAY [0..13] OF CHAR;  (* Identifier            *)
   SampleInterval     : LONGREAL;       (* Seconds                       *)
   FilterFactor       : LONGREAL;       (* oversampling factor           *)
   SweepInterval      : LONGREAL;       (* Repetition-Interval           *)
   NumberSweeps       : INT32;          (* Number of sweeps in Series    *)
   NumberRepeats      : INT32;          (* Number of Seq. repeats        *)
   RepeatWait         : LONGREAL;       (* Wait between repeats          *)
   LinkedSequence     : ARRAY [0..13] OF CHAR;
   LinkedWait         : LONGREAL;       (* Wait between linked sequences  *)

   LeakCount          : INT32;          (* number of leak sweeps         *)
   LeakSize           : LONGREAL;       (* rel. amplitude of leak pulse  *)
                                        (* e.g. 0.25 for P/4 protocol    *)
   LeakHolding        : LONGREAL;
   LeakAlternate      : BOOLEAN;        (* norm. or alt. leak protoc.    *)
   AltLeakAveraging   : BOOLEAN;        (* norm. or alt. leak while aver. *)
   LeakDelay          : LONGREAL;       (* Seconds                       *)
   Trig               : ARRAY [0..2] OF Trigger;
   NumberOfTriggers   : INT16;          (* Number actually used; 0<x<4   *)
   RelevantXSegment   : INT16;          (* usually that which changes    *)
   RelevantYSegment   : INT16;          (* usually that which changes    *)

   WriteMode          : WriteModeType;
   IncrementMode      : IncrementModeType;

   TotalSweepLength   : INT32;        (* Total sweeplength including a   *)
                                      (* possible continuous segment,    *)
                                      (* in units of sample intervals.   *)

   MaxSweepLength     : INT32;        (* max length of sweep to be shown. *)
                                      (* For a pulse without a continuous *)
                                      (* segment it is identical to the  *)
                                      (* max sweeplength from trig #1,   *)
                                      (* in units of sample intervals    *)

   InputChannels      : INT16;        (* # input channels. Default=1     *)
   GUpdate            : GUpdateType;  (* make C-slow/fast bef. each pulse *)
                                      (* only for EPC9                   *)
   RelAbsPot          : BOOLEAN;      (* absolute or relativ potentials  *)

   HasContinuous      : BOOLEAN;
   LogIncrement       : BOOLEAN;

   StimDac            : INT16;
   Adc1               : INT16;
   Adc2               : INT16;
   YUnit1             : ARRAY[0..1] OF CHAR;
   YUnit2             : ARRAY[0..1] OF CHAR;

   VmembIncrement     : REAL;

   ExtTrigger         : ExtTriggerType;
   FileTemplate       : BOOLEAN;

   StimKind           : SET16;    (* meaning of bits:
                                          0 => extended fields defined
                                          1 => LockIn.Active
                                          2 => Fura.Active
                                         15 => set: use scan-rate!
                                     *)

   LockInCycle           : LONGREAL;
   LockInAmplitude       : LONGREAL;

   FuraOn                : BOOLEAN;
   VmembMode             : PostVmembType;
   FuraTotLength         : LONGREAL;
   FuraDelay             : LONGREAL;
   FuraLength1           : LONGREAL;
   FuraLength2           : LONGREAL;
   FuraWaveLength0       : LONGREAL;
   FuraWaveLength1       : LONGREAL;
   FuraWaveLength2       : LONGREAL;
   FuraRepeats           : INT16;

   LockInSkip            : INT32;
   LockInVReversal       : LONGREAL;
   LockInMode            : LockInType;
   LockInShow            : BOOLEAN;

   ConfigMacro           : ARRAY[0..15] OF CHAR;
   EndMacro              : ARRAY[0..15] OF CHAR;

   AmplModeKind          : AmplModeType;
   NoStartWait           : BOOLEAN;

   ActualInChannels      : INT16;
   ActualOutChannels     : INT16;
   ActualAdc1            : INT16;
   ActualAdc2            : INT16;
 END (* RECORD *);

TYPE Stimulation         = POINTER TO StimulationRecord;


TYPE RootRecord          = RECORD
   Version               : INT16;
END (* RECORD *);

TYPE Root                = POINTER TO RootRecord;

END Stim.
```

Pulsed.de

```
DEFINITION MODULE Pulsed;

(*
 * Pulsed data file format
 *
 * This module defines the data types to be used for control information on
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'PulsedTree'.
 *)

FROM SYSTEM IMPORT ADDRESS, BYTE; FROM SYSTEMp1 IMPORT INT16, INT32, SET16;

CONST
   VersionNumber     = 7;

(* Structure of the trees
 *)

CONST
   TreeLevels        = 4;  (* 4 Levels: PHeader, Group, Series, Sweep *)

VAR
   RamSizes          : ARRAY [0..TreeLevels-1] OF INT32;

(*    RecordTypes    : Root, Group, Series, Sweep
 *
 * Sweep
 * SweepRecord
 *
 * A SweepRecord describes the sampled data from a single stimulation/
 * acquisition.
 *)

CONST
   ExtendedBit      = 0;
   FuraBit          = 1;
   LockInBit        = 2;
   DataFormatBit    = 3;
   LittleEndianBit  = 4;
   ScanRateBit      = 15;

TYPE
   StringType       = ARRAY [0..13] OF CHAR;
   CommentType      = ARRAY [0..79] OF CHAR;
   RootTextType     = ARRAY [0..4],[0..79] OF CHAR;

   DataAbscissaType = ( Time,
                        Amplitude, LnAmplitude, LogAmplitude,
                        Frequency, LnFrequency, LogFrequency );

   DataFormatType   = ( int16, int32, real32, real64 );

TYPE SweepRecord    = RECORD
   Time             : LONGREAL;       (* UNIX (seconds)+ Rest            *)
   StimCount        : INT32;          (* relevant entry in StimTree      *)
   SweepCount       : INT32;          (* .. at time of acquisition       *)
   AverageCount     : INT32;          (* number of on-line averages      *)
   Leak             : BOOLEAN;        (* TRUE, if Record has a leaksweep *)
   SecondTrace      : BOOLEAN;        (* TRUE, if Record has a 2. trace  *)
   Label            : StringType;
   DataPoints       : INT32;          (* Number of data points in RAM    *)
   Data             : INT32;          (* Offset in raw data file        *)
   DataPointer      : ADDRESS;        (* Raw data ADDRESS in memory     *)
       (* EPC 7/9 settings *)
   DataFactor1      : LONGREAL;       (* Amperes/ADC-unit               *)
   DataFactor2      : LONGREAL;       (* Volts/ADC-unit                 *)
   CSlow            : LONGREAL;       (* Capacitance, Farad             *)
   GSeries          : LONGREAL;       (* Series conductance, Siemens    *)
   RsValue          : LONGREAL;       (* Series resistance setting, Ohms *)
   Mconductance     : LONGREAL;
       (* Results of pre-analysis  *)
   ZeroCurrent      : LONGREAL;       (* Amperes                        *)
   OnlineYResult    : LONGREAL;       (* Param. determined by last anal. *)
   OnlineXResult    : LONGREAL;       (* Param. determined by last anal. *)

   TotalPoints      : INT32;          (* Total data points in file      *)
   Offset           : INT32;          (* Offset of loaded data          *)

   SweepKind        : SET16;   (* meaning of bits:
                                  ExtendedBit   => extended fields defined
                                  FuraBit       => Fura.Active
                                  LockInBit     => LockIn.Active
                                  DataFormatBit => "DataFormat" field valid
                                  LittleEndianBit => byte sequence
                                                   "Mac" = cleared
                                                   "DOS" = set
                                  ScanRateBit   => set: use scan-rate!
                                *)

   FuraPoints       : INT32;          (* Number of data points in RAM   *)
   FuraData         : INT32;          (* Offset in raw data file        *)
   FuraPointer      : ADDRESS;        (* Raw data ADDRESS in memory     *)
   OnlineYResult2   : LONGREAL;       (* Param. determined by last anal. *)
   OnlineXResult2   : LONGREAL;       (* Param. determined by last anal. *)

   DispFactor1      : LONGREAL;       (* reserved by PULSE              *)
   DispFactor2      : LONGREAL;       (* reserved by PULSE              *)

     (*
       Since we introduced the new "DataFormat" field in a field
       which before was without importance, we are interpreting it
       ONLY if "SweepKind" has ExtendedBit and DataFormatBit set!
       Thus, the procedure "StimIO.InitializeSweep" scans
       the "*.pul" tree upon loading and enforces that check.
      *)
   DataFormat       : DataFormatType;   (* Kind of abscissa, INT or REAL *)
   DataAbscissa     : DataAbscissaType;
   Timer            : LONGREAL;

   FuraStart        : INT32;
   VideoTime        : LONGREAL;
   Spares           : ARRAY[0..3] OF CHAR;
END (* RECORD *);

TYPE Sweep = POINTER TO SweepRecord;

(*
 * Series
 * SeriesRecord
 *
 * A SeriesRecord describes a Series of Sweeps; this is characterized
 * by the StimulationRecord, that generates it; also the record in-
 * cludes a number of environmental parameters that are obtained
 * before the series is started.
 *)

TYPE
  RecordingModeType     = ( InOut,
                            OnCell,
                            OutOut,
                            WholeCell,
                            PCClamp,
                            VClamp,
                            NoRec,
                            TestInt,
                            TestExt );

  UserParamType         = RECORD
    Value               : LONGREAL;
    Name                : StringType;
    Unit                : ARRAY [0..1] OF CHAR;
  END;

  EPC9StateType         = ARRAY[0..103] OF BYTE;   (* E9Panel.StateType *)

  SeriesRecord          = RECORD
    Time                : LONGREAL;       (* seconds                 *)
        (* Patch parameters, usually obtained before series *)
    Bandwidth           : LONGREAL;       (* Hertz                   *)
    PipettePotential    : LONGREAL;       (* Volts                   *)
    CellPotential       : LONGREAL;       (* Volts                   *)
    PipetteResistance   : LONGREAL;       (* Ohms                    *)
    SealResistance      : LONGREAL;       (* Ohms                    *)
    BackgroundNoise     : LONGREAL;       (* Amperes, RMS            *)
    Temperature         : LONGREAL;       (* Degrees C               *)
    PipettePressure     : LONGREAL;       (* in cm H20               *)
    UserParam1          : UserParamType;
    UserParam2          : UserParamType;
    RecordingMode       : RecordingModeType;
    VideoMode           : CHAR;
    Comment             : CommentType;
      (* EPC9 state record *)
    EPC9State           : EPC9StateType;
    InternalSolution,
    ExternalSolution    : INT32;        (* solutions according to solution
                                           code as stored in '*.sol'  *)
    ExtraYUnit1         : ARRAY [0..1] OF CHAR;
    ExtraYUnit2         : ARRAY [0..1] OF CHAR;

    DispYUnit1          : ARRAY [0..3] OF CHAR; (* reserved by PULSE    *)
    DispYUnit2          : ARRAY [0..3] OF CHAR; (* reserved by PULSE    *)

    FuraK               : LONGREAL;
    FuraMin             : LONGREAL;
    FuraMax             : LONGREAL;

    LockInExtPhase      : LONGREAL;

    Timer               : LONGREAL;

      (* Reserved for extensions *)

    ExtCalPhase         : LONGREAL;
    ExtCalAttenuation   : LONGREAL;
    PLPhase             : LONGREAL;
    PLPhaseY1           : LONGREAL;
    PLPhaseY2           : LONGREAL;
    ExtCalValid         : BOOLEAN;
    PLPhaseValid        : BOOLEAN;

      (* EPC9 state record *)
    EPC9State           : EPC9StateType;

  END (* RECORD *);

TYPE Series = POINTER TO SeriesRecord;

(*
 * Group
 * GroupRecord
 *
 * A GroupRecord describes a group of series, such as patch and
 * whole cell currents obtained simultanuously, or groups of series
 * obtained in sequence under different sets of conditions
 *)

TYPE GroupRecord        = RECORD
   Label                : StringType;
   Text                 : CommentType;
   ExperimentNumber     : INT32;
   ExtraLongReal        : LONGREAL;
END (* RECORD *);

TYPE Group = POINTER TO GroupRecord;

(*
 * Root
 * RootRecord
 *
 * A RootRecord describes a complete experiment.
 *)

TYPE RootRecord         = RECORD
   Version              : INT16;
   VersionName          : StringType;
   FileName             : StringType; (* the part common to all *)
   Comments             : RootTextType;
   StartTime            : LONGREAL;       (* UNIX Time  +Rest     *)
END (* RECORD *);

TYPE Root = POINTER TO RootRecord;

END Pulsed.
```

Solution.de

```
DEFINITION MODULE Solution;

(*
 * Purpose: Handling of Pulsed solution files and database.
 *)
```

```
FROM SYSTEM IMPORT ADDRESS; FROM SYSTEMp1 IMPORT INT16, INT32;

CONST
   VersionNumber  = 1;

   TreeLevels     = 3; (* Root, Sol, Chemical *)

VAR
   RamSizes       : ARRAY [0..TreeLevels-1] OF INT32;

(*
   RecordTypes: Root, Sol, Chemical

   Chemical - Description of one component of the solution
*)

CONST
   MaxChemicalNameLength  = 29;

TYPE
   ChemicalNameType = ARRAY [0..MaxChemicalNameLength] OF CHAR;

TYPE ChemicalRecord = RECORD
   Concentration : REAL;              (* molar concentration *)
   Name          : ChemicalNameType;  (* ASCII identifier, e.g. "NaCl" *)
END (* RECORD *);

TYPE Chemical = POINTER TO ChemicalRecord;

(*
   Sol - Description of a solution
*)

CONST
   MaxSolutionNameLength  = 79;

TYPE
   SolutionNameType = ARRAY [0..MaxSolutionNameLength] OF CHAR;

TYPE SolutionRecord = RECORD
    Number      : INT32;              (* number in common data base      *)
    Name        : SolutionNameType;   (* ASCII identifier                *)
    Numeric     : REAL;               (* numerical value, like Ca Conc   *)
    NumericName : ChemicalNameType;   (* description of numeric,         *)
                                      (* e.g. "Free Ca"                  *)
    pH          : REAL;               (* optimum pH                      *)
    pHCompound  : ChemicalNameType;   (* adjusted the pH with            *)
    Osmol       : REAL;               (* measured osmolarity, mosm       *)
END (* RECORD *);

TYPE Sol = POINTER TO SolutionRecord;

(*
   Root - File description
*)

TYPE RootRecord = RECORD
   Version       : INT16;
   DataBaseName  : ARRAY [0..79] OF CHAR; (* name of Solution Data Base
that is
                                              used to define Sol.Number *)
```

```
END (* RECORD *);

TYPE Root = POINTER TO RootRecord;

END Solution.
```

Analysis.de

```
DEFINITION MODULE Analysis;

(*
   Purpose:    Structure of analyzed data.

   Pulse Analysis Format:

   --------      ---------     ---------     -----------
   |Header| -->  |Group  | --> |Series | --> | Sweep 1 |
   |      | .    |       | .   |       | .   |         |
   -------- .    |Group- | .   |Series-| .   | SweepRec|
            .    | Descr | .   | Descr | .   -----------
                 |       | .   |       | .
                 |Group- | .   |Series-| --> -----------
                 | Record| .   | Record| .   | Sweep 2 |
                 --------- .   |       | .   |         |
                           .   |Sweep- | .   | SweepRec|
                           .   | Descr | .   -----------
                           .   --------- .      ...
                           .             --> -----------
                           .                 | Sweep n |
                           .                 |         |
                           .                 | SweepRec|
                           .                 -----------
                           .
                           .   ---------     -----------
                           --> |Series | --> | Sweep 1 |
                           .   |       | .   |         |
                           .   |Series-| .   | SweepRec|
                           .   | Descr | .   -----------
                               |       | .
                               |Series-| --> -----------
                               | Record| .   | Sweep 2 |
                               |       | .   |         |
                               |Sweep- | .   | SweepRec|
                               | Descr | .   -----------
                               --------- .      ...
                                         --> -----------
                                             | Sweep m |
                                             |         |
                                             | SweepRec|
                                             -----------
*)

FROM SYSTEMp1 IMPORT INT16, INT32;

CONST
   VersionNumber        = 5;

   TreeLevels           = 4;  (* Root , Group , Series , Sweep *)

VAR
   RamSizes             : ARRAY [0..TreeLevels-1] OF INT32;

 CONST
   LABELLENGTH          = 12;
   MAXENTRIES           = 15;

   SWEEPLABELLENGTH     = 12;
   MAXSWEEPENTRIES      = 15;

TYPE
   GeneralInfo = RECORD
      Time              : LONGREAL;     (* time of analysis *)
      AnalysisType      : INT16 ;       (* type of analysis *)
   END;

   TYPE
      StatusType = ( StatusFitted, StatusFixed, StatusSkipped );

   EntryRecord          = RECORD
      Value             : LONGREAL;
      Status            : StatusType;
      Filler            : BOOLEAN;
   END;

   DescriptionType = ARRAY [0..MAXENTRIES-1],
                           [0..LABELLENGTH-1] OF CHAR;
   DescrPointer    = POINTER TO DescriptionType;

   AnalysisType    = ARRAY [0..MAXENTRIES-1] OF EntryRecord;
   AnalysisPointer = POINTER TO AnalysisType ;

(*
   Root
*)

TYPE

   RootRecord           = RECORD
      Version           : INT16;

      SeriesLabelLength : INT16;
      MaxSeriesEntries  : INT16;

      SweepLabelLength  : INT16;
      MaxSweepEntries   : INT16;

      GroupLabelLength  : INT16;
      MaxGroupEntries   : INT16;
   END;

   Root                 = POINTER TO RootRecord;

(*
  Group
*)

   GroupRecord          = RECORD
      General           : GeneralInfo ;
      GroupDescription  : DescriptionType ;
      GroupAnalysis     : AnalysisType;
   END;

   Group                = POINTER TO GroupRecord;

(*
   Series
*)

   SweepDescriptionType  = ARRAY [0..MAXSWEEPENTRIES-1],
                                 [0..SWEEPLABELLENGTH-1] OF CHAR;
   SweepDescrPointer     = POINTER TO SweepDescriptionType;

   SeriesRecord         = RECORD
      General           : GeneralInfo;
      SeriesDescription : DescriptionType;
      SeriesAnalysis    : AnalysisType;
      SweepDescription  : SweepDescriptionType;
   END;

   Series               = POINTER TO SeriesRecord;

(*
   Sweep
*)

TYPE
   AnalysisRangeType = RECORD
     YSegmentOffset : INT16 ;
     LeftCursor     : LONGREAL ;
     RightCursor    : LONGREAL ;
   END ;

TYPE

   GeneralSweepInfo     = RECORD
      SweepCount        : INT32;       (* Number in parent Pulsed Series *)
      AnalysisType      : INT16 ;
      AnalysisRange1    : AnalysisRangeType ;
      AnalysisRange2    : AnalysisRangeType ;
   END;

   SweepAnalysisType        = ARRAY [0..MAXSWEEPENTRIES-1] OF EntryRecord;
   SweepAnalysisTypePointer = POINTER TO SweepAnalysisType;

   SweepRecord          = RECORD
      General           : GeneralSweepInfo;
      SweepAnalysis     : SweepAnalysisType;
   END;

   Sweep                = POINTER TO SweepRecord;

END Analysis.
```

Appendix II: Record Offset Bytes

These are the record offset bytes of the various data files created by *POTPULSE*:

StimDef.de

```
DEFINITION MODULE StimDef;

(*
 * Stim data file format, see module Stim.de in Appendix I.
 *
 * This module defines the data types to be used for the stimulation
 * in experiments involving pulsed data; it uses the tree structures
 * as defined in module Tree and provides types for a 'StimTree'.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST
   Unit2Size             =  2;
   StringSize            = 14;
   MacroNameSize         = 16;

   (* StimSegmentRecord = RECORD *)
   SeClass               =   0;
   SeIsHolding           =   1;
   SeVoltage             =   2;
   SeDuration            =  10;
   SeDeltaVFactor        =  18;
   SeDeltaVIncrement     =  26;
   SeDeltaTFactor        =  34;
   SeDeltaTIncrement     =  42;
   StimSegmentSize       =  50;
   (* END StimSegmentRecord *)

   (* StimulationRecord = RECORD *)
   StFileName            =   0;
   StEntryName           =  14;
   StSampleInterval      =  28;
   StFilterFactor        =  36;
   StSweepInterval       =  44;
   StNumberSweeps        =  52;
   StNumberRepeats       =  56;
   StRepeatWait         =  60;
   StLinkedSequence     =  68;
   StLinkedWait         =  82;
   StLeakCount          =  90;
   StLeakSize           =  94;
   StLeakHolding        = 102;
   StLeakAlternate      = 110;
   StAltLeakAveraging   = 111;
   StLeakDelay          = 112;
   StTriggerSegment1    = 120;
   StTriggerTime1       = 122;
   StTriggerLength1     = 130;
   StTriggerAmplitude1  = 138;
   StTriggerDac1        = 146;
   StTriggerSegment2    = 148;
   StTriggerTime2       = 150;
   StTriggerLength2     = 158;
   StTriggerAmplitude2  = 166;
   StTriggerDac2        = 174;
   StTriggerSegment3    = 176;
   StTriggerTime3       = 178;
   StTriggerLength3     = 186;
   StTriggerAmplitude3  = 194;
   StTriggerDac3        = 202;
   StNumberOfTriggers   = 204;
   StRelevantXSegment   = 206;
   StRelevantYSegment   = 208;
   StWriteMode          = 210;
   StIncrementMode      = 211;
   StTotalSweepLength   = 212;
   StMaxSweepLength     = 216;
   StInputChannels      = 220;
   StGUpdate            = 222;
   StRelAbsPot          = 223;
   StHasContinuous      = 224;
   StLogIncrement       = 225;
   StStimDac            = 226;
   StAdc1               = 228;
   StAdc2               = 230;
   StYUnit1             = 232;
   StYUnit2             = 234;
   StVmembIncrement     = 236;
   StExtTrigger         = 240;
   StFileTemplate       = 241;
   StStimKind           = 242;
   StLockInCycle        = 244;
   StLockInAmplitude    = 252;
   StFuraOn             = 260;
   StVmembMode          = 261;
   StFuraTotLength      = 262;
   StFuraDelay          = 270;
   StFuraLength1        = 278;
   StFuraLength2        = 286;
   StFuraWaveLength0    = 294;
   StFuraWaveLength1    = 302;
   StFuraWaveLength2    = 310;
   StFuraRepeats        = 318;
   StLockInSkip         = 320;
   StLockInVReversal    = 324;
   StLockInMode         = 332;
   StLockInShow         = 333;
   StConfigMacro        = 334;
   StEndMacro           = 350;
   StAmplModeKind       = 366;
   StNoStartWait        = 367;
   StActualInChannels   = 368;
   StActualOutChannels  = 370;
   StActualAdc1         = 372;
   StActualAdc2         = 374;
   StimulationSize      = 376;
   (* END StimulationRecord *)

   (* RootRecord        = RECORD *)
   RoVersion            =   0;
   RootSize             =   2;
   (* END RootRecord *)

END StimDef.
```

PulsedDef.de

```
DEFINITION MODULE PulsedDef;

(*
 * Pulsed data file format, see module Pulsed.de in Appendix I.
 *
 * This module defines the data types to be used for control information on
 * experiments involving pulsed data; it uses the tree structures as
 * defined in module Tree and provides types for a 'PulsedTree'.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE          1
 *    CHAR          1
 *    enumeration   1
 *    SET16         2
 *    INT16         2
 *    INT32         4
 *    ADDRESS       4
 *    REAL          4
 *    LONGREAL      8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST

   Unit2Size            =   2;
   Unit4Size            =   4;
   SwSpareSize          =  10;
   StringSize           =  14;
   CommentSize          =  80;
   EPC9StateSize        = 104;
   RootTextSize         = 400;
   SeExtraLongReals     =   4;

   (* SweepRecord       = RECORD *)
   SwTime               =   0;
   SwStimCount          =   8;
   SwSweepCount         =  12;
```

```
SwAverageCount       =  16;
SwLeak               =  20;
SwSecondTrace        =  21;
SwLabel              =  22;
SwDataPoints         =  36;
SwData               =  40;
SwDataPointer        =  44;
SwDataFactor1        =  48;
SwDataFactor2        =  56;
SwCSlow              =  64;
SwGSeries            =  72;
SwRsValue            =  80;
SwMConductance       =  88;
SwZeroCurrent        =  96;
SwOnlineYResult      = 104;
SwOnlineXResult      = 112;
SwTotalPoints        = 120;
SwOffset             = 124;
SwSweepKind          = 128;
SwFuraPoints         = 130;
SwFuraData           = 134;
SwFuraPointer        = 138;
SwOnlineYResult2     = 142;
SwOnlineXResult2     = 150;
SwDispFactor1        = 158;
SwDispFactor2        = 166;
SwDataFormat         = 174;
SwDataAbscissa       = 175;
SwTimer              = 176;
SwFuraStart          = 184;
SwVideoTime          = 188;
SwSpares             = 196;
SweepSize            = 200;
(* END SweepRecord *)

(* SeriesRecord      = RECORD *)
SeTime               =   0;
SeBandwidth          =   8;
SePipettePotential   =  16;
SeCellPotential      =  24;
SePipetteResistance  =  32;
SeSealResistance     =  40;
SeBackgroundNoise    =  48;
SeTemperature        =  56;
SePipettePressure    =  64;
SeUserParam1Value    =  72;
SeUserParam1Name     =  80;
SeUserParam1Unit     =  94;
SeUserParam2Value    =  96;
SeUserParam2Name     = 104;
SeUserParam2Unit     = 118;
SeRecordingMode      = 120;
SeVideoMode          = 121;
SeComment            = 122;
SeEPC9State          = 202;
SeInternalSolution   = 306;
SeExternalSolution   = 310;
SeExtraYUnit1        = 314;
SeExtraYUnit2        = 316;
SeDispYUnit1         = 318;
SeDispYUnit2         = 322;
```

```
   SeFuraK               = 326;
   SeFuraMin             = 334;
   SeFuraMax             = 342;
   SeLockInExtPhase      = 350;
   SeTimer               = 358;
   SeExtraLongReal       = 366;
   SeExtCalPhase         = 398;
   SeExtCalAttenuation   = 406;
   SePLPhase             = 414;
   SePLPhaseY1           = 422;
   SePLPhaseY2           = 430;
   SeExtCalValid         = 438;
   SePLPhaseValid        = 439;
   SeriesSize            = 440;
(* END SeriesRecord *)

   (* GroupRecord        = RECORD *)
   GrLabel               =   0;
   GrText                =  14;
   GrExperimentNumber    =  94;
   GrExtraLongReal       =  98;
   GroupSize             = 106;
   (* END GroupRecord *)

   (* RootRecord         = RECORD *)
   RoVersion             =   0;
   RoVersionName         =   2;
   RoFileName            =  16;
   RoComments            =  30;
   RoStartTime           = 430;
   RootSize              = 438;
   (* END RootRecord *)

END PulsedDef.
```

SolutionDef.de

```
DEFINITION MODULE SolutionDef;

(*
 * Handling of Pulsed solution files and database, see module Solution.de
 * in Appendix I.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST
```

```
   ChemicalNameLength   =  30;
   SolutionNameLength   =  80;

   (* ChemicalRecord    = RECORD *)
   ChConcentration      =   0;
   ChName               =   4;
   ChemicalSize         =  34;
   (* END ChemicalRecord *)

   (* SolutionRecord    = RECORD *)
   SoNumber             =   0;
   SoName               =   4;
   SoNumeric            =  84;
   SoNumericName        =  88;
   SopH                 = 118;
   SopHCompound         = 122;
   SoOsmol              = 152;
   SolutionSize         = 156;
   (* END SolutionRecord *)

   (* RootRecord        = RECORD *)
   RoVersion            =   0;
   SoDataBaseName       =   2;
   RootSize             =  82;
   (* END RootRecord *)

END SolutionDef.
```

AnalysisDef.de

```
DEFINITION MODULE Analysis;

(*
 * Structure of analyzed data file, see module Analysis.de in Appendix I.
 *
 * The sizes in bytes of the used variables are:
 *    BYTE           1
 *    CHAR           1
 *    enumeration    1
 *    SET16          2
 *    INT16          2
 *    INT32          4
 *    ADDRESS        4
 *    REAL           4
 *    LONGREAL       8
 *
 * Here, we define the record fields by their offsets from the
 * beginning of their respective record blocks:
 *)

CONST

   MAXENTRIES         = 15;
   AnalysisSize       = MAXENTRIES * EntrySize;
   DescriptionSize    = 180;

   (* EntryRecord     = RECORD *)
   Value              =   0;
   Status             =   8;
   Filler             =   9;
   EntrySize          =  10;
   (* END EntryRecord *)

   (* RootRecord      = RECORD *)
   Version            =   0;
   SeriesLabelLength  =   2;
   MaxSeriesEntries   =   4;
   SweepLabelLength   =   6;
   MaxSweepEntries    =   8;
   GroupLabelLength   =  10;
   MaxGroupEntries    =  12;
   RootSize           =  14;
   (* END RootRecord *)

   (* GroupRecord     = RECORD *)
   GrGenTime          =   0;
   GrGenAnalysisType  =   8;
   GroupDescription   =  10;
   GroupAnalysis      = 190;
   GroupSize          = 340;
   (* END GroupRecord *)

   (* SeriesRecord    = RECORD *)
   SeGenTime          =   0;
   SeGenAnalysisType  =   8;
   SeriesDescription  =  10;
   SeriesAnalysis     = 190;
   SweepDescription   = 340;
   SeriesSize         = 520;
   (* END SeriesRecord *)

   (* SweepRecord     = RECORD *)
   SweepCount         =   0;
   SwAnalysisType     =   4;
   YSegmentOffset1    =   6;
   LeftCursor1        =   8;
   RightCursor1       =  16;
   YSegmentOffset2    =  24;
   LeftCursor2        =  26;
   RightCursor2       =  34;
   SweepAnalysis      =  42;
   SweepSize          = 192;
   (* END SweepRecord *)

END AnalysisDef.
```

Appendix III: ‘C’-Headers

These are header files for the ‘C’ programming language.

Warning: The variables on disk are **packed**, i.e., there are no additional padding bytes anywhere. You must keep in mind that most compilers will align the variables in the structures differently!

Stim.h

```
// Stim.h

typedef unsigned char BOOLEAN;
typedef unsigned char CHAR;
typedef float REAL;
typedef double LONGREAL;
typedef void *ADDRESS;
typedef signed char INT8;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned char SET8;
typedef unsigned short SET16;
typedef unsigned char ENUM_8;

typedef struct String2Type_Struct {CHAR a [2];} String2Type;
typedef struct String14Type_Struct {CHAR a [14];} String14Type;
typedef struct String16Type_Struct {CHAR a [16];} String16Type;

typedef ENUM_8 SegmentClass;

typedef struct StimSegment_Struct
{
    SegmentClass Class;
    BOOLEAN IsHolding;
    LONGREAL Voltage;
    LONGREAL Duration;
    LONGREAL DeltaVFactor;
    LONGREAL DeltaVIncrement;
    LONGREAL DeltaTFactor;
    LONGREAL DeltaTIncrement;
} StimSegment;

typedef ENUM_8 IncrementModeType;
typedef ENUM_8 GUpdateType;
typedef ENUM_8 WriteModeType;
typedef ENUM_8 ExtTriggerType;
typedef ENUM_8 PostVmembType;
typedef ENUM_8 LockInShowType;
typedef ENUM_8 AutoRangingType;
typedef ENUM_8 AmplModeType;
typedef ENUM_8 RelAbsPotType;

typedef struct Stimulation_Struct
{
    String14Type FileName;
    String14Type EntryName;
    LONGREAL SampleInterval;
    LONGREAL FilterFactor;
    LONGREAL SweepInterval;
    INT32 NumberSweeps;
    INT32 NumberRepeats;
    LONGREAL RepeatWait;
    String14Type LinkedSequence;
    LONGREAL LinkedWait;
    INT32 LeakCount;
    LONGREAL LeakSize;
    LONGREAL LeakHolding;
    BOOLEAN LeakAlternate;
    BOOLEAN AltLeakAveraging;
    LONGREAL LeakDelay;
    INT16 Trigger1Segment;
    LONGREAL Trigger1Time;
    LONGREAL Trigger1Length;
    LONGREAL Trigger1Amplitude;
    INT8 Trigger1Dac;
    SET8 Trigger1DigValue;
    INT16 Trigger2Segment;
    LONGREAL Trigger2Time;
    LONGREAL Trigger2Length;
    LONGREAL Trigger2Amplitude;
    INT8 Trigger2Dac;
    SET8 Trigger2DigValue;
    INT16 Trigger3Segment;
    LONGREAL Trigger3Time;
    LONGREAL Trigger3Length;
    LONGREAL Trigger3Amplitude;
    INT8 Trigger3Dac;
    SET8 Trigger3DigValue;
    INT16 NumberOfTriggers;
    INT16 RelevantXSegment;
    INT16 RelevantYSegment;
    WriteModeType WriteMode;
    IncrementModeType IncrementMode;
    INT32 TotalSweepLength;
    INT32 MaxSweepLength;
    INT16 InputChannels;
    GUpdateType GUpdate;
    RelAbsPotType RelAbsPot;
    BOOLEAN HasContinuous;
    BOOLEAN LogIncrement;
    INT16 StimDac;
    INT16 Adc1;
    INT16 Adc2;
    String2Type YUnit1;
    String2Type YUnit2;
    REAL VmembIncrement;         // warning: this is only a REAL!
    ExtTriggerType ExtTrigger;
    BOOLEAN FileTemplate;
    SET16 StimKind;
    LONGREAL LockInCycle;
    LONGREAL LockInAmplitude;
    BOOLEAN FuraOn;
    PostVmembType VmembMode;
    LONGREAL FuraTotLength;
    LONGREAL FuraDelay;
    LONGREAL FuraLength1;
    LONGREAL FuraLength2;
    LONGREAL FuraWaveLength0;
    LONGREAL FuraWaveLength1;
    LONGREAL FuraWaveLength2;
    INT16 FuraRepeats;
    INT32 LockInSkip;
    LONGREAL LockInVReversal;
    AutoRangingType AutoRanging;
    LockInShowType LockInShow;
    String16Type ConfigMacro;
    String16Type EndMacro;
    AmplModeType AmplModeKind;
    BOOLEAN NoStartWait;
    INT16 ActualInChannels;
    INT16 ActualOutChannels;
    INT16 ActualAdc1;
    INT16 ActualAdc2;
} Stimulation;

typedef struct Root_Struct
{
    INT16 Version;
} Root;
```

Pulsed.h

```
// Pulsed.h

typedef unsigned char BOOLEAN;
typedef unsigned char CHAR;
typedef double LONGREAL;
typedef void *ADDRESS;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned short SET16;
typedef unsigned char ENUM_8;

typedef struct String2Type_Struct {CHAR a [2];} String2Type;
typedef struct String4Type_Struct {CHAR a [4];} String4Type;
typedef struct String10Type_Struct {CHAR a [10];} String10Type;
typedef struct String14Type_Struct {CHAR a [14];} String14Type;
typedef struct CommentType_Struct {CHAR a [80];} CommentType;
typedef struct RootTextType_Struct {CHAR a [400];} RootTextType;
typedef ENUM_8 DataAbscissaType;
typedef ENUM_8 DataFormatType;

typedef struct Sweep_Struct
{
    LONGREAL Time;
    INT32 StimCount;
    INT32 SweepCount;
    INT32 AverageCount;
    BOOLEAN Leak;
    BOOLEAN SecondTrace;
    String14Type Label;
    INT32 DataPoints;
    INT32 Data;
    ADDRESS DataPointer;
    LONGREAL DataFactor1;
    LONGREAL DataFactor2;
    LONGREAL CSlow;
    LONGREAL GSeries;
    LONGREAL RsValue;
    LONGREAL MConductance;
    LONGREAL ZeroCurrent;
    LONGREAL OnlineYResult;
    LONGREAL OnlineXResult;
    INT32 TotalPoints;
    INT32 Offset;
    SET16 SweepKind;
    INT32 FuraPoints;
    INT32 FuraData;
    ADDRESS FuraPointer;
    LONGREAL OnlineYResult2;
    LONGREAL OnlineXResult2;
    LONGREAL DispFactor1;
    LONGREAL DispFactor2;
    DataFormatType DataFormat;
    DataAbscissaType DataAbscissa;
    LONGREAL Timer;
    INT32 FuraStart;
    LONGREAL VideoTime;
    String4Type Spares;
} Sweep;

typedef ENUM_8 RecordingModeType;
typedef struct EPC9Type_Struct {unsigned char a [104];} EPC9Type;

typedef struct Series_Struct
{
    LONGREAL Time;
    LONGREAL Bandwidth;
    LONGREAL PipettePotential;
    LONGREAL CellPotential;
    LONGREAL PipetteResistance;
    LONGREAL SealResistance;
    LONGREAL BackgroundNoise;
    LONGREAL Temperature;
    LONGREAL PipettePressure;
    LONGREAL UserParam1Value;
    String14Type UserParam1Name;
    String2Type UserParam1Unit;
    LONGREAL UserParam2Value;
    String14Type UserParam2Name;
    String2Type UserParam2Unit;
    RecordingModeType RecordingMode;
    BYTE SeVideoMode;
    CommentType Comment;
    EPC9Type EPC9State;
    INT32 InternalSolution;
    INT32 ExternalSolution;
    String2Type ExtraYUnit1;
    String2Type ExtraYUnit2;
    String4Type DispYUnit1;
    String4Type DispYUnit2;
    LONGREAL FuraK;
    LONGREAL FuraMin;
    LONGREAL FuraMax;
    LONGREAL LockInExtPhase;
    LONGREAL Timer;
    LONGREAL ExtraLongReal1;
    LONGREAL ExtraLongReal2;
    LONGREAL ExtraLongReal3;
    LONGREAL ExtraLongReal4;
    LONGREAL ExtCalPhase;
    LONGREAL ExtCalAttenuation;
    LONGREAL PLPhase;
    LONGREAL PLPhaseY1;
    LONGREAL PLPhaseY2;
    BOOLEAN ExtCalValid;
    BOOLEAN PLPhaseValid;
} Series;

typedef struct Group_Struct
{
    String14Type Label;
    CommentType Text;
    INT32 ExperimentNumber;
    LONGREAL ExtraLongReal;
} Group;

typedef struct Root_Struct
{
    INT16 Version;
    String14Type VersionName;
    String14Type FileName;
    RootTextType Comments;
    LONGREAL StartTime;
} Root;
```

Solution.h

```
// Solution.h

typedef unsigned char BOOLEAN;
typedef unsigned char CHAR;
typedef float REAL;
typedef double LONGREAL;
typedef signed short INT16;
typedef signed long INT32;
typedef unsigned char ENUM_8;

typedef struct String30Type_Struct {CHAR a [30];} String30Type;
typedef struct String80Type_Struct {CHAR a [80];} String80Type;

typedef struct Chemical_Struct
{
    REAL Concentration;
    String30Type Name;
} Chemical;

typedef struct Solution_Struct
{
    INT32 Number;
    String80Type Name;
    REAL Numeric;
    String30Type NumericName;
    REAL pH;
    String30Type pHCompound;
    REAL Osmol;
} Solution;

typedef struct Root_Struct
{
    INT16 Version;
    String30Type DataBaseName;
} Root;
```

Appendix IV: Loading Files

Here comes the actual source of a module that could be used to read a tree:

Sample Program

```
MODULE FileFormat;

FROM SYSTEM IMPORT ADR, ADDRESS, BYTE, LONG, SHORT;
FROM SYSTEMp1 IMPORT INT32, ADDADR;

IMPORT Alert, FileSelect, IOBytes, IOFiles, Strings, TermIO, Buffer;

(*
 * MagicNumber - This is a special value used as a prefix for a tree
 * stored to a file.  The value contains the four byte values of the
 * characters 'Tree' in order.
 *
 * SwappedMagicNumber - This is the MagicNumber written by a CPU
 * which used the opposite byte ordering.
 *)

CONST
   MagicNumber        = 054726565H; (* i.e. "Tree" in ASCII *)
   SwappedMagicNumber = 065657254H; (* i.e. "eerT" in ASCII *)

(*
 * SwappedInt32 - Swaps the byte of a 32 bit long variable.
 *)

PROCEDURE SwappedInt32( Value : INT32 ): INT32;
VAR
   Source,
   Target      : POINTER TO ARRAY[0..3] OF BYTE;
   Result      : INT32;
BEGIN
   Source      := ADR( Value );
   Target      := ADR( Result );
   Target^[0]  := Source^[3];
   Target^[1]  := Source^[2];
   Target^[2]  := Source^[1];
   Target^[3]  := Source^[0];
   RETURN Result;
END SwappedInt32;

(*
 * LoadOneRecord
 *
 *    Loads one data block.
 *    All "TermIO" statements are for demonstration purpose only.
 *
 *    The variables are
 *           Stream       : the file handle to the open file
 *           FileSize     : the number of bytes the data block has in
 *                          the file.
 *           MemorySize   : the byte length of the memory block where
 *                           the data is going to be stored.
 *           WhereToStore  : the address of where the data block is
 *                           going to be stored.
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadOneRecord(
             Stream       : IOFiles.FileHandleType;
             FileSize     : LONGINT;
             MemorySize   : LONGINT;
             WhereToStore : ADDRESS )
             : BOOLEAN;

VAR
   Excess    : LONGINT;
   FileBytes : LONGINT;

BEGIN

   (* Here we load the next block of data into memory.
    *
    * First, we have to compare the number of bytes we can load from
    * the file with the bytes of allocated memory for that record.
    *
    * There are 3 possibilities:
    *    1. The size of the allocated memory ("MemorySize") equals the
    *       number of bytes of the data block in the file ("FileSize").
    *       Thus, we can load the complete block.
    *    2. There are fewer bytes in the file than we expect. This can
    *       occur, e.g., when the file has been written by an earlier
    *       version which used fewer parameters than the present one.
    *       In this case, we would have to zero out those fields which
    *       are not filled with data from the file.
    *    3. There are more bytes in the file than we expect. This can
    *       happen, when the program which created that tree file was
    *       using more parameters than we presently know of. In that
    *       case, we would load only as much byte as we had reserved
    *       RAM for.
    *)

   Excess    := MemorySize - FileSize;

   IF Excess = 0D THEN
      (* The file record has as many bytes as there is space in RAM *)

      FileBytes := MemorySize;

   ELSIF Excess < 0D THEN
      (* The file record has more many bytes than there is space in RAM.
       * Load only as many bytes as there is space in RAM.
       *)

      FileBytes := MemorySize;

   ELSE (* i.e., Excess > 0D *)
      (* The file record has fewer bytes than there is space in RAM.
       * Load only as many bytes as there are in the file.
       *)

      FileBytes := FileSize;
      (* Do not forget to clear the remaining fields which are not going
       * to be filled from the file.
       *)

      Buffer.Set( ADDADR( WhereToStore, FileSize ), Excess, 0 );

   END (* IF *);

   RETURN IOBytes.Read( Stream, FileBytes, WhereToStore );

END LoadOneRecord;

(*
 * LoadOneLevel
 *
 *    Processes the loading of one data block from a "Tree", and all
 *    its "children".
 *    All "TermIO" statements are for demonstration purpose only.
 *
 *    The variables are
 *           Stream        : the file handle to the open file
 *           Sizes         : the array containing the level sizes
 *           Levels        : the number of levels in the tree
 *           NeedsByteSwap : the flag telling, whether byte-swapping is
 *                           needed
 *           Level         : the actual tree level to load
 *           IsFirst       : a flag telling, whether it is the first child
 *                           loaded. This is only required for text output!
 *           Position      : the variable containing the position in the
 *                           file.
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadOneLevel(
             VAR Stream    : IOFiles.FileHandleType;
             VAR Sizes     : ARRAY OF INT32;
             VAR Levels    : LONGINT;
             NeedsByteSwap : BOOLEAN;
             Level         : LONGINT;
             IsFirst       : BOOLEAN;
             VAR Position  : LONGINT )
             : BOOLEAN;

VAR
   Count     : INT32;
   Size      : LONGINT;
   Children  : LONGINT;
   i         : INTEGER;
   WriteInfo : BOOLEAN;

BEGIN

   WriteInfo := IsFirst OR ( Level < Levels - 1D );

   IF WriteInfo THEN
      FOR i := 1 TO SHORT( Level ) DO
         TermIO.WriteString( '  ' );
      END; (* FOR *)
      TermIO.WriteString( 'level: ' );
      TermIO.WriteLongInt( Level, 0 );
   END; (* IF *)
   (* Here would normally be the code which loads the next block of
    * data somewhere into memory. In the present example, we just skip
    * the bytes containing these data.
    *
    * In case we would load the data block from the file, we would call
    * the following procedure:

   IF NOT
      LoadOneRecord
         ( Stream, Sizes[SHORT(Level)], MemorySize, WhereToStore )
   THEN
      Alert.IOError( '7-Error' );
      RETURN FALSE;
   END;

   (* If byte-swapping is required, we would now have to swap the bytes
      of all fields in the loaded record!
    *)

   IF NeedsByteSwap THEN ( go and swap the record fields ... ) END;

    * End of code we would call.
    *)


   (* Increase the file pointer by "Sizes[Level]" bytes and set the
    * file position just beyond the next data block:
    *)

   INC( Position, Sizes[SHORT(Level)] );

   IF NOT
      IOBytes.SetPosition( Stream, IOFiles.FromStart, Position )
   THEN
      Alert.IOError( '8-Error' );
      RETURN FALSE;
   END;


   (* The next 4 bytes contain the number of children of the present
    * level.
    *)

   Size := SIZE( INT32 );

IF NOT IOBytes.Read( Stream, Size, ADR(Count) ) THEN
      Alert.IOError( '9-Error' );
      RETURN FALSE;
   END;

   (* The file pointer increased by 4 bytes: *)
   INC( Position, 4 );

   (* And we swap the bytes, if needed: *)

   IF NeedsByteSwap THEN Count := SwappedInt32( Count ); END;

   IF WriteInfo THEN
      TermIO.WriteString( ';  children: ' );
      TermIO.WriteLongInt( Count, 0 );
      TermIO.WriteString( ';  file offset: ' );
      TermIO.WriteLongInt( Position, 0 );
      TermIO.WriteLn;
   END; (* IF *)

   (* Now, we can proceed to load all the children of the present
    * level, if there are any:
    *)

   INC( Level );

   Children := 0D;

   IF Level < Levels THEN

      WHILE Children < Count DO

         IF NOT
            LoadOneLevel(
               Stream,
               Sizes,
               Levels,
               NeedsByteSwap,
               Level,
               Children = 0D,
               Position )
         THEN
            RETURN FALSE;
         END; (* IF *)

         INC( Children );

      END (* WHILE *);

   END (* IF *);

   RETURN TRUE;

END LoadOneLevel;

(*
 * LoadTree
 *
 *    Scans a complete Tree.
 *    All "TermIO" statements are for demonstration purpose only.
 *    The variables are
 *           Stream        : the file handle to the open file
 *           Sizes         : the array returns the level sizes in
 *                           the Tree on disk.
 *           Levels        : the number of levels in the tree
 *           NeedsByteSwap : the flag telling, whether byte-swapping
 *                           is needed
 *
 *    The procedure returns TRUE, if it encountered no errors.
 *)

PROCEDURE LoadTree(
             VAR Stream          : IOFiles.FileHandleType;
             VAR Sizes           : ARRAY OF INT32;
             VAR Levels          : LONGINT;
             VAR NeedsByteSwap   : BOOLEAN )
             : BOOLEAN;
```

```
VAR
   Value      : INT32;
   Position   : LONGINT;
   Size       : LONGINT;
   i          : INTEGER;
   Success    : BOOLEAN;

BEGIN

   (* We start at the beginning of the file. We keep the variable
    * "Position" containing the actual position in the file.
    *)

   Position := 0D;

   (* The first 4 bytes should contain the "MagicNumber", see above.
    * a variable of type INT32 is a 32-bit long, signed word.
    *)

   Size := SIZE( INT32 );

   IF NOT IOBytes.Read( Stream, Size, ADR(Value) ) THEN
      Alert.IOError( '2-Error' );
      RETURN FALSE;
   END;

   IF Value = MagicNumber THEN
      NeedsByteSwap := FALSE;
   ELSIF Value = SwappedMagicNumber THEN
      NeedsByteSwap := TRUE;
   ELSE
      Alert.OK( '3-Error: File does not start with "Tree" !' );
      RETURN FALSE;
   END; (* IF *)

   (* The file pointer increased by 4 bytes: *)
   INC( Position, 4 );

   (* Next we load the number of levels in the Tree, which is stored
    * in the next 4 bytes (at offset 4):
    *)

   Size := SIZE( INT32 );

   IF NOT IOBytes.Read( Stream, Size, ADR(Levels) ) THEN
      Alert.IOError( '4-Error' );
      RETURN FALSE;
   END;

   (* The file pointer increased by 4 bytes: *)
   INC( Position, 4 );

   (* If the file originates from a platform with opposite byte ordering,
    * then we have to swap the bytes:
    *)

   IF NeedsByteSwap THEN Levels := SwappedInt32( Levels ); END;

   TermIO.WriteString( '  -> levels: ' );
   TermIO.WriteLongInt( Levels, 0 );

   (* The next bytes contain the sizes of all levels. Thus, there is
    * one 4-byte variable for each level, totaling in "Levels" times
    * 4 bytes.
    *
    * First, we check, if the array "Sizes" passed to this procedure
    * is large enough to contain all level sizes:
    *)

   IF ( Levels <= 0D ) OR ( Levels > LONG(HIGH(Sizes)+1) ) THEN
      Alert.OK( '5-Error: number of level either <= 0 or too large!' );
      RETURN FALSE;
   END (* IF *);

   (* Next, we load the "Level Size": *)

   Size := Levels * LONG( SIZE( INT32 ) );

   IF NOT IOBytes.Read( Stream, Size, ADR(Sizes) ) THEN
      Alert.IOError( '6-Error' );
      RETURN FALSE;
   END;

   (* The file pointer increased by "Size" bytes: *)
   INC( Position, Size );

   (* And we swap the bytes, if needed: *)

   IF NeedsByteSwap THEN
      FOR i := 0 TO SHORT( Levels - 1D ) DO
         Sizes[i] := SwappedInt32( Sizes[i] );
      END; (* FOR *)
   END; (* IF *)

   TermIO.WriteString( ';  sizes: ' );
   FOR i := 0 TO SHORT( Levels - 1D ) DO
      TermIO.WriteLongInt( Sizes[i], 0 );
      IF i < SHORT( Levels - 1D ) THEN
         TermIO.WriteString( ', ' );
      END; (* IF *)
   END; (* FOR *)

   TermIO.WriteString( ';  swap: ' );
   TermIO.WriteBoolean( NeedsByteSwap );
   TermIO.WriteLn;

   (* Now, the tree data follow.
    * We can load them by a recursive procedure:
    *)

   Success :=
      LoadOneLevel(
         Stream,
         Sizes,
         Levels,
         NeedsByteSwap,
         0D,
         TRUE,
         Position );

   IF Success THEN
      TermIO.WriteString( 'total file length: ' );
      TermIO.WriteLongInt( Position, 0 );
   END; (* IF *)
   TermIO.WriteLn;
   TermIO.WriteLn;

   RETURN Success;

END LoadTree;

VAR
   FileName        : IOFiles.FileNameType;
   Path            : IOFiles.FileNameType;
   Stream          : IOFiles.FileHandleType;
   Sizes           : ARRAY[0..9] OF INT32;
   Levels          : LONGINT;
   NeedsByteSwap   : BOOLEAN;
   Success         : BOOLEAN;
   Dummy           : BOOLEAN;

BEGIN

   (* Get a filename of a tree file to load: *)

   FileName[0]     := 0C;
   Path            := 0C;

   IF NOT
      FileSelect.Select(
         FileName,
         Path,
         '*.*',
         FileSelect.ExistingFile,
         'Select the TREE file to scan:' )
   THEN
      RETURN;
   END; (* IF *)

   Strings.Insert( FileName, Path, 0 );

   TermIO.DoBuffer := TRUE;
   TermIO.WriteLn;
   TermIO.WriteLine( FileName );

   (* Open the file : *)

   IF NOT IOBytes.Open( FileName, IOFiles.Read, Stream ) THEN
      Alert.IOError( '1-Error' );
      RETURN;
   END;

   (* Now, load the "Tree" : *)

   Success := LoadTree( Stream, Sizes, Levels, NeedsByteSwap );

   (* And, finally, we are done and can close the file. *)

   Dummy   := IOBytes.Close( Stream );

END FileFormat.
```

Appendix V: Lookup Tables

If you are using a *"Telegraphing"* amplifier like the *Axoclamp 2A*, the software reads all possible gain settings from the so called *"I-Gain"* table and all filter settings from the *"Filter Bandwidth"* table. These tables are ASCII text files located inside the folder Pulse\LookupTables by default, with each line representing a pair of *minimal voltage* the amplifier sends to the program plus the *actual value* of the corresponding amplifier gain or filter setting. For example the following two lines in an *I-Gain* lookup table

```
6.3 500.0
5.8 200.0
```

will be interpreted the following way: a voltage **higher** than *6.3 V* corresponds to the setting *500 mV/pA*, a value **higher** than *5.8 V* (and lower 6.3 V) is *200 mV/pA*. For more details please refer to Chapter *Configuration*.

I-Gain Lookup Tables

EPC7 Amplifier (default table):

File IGainTable_EPC7

```
  9.5      0.5
  7.0      1.0   ; i.e. 1 mV/pA (milli-volts per pico-amperes)
  5.0      2.0
  3.0      5.0
  1.0     10.0
 -1.0     20.0
 -3.0     50.0
 -5.1    100.0
 -7.0    200.0
 -9.5    500.0
-12.0   1000.0
```

Dagan 3900A Amplifier:

File IGainTable_Degan 3900A

```
3.0   500   ; i.e. 500 mV/pA (milli-volts per pico-amperes)
2.6   100
2.2   50
1.8   20
1.4   10
1.0   5
0.6   2
0.2   1
```

Warner PC-501A Amplifier:

File IGainTable_Warner PC-501A

```
4.1   1000     ; switch position 100, 10 GΩ, in mV/pA
3.9   500      ; switch position  50, 10 GΩ
3.7   200      ; switch position  20, 10 GΩ
3.5   100      ; switch position  10, 10 GΩ
3.3   50       ; switch position   5, 10 GΩ
3.1   20       ; switch position   2, 10 GΩ
2.9   10       ; switch position   1, 10 GΩ
2.7   100      ; switch position 100, 1 GΩ
2.5   50       ; switch position  50, 1 GΩ
2.3   20       ; switch position  20, 1 GΩ
2.1   10       ; switch position  10, 1 GΩ
1.9   5        ; switch position   5, 1 GΩ
1.7   2        ; switch position   2, 1 GΩ
1.5   1        ; switch position   1, 1 GΩ
1.3   10       ; switch position 100, 100 MΩ
1.1   5        ; switch position  50, 100 MΩ
0.9   2        ; switch position  20, 100 MΩ
0.7   1        ; switch position  10, 100 MΩ
0.5   0.5      ; switch position   5, 100 MΩ
0.3   0.2      ; switch position,  2, 100 MΩ
0.1   0.1      ; switch position   1, 100 MΩ
```

I-Gain (V-Clamp) Lookup Tables

Default Table:

```
         7.3     500.0   ; i.e. 500 mV/nA
         6.3      50.0
         5.4       5.0
         4.5     200.0
         3.6      20.0
         2.7       2.0
         1.9     100.0
         0.9      10.0
         0.0       1.0
        -0.9      50.0
        -1.9       5.0
        -2.7       0.5
        -3.5      20.0
        -4.5       2.0
        -5.4       0.2
        -6.2      10.0
        -7.2       1.0
       -10.3       0.1
```

TEC-5 Amplifier:

File IGainVClampTable_TEC-05

```
         6.5     10.0   ; i.e. 10 mV/nA (milli-volts per nano-amperes)
         5.5      5.0
         4.5      2.0
         3.5      1.0
         2.5      0.5
         1.5      0.2
         0.5      0.125
        -1.0      0.1
```

TEC-10 Amplifier:

File IGainVClampTable_TEC-10

```
         4.5     10.0  ; i.e. 10 mV/nA (milli-volts per nano-amperes)
         3.5      5.0  ; this lookup table is for the "I-Gain, V-Clamp"
                         (not "I-Gain")
         2.5      1.0
         1.5      0.5
         0.5      0.2
       -10.0      0.1  ; this is the "else" case
```

Filter Bandwidth Lookup Tables

Dagan 3900A Amplifier:

File BandwidthTable_Dagan 3900A

```
    4.6   100000   ; in Hz, Dagan Integrating Patch Clamp 3900A
    4.2    50000
    3.8    20000
    3.4    10000
    3.0     5000
    2.6     2000
    2.2     1000
    1.8      500
    1.4      200
    1.0      100
    0.6       50
    0.2       20
```

Appendix VI: Controlling *POTPULSE*

Controlling POTPULSE from another Program

POTPULSE can be controlled from another program by a simple "batch file control" protocol. This "batch file control" protocol is simple, fast, and platform independent. The new control protocol allows to control the potentiostat over a network, even one with different platforms, such as Windows, OS/2, MacOS, or workstations. Thus, it is now possible to control the potentiostat from computers running a multitasking operating system which can create and read shared files (e.g., Windows 95, Windows NT, MacOS, etc.).

Controlling *POTPULSE* from another program is possible by communicating via two ASCII-files. The user writes the commands to one file (the "command" file) and *POTPULSE* communicates back by writing to a second file (the "response" file). The user program has write permission (plus sharing permission) on the "command" file it will write to. *POTPULSE* will access that file with read and shared permission only. The reverse is used on the second file, the "response" file: *POTPULSE* will have write and sharing permission, and the user program read permission only (plus sharing permission, of course).

The first line in the "command" file must contain one positive number (as ASCII, e.g., "+1234"). This "command index" is interpreted by *POTPULSE* as follows:

- If this number is zero or negative, *POTPULSE* does not execute the commands in the "command" file.
- If the number is larger than zero, *POTPULSE* will execute the instructions immediately. *POTPULSE* will write that number to the "response" file to flag execution once all commands have been executed.
- To prevent *POTPULSE* to execute the instructions more than once, *POTPULSE* will not execute any further commands until the "command index" value is changed by the user program.
- Every command plus the required parameters must be in one text line, i.e., terminated by a "CR" character code (any following linefeed character will be ignored).
- An empty string (i.e. a string starting with 0x000) ends the list of commands.
- *POTPULSE* writes the responses to the "response" file. In the first line of that file, *POTPULSE* writes the "command index". The following lines will contain the responses, if any, one response per line.
- The name of the "command" file must be "PotBatch.In", the one of the "response" file "PotBatch.Out". Both files will be inside the default folder of *POTPULSE*, usually the folder " *POTPULSE* ".

- A text string must be set within double quotes, when it contains non-alphanumeric characters (e.g. commas, colons, blank spaces, etc.). A filename with path should always be within double quotes!

Thus, communication would proceed as follows:

1. The user program is started first: It has to create a file in the " *POTPULSE* " folder named " PotBatch.In". It has to keep this file open with "write" and "shared" access permission.

2. Then *POTPULSE* is started: Enable the "Enable Batch Control" option. *POTPULSE* will open the file " PotBatch.In" with "read" and "shared" access permission. Also, *POTPULSE* will now create the "PotBatch.Out" file with "write" and "shared" access permission.

3. Now, *POTPULSE* will immediately execute the commands in the command file, provided that the "command index" is larger than zero. *POTPULSE* writes the "command index" and eventually any error and requested answer to the "response" file.

4. Next, the user switches back to the user program.

5. Any time the user program writes to the "command" file, *POTPULSE* will scan the command file and execute the commands, if the "command index" changed. Windows 95 and Windows NT: On computers running Windows 95 or Windows NT, *POTPULSE* will immediately execute the commands.
   MacOS: On computers running MacOS, *POTPULSE* will immediately execute the commands, if it is the front application. If *POTPULSE* is not the front application, then *POTPULSE* will get active when the front application calls the "Toolbox" routine "GetNextEvent" or "WaitNextEvent".

6. The user program can now read the "response" file. The first line should mirror the "command index". Subsequent text lines may contain responses and error messages (see list below). There are the following possibilities:
   - ? There is no further line in the response file. This means, that no error occurred during execution and that no response was requested.
   - ? There is additional text in the file. The user can easily recognize error messages, because the first character in an error message is a lower case letter. All other responses start with an upper case letter.

7. When the user program wants to issue new commands, it writes a new "command index" and the new commands to the "command" file, then continues with step 5.

Here is an example of such a command file:

```
1. line: "1234"
2. line: "SetVHold -0.080"
3. line: "GetCurrentRange"
4. line: "MakeAnError"
```

And this would be the content of the "response" file:

```
1. line: "1234"
```

```
2. line: "CurrentGain 1.000E+09"
3. line: "error_not_found"
```

Error Messages

Errors begin with lower case letters:

'error_syntax':	parameter missing or misspelled
'error_range':	parameter is out of allowed range
'error_not_found':	command is unknown
'error_ioerror':	error during an I/O-operation
'error_acquiring':	command not allowed while acquiring
‘error_playback':	command not allowed while macro playback
'error_unknown':	an unidentified error occurred

Implemented Commands and Messages for the PG 300 series

Messages send by *POTULSE*. They do not contain a response string.

Started	*POTPULSE* started communication link
Terminated	*POTPULSE* terminated communication
SeriesStart [gr se]	*POTPULSE* started acquiring a Series
SeriesEnd [gr se]	*POTPULSE* finished acquiring a Series [gr se] is the group and series indexes of the series. The 2 indexes are replaced by the string "NIL", if the acquisition was aborted.
SweepStart [gr se sw]	*POTPULSE* started acquiring a Sweep
SweepEnd [gr se sw]	*POTPULSE* finished acquiring a Sweep [gr se sw] is the group, series, and sweep indexes of the sweep. The 3 indexes are replaced by the string "NIL", if the acquisition was aborted.

“Get” operations: They have no parameters. The response always repeats the command string without the “Get” sub-string.

```
GetAmplifier
GetAmplifierMode
GetAmplModeSwitch
GetAnodicQ
GetAutoRangeMaximum
GetAutoRangeMinimum
GetCathodicQ
GetCellMode
GetCtrlBandwidth
GetCurrent
GetCurrentDensity
GetCurrentDensityRange
GetCurrentGain
GetCurrentRange
GetCurrFilter
GetCurrFilterResponse
GetECell
GetElectrodeMode
GetExternalInput
GetExtPreAmpActive
GetExtPreAmpPresent
GetICell
GetIRCompensation
```

```
GetIRCompensationOn
GetNotchFilter
GetOCPValue
GetOverload
GetOverloadDisk
GetOverloadRing
GetPG300SerialNo
GetPotential
GetRemoteOn
GetRingAmplPresent
GetStandbyPressed
GetStimulusFilter
GetVoltageFilter
```

"Set" operations: They have no or one parameter and do not return a response.

```
SetAmplifierMode         integer: 0,1,3
SetAmplModeSwitch        boolean: 0 or 1
SetAutoRangeMaximum      real:    0.0 to 1.0
SetAutoRangeMinimum      real:    0.0 to 1.0
SetCell
SetCtrlBandwidth         integer: 0 to 9
SetCurrent               real:    -10 to +10 [A]
SetCurrentDensity        real:    -10 to +10 [A]
SetCurrentDensityRange   real: -10 to +10 [A]
SetCurrentRange          integer: 0 to 12
SetCurrFilter            real:    70 to 16000 [Hz]
SetCurrFilterResponse    integer: 0,1
SetDiskAmplifier
SetElectrodeMode         integer: 0,1
SetExternalInput         integer: 0,1,2
SetIRCompensation        real:    0 to 1 MOhm
SetIRCompensationOn      boolean: 0 or 1
SetNotchFilter           boolean: 0 or 1
SetOCP
SetOCPValue
SetPotential             -10 to +10 [V]
SetRingAmplifier
SetStandby
SetStimulusFilter        integer: 0,1,2,3
SetStopAndStandby
SetVoltageFilter         integer: 0,1,2
```

Miscellaneous Operations

```
acknowledged (re-synchronize command index)
   syntax: "acknowledged"
      no parameters
      no response

ADRead
   syntax: "ADRead AD-channel"
      AD-channel: integer 0 to 7
      returns a real (read AD-value in volt)

ADRelease
   syntax: "ADRelease"
       no parameters
       returns "ADReleased" if board could be released
```

```
ADReserve
   syntax: "ADReserve"
       no parameters
       returns "ADReserved" if board could be reserved
       returns "ADReleased" otherwise

AutoZero
   syntax: "AutoZero"
      no parameters
      no response

Break
   syntax: "Break"
      no parameters
      no response

CloseFile
   syntax: "CloseFile"
      no parameters
      response:
         if POTPULSE is still acquiring, the command is ignored
         and the string "error_acquiring" is returned.
         Otherwise, the data file is closed, and all files
         are flushed to disk.

DAWrite
   syntax: "DAWrite  AD-channel  Volts"
      DA-channel: integer, 0 to 3
      Volts:      real,    +/-10.24 volt
      no response

DigGet
   syntax: "DigGet"
      no parameters
      returns an integer: 0..65535

DigSet
   syntax: "DigSet  Mask  Word"
      Mask:       integer, 0 to 16383
      Word:       integer, 0 to 16383
      no response

DoExport
   syntax: "DoExportFull  Overwrite  Filename"
      Overwrite:  integer, 0 or 1
      Filename:   string, may include a path
      no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

DoExportFull
   syntax: "DoExportFull  Overwrite  Filename"
      Overwrite:  integer, 0 or 1
      Filename:   string, may include a path
      no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

ExecMacro
   syntax: "ExecMacro  MacroIndex"
      MacroIndex: integer, 0 to 19
      no response
```

```
GetOnline1
   syntax: "GetOnline1"
      no parameters
      no response

GetOnline2
   syntax: "GetOnline2"
      no parameters
      no response

GetSeriesOnline
   syntax: "GetSeriesOnline"
      no parameters
      returns as many text lines as there are sweeps in the target
         series. Each line has 4 reals: X1, Y1, X2, Y2

GetSweepName
   syntax: "GetSweepName"
      no parameters
      returns the name of the sweep

GetSweepStart
   syntax: "GetSweepStart"
      no parameters
      returns

GetTime
   syntax: "GetTime"
      no parameters
      returns system time

GetVersion
   syntax: "GetVersion"
      no parameters
      returns version of Potpulse

MoveTarget
   syntax: "MoveTarget  Direction  Moves"
      Direction:  integer, 0 to 5 -> left, right, up, down, HOME, END
      Moves:      integer, 1 to 5, optional, default = 1
      no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

NewGroup
   syntax: "NewGroup"
      no parameters
      response:
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, a new Group is created in the Tree (if the
         last group is not empty).

NotebookClear
   syntax: "NotebookClear"
      no parameters
      no response
```

```
NotebookWrite
   syntax: "NotebookWrite “Text”"
      Text: the string (within double quotes) to write to the
         Notebook
      no response

OpenFile
   syntax: "OpenFile “FileName”"
      parameter: file name (within double quotes),
         with or without a path.
      response:
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, "CloseFile" is called to close all files, and
         the file with the given name is opened. The file is
         opened in "modify" mode, if it exists. Otherwise a new
         file is created.
         It returns the error "error_ioerror", if the file could
         not be opened.

Query
   syntax: "Query"
      no parameters
      returns:
         "Acquiring" -> when still acquiring sweeps
         "Disabled"  -> when the amplifier window is active, but
                        POTPULSE is not the foremost application and
                        "Enable Background" is disabled.
         "Waiting"   -> when "RestrictWindows" is set and the active
                        window of POTPULSE is NOT one of the fol-
lowing:
                        - the amplifier window
                        - the oscilloscope window
                        - the online window
                        - the parameters window.
         "Ready"     -> otherwise.

Reset
   syntax: "Reset"
      no parameters
      no response

Reset Charge
   syntax: "ResetCharge"
      no parameters
      no response

Resume
   syntax: "Resume"
      no parameters
      no response

SetExportMode
   syntax: "SetExportMode Mode"
      Mode: integer 0 to 2
      no response

SetExportType
   syntax: "SetExportType Type"
      Type: integer 0 to 8
      no response
```

```
SetRestrictWindows
   syntax: "SetRestrictWindows Mode"
      Mode: boolean: 0 or 1
      no response

SetSleep
   syntax: "SetSleep Time"
      Time: real:    [s]
      no response

SetSubtract
   syntax: "SetSubtract Mode"
      Mode: integer: 0..3
      no response

SetWaitResume
   syntax: "SetWaitResume Wait"
      Mode: integer: 0..2
      no response

SetOnline1           2 required and 1 optional integer
   syntax: "SetOnline1  mode  abscissa  math"
      mode:       integer (required): 0..24
      abscissa:   integer (required): 0..9
      math:       integer (optional)" 0..4
   no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

SetOnline2           2 required and 1 optional integer
   syntax: "SetOnline2  mode  abscissa  math"
      mode:       integer (required): 0..24
      abscissa:   integer (required): 0..9
      math:       integer (optional)" 0..4
   no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.

SetTarget            3 optional integer
   syntax: "SetTarget  group-no  series-np  sweep-no"
      group-no:   integer (optional), 0 means “previous selection”
      series-no:  integer (optional), 0 means “previous selection”
      sweep-no:   integer (optional), 0 means “previous selection”
   no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
   Examples:
      "SetTarget"          -> select the root
      "SetTarget 3"        -> select 3. group
      "SetTarget 3, 2"     -> select 2. series of 3. group
      "SetTarget 3, 2, 1"  -> select 1. sweep of 2. series of
                              3. group
ShowTarget
   syntax: "ShowTarget "
      no parameters
      no response
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
```

```
Shutdown
   syntax: "Shutdown"
      no parameters
      response:
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, the string "Shutdown" is returned, and then
         POTPULSE shuts down.

SwitchToEpc9
SwitchToOnline
SwitchToOsci
SwitchToParams
   syntax: "SwitchToXXXX"
      no parameters
      no response

SwitchToPG300


Terminate
   syntax: "Terminate"
      no parameters
      returns the string "Terminated"

UpdateFile
   syntax: "UpdateFile"
      no parameters
      response:
         if POTPULSE is still acquiring, the command is ignored and
         the string "error_acquiring" is returned.
         Otherwise, the all files are flushed to disk.

Key
   (only "dialog" keys, not menu keys)
   syntax: "Key  Key-Code  ASCII-code"

   Key-Codes:
      2 -> Normal character input

      cursor keys:
      3 -> KeyCursorUp
      4 -> KeyCursorDown
      5 -> KeyCursorLeft
      6 -> KeyCursorRight

      numeric keypad:
      7 -> 0 (zero) on numeric keypad
      8 -> 1 on numeric keypad
      9 -> 2 on numeric keypad
     10 -> 3 on numeric keypad
     11 -> 4 on numeric keypad
     12 -> 5 on numeric keypad
     13 -> 6 on numeric keypad
     14 -> 7 on numeric keypad
     15 -> 8 on numeric keypad
     16 -> 9 on numeric keypad
     17 -> Enter
     18 -> minus  on numeric keypad
     19 -> plus   on numeric keypad
     20 -> period on numeric keypad
     21 -> clear  on numeric keypad (above '7')
```

```
22 -> '='    on numeric keypad (above '8')
23 -> '/'    on numeric keypad (above '9')
24 -> '*'    on numeric keypad

 function keys (F1 to F15):
24 + function key number, e.g., 25 for F1

 key block above cursor keys:
40 -> HOME
41 -> END
42 -> PageUp
43 -> PageDown
44 -> HELP
45 -> Delete Left
46 -> Delete Right
```

Notes for Programmers

One has to diligently select when to open the message file. That file is created by *POTPULSE*. Therefore it cannot and should not be opened before *POTPULSE* is running. Thus, the user program has to delete any message file it finds upon starting. That file may still be left around from a previous session. A good option is to create an empty command file in the directory where *POTPULSE* is located and to wait for the message file to appear in the *POTPULSE* directory. At that moment one can start *POTPULSE* and enable the option "Enable Batch Control" in the Potpulse drop-down menu. The message file will then be created and the user program can now open it for reading and sharing.

The first response *POTPULSE* writes to the message file is "started". Thereafter the user can proceed to send commands to *POTPULSE* and read back the response.

It is advisable to write to the command file in an analogous way as it is done in the example code below. In principle, one should pr oceed as follows:

1. Write a minus sign ("-") to the first byte of the command file. This prevents *POTPULSE* from interpreting anything in the file while the user programs writes to it. Please, recall that some operating systems are multitasking. This means that both programs, *POTPULSE* as well as the user program, run concurrently. Thus, *POTPULSE* may attempt to read from the message file while the user program is writing to it!

2. Write the remaining of the first text line. The first text line must be the signature number. The sample code below writes the negative value of the signature to achieve:

   - a negative sign is placed in the first byte of the file;
   - the signature value is written; and
   - the final, positive signature can be obtained by replacing the negative sign with a plus ("+").

3. Write to following text lines the required instructions, one instruction per text line.

4. Finally, replace the negative sign in the first byte of the file with a plus ("+"). This will signal to *POTPULSE* to proceed to read and interpret the command file.

5. Monitor the content of the message file. Do not interpret the content of the message file, as long as the first byte is a minus sign ("-") or the signature value in the first line is the one used in writing the last command.

6. *POTPULSE* writes responses to the message files in the following situations:
   - On starting "batch" processing, it messages "Started".
   - Responding to a "Get", "Query", or "Terminate" instruction.
   - When an error occurred while scanning the message file.

Glossary

term	synonymous	description
Electrochemical Convention		Anodic charge is due to the positive current flow through the cell. Cathodic charge, which is displayed as absolute value, is due to the negative current flow.
Experiment		Arbitrary number of groups that belong to the same series of measurement (organizational unit)
Group		Number of individual series with different sequences
Net Charge		Calculated by anodic minus cathodic charge
Pulse Generator Pool	Sequence Pool	Total amount of sequences in a pgf file
Relevant Segment		Part of a sequence on which a basic online analysis is done
Sample interval		Distance of the points in a measurement
Segment		Part of a sequence
Sequence	Pulse Pattern	Arbitrary number of pulse segments of constant, ramp, or sinusoidal voltage
Series		Number of individual sweeps based upon the same sequence
Stimulus template	Pulse Protocol Pulse Generator Protocol	Sequence plus additional settings
Sweep	Data Trace	One execution of a sequence
Target		Highlighted part of the data tree in the Replay window

Index